
بسم الله الرحمن الرحيم

دانشگاه حکیم سبزواری

دانشکده ریاضی و علوم کامپیوتر

پایان نامه جهت اخذ درجه کارشناسی ارشد در

ریاضی کاربردی، (گرایش تحقیق در عملیات)

عنوان

بدست آوردن مسیرهای بهینه روی یک شبکه درختی

استاد راهنما

آقای دکتر مهدی زعفرانیه

استاد مشاور

آقای دکتر محمود امین طوسی

پژوهش و نگارش

مهناز ابارشی

مهر ۹۲

تقدیم به

* شمع فروزان زندگی

پدرم

* سنگ صبور لحظه‌ها

مادرم

* یار همیشگی

خواهرم

تشکر و قدردانی

حمد و سپاس به درگاه خداوندی که وجود انسان را به زیور علم و معرفت بیاراست و با شکرگزاری به درگاه ایزد منان که مرا در تهیه این رساله یاری نمود.
بر خود لازم می‌دانم که از زحمات بی‌دریغ پدر، مادر و خواهرم که در این مسیر محیطی آرام را برایم فراهم کردند تشکر کنم و همچنین از راهنمایی‌های ارزشمند استاد راهنمای بزرگوار جناب آقای دکتر مهدی زعفرانیه و استاد مشاور جناب آقای دکتر محمود امین طوسی که همچون چراغی راهنمای راهم بودند نهایت تشکر و قدردانی را داشته باشم .
در پایان نیز از خواهر عزیزم که در تمام مراحل مختلف این رساله مرا یاری‌کردند، قدردانی می‌نمایم.

دانشگاه حکیم سبزواری

سوگندنامه

به نام خداوند جان و خرد　　کزین برتر اندیشه بر نگذرد

اینک که به خواست آفریدگار پاک ، کوشش خویش و بهره‌گیري از دانش استادان و سرمایه‌هاي مادي و معنوي این مرز و بوم، توشه‌اي از دانش و خرد گردآورده‌ام، در پیشگاه خداوند بزرگ سوگند یاد مي‌کنم که در به کارگیري دانش خویش، همواره بر راه راست و درست گام بردارم. خداوند بزرگ، شما شاهدان، دانشجویان و دیگر حاضران را به عنوان داوراني امین گواه مي‌گیرم که ازهمه دانش و توان خود براي گسترش مرزهاي دانش بهره گیرم و از هیچ کوششي براي تبدیل جهان به جایي بهتر براي زیستن، دریغ نورزم. پیمان مي‌بندم که همواره کرامت انساني را در نظر داشته باشم و همنوعان خود را در هر زمان و مکان تا سر حد امکان یاري دهم. سوگند مي‌خورم که در به کارگیري دانش خویش به کاري که با راه و رسم انساني، آیین پرهیزگاري، شرافت و اصول اخلاقي برخاسته از ادیان بزرگ الهي، به ویژه دین مبین اسلام، مباینت دارد دست نیازم. همچنین در سایه اصول جهان شمول انساني و اسلامي، پیمان مي‌بندم از هیچ کوششي براي آباداني و سرافرازي میهن و هم میهنانم فروگذاري نکنم و خداوند بزرگ را به یاري طلبم تا همواره در پیشگاه او و در برابر وجدان بیدار خویش و ملت سرافراز، بر این پیمان تا ابد استوار بمانم.

تاییدیه صحت و اصالت نتایج

بسمه تعالی

اینجانب مهناز ابارشی به شماره دانشجویی ۹۰۲۳۱۳۳۰۲۱ رشته ریاضی کاربردی مقطع تحصیلی کارشناسی ارشد

تأیید می‌نمایم که کلیه نتایج این پایان‌نامه حاصل کار اینجانب و بدون هرگونه دخل و تصرف و موارد نسخه برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام در صورت اثبات خلاف مندرجات فوق به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مولفان و مصنفان. قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی ضوابط و مقررات آموزشی پژوهشی و انضباطی) با اینجانب رفتار خواهد شد و حق هر گونه اعتراض در خصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن مسئولیت هر گونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی صلاح (اعم از اداری و قضایی) به عهده اینجانب خواهد بود و دانشگاه هیچ‌گونه مسئولیتی در این خصوص نخواهد داشت .

نام و نام خانوادگی: مهناز ابارشی

تاریخ و امضا :

مجوز بهره برداری از پایان نامه

بهره برداری از این پایان‌نامه در چهار چوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود بلامانع است:

- بهره برداری از این پایان‌نامه برای همگان بلامانع است.
- بهره برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما بلامانع است.
- بهره برداری از این پایان نامه تا تاریخ ..... ممنوع است.

استاد راهنما : استاد راهنمای اول
تاریخ :
امضاء:

فرم چکیده پایان نامه دوره تحصیلات تکمیلی
دفتر تحصیلات تکمیلی

نام خانوادگی: ابارشی	**نام:**مهناز	**ش دانشجویی:** ۹۰۲۳۱۳۳۰۲۱
استاد راهنما: دکتر مهدی زعفرانیه	**استاد مشاور:**دکتر محمود امین طوسی	
دانشکده: ریاضی و علوم کامپیوتر	**رشته:** ریاضی کاربردی	**گرایش:** تحقیق در عملیات
مقطع: کارشناسی ارشد	**تاریخ دفاع:** ۹۲/۷/۱۷	**تعداد صفحه:** ـــ

عنوان پایان‌نامه: بدست آوردن مسیرهای بهینه روی یک شبکه درختی

کلید واژه‌ها: مکان‌یابی ، مسیر میانه، مسیر مرکزی، مسیر میانه مرکزی،هسته درختی

چکیده: مسائل مکانیابی یکی از مباحث بسیار مهم و کاربردی در حوزه مدیریت خدمات است. بسیاری از مراکز دولتی و غیر دولتی برای انجام پروژه‌های اجرایی خود از علم مکان‌یابی استفاده می‌کنند. برای مثال ساختن یک بیمارستان و یا یک کارخانه در مرحله اول نیازمند تعیین مکان ساخت است. همچنین تعیین مسیر عبور شبکه فاضلاب شهری یا شبکه‌های حمل و نقل مانند مترو در زمره مسائل مکان‌یابی قرار می‌گیرند. در این‌گونه مسائل هدف تعیین مسیری است که تا حد امکان کم‌ترین فاصله ممکن را تا مشتری‌های موجود داشته باشد. مسائل مکان‌یابی شبکه در حقیقت مطالعه چگونگی قرار دادن یک یا چند سرویس‌دهنده و یا مسیر عبور یک سرویس‌دهنده در یک شبکه است به گونه‌ای که مشتری‌های موجود را به بهترین شکل سرویس‌دهی کند. در این پایان‌نامه چهار نوع مسئله مکان‌یابی تحت عنوان مسائل مسیر میانه، مسیر مرکزی، مسیر میانه مرکزی و مسیر میانه درختی را در یک شبکه با یال‌ها و رئوس وزن‌دار و غیر وزن‌دار بررسی می‌کنیم. در مسئله مسیر میانه، هدف پیدا کردن کوتاهترین مسیری است که مجموع فاصله وزنی رئوس شبکه از آن کمترین مقدار ممکن باشد. در مسئله مسیر مرکزی هدف پیدا کردن مکان بهینه مسیر به‌گونه‌ای است که حداًکثر فاصله مشتری‌ها از آن کمترین مقدار ممکن باشد. در مسئله مسیر میانه مرکزی، هدف پیدا کردن محل بهینه یک مسیر با استفاده از ترکیب معیارهای ماکزیمم فاصله و میانگین فاصله مشتری‌ها از سرویس‌دهنده‌ها است. مسیر میانه درختی عبارت است از پیدا کردن زیر درختی با k برگ و طول حداًکثر l به گونه‌ای که مجموع فاصله وزنی رئوس درخت تا آن کم‌ترین مقدار ممکن باشد.

امضای استاد راهنما

فهرست مطالب

فهرست اشکال

مقدمه

فرض کنیم یک مجموعه از مشتری‌ها در یک ناحیه جغرافیایی موجودند و هرکدام از آن‌ها میزان تقاضایی دارند که باید برآورده شود. هدف ما در مسائل مکان‌یابی در حالت کلی تعیین موقعیت یک و یا گاهی چند سرویس‌دهنده در این ناحیه است، به‌گونه‌ای که تابع خاصی از فاصله ما بین سرویس‌دهنده و نقاط تقاضا را بهینه کند.به‌عنوان مثالی از این سرویس‌دهنده‌ها می‌توان به طراحی مسیر حرکت شهری، خیابان‌های داخل شهر یا جاده‌های برون‌شهری، مسیر لوله‌های آب و فاضلاب اشاره کرد.

یکی از مهم‌ترین مسائل مکانیابی شبکه، مسئله هسته است که در این مسأله هدف پیدا کردن کوتاه‌ترین مسیری است که مجموع فاصله وزنی رئوس شبکه از آن کم‌ترین مقدار ممکن باشد [۱]. در نتیجه با واقع کردن یک سرویس‌دهنده بر روی آن دسترسی مشتری‌ها به سرویس‌دهنده تا حد امکان تسهیل می‌شود. در واقع بر خلاف مسئله میانه-p که سرویس‌دهنده‌ها بر روی چند نقطه جدا از هم قرار می‌گیرند در مسئله هسته سرویس دهنده یک مجموعه پیوسته است که بر روی یک مسیر قرار می‌گیرد.

گاهی در برخی مواقع در حل مسائل مکان‌یابی، هدف تسهیل دسترسی مشتری‌هایی است که در فاصله زیادی از سرویس‌دهنده‌ها قرار دارند. این نوع مسئله مکان‌یابی عبارت است از پیدا کردن مکان بهینه مسیر به‌گونه‌ای که حداکثر فاصله مشتری‌ها از آن کم‌ترین مقدار ممکن باشد. چنین مسیری را مسیرمرکزی می‌نامیم. این مسئله در سال ۱۹۸۱ توسط کوکاینه و همکارانش بررسی شد و آن‌ها توانستند الگوریتمی خطی برای حل آن ارائه دهند [۲]. همچنین این مسئله در سال ۱۹۸۲ توسط اسلیتر نیز بررسی شد و او نیز یک الگوریتم خطی برای آن ارائه داد [۳].

دو معیار اصلی که اغلب در تئوری مکان‌یابی شبکه برای تعیین محل بهینه یک وسیله مورد استفاده قرار می‌گیرد عبارت است از: ماکزیمال فاصله بین وسیله تا مشتری‌ها و میانگین فاصله بین وسیله و مشتری‌های مورد نظر[۴].

نوع دیگری از مسائل مکان‌یابی که به آن پرداخته شده، پیدا کردن محل بهینه یک مسیر با استفاده از ترکیب معیارهای ماکزیمم فاصله و میانگین فاصله مشتری‌ها از سرویس‌دهنده‌ها است. چنین مسیری،مسیر میانه مرکزی نام دارد [۵]. در واقع این مسئله ترکیبی از مسائل مسیر میانه و مسیر مرکزی است. به عنوان مثالی از چنین مسائلی می‌توان به طراحی مسیر خط‌های ریل راه آهن، مسیر راه‌هایی هوایی، مسیر خط‌های لوله آب و فاضلاب اشاره کرد. در مقالات بسیاری ثابت شده است که با وجود این‌که تعداد رأس‌های درخت (n) است، تعداد نمایش‌های مسیرهای بهینه $(n-۱)$ است.

مسئله هسته درختی نوع دیگری از مسائل مکان‌یابی شبکه است که در این نوع مسائل هدف پیدا کردن زیر درختی با دقیقاً k برگ است به‌گونه‌ای که مجموع فاصله رئوس شبکه تا آن کم‌ترین مقدار ممکن باشد. این مسئله در سال ۱۹۹۲ توسط پنگ بررسی شد و او توانست الگوریتمی خطی برای آن ارائه دهد. نوع دیگری از مسئله هسته درختی نیز وجود دارد که در آن هدف پیدا کردن زیر درختی با دقیقاً k برگ و طول مقید l است. این مسئله در سال ۲۰۰۲ توسط بکر و همکارانش بررسی شد و آن‌ها توانستند دو الگوریتم با مرتبه‌های زمانی مختلف برای پیدا کردن آن ارائه دهند. مسئله هسته درختی در طراحی شبکه‌های ارتباطی با سرعت بالا مانند شبکه‌های اینترنتی بسیار کاربرد دارد.

- ایده اصلی این پایان‌نامه از مقاله‌های [۶]، [۷]، [۸] و [۹] گرفته شده است.

تاریخچه

مکان‌یابی یک مبحث مهم در علم مدیریت وتخصیص منابع است. مطالعات در علم مکان‌یابی مدرن رسماً در سال ۱۹۰۹ توسط آلفرد وبر [۱] آغاز شد[۱۰]. وی مسئله پیدا کردن مکان یک سرویس دهنده راکه مجموع فاصله آن تا چند مشتری کم‌ترین مقدار ممکن است را مورد بررسی قرار داد. ایزفلد [۲] یک روش تکراری برای مسئله وبر در سال ۱۹۳۷ پیشنهاد کرد. مکان‌یابی در سال ۱۹۶۴ تولد مجددی را توسط حکیمی [۳] تجربه کرد [۱]. حکیمی مسئله مکان‌یابی را برای پیدا کردن مکان گشت‌های پلیس در بزرگ‌راه‌ها و مناطق شهری مورد استفاده قرار داد. همچنین حکیمی در سال ۱۹۹۳ نشان داد که پیدا کردن مسیر بر روی شبکه یک مسئله NP-سخت است. مورگان واسلیتر [۴] در سال ۱۹۸۰ الگوریتمی خطی برای آن ارائه دادند [۲]. لازم به ذکر است که برخی از محققین طبقه‌بندهایی از مدل‌های مکان‌یابی ارائه داده‌اند. اولین طبقه‌بندی مدل‌های مختلف مکان‌یابی توسط هندلر [۵] ومیرچندانی [۶] ارائه شد[۱۱] و پس از آن هاماخر [۷]و نیکل [۸] و هانسن [۹] مدل‌هایی را ارائه دادند[۱۲]. هندلر الگوریتمی برای پیدا کردن مرکز درخت ارائه داد. مسئله پیدا کردن مسیر مرکزی یک درخت را کوکاینه [۱۰] و همکاران در سال ۱۹۸۱ مورد بررسی قرار داده و الگوریتم خطی برای آن ارائه دادند[۱۳]. در سال ۱۹۸۲ نیز این مسئله توسط اسلیتر [۱۱] بررسی شد. همچنین در سال ۱۹۹۰ بکر الگوریتمی برای پیدا کردن هسته درخت ارائه داد [۱۴]. در سال ۱۹۹۹ برمن و اورباخ [۱۲] روشی برای پیدا کردن مسیر میانه مرکزی یک درخت ارائه دادند [۱۵].

مسئله هسته درختی که در سال ۱۹۹۳ توسط پنگ [۱۳] معرفی شد و او الگوریتمی با مرتبه زمانی

[۱] $Weber$ [۲] $Weiszfeld$ [۳] $Hakimii$ [۴] $Mirchandani$ [۵] $Handler$ [۶] $Mirchandani$ [۷] $Hamacher$
[۸] $Nichel$ [۹] $Hansen$ [۱۰] $Cocckayne$ [۱۱] $Slater$ [۱۲] $Berman\ and\ Averbakh$ [۱۳] $Peng$

$O(kn)$ برای پیدا کردن آن ارائه داد [۱۶]. همچنین این مسئله در سال ۱۹۹۷ توسط شئورا و یونو [۱] نیز بررسی شد و آن‌ها نیز الگوریتمی خطی برای یافتن آن ارائه دادند [۱۷].

[۱] *Shioura and Uno*

فصل ۱

تعاریف و مفاهیم اولیه

۱.۱ تعاریف مکان‌یابی

تعریف ۱.۱. (شبکه[۱]) یک شبکه $G = (V, E)$ شامل یک جفت مجموعه مجزا مانند V شامل مجموعه رئوس [۲] شبکه و E مجموعه یال‌های [۳] شبکه است. وزن [۴] هر رأس $v \in V$ را با $w(v)$ نشان می‌دهیم . یال $< v_i, v_j >$ یالی است که رئوس v_i و v_j را به یکدیگر متصل می‌کند و طول [۵] آن را با l_{ij} نشان می‌دهیم.

تعریف ۲.۱. (مسیر[۶]) به دنباله‌ای از رئوس غیر تکراری مانند $v_۱, v_۲, \cdots, v_n$ که توسط یال‌هایشان به یکدیگر متصل هستند مسیر می‌گوییم.

تعریف ۳.۱. (فاصله[۷]) فاصله دو رأس u و v را با $d(u, v)$ نشان می‌دهیم، که برابر با طول کوتاهترین مسیری است که این دو رأس را به هم متصل می‌کند.

تعریف ۴.۱. (دور[۸]) مسیری است که ابتدا و انتهایش بر هم منطبق باشد.

تعریف ۵.۱. (شبکه همبند[۹]) شبکه G را همبند می‌گوییم، اگر بین هر دو رأس متمایزش حداقل یک مسیر وجود داشته باشد.

تعریف ۶.۱. (درخت[۱۰]) شبکه همبند G را که فاقد دور باشد، درخت گوییم.

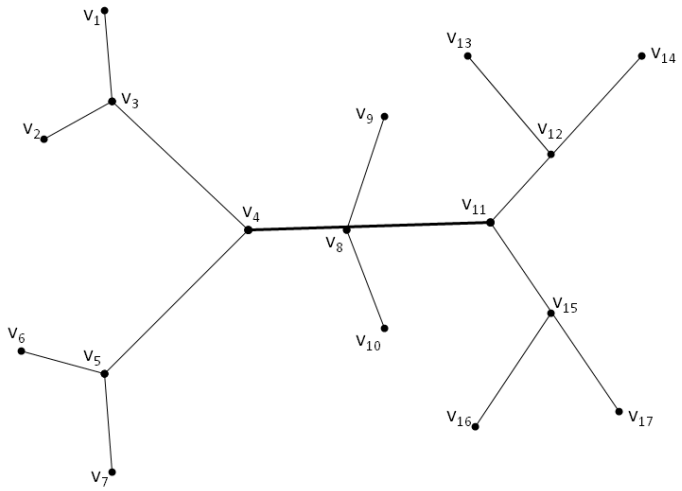
[۱] $Graph$ [۲] $Vertex$ [۳] $Edge$ [۴] $Weight$ [۵] $Length$ [۶] $Path$ [۷] $Distance$ [۸] $Cycle$ [۹] $Connected$ [۱۰] $Tree$

تعریف ۱.۷. (برگ[۱]) رأسی از درخت که تنها متصل به یک یال باشد را برگ نامیم.

تعریف ۱.۸. فرض کنید $T = (V, E)$ یک شبکه درختی باشد که V مجموعه رئوس آن با $|V| = n$ و E مجموعه یال‌های آن باشد، بطوری‌که طول تمام یال‌ها غیر صفر و وزن‌های نامنفی $w(v)$ متناظر با رأس‌های v باشد. اکنون برای هر رأس دلخواه v مجموعه ای پیوسته از $T - v$ را با v-شاخه [۲] نشان می‌دهیم. برای هر رأس دیگر مانند q مجموعه $B(v, q)$ رابه عنوان یک v- شاخه منحصر بفرد در نظر می‌گیریم که شامل q است و $N(v, q)$ را مجموعه رأس‌های آن تعریف می‌کنیم.

تعریف ۱.۹. (درجه رأس[۳]) درجه هر رأس برابر با تعداد رأس‌هایی است که با آن همسایه [۴] هستند و مجموعه تمام این رأس‌ها را با $NEI(v)$ نشان می‌دهیم .

تعریف ۱.۱۰. (قطر[۵]) قطر یک درخت برابر با طول بلندترین مسیر در درخت است .



شکل ۱.۱ : مثال یافتن v-شاخه در یک درخت

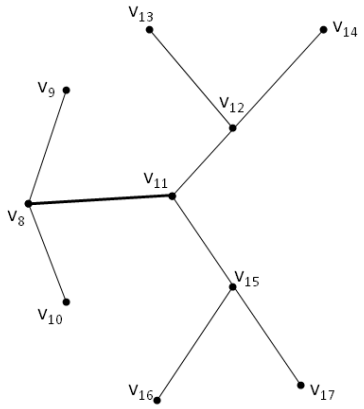
مثال ۱. برای درک بهتر تعاریف قبل به شکل ۱.۱ توجه کنید . در این شکل درختی با ۱۷ رأس و ۱۶ یال وجود دارد. رأس $v_۴$ را در نظر بگیرید. $v_۴$-شاخه مجموعه‌ای بصورت زیر است :

$$T - v_۴ = \{v_۱, v_۲, v_۳, v_۵, v_۶, v_۷, v_۸, v_۹, v_{۱۰}, v_{۱۱}, v_{۱۲}, v_{۱۳}, v_{۱۴}, v_{۱۵}, v_{۱۶}, v_{۱۷}\}$$

[۱] $Leaf$ [۲] $V - Branch$ [۳] $Degree$ [۴] $Adjacent$ [۵] $Diameter$

اکنون به عنوان کاندیدا برای رأس $v_۸$ ، $B(v_۴, v_۸)$ زیر درختی است شامل رأس $v_۸$ که رأس $v_۴$ را در بر ندارد. شکل ۲.۱ را ببینید. $N(v_۴, v_۸)$ مجموعه رأس‌های این زیر درخت است که در زیر آمده است:

$$N(v_۴, v_۸)=\{v_۸, v_۹, v_{۱۰}, v_{۱۱}, v_{۱۲}, v_{۱۳}, v_{۱۴}, v_{۱۵}, v_{۱۶}, v_{۱۷}\}$$



شکل ۲.۱: $B(v_۴, v_۸)$

تعریف ۱.۱۱. (مسیر مرکزی درخت[۱]) مسیر مرکزی درخت ؛ کوتاهترین مسیر مرکزگیری مینیمال است و آن را با PC که منحصر بفرد است نشان می‌دهیم .

تعریف ۱.۱۲. (مرکز مطلق[۲]) مرکز مطق درخت، نقطه میانی قطر درخت است و آن‌را با c نشان می‌دهیم.

نکته ۱. مرکز مطلق درخت (c) درون مسیر مرکزی (PC) درخت قرار دارد به عبارت دیگر $c \in PC$.

تعریف ۱.۱۳. فرض کنید Φ مجموعه‌ای از همه مسیرهای ساده درخت باشد ، برای هر مسیر $P \in \Phi$ و هر رأس $v \in V$ فاصله بین رأس و مسیر را بصورت زیر تعریف می‌کنیم :

$$d(P, v) = \min_{u \in P} d(u, v) \qquad (۱.۱)$$

[۱] $Path - Center$ [۲] $Absolut\ Center$

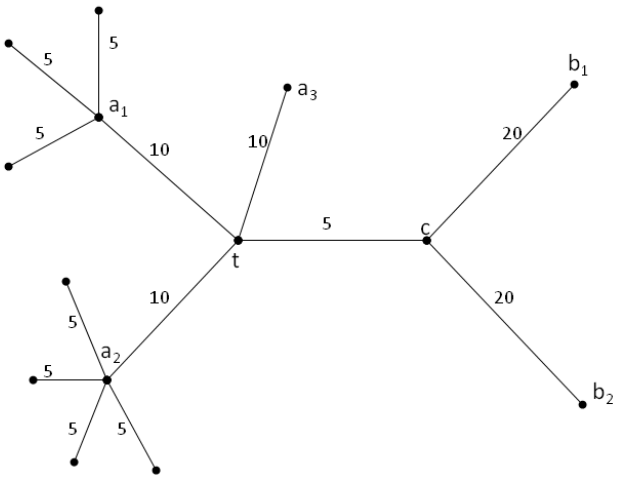
تعریف ۱.۱۴. به ازاء هر مسیر دلخواه $P \in \Phi$ فاصله ماکزیمال و فاصله متوسط را به ترتیب بصورت زیر تعریف می‌کنیم:

$$F(P) = Max(P) = \max_{v \in V} d(P, v) \tag{۲.۱}$$

$$G(P) = Mean(P) = \sum_{v \in V} w(v) d(P, v) \tag{۳.۱}$$

تعریف ۱.۱۵. (مسیر میانی[۱]) فرض کنید c_1 و c_2 رأس‌های ابتدا و انتهایی مسیر PC باشند، یعنی $PC = P(c_1, c_2)$. اکنون مسیر P یک مسیر میانی است اگر برای هر مسیر دیگری مانند P' داشته باشیم:

$$Mean(P) \leqslant Mean(P') \tag{۴.۱}$$



شکل ۳.۱: مثال تعیین رأس بحرانی برای یک مسیر

تعریف ۱.۱۶. (رأس بحرانی[۲]) به ازاء یک مسیر دلخواه مانند P رأس $a \in P$ یک رأس بحرانی برای مسیر P است، اگر رأس دیگری مانند $b \in V$ به قسمی وجود داشته باشد که:

$$d(b, a) = d(P, b) = Max(P) \tag{۵.۱}$$

[۱] $Path - Median$ [۲] $Critical - Vertex$

مثال ۲. رأس t در درخت شکل ۳.۱ یک رأس بحرانی برای مسیر $P(a_۱, a_۲)$ می‌باشد زیرا:

$$d(t, b_۱) = d(P(a_۱, a_۲), b_۱) = Max(P) = ۲۵$$

تعریف ۱.۱۷. برای رأس دلخواه a و a-شاخه، مقادیر زیر را تعریف می‌کنیم:

$$Q_۱(a, b) = \sum_{v \in N(a,b)} w(v) \tag{۶.۱}$$

عبارتست از مجموع وزنی رئوس متعلق به $B(a, b)$

$$Q_۲(a, b) = \sum_{v \in N(a,b)} d(v, a)w(v) \tag{۷.۱}$$

عبارتست از مجموع فاصله وزنی رئوس متعلق به $B(a, b)$ تا رأس a.

$$Q_۳(a, b) = \max_{v \in N(a,b)} d(v, a)w(v) \tag{۸.۱}$$

عبارتست از ماکزیمم فاصله وزنی رئوس متعلق به $B(a, b)$ تا رأس a

$$Q_۴(a, b) = \min_{v \in N(a,b)} \sum_{t \in N(a,b)} w(t)d(t, P(a, v)) \tag{۹.۱}$$

عبارتست از مینیمم فاصله وزنی بین رئوس متعلق به $B(a, b)$ و مسیرهای گذشته از رأس a

$$Q_۵(a, b) = Q_۲(a, b) - Q_۴(a, b) \tag{۱۰.۱}$$

عبارتست از فاصله ذخیره شده به وسیله مسیرهای گذشته از رأس a و رئوس متعلق $B(a, b)$

$$R(a) = \sum_{b \in NEI(a)} Q_۲(a, b) \tag{۱۱.۱}$$

عبارتست از مجموع فاصله‌ها تا رأس a

مثال ۳. اکنون به عنوان مثالی از فرمول‌های فوق، در درخت شکل ۳.۱ مقادیر $Q_i(t, c)$ و $R(t)$ را تعیین می‌کنیم. ابتدا باید $N(t, c)$ را تعیین کنیم که بصورت زیر است:

$N(t,c) = \{c, b_۱, b_۲\}$

$Q_۱(t,c) = \sum_{v \in N(t,c)} w(v) = ۳$

$Q_۲(t,c) = \sum_{v \in N(t,c)} d(v,a)w(v) = ۵ + ۲۵ + ۲۵ = ۵۵$

$Q_۳(t,c) = \max_{v \in N(t,c)} d(t,v)w(v) = ۲۵$

$Q_۴(t,c) = \min_{v \in N(t,c)} \sum_{u \in N(u,b)} w(u)d(u,P(u,v)) = ۲۰$

$Q_۵(t,c) = Q_۲(t,c) - Q_۴(t,c) = ۵۵ - ۲۰ = ۳۵$

$R(t) = \sum_{c \in NEI(t)} Q_۲(t,c) = Q_۲(t,a_۱) + Q_۲(t,a_۲) + Q_۲(t,a_۳) + Q_۲(t,c) =$

$۵۵ + ۷۰ + ۱۰ + ۵۵ = ۱۹۰$

تعریف ۱.۱۸. (مسیر میانه مرکزی[۱]) مسیر $P' \in \Phi$ یک مسیر پرتو بهینه [۲] یا یک مسیر میانه مرکزی است، اگر مسیر دیگری مانند $P \in \Phi$ وجود نداشته باشد به قسمی‌که:

$$Max(P) \leqslant Max(P') \qquad (۱۲.۱)$$

$$Mean(P) \leqslant Mean(P') \qquad (۱۳.۱)$$

یا حدأقل یکی از نامساوی‌ها اکید باشد .

تعریف ۱.۱۹. فرض کنید Π مجموعه‌ای از تمام مسیرهای پرتو بهینه باشد، برای هر مجموعه Q از این مسیرها مجموعه $\phi - representaion$ را به‌صورت زیر تعریف می‌کنیم:

$$\phi(Q) = \{(F,G) \in R^۲ | \exists P \in Q; F = Max(P), G = Mean(P)\} \qquad (۱۴.۱)$$

[۱] $Path - Medi - Center$ [۲] $Pareto - Optimal$

فصل ۲

مسائل مسیر میانه، مسیر مرکزی و مسیر میانه مرکزی

۱.۲ مقدمه و تاریخچه

این فصل شامل سه بخش است . در بخش اول به معرفی مسئله هسته می‌پردازیم که یکی از مهم‌ترین مسائل مکان‌یابی شبکه است. در واقع در این مسأله هدف پیدا کردن کوتاه‌ترین مسیری است که مجموع فاصله وزنی رئوس شبکه از آن کم‌ترین مقدار ممکن باشد. در نتیجه با واقع کردن یک سرویس‌دهنده بر روی آن دسترسی مشتری‌ها به سرویس‌دهنده تا حد امکان تسهیل می‌شود. در واقع بر خلاف مسئله میانه-p که سرویس‌دهنده‌ها بر روی چند نقطه جدا از هم قرار می‌گیرند در مسئله هسته سرویس دهنده یک مجموعه پیوسته است که بر روی یک مسیر قرار می‌گیرد. در سال ۱۹۸۰ مورگان و اسلیتر الگوریتمی را برای پیدا کردن هسته درخت ارائه دادند [۱۸] .در سال ۱۹۹۰ بکر الگوریتمی برای پیدا کردن هسته درخت ارائه داد. در سال ۱۹۹۲ نیز این مسئله توسط پنگ و همکارانش بررسی شد و آن‌ها نیز الگوریتمی خطی برای پیدا کردن هسته درخت ارائه دادند که دارای محاسبات کم‌تری نسبت به الگوریتم مورگان و اسلیتر است [۱۹].

در بخش دوم مسئله مسیر مرکزی درخت را مورد بررسی قرار می‌دهیم که در این مسئله هدف تسهیل دسترسی مشتری‌هایی است که در فاصله زیادی از سرویس‌دهنده قرار دارند. این نوع مسئله مکان‌یابی عبارت است از پیدا کردن مکان بهینه مسیر به‌گونه‌ای که حداکثر فاصله مشتری‌ها از آن کم‌ترین مقدار ممکن باشد. چنین مسیری را مسیرمرکزی می‌نامیم. این مسئله در سال ۱۹۸۱ توسط

کوکاینه و همکارانش بررسی شد و آن‌ها توانستند الگوریتمی خطی برای آن ارائه دهند. همچنین این مسئله در سال ۱۹۸۲ توسط اسلیتر نیز بررسی شد و او نیز یک الگوریتم خطی برای آن ارائه داد. هندلر الگوریتمی برای پیدا کردن مرکز درخت ارائه داد. مسئله پیدا کردن مسیر مرکزی یک درخت را کوکاینه [۱] و همکاران در سال ۱۹۸۱ مورد بررسی قرار داده و الگوریتم خطی برای آن ارائه دادند. در سال ۱۹۸۲ نیز این مسئله توسط اسلیتر [۲] بررسی شد.

در بخش سوم مسیر میانه مرکزی درخت را معرفی می‌کنیم، در واقع در این مسئله هدف پیدا کردن محل بهینه یک مسیر با استفاده از ترکیب معیارهای ماکزیمم فاصله و میانگین فاصله مشتری‌ها از سرویس‌دهنده‌ها است. چنین مسیری،مسیر میانه مرکزی نام دارد. در واقع این مسئله ترکیبی از مسائل مسیر میانه و مسیر مرکزی است. به عنوان مثالی از چنین مسائلی می‌توان به طراحی مسیر خطهای ریل راه آهن، مسیر راه‌هایی هوایی، مسیر خطهای لوله آب و فاضلاب اشاره کرد.

در سال ۱۹۹۹ برمن و اورباخ [۳] روشی برای پیدا کردن مسیر میانه مرکزی یک درخت ارائه دادند.

۲.۲ مسیر میانه(هسته)

۱.۲.۲ معرفی

یکی از انواع مسائل مکان‌یابی مسئله مسیر میانه(هسته) است. در این‌گونه مسائل هدف تعیین مکان قرار دادن یک سرویس دهنده مانند مسیر حرکت یک اتوبوس یا قطار بین شهری در شبکه است بطوری‌که تا حد امکان دسترسی نقاط متقاضی به آن تسهیل شود.

به‌عبارت دیگر در این مسأله، هدف تعیین مکان قرار دادن یک سرویس‌دهنده در شبکه $G = (V, A)$ به گونه‌ای است که، مجموع فاصله وزنی آن تا مشتری‌ها کم‌ترین مقدار ممکن باشد.

قبل از بیان الگوریتمی برای بدست آوردن هسته، به بیان برخی از تعاریف و نکات مورد نیاز می‌پردازیم.

تعریف ۱.۲. (مسیر میانه[۴]) هسته یک درخت عبارت از مسیری است که مجموع فاصله وزنی

[۱] $Cocckayne$ [۲] $Slater$ [۳] $Berman\ and\ Averbakh$ [۴] $Path - Median$

رئوس از آن کم‌ترین مقدار ممکن باشد.

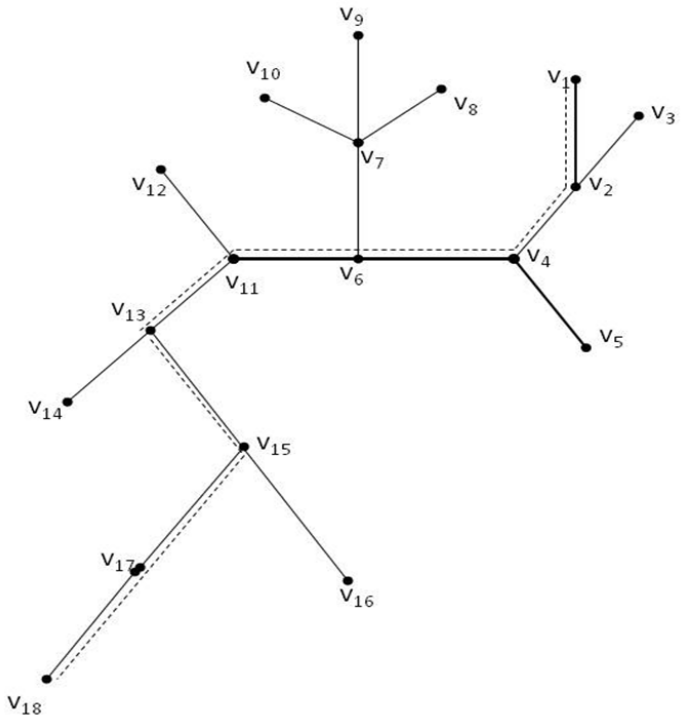
در نتیجه با واقع کردن یک سرویس‌دهنده بر روی آن دسترسی مشتری‌ها تا حد امکان تسهیل می‌شود. تابع هدف مسئله هسته بصورت زیر است:

$$F(P) = min \sum_{i=1}^{n} w_i d(P, v_i) \qquad (1.2)$$

که در آن P یک مسیر و وزن رئوس $w_i \geq 0$ است.

۲.۲.۲ خواص مسأله هسته

اگر رأس ابتدایی v_1 و رأس انتهایی v_n باشد، هسته را با P_{v_1,v_n} نمایش می‌دهیم .



شکل ۱.۲: هسته درخت

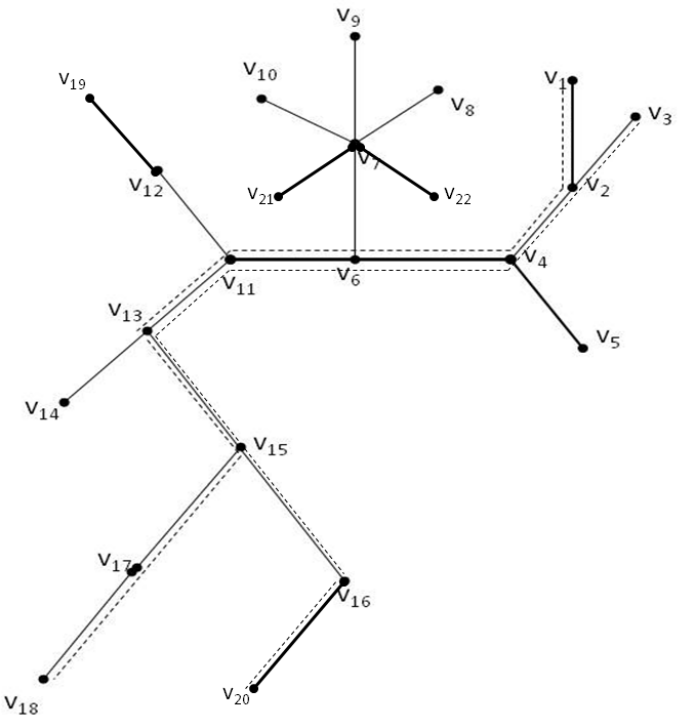
مثال ۴. در شکل ۱.۲ هسته درخت $P_{v_1,v_{18}}$ است که مقدار تابع هدف $F(P) = min \sum_{i=1}^{n} w_i d(v_i, P)$ به ازاء آن برابر با ۱۲ است. چون وزن رئوس و طول یال‌ها برابر یک است لذا فاصله دو رأس برابر با تعداد یال‌های بین آن دو است. توجه داشته باشید که فاصله رأس‌های متعلق به مسیر برابر صفر است.

$$F(P) = ۱ + ۱ + ۱ + ۱ + ۱ + ۱ + ۲ + ۲ + ۲ = ۱۲$$

لم ۱. رئوس ابتدایی و انتهایی هسته درخت، برگ هستند.

اثبات. اگر $P_۱$ یک زیر مسیر محض از مسیر $P_۲$ باشد، آن‌گاه $F(P_۱) < F(P_۲)$ است. در نتیجه هسته، مسیری است که از یک برگ شروع شده و تا یک برگ دیگر ادامه دارد. فرض کنیم در هسته P رأس ابتدایی برگ نباشد، بنابراین رأسی با درجه بیشتر از یک وجود دارد که v رأس همسایه آن است، لذا با گسترش مسیر به این رأس، مسیر P' ی بدست می‌آوریم که P زیر مسیر آن است. بنابراین $F(P') < F(P)$ و این با هسته بودن P در تناقض است. □

نکته ۲. هسته یک درخت لزوما یکتا نیست، یعنی امکان دارد دو مسیر متفاوت وجود داشته باشد در حالی‌که مجموع فاصله رئوس از هر دو آن‌ها به یک اندازه باشد.



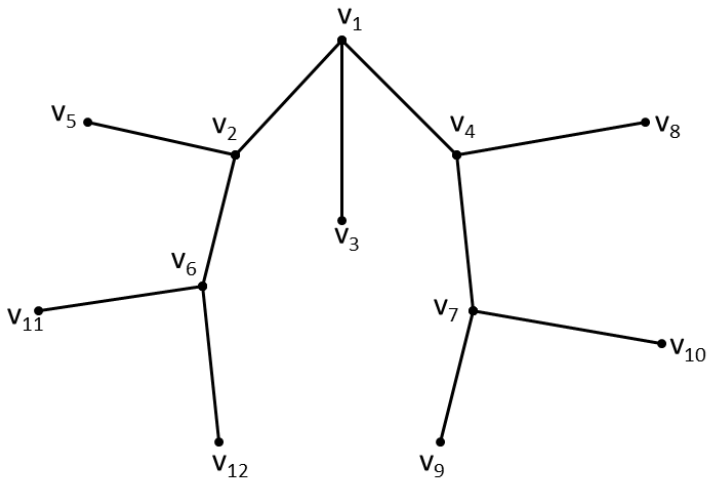
شکل ۲.۲: مثال هسته‌های متفاوت

مثال ۵. در شکل ۲.۲ دو هسته متفاوت $P_{v_۱,v_{۱۸}}$ ، $P_{v_۳,v_۲.}$ نشان داده شده است که مجموع فاصله رئوس از هر دو آن‌ها برابر ۲۰ است.

در سال ۱۹۸۰ مورگان و اسلیتر الگوریتمی را برای پیدا کردن هسته درخت ارائه دادند [۱۷]. با وجود این‌که این الگوریتم نسبت به تعداد رأس‌ها و یال‌ها خطی است اما دارای دو پیمایش است. در این فصل الگوریتم جدیدی را معرفی می‌کنیم که دارای یک پیمایش است و به محاسبات کم‌تری نسبت به الگوریتم مورگان و اسلیتر نیاز دارد [۱۲]. که قبل از ارائه آن برخی از تعاریف مورد نیاز را بیان می‌کنیم.

تعریف ۲.۲. (ریشه[۱]) رأسی که کلیه رأس‌های دیگر در پایین آن قرار دارند، ریشه نام دارد.

تعریف ۳.۲. (درخت ریشه‌دار شده[۲]) درخت Tرا در رأس دلخواه v ریشه‌دار شده می‌گوییم، اگر رأس v به عنوان ریشه و تمام رئوس دیگر به عنوان فرزندان آن قرار گرفته باشند. به عنوان مثال درخت ۳.۲ در رأس v_1 ریشه‌دار شده است.



شکل ۳.۲: درخت ریشه‌دار شده

تعریف ۴.۲. فرض کنید T یک درخت ریشه‌دار شده در یک رأس دلخواه باشد، در این‌صورت $V(T)$ معرف مجموعه رأس‌های درخت T و $|T|$ معرف تعداد رأس‌های درخت است.

تعریف ۵.۲. به ازای هر رأس $v \in V(T)$ فاصله از v را با $d(v) = \sum_{u \in V(T)} d(u,v)$ نشان می‌دهیم که $d(u,v)$ فاصله دو رأس دلخواه است.

[۱] *Roote* [۲] *Rooted Tree*

تعریف ۶.۲. به ازای هر مسیر دلخواه P فاصله از مسیر را به‌صورت زیر تعریف می‌کنیم:

$$d(P) = \sum_{u \in V(T)} d(P, u) \qquad (۲.۲)$$

که $d(P, u) = \min_{v \in P} d(u, v)$ است.

با توجه به این تعریف واضح است که هسته یک درخت، مسیری است که فاصله از آن کم‌ترین مقدار ممکن است.

تعریف ۷.۲. اگر $P_{v,l}$ مسیری از رأس v تا یک برگ مانند l باشد، آن‌گاه فاصله ذخیره شده به‌وسیله این مسیر را به‌صورت زیر تعریف می‌کنیم:

$$DSAV(P_{v,l}) = d(v) - d(P_{v,l}) \qquad (۳.۲)$$

که $DSAV(P_{v,l})$ بهترین مقدار برای انتخاب مسیر گسترش‌یافته از رأس v است.

تعریف ۸.۲. فرض کنید $P_{v,l}$ مسیری به‌صورت $P_{v,l} = v_۰, v_۱, \ldots, v_r$ باشد، بطوری‌که $v = v_۰$ ، $l = v_r$. دراین‌صورت زیر درخت آویزان شده روی v_i را با $Tr(v_i)$ نشان می‌دهیم. در واقع $Tr(v_i)$ زیر درخت تولید شده به‌وسیله مجموعه رأس‌های متصل شده به v_i به‌وسیله مسیرهای است که تنها شامل v_i و آن رأس‌هایی است که در مسیر $P_{v,l}$ نیستند. از این‌رو داریم:

$$DSAV(P_{v,l}) = \sum_{i=۱}^{r} i|Tr(v_i)| \qquad (۴.۲)$$

تعریف ۹.۲. در یک درخت ریشه‌دار، رأسی که بلافاصله زیر یک رأس قرار می‌گیرد، فرزند[۱] آن رأس نامیده می‌شود. رأس والد[۲] ، رأس دیگری است که بلافاصله بالاتر از آن نزدیک ریشه قرار داشته باشد.

قبل از فرمول‌بندی الگوریتم نیاز است توصیفی از چگونگی کارکرد آن ارائه دهیم. الگوریتم از برگ‌ها شروع کرده و به سمت داخل پیش می‌رود. در هرتکرار دو عمل برش درختی[۳] و به هنگام کردن [۴] انجام می‌شود که در ادامه آن‌ها را بیان می‌کنیم.

[۱] $Child$ [۲] $Parent$ [۳] $Tree - Cut$ [۴] $Update$

۳.۲.۲ عملیات برش درختی

این عملیات بر روی تمام برگ‌های درخت انجام می‌شود. به‌این‌صورت که برگ‌هایی که والدینشان فقط یک همسایه غیر برگی دارند، حذف می‌شوند. با این‌کار والدها به برگ‌های جدید تبدیل می‌شوند. مرتبط با هر رأس که این عمل روی آن انجام می‌شود یک بهترین مسیر انتخاب می‌شود، که این مسیر یا شامل رأس مورد نظر است یا در زیر درخت وابسته به آن رأس قرار دارد.
این عملیات تا زمانی ادامه می‌یابد که فقط ۱ یا ۲ رأس در درخت باقی بماند.

۴.۲.۲ عملیات به‌هنگام کردن

ابتدا برخی پارامترهای مورد نیاز عملیات به‌هنگام کردن را تعریف می‌کنیم.
فرض کنید $T(u)$ زیر درخت تولید شده به‌وسیله مجموعه رأس‌های متصل شده به u به وسیله مسیرهایی باشد که فقط شامل u و آن رأس‌هایی هستند که زمانی‌که رأس u در حال بهنگام شدن است، کنار گذاشته می‌شوند. در واقع $T(u)$ درخت شامل رأس u و مجموعه رأس‌هایی است که تا قبل از عملیات برش درختی به رأس u متصل هستند.
اکنون ۵ پارامتر وابسته به رأس دلخواه u را به‌صورت زیر تعریف می‌کنیم:

۱) $tT(u)$ معرف اندازه زیر درخت $T(u)$ است.

۲) $DSAV(u)$ و $SDSAV(u)$ به‌ترتیب بزرگ‌ترین و دومین بزرگ‌ترین فاصله ذخیره شده بوسیله مسیر $P_{u,l}$ از رأس u تا یک برگ مانند $l \in V(T)$ هستند که بصورت زیر تعریف می‌شوند:

$$DSAV(u) = \max_{l \in V(T(u))} \{DSAV(P_{u,l})\} = DSAV(P_{u,l_*}) \qquad (۵.۲)$$

$$SDSAV(u) = \max_{l \in V(T(u)), P_{u,l} \cap P_{u,l_*} = u} \{DSAV(P_{u,l})\} \qquad (۶.۲)$$

۳) $W(u)$ رأسی از مجموعه $V(T(u))$ است که برای پیدا کردن مسیری با بیشترین فاصله ذخیره شده در زیر درخت $T(u)$ استفاده می‌شود. اگر این مسیر شامل رأس u باشد $W(u) = u$ است.

۴) $PSAV(u, W(u))$ ماکزیمم فاصله ذخیره شده به‌وسیله مسیری در $T(u)$ است که شامل $W(u)$ نیز باشد و بصورت زیر تعریف می‌شود:

$$PSAV(u, W(u)) = \max\{DSAV(P_{W(u),l_1}) + DSAV(P_{W(u),l_2})\} \qquad (۷.۲)$$

که $DSAV(P_{W(u),l_i}) = d(W(u)) - d(P_{W(u),l_i}); i = ۱, ۲$ همچنین مسیرهای $P_{W(u),l_1}$، $P_{W(u),l_2}$; $l_1, l_2 \in V(T(u))$ مسیرهای مجزا هستند و فقط در رأس $W(u)$ با یکدیگر اشتراک دارند.

با استفاده از تعریف به راحتی می‌توان دید، مسیری که مقدار $PSAV(u, W(u))$ را نتیجه بدهد کمترین مقدار $d(P)$ را در میان همه مسیرهای P در $T(u)$ که از $W(u)$ عبور می‌کند دارد.

۵) $ADSA(u)$ اگر v_k رأس همسایه u باشد به قسمی‌که $W(u) \in V(T(v_k))$ است در این‌صورت :

$$ADSAV(u) = DSAV(P_{W(u),u} \cup P_{u,w}) - DSAV(P_{u,w}) \qquad (۸.۲)$$

که $P_{u,w}$ مسیر ماکزیمم‌سازی $d(u) - d(P_{u,l})$ در بین همه مسیرهای گذرنده از رأس u تا یک برگ مانند $l \in T(u) - T(v_k)$ است. همچنین باید داشته باشیم: $W(u) \neq u$.

۵.۲.۲ الگوریتم $Peng$ برای یافتن هسته درخت

این الگوریتم شامل ۳ گام اصلی بصورت زیر است:

عملیات بهنگام کردن رأس دلخواه u.

عملیات برش درختی.

برنامه اصلی.

۱) عملیات به‌هنگام کردن رأس u.

فرض کنید $(v_1, \ldots, v_r)$ مجموعه برگ‌های همسایه با رأس u باشند، در این‌صورت قرار می‌دهیم:

$$|T(u)| := \sum_{i=1}^{r} |T(v_i)| + 1$$

$$CSET(u) := \{|T(v_i)| + DSAV(v_i), 1 \leq i \leq r\}$$

همچنین فرض کنید $DSAV(u)$ و $SDSAV(u)$ به‌ترتیب اولین و دومین مقدار در $CSET(u)$ باشند. بویژه اگر $r = 1$ باشد آن‌گاه $SDSAV(u) = 0$ است.

اکنون فرض کنید v_k رأسی باشد که به ازای آن $|T(v_k)| + DSAV(v_k)$ بزرگ‌ترین مقدار در مجموعه $CSET(u)$ باشد. در این‌صورت دو حالت زیر را خواهیم داشت:

(a) اگر $SDSAV(u) + (n - |T(v_k)|) + ADSAV(v_k) < SDSAV(W(v_k))$ سپس قرار دهید:

$W(u) := W(v_k)$ و $PSAV(u, W(u)) := PSAV(v_k, W(v_k))$ و

$$ADSAV(u) := ((n - |T(v_k)|) + ADSAV(v_k)$$

(b) در غیر اینصورت قرار دهید $W(u) := u$ و $PSAV(u, W(u)) := DSAV(u)+SDSAV(u)$ و $ADSAV(u) = 0$ و یال e_{u,v_k} را از درخت T حذف کنید.

(۲) عملیات برش درختی به ازاء هر $u \in S$ عملیات بهنگام کردن u را انجام دهید.

(۳) برنامه اصلی. به ازاء هر برگ $l \in T$ که $|T(l)| = 1$ ، $W(l) = l$ و

$n := |T|$ ، $DSAV(l) = SDSAV(l) = PSAV(l, W(l)) = ADSAV(l) = 0$

عملیات زیر را تکرار کنید:

(a) قرار دهید: $S = \{u | u$ *یک رأس غیر برگی است و بیشتر از یک همسایه برگی غیر ندارد* $\}$

عملیات برش درختی را تا زمانی‌که $S \neq \phi$ روی مجموعه S انجام دهید. اگر تنها یک رأس v در T باقی مانده است آنگاه مسیری را که مقدار $PSAV(v, W(v))$ را بدست می‌دهد به عنوان هسته درخت معرفی کنید.

در غیر این‌صورت فرض کنید v_1 و v_2 دو رأس باقی مانده در درخت باشند و $P_{v_i,l_i}; i = 1, 2$ مسیرهایی باشند که به ترتیب مقادیر $DSAVE(v_i)$ را مشخص می‌کنند. در این صورت:

a) اگر $DSAV(v_1) + |T(v_1)| < DSAV(v_2)$

a۱) اگر $DSAV(v_1) + |T(v_1)| + ADSAV(v_2) < SDSAV(W(v_2))$ مسیری را که مقدار $PSAV(v_2, W(v_2))$ را بدست می دهد به عنوان هسته معرفی کنید.

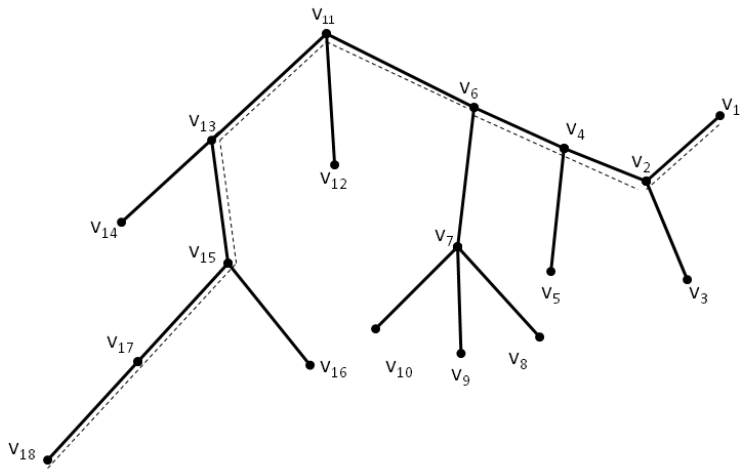
a۲) در غیر این صورت مسیر $P_{v_1,l_1} \cup e_{v_1,v_2} \cup P_{v_2,l_2}$ را به عنوان هسته معرفی کنید.

b) اگر $DSAV(v_1) + |T(v_1)| > DSAV(v_2)$

b۱) اگر $DSAV(v_2) + |T(v_2)| + ADSAV(v_1) < SDSAV(W(v_1))$ آنگاه مسیری را که مقدار $PSAV(v_2, W(v_2))$ را بدست می‌دهد به عنوان هسته معرفی کنید.

b۲) در غیر این صورت مسیر $P_{v_1,l_1} \cup e_{v_1,v_2} \cup P_{v_2,l_2}$ را به‌عنوان هسته معرفی کنید.

گام‌های الگوریتم کامل است.



شکل ۴.۲: مثال هسته درخت

مثال ۶. با استفاده از الگوریتم ۵.۲.۲، هسته درخت شکل ۴.۲ را پیدا کنید.

در تکرار اول، مجموعه $S = \{v_2, v_7, v_{17}\}$ در نتیجه عملیات برش درختی را روی این سه رأس اجرا می‌کنیم. نتایج بدست آمده در جدول ۱.۲ نشان داده شده است.در انتهای این گام یال‌های $(v_2, v_1), (v_2, v_3)$ و $(v_7, v_{10}), (v_7, v_9), (v_7, v_8)$ و (v_{17}, v_{18}) از درخت حذف می‌شوند.

جدول ۱.۲ : نتایج عملیات برش درختی در تکرار اول

v_i	$DSAV$	$SDSAV$	$W(v_i)$	$PSAV$	$ADSAV$
$v_۲$	۱	۱	$v_۲$	۲	۰
$v_۷$	۱	۱	$v_۷$	۲	۰
$v_{۱۷}$	۱	۰	$v_{۱۷}$	۱	۰

در تکرار دوم، مجموعه $S = \{v_۴, v_{۱۵}\}$ و عملیات برش درختی را روی این دو رأس انجام می‌دهیم. نتایج بدست آمده در جدول ۲.۲ نشان داده شده است. در نتیجه یال‌های $(v_۴, v_۲), (v_۴, v_۵)$ و $(v_{۱۵}, v_{۱۶}), (v_{۱۵}, v_{۱۷})$ از درخت حذف می‌شوند.

جدول ۲.۲ : نتایج عملیات برش درختی در تکرار دوم

v_i	$DSAV$	$SDSAV$	$W(v_i)$	$PSAV$	$ADSAV$
$v_۴$	۴	۱	$v_۴$	۵	۰
$v_{۱۵}$	۳	۱	$v_{۱۵}$	۴	۰

در تکرار سوم، مجموعه $S = \{v_۶, v_{۱۳}\}$ و عملیات برش درختی را روی این دو رأس انجام می‌دهیم. نتایج بدست آمده در جدول ۳.۲ نشان داده شده است. در نتیجه یال‌های $(v_۶, v_۴), (v_۶, v_۷)$ و $(v_{۱۳}, v_{۱۵})$،$(v_{۱۳}, v_{۱۴})$ از درخت حذف می‌شوند.

جدول ۳.۲ : نتایج عملیات برش درختی در تکرار سوم

v_i	$DSAV$	$SDSAV$	$W(v_i)$	$PSAV$	$ADSAV$
$v_۶$	۹	۵	$v_۶$	۱۴	۰
$v_{۱۳}$	۷	۱	$v_{۱۳}$	۸	۰

در نهایت در تکرار سوم هسته درخت $P_{v_۱, v_{۱۸}}$ با مقدار تابع هدف $PSAV(v_{۱۱}, W(v_{۱۱})) = ۱۲$ بدست خواهد آمد.

در ادامه درستی الگوریتم هسته را بیان می‌کنیم که قبل از آن نیاز به بیان لم زیر داریم.

لم ۲. بعد از اجرای حلقه تکرار در برنامه اصلی الگوریتم هسته، تنها یک یا دو رأس باقی می‌ماند.

اثبات. فرض کنید $P = \{v_۱, v_۲, \ldots, v_r\}$ مسیری با حداکثر طول در درخت T باشد. اثبات به برهان خلف: ادعا می‌کنیم که اگر $r > ۲$ آنگاه $v_۲ \in S$ است . در غیر این صورت $v_۲$ باید همسایه

یک رأس غیر برگی باشد که عضو مسیر P نیست. بنابراین می‌توان مسیری با طول بزرگ‌تر از P را پیدا کرد که تناقض است. بنابراین $S = \varnothing$ نتیجه می‌دهد که $r \leq ۲$. و این بدان معناست که بعد از اجرای حلقه تکرار، درخت T تنها شامل یک یا دو رأس است. □

قضیه ۱.۲. الگوریتم ۵.۲.۲ مسیری با بهترین مقدار تابع هدف(هسته) را پیدا می‌کند.

اثبات. در لم ۲ دیدیم که بعد از اجرای حلقه تکرار یک یا دو رأس باقی می ماند، لذا اثبات قضیه به دو قسمت تقسیم می‌شود.

a) حالتی که تنها یک رأس باقی بماند. در ابتدا ثابت می‌کنیم که در فرآیند بهنگام کردن، مسیر P که مقدار $PSAV(u, W(u))$ را نتیجه می‌دهد دارای بیشترین فاصله ذخیره شده در میان همه مسیرهای موجود در زیر درخت $T(u)$ است. همچنین توجه می‌کنیم که لزومی ندارد این مسیر، هسته درخت باشد زیرا فاصله ذخیره شده را نسبت به تمامی رئوس درخت T پیدا می‌کنیم.
فرض کنید P_{max} مسیری با بیشترین فاصله ذخیره شده باشد. اگر این مسیر شامل رأس u باشد آنگاه چون $W(u) = u$ واضح است که $P_{max} = P = P_۱ \cup P_۲$ که $P_۱$ و $P_۲$ به ترتیب مسیرهایی هستند که مقادیر $DSAV(u)$ و $SDSAV(u)$ را بدست می‌دهند. اکنون فرض کنید k اندیسی باشد که به ازاء آن $PSAV(v_i, W(v_i))$ ماکزیمم باشد. در نتیجه P_{max} نمی‌تواند در زیر درخت‌هایی $T(v_i)$ باشد که دارای مقدار کمتری از $PSAV$ هستند. در نتیجه P_{max} فقط در صورتی می‌تواند در زیر درخت T_{v_k} باشد که شامل رأس u نباشد. لذاتنها داوطلب‌های مسیر P_{max} مسیر P و آن مسیرهایی هستند که مقدار $PSAV(v_k, W(v_k))$ را بدست بدهند.

b) اکنون فرض کنید بعد از اجرای حلقه تکرار رأس‌های $v_۱$ و $v_۲$ باقی بمانند. لذا یکی از سه داوطلب زیر کاندید هسته خواهد بود.
(۱) مسیرهایی که مقدار $PSAV(v_۱, W(v_۱))$ و $PSAV(v_۱, W(v_۱))$ را بدهند.
(۲) مسیر $P_{v_۱,l_۱} \cup e_{v_۱,v_۲} \cup P_{v_۲,l_۲}$ که مسیرهای $P_{v_۱,l_۱}$ و $P_{v_۲,l_۲}$ به ترتیب مسیرهایی هستند که مقادیر $DSAV(v_۱)$ و $DSAV(v_۲)$ را بدست می‌دهند. □

۶.۲.۲ الگوریتم مورگان و اسلیتر

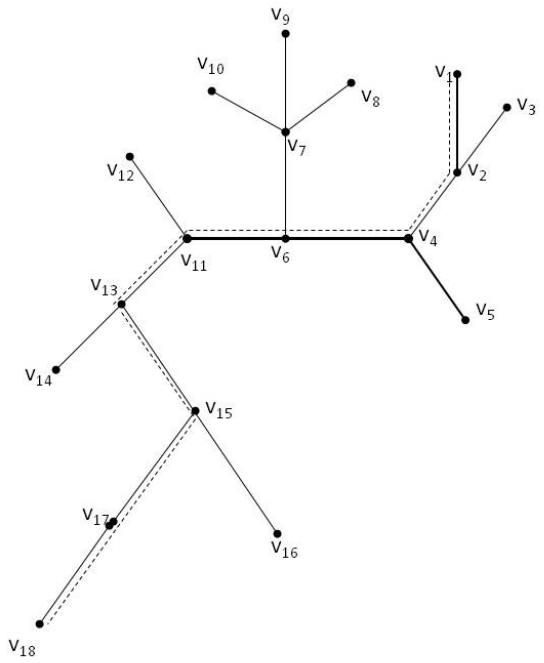
این الگوریتم برای پیدا کردن هسته درخت وزن‌دار و یا بی‌وزن استفاده می‌شود و دارای مرتبه محاسباتی $O(n)$ است و دارای تعداد محاسبات بیشتری نسبت به الگوریتم هسته ۵.۲.۲ است.

برای معرفی این الگوریتم ابتدا نمادهای زیر را تعریف می‌کنیم.

۱) تعداد رئوس درخت را با N نشان می‌دهیم .

۲) (لیست نقطه بعدی[۱]) لیستی از رئوس درخت که به ترتیب خاصی چیده شده‌اند و عضو i ام آن با $epl(i)$ نشان داده می‌شود را لیست نقطه بعدی می‌نامیم و با $endpt$ نمایش می‌دهیم دراین لیست فقط یکی از همسایه‌های رأس $epl(i)$ می‌تواند اندیس $j > i$ داشته باشد و دیگر همسایه‌ها باید قبل از آن در لیست آورده شوند .

۳) (لیست رئوس مجاور[۲]) لیستی از رئوس که همسایه $epl(i)$ هستند و در لیست نقطه بعدی دارای اندیس بزرگ‌تر از i هستند را لیست رئوس مجاور می‌گوییم .



شکل ۵.۲: لیست رئوس بعدی و مجاور

[۱] *End Point List* [۲] *Adjacent Point List*

مثال ۷. در شکل ۵.۲ لیست نقطه بعدی و لیست رئوس مجاور را مشخص کنید.

برای تعیین لیست نقطه بعدی با توجه به تعریف آن توجه می‌کنیم که بعد از رأس $v_۱$ نمی‌تواند رأس $v_۲$ قرار گیرد زیرا در این صورت بعد از رأس $v_۲$ دو رأس $v_۳$ و $v_۴$ که همسایه $v_۲$ هستند قرار خواهند گرفت، یعنی در لیست نقطه بعدی اندیس دو رأس $v_۳$ و $v_۴$ بزرگ‌تر از اندیس رأس $v_۲$ خواهد بود و این با تعریف لیست نقطه بعدی تناقض دارد. بنابراین بعد از رأس $v_۱$ رأس $v_۳$ قرار خواهد گرفت و همچنین رأس $v_۴$ نمی‌تواند قبل از رأس $v_۵$ قرارگیرد:

$$endpt = \{v_۱, v_۳, v_۲, v_۵, v_۴, v_{۱۴}, v_{۱۶}, v_{۱۸}, v_{۱۷}, v_{۱۵}, v_{۱۳}, v_{۱۲}, v_{۱۱}, v_{۱۰}, v_۹, v_۸, v_۷, v_۶\}$$

لیست بعدی لیست رئوس مجاوراست . برای هررأس ، رأس مجاور ، رأس همسایه‌ای است که در لیست نقطه بعدی اندیس آن بزرگ‌تر از اندیس رأس مورد نظر باشد . توجه کنید که اندیس هر رأس در لیست نقطه بعدی با اندیس رأس مجاورش در لیست رئوس مجاور یکسان است .

مثلاً همسایه‌های رأس $v_۲$ رئوس $v_۱$ و $v_۴$ هستند و چون اندیس رأس $v_۴$ بزرگ‌تر از اندیس رأس $v_۱$ است، لذا رأس $v_۴$ در لیست رئوس مجاور به عنوان رأس مجاور $v_۱$ قرارمی‌گیرد. بنابراین رأس $v_۲$ درلیست نقطه بعدی و رأس $v_۴$ درلیست رئوس مجاور هر دو در مکان سوم قرار می‌گیرند.

$$adj = \{v_۲, v_۲, v_۴, v_۴, v_۶, v_{۱۳}, v_{۱۵}, v_{۱۷}, v_{۱۵}, v_{۱۳}, v_{۱۱}, v_{۱۱}, v_۶, v_۷, v_۷, v_۷, v_۶, v_۶\}$$

برای هر رأس $epl(i)$ مشخصات زیر را تعریف می‌کنیم .

۱) (همسایه داخلی[۱]) به ازاء هر رأس $epl(i)$ یک رأس منحصربفرد مانند $epl(j)$ ؛ $j > i$ و همسایه $epl(i)$ است وجود دارد که آن‌را همسایه داخلی $epl(i)$ گوییم .

۲) (شاخه داخلی[۲]) زیر درختی که از حذف $epl(i)$ به وجود می‌آید و شامل همسایه داخلی $epl(i)$ است را شاخه داخلی گوییم .

۳) (مجموعه بیرونی[۳]) مجموعه رأس‌هایی که به وسیله یک مسیر به رأس $epl(i)$ متصل هستند و در مجموعه $epl(۱), \ldots, epl(i)$ قرار دارند را مجموعه بیرونی رأس $epl(i)$ گوییم . به عبارت

[۱] $In - Neighbor$ [۲] $In - Branch$ [۳] $Out\ Set$

دیگر مجموعه بیرونی شامل رأس‌های همه شاخه‌های یک رأس به جز شاخه داخلی است . توجه کنید که هر رأس عضوی ازمجموعه بیرونی خود است .

مثال ۸. درشکل ۵.۲ مجموعه بیرونی رأس $v_۴$ را مشخص کنید.

همسایه داخلی رأس $v_۴$ رأس $v_۶$ است ، با حذف رأس $v_۴$ سه زیر درخت خواهیم داشت. به زیر درختی که شامل رأس $v_۶$ است یعنی $\{v_{۱۴}, v_{۱۶}, v_{۱۸}, v_{۱۷}, v_{۱۵}, v_{۱۳}, v_{۱۲}, v_{۱۱}, v_{۱۰}, v_۹, v_۸, v_۷, v_۶\}$ شاخه داخلی رأس $v_۴$ می‌گوییم . لذا مجموعه بیرونی رأس $v_۴$ مجموعه $\{v_۱, v_۲, v_۳, v_۴, v_۵\}$ می‌باشد .

۷.۲.۲ پیمایش‌های الگوریتم مورگان و اسلیتر

الگوریتم مورگان و اسلیتر شامل دو پیمایش پیشرو[۱] وپسرو[۲] است:

۱- ***پیمایش پیشرو :*** این پیمایش روی مجموعه بیرونی هر رأس انجام می‌گیرد و در طی آن بهترین مسیری که از هر رأس در مجموعه بیرونی آن می‌گذرد بدست می‌آید ، برای رسیدن به این مسیر محاسبه مقادیر زیر ضروری است .

۱) برای هررأس $epl(i)$ ، $\bar{w}(i)$ برابر با مجموع وزن رئوسی است که توسط مسیری به $epl(i)$ متصل‌اند و در لیست نقطه بعدی قبل از آن آمده اند به عبارت دیگر $\bar{w}$ برابر با مجموع وزن رأس‌های مجموعه بیرونی رأس مورد نظر است ، بنابراین می‌توان نوشت:

$$\bar{w}(epl(i)) = \sum_{v \in outset} w(v)d(v, epl(i))$$

۲) مجموع فاصله‌های رئوس متعلق به مجموعه بیرونی رأس $epl(i)$ از این رأس را با $outdst(i)$ (فاصله بیرونی) نشان می‌دهیم پس داریم :

$$outdst(epl(i)) = \sum_{v \in outset} w(v)d(v, epl(i))$$

[۱] $Forward\ Pass$ [۲] $Backward\ Pass$

۳) اگررأس $epl(i)$ برگ نباشد در این‌صورت با شروع از آن می‌توان در دو شاخه متفاوت دو مسیر متمایز زیر را مشخص کرد . $P_۱ : P_{epl(i),epl(j)}, P_۲ : P_{epl(i),epl(j)}$ که $\bar{F}(P_۱)$ و $\bar{F}(P_۲)$ به ازاء آن ها کمترین مقدار ممکن را داشته باشد ، دراین‌صورت داریم:

$$Disave۱(epl(i)) = d(epl(i) - \bar{F}(P_۱) \quad (a)$$

$$Disave۲(epl(i)) = d(epl(i) - \bar{F}(P_۲) \quad (b)$$

(c) همسایه $epl(i)$ روی شاخه‌ای که $Disave۱(epl(i))$ از آن بدست آمده را $vrt۱(epl(i))$ می‌نامیم.

(d) همسایه $epl(i)$ روی شاخه‌ای که $Disave۲(epl(i))$ از آن بدست آمده را $vrt۲(epl(i))$ می‌نامیم.

۴) اگر رأس $epl(i)$ برگ باشد دارای یک شاخه است که با گسترش مسیر به این شاخه مسیر $P_۱$ ی بدست می‌آید که :

$$Disave۱(epl(i)) = d(epl(i) - \bar{F}(P_۱) \quad (a)$$

(b) همسایه $epl(i)$ روی شاخه‌ای که $Disave۱$ از آن به‌دست آمده را $vrt۱(epl(i))$ می‌نامیم

۵) ماکزیمم مقداری که $d(epl(i)$ می‌تواند کاهش یابد را مقدار مسافت ذخیره شده گویند و با $save(epl(i))$ (ذخیره) نشان می‌دهیم. واضح است که:

$$save(epl(i)) = max\{Disave۱(epl(i)), Disave۲(epl(i))\}$$

۲- ***پیمایش پسرو :*** این پیمایش محاسبات را روی تمام رئوس درخت تعمیم می‌دهد . در پایان این پیمایش مقادیر زیر بدست می‌آیند.

۱) مجموع فاصله تمام رئوس درخت از رأس $epl(i)$ را با $vrtdst(epl(i))$ نشان می‌دهیم و داریم:

$$vrtdst(epl(i)) = d(epl(i))$$

۲) مقادیر $Disave۱(epl(i))$ و $vrt۱(epl(i))$ ، $Disave۲(epl(i))$ ، $vrt۲(epl(i))$ همانند پیمایش پیشرو محاسبه می‌شوند ، با این تفاوت که این بار محاسبات روی کل رئوس درخت انجام می‌گیرد.

۳) فاصله از مسیر P را با $pdst(epl(i))$ نمایش می‌دهیم که P شامل رأس $epl(i)$ ودو مسیر P_1 و P_2 ی است که $Disave1(epl(i))$ و $Disave2(epl(i))$ در آن‌ها اتفاق می‌افتد و داریم $\bar{F}(p) = pdst(epl(i))$

۴) کمترین مقدار $pdst(epl(i))$ را با $mindst$ (مینیمم مسافت) و رأس $epl(i)$ مربوط به آن را با $locmin$(محل مینیمم) نمایش می‌دهیم . توجه کنید که در هر مرحله از پیمایش مینیمم مسافت و به تبع آن محل مینیمم باید بهنگام شود .

۵) همسایه $epl(i)$ روی شاخه‌ای که $Disave2(epl(i))$ اتفاق می‌افتد را $vrt2(epl(i))$ می‌نامیم . از آن‌جا که مجموعه بیرونی شامل همه رئوس به جز شاخه داخلی است اختلاف دو پیمایش در شاخه داخلی است . لذا برای محاسبه ذخیره در شاخه داخلی به صورت زیر عمل می‌کنیم .

(a) اگر $epl(i)$ روی هیچ‌کدام از دو شاخه ماکزیمم همسایه‌اش نبود یعنی

$$vrt1(adj(i)) \neq i, vrt2(adj(i)) \neq i$$

آن‌گاه $save(epl(i)) = max\{Disave1(adj(i)), Disave2(adj(i))\}$

(b) اگر $epl(i)$ روی اولین مسیر ماکزیمم باشد یعنی $vrt1(adj(i)) = epl(i)$ آن‌گاه داریم:

$$save(epl(i)) = Disave2(adj(i))$$

(c) اگر $epl(i)$ روی دومین مسیر ماکزیمم باشد یعنی $vrt2(adj(i)) = epl(i)$ آن‌گاه داریم:

$$. \ save(epl(i)) = Disave1(adj(i))$$

بعد از این مقدمات اکنون الگوریتمی را که مورگان و اسلیتر برای پیدا کردن هسته درخت ارائه داده‌اند بیان می‌کنیم .

الگوریتم مورگان واسلیتر

ورودی: درخت $T = (V, E)$.

خروجی: هسته.

گام ۱ : ورودی‌های درخت

(a) N تعداد رئوس درخت

(b) $epl(i), i = ۱, \dots N$ لیست رئوس مجاور

(c) $adj(i), i = ۱ \dots N$ لیست رئوس مجاور

گام ۲ : پیمایش پیشرو

(a) به ازاء $I = ۱, \dots .N$ مقادیر اولیه زیر را قرار دهید :

$$\bar{w}(i) = w(epl(i)) \text{ ، } Disave۲(i) = ۰ \text{ ، } Disave۱(i) = ۰$$

$$outdst(i) = ۰ \text{ ، } vrt۲(i) = \{\} \text{ ، } vrt۱(i) = \{\}$$

به ازاء $i = ۱, \dots N$ گام‌های ذیل را انجام دهید :

(b) قرار دهید $save(epl(i)) = max\{Disave۱(epl(i)), Disave۲(epl(i))\}$

(c) قرار دهید $\bar{w}(adj(i)) = \bar{w}(adj(i)) + \bar{w}(epl(i))$

(d) قرار دهید

$$outdst(adj(i)) = outdst(adj(i)) + outdst(epl(i)) + \bar{w}(epl(i)) * d(epl(i), adj(i))$$

(e) قرار دهید $newmax = save(epl(i)) + \bar{w}(epl(i)) * d(epl(i), adj(i))$

(f) اگر $newmax > min\{Disave۱(adj(i)), Disave۲(adj(i))\}$

آن‌گاه قرار دهید $newmax = min\{Disave۱(adj(i)), Disave۲(adj(i))\}$ و بسته به این‌که کدام یک مینیمم می‌شود قرار دهید :

$vrt۱(adj(i)) = epl(i)$ یا $vrt۲(adj(epl(i)) = epl(i)$

گام ۳ : (پیمایش پسرو)مقادیر زیر را برای رأس N ام به‌دست آورید

(a) $vrtdst(epl(N)) = outdst(epl(N))$

(b) $pdst(N) = outdst(epl(N)) - disave۱(epl(N)) - Disave۲(epl(N))$

(c) $mindst = pdst(epl(N))$

(d) $locmin = epl(N)$

با ازاء $i = N - ۱, \ldots, ۱$ گام‌های زیر را انجام دهید:

(e) قرار دهید :

$$vrtdst(epl(i)) = vrtdst(adj(i)) + (W(T)) - ۲(\bar{w}(epl(i))) * d(epl(i), adj(i))$$

(f) اگر $olddst = Disave۲(adj(i))$ قرار دهید $vrt۱(adj(i)) = epl(i)$

اگر $olddst = Disave۱(adj(i))$ قرار دهید $vrt۱(adj(i)) = epl(i)$

درغیر این‌صورت قرار دهید $olddst = max\{Disave۱(epl(i)), Disave۲(epl(i))$

(g) قرار دهید $newdst = olddst + (W(T) - \bar{w}(epl(i)) * d(epl(i), adj(i))$

(h) اگر $newdst > min\{Disave۱(epl(i), Disave۲(epl(i)\}$ آن‌گاه قرار دهید :

$vrt۲(epl(i)) =$ و مطابق آن قرار دهید $min\{Disave۱(epl(i), Disave۲(epl(i)\} = newdst$

$vrt۱(epl(i)) = adj(i)$ یا $adj(i)$

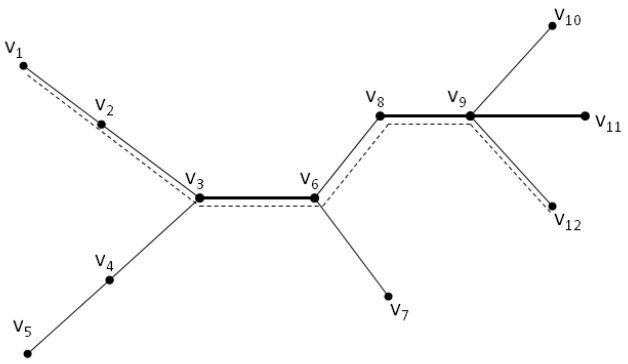
(i) قرار دهید $pdst(epl(i)) = vrtdst(epl(i)) - Disave۱(epl(i)) - Disave۲(epl(i))$

اگر $mindst \geq pdst(epl(i))$ آن‌گاه قرار دهید $mindst = pthdst(epl(i))$ و همچنین

قراردهید: $locmin = epl(i)$

گام‌های الگوریتم کامل است.

مثال ۹. با استفاده از الگوریتم MS هسته درخت شکل ۶.۲ را پیدا کنید. وزن رئوس و طول یال‌ها را برابر یک در نظر بگیرید.



شکل ۶.۲: پیدا کردن هسته درخت با استفاده از الگوریتم MS

ابتدا لیست نقطه بعدی و لیست رئوس مجاور را تعیین می‌کنیم.

$$endpt = \{v_۱, v_۲, v_۵, v_۴, v_۳, v_{۱۰}, v_{۱۱}, v_{۱۲}, v_۹, v_۷, v_۸, v_۶\}$$

$$adj = \{v_۲, v_۳, v_۴, v_۳, v_۶, v_۹, v_۹, v_۹, v_۸, v_۶, , v_۶, v_۶\}$$

نتایج ارائه شده در جدول ۴.۲ به وسیله الگوریتم MS به دست آمده است. در پایان پیمایش پسرو رأس $v_۱$ به عنوان $locmin$ به دست آمده است . برای رسیدن به هسته خروجی باید از این رأس شروع کنیم ، سپس به رأس $vrt۱(v_۱)$ یعنی رأس $v_۲$ برویم بعد از آن می‌توانیم به $vrt۱(v_۲) = v_۱$ یا $vrt۲(v_۲) = v_۳$ برویم اما $Disave۱(v_۲) = ۴۹ > Disave۱(v_۲)$ ۱ لذا باید به رأس $v_۳$ برویم . به همین ترتیب به رأس $v_۶$ و $v_۸$ و $v_۹$ و $v_{۱۲}$ می‌رویم . رأس $v_{۱۲}$ یک برگ است ، لذا الگوریتم خاتمه می‌یابد و هسته مطلوب $p_{v_۱,v_{۱۲}}$ با مقدار تابع هدف $F_{p_{v_۱,v_{۱۲}}} = ۶$ است.

جدول ۴.۲: نتایج گام‌های الگوریتم MS

points	adj	save	$\bar{w}$	outdst	Disave۱	vrt۱	Disave۲	vrt۲	vrtdst	pdst
$v_۱$	$v_۲$	۰	۱	۰	۳۴	$v_۲$	۰	۰	۴۰	۶
$v_۲$	$v_۳$	۱	۲	۱	۱	$v_۱$	۲۷	$v_۳$	۳۴	۶
$v_۵$	$v_۴$	۰	۱	۰	۳۴	$v_۴$	۰	۰	۴۰	۶
$v_۴$	$v_۳$	۱	۲		۱ ۱	$v_۵$	۲۷	$v_۳$	۳۴	۷
$v_۳$	$v_۶$	۳	۶	۶	۳	$v_۲$	۱۷	$v_۶$	۲۶	۶
$v_{۱۰}$	$v_۹$	۰	۱	۰	۳۴	$v_۹$	۰	۰	۴۰	۷
$v_{۱۱}$	$v_۹$	۰	۱	۰	۳۴	$v_۹$	۰	۰	۴۰	۷
$v_{۱۲}$	$v_۹$	۰	۱	۰	۳۴	$v_۹$	۰	۰	۴۰	۶
$v_۹$	$v_۸$	۱	۴	۳	۱	$v_۸$	۲۴	$v_{۱۲}$	۳۰	۶
$v_۷$	$v_۶$	۰	۱	۰	۲۱	$v_۶$	۰	۰	۳۴	۱۳
$v_۸$	$v_۶$	۵	۵	۷	۱۵	$v_۶$	۱۵	$v_۹$	۲۶	۶
$v_۶$	$v_۶$	۰	۱۲	۲۴	۶	$v_۳$	۱۰	$v_۸$	۲۴	۶

۳.۲ مرکز و مسیر مرکزی

گاهی در مسائل مکان‌یابی به مواردی برخورد می‌کنیم که مشتری‌ها در نقاط دوردست هستند و دسترسی آن‌ها به سرویس‌دهنده مشکل‌تر است و هدف تسهیل دسترسی این‌گونه مشتری‌ها به مسیر است. اگر در این مسئله مجموعه مشتری‌ها را رئوس شبکه و مقدار تقاضای هر مشتری را وزن رأس مربوط به آن در نظر بگیریم، هدف پیدا کردن مسیری است که بیشترین فاصله رأس‌های موجود تا آن کم‌ترین مقدار ممکن باشد.

۱.۳.۲ مرکز

هنگامی‌که طول مسیر صفر باشد مسیر یک نقطه است که به آن مرکز مرکز مطلق[۱] می‌گویند. بنابراین مرکز نقطه‌ای است که فاصله دورترین رأس از آن دارای کم‌ترین مقدار ممکن باشد.

۲.۳.۲ الگوریتم هندلر برای بدست آوردن مرکز درخت

الگوریتم زیر توسط هندلر برای یافتن مرکز یک درخت بی‌وزن ارائه شده است[۸].

[۱] *Absolute Center*

الگوریتمheight $Handler$ برای بدست آوردن مرکز درخت

ورودی: درخت $T = (V, E)$.
خروجی: مرکز درخت.

گام ۱ : یک رأس v_1 را به دلخواه در درخت انتخاب کنید.

گام ۲ : دورترین رأس از آن مثلاً v_2 را پیدا کنید.

گام ۳ : رأس v_3 دورترین رأس از v_2 را پیدا کنید.

گام ۴ : نقطه میانی مسیر v_2 به v_3 مرکز درخت است.

۳.۳.۲ مسیر مرکزی

گاهی در مسائل مکان‌یابی به مواردی برمی‌خوریم که مشتری‌ها در نقاط دوردست هستند و دسترسی آن‌ها به سرویس دهنده مشکل‌تر است و ما به‌دنبال تسهیل دسترسی این گونه مشتری‌ها به مسیر هستیم. اگر در این مسأله مجموعه مشتری‌ها را نقاط یک شبکه و مقدار تقاضای هر فرد را وزن رأس مربوط به آن در نظر بگیریم، هدف پیدا کردن مسیری است که بیشترین فاصله رأس‌های موجود تا آن کم‌ترین مقدار ممکن باشد. مدل این مسأله بصورت زیر است که در آن P یک مسیر است:

$$G(P) = \min_{p}(\max_{1 \le i \le n} w_i d(v_i, p)) \qquad (۹.۲)$$

تعریف ۱۰.۲. (مسیر مرکزی درخت[1]) مسیر مرکزی درخت، کوتاه‌ترین مسیر مرکزگیری مینیمال است و آن را با PC که منحصر بفرد است نشان می‌دهیم .

۴.۳.۲ الگوریتم مینیکا برای بدست آوردن مسیر مرکزی درخت

برای مسیرهایی با طول بیشتر از صفر الگوریتمی توسط مینیکا در سال ۱۹۸۵ ارائه شد که دارای مرتبه زمانی $O(poly(n))$ (چند جمله‌ای بر حسب n) است که در ادامه به بررسی و معرفی این الگوریتم می‌پردازیم [۱۸].

[1] $Path - Center$

الگوریتم مینیکا برای بدست آوردن مسیر مرکزی درخت
ورودی: درخت $T = (V, E)$.
خروجی: مسیر مرکزی درخت.

گام ۱ : مسیر اولیه P_A را نقطه c (مرکز درخت) بنامید.

گام ۲ : مجموعه S_1 شامل دورترین برگ‌ها از P_A و مجموعه S_2 شامل برگ‌هایی با دومین بیشترین فاصله از P_A را بیابید.

گام ۳ : تفاوت این دو فاصله را dif بنامید.

گام ۴ : مسیر P_A را در جهت رئوس S_1 گسترش دهید که در این صورت یکی از سه حالت زیر رخ می‌دهد:

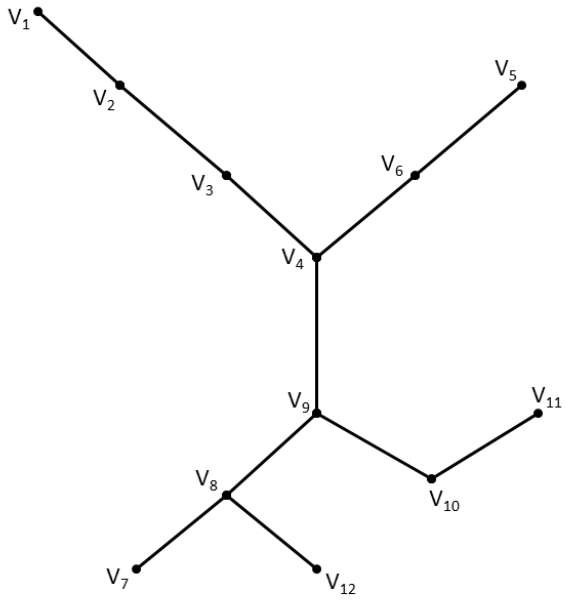
(a) اندازه P_A برابر با L می‌شود که در این صورت باید متوقف شوید.

(b)مسیر P_A به اندازه dif به رئوس S_1 نزدیک می‌شود در این حالت به گام **۲** بروید.

(c) اگر جهت گسترش بیش از دو تا باشد که با مسیر بودن تناقض دارد در این حالت نیز توقف کنید.

مثال ۱۰. در درخت شکل ۷.۲ مرکز و مسیر مرکزی را پیدا کنید.

ابتدا برای پیدا کردن مرکز درخت رأس v_5 را در نظر می‌گیریم و سپس دورترین رأس از آن یعنی v_1 را پیدا می‌کنیم اکنون دوباره باید دورترین رأس از v_7 را بیابیم که رأس v_1 است. رأس v_4 یعنی رأس میانی مسیر بین رأس v_7 و v_1 مرکز درخت است.
برای یافتن مسیر مرکزی درخت در ابتدا قرار می‌دهیم $P_A = v_4$. مجموعه s_1 و s_2 بصورت زیر خواهند بود: $S_1 = \{v_1, v_7, v_{11}, v_{12}\}$ و $S_2 = \{v_5\}$ تفاوت فاصله دو مجموعه S_1 و S_2 را برابر dif قرار می‌دهیم. $dif = 3 - 2 = 1$ سپس P_A را از رأس v_4 به طرف رئوس مجموعه S_1 گسترش می‌دهیم تا زمانی‌که $P_A = P_{V_1,V_2}$ که در آنجا P_A به اندازه dif به



شکل ۷.۲: یافتن مرکز و مسیر مرکزی

$S_۱$ نزدیک می‌شود سپس به گام ۲ می‌رویم. در این گام $S_۱ = \{v_۱, v_۷, v_{۱۱}, v_{۱۲}, v_۵\}$ و $S_۲ = \{\}$ بنابراین $dif = ۰$ می‌خواهیم P_A را به اندازه dif به رئوس $S_۱$ گسترش دهیم اما از طرف رأس $v_۹$ دو جهت داریم (یک جهت به سمت $v_{۱۲}$ و جهت دیگر به سمت $v_{۱۲}$ از طرفی از سمت رأس $v_۳$ نیز باید مسیرگسترش یابد در نتیجه ۳ جهت گسترش داریم که با تعریف مسیر تناقض دارد. لذا الگوریتم خاتمه می‌یابد و مسیر مرکزی $P_A = P_{v_۳,v_۹}$ با مقدار تابع هدف $\bar{F} = ۱۴$ است.

درادامه الگوریتم دیگری را برای به دست آوردن مسیر مرکزی درخت (PC) بیان خواهیم کرد .

۵.۳.۲ الگوریتم PCT برای بدست آوردن مسیر مرکزی درخت

در این بخش قصد داریم الگوریتم دیگری را برای بدست آوردن مسیر مرکزی درخت ارائه دهیم که قبل از آن به بیان برخی تعاریف و نکات مورد نیاز این الگوریتم می‌پردازیم.

تعریف ۱۱.۲. اگر T یک درخت و u رأسی از آن باشد ، آن‌گاه $e(u,T)$ و $d(u,T)$ را

بصورت زیر تعریف می‌کنیم :

$$e(u,T) = \max_{v \in V(T)} d(u,v) \qquad (۱۰.۲)$$

$$d(u,T) = \sum_{v \in V(T)} d(u,v) \qquad (۱۱.۲)$$

اگر T یک درخت و P مسیری در آن باشد آن‌گاه داریم :

$$e(P,T) = e(V(P),T) \qquad (۱۲.۲)$$

$$d(P,T) = d(V(P),T) \qquad (۱۳.۲)$$

که $V(P)$ مجموعه رأس‌های مسیر P می‌باشد .

تعریف ۱۲.۲. برای مسیر P و رأس v_i در درخت T ، $e_P(v_i)$ را به‌صورت زیر تعریف می‌کنیم :

$$e_P(v_i) = max\{d(w,v_i); w \in V(T), d(w,P) = d(w,v_i)\} \qquad (۱۴.۲)$$

الگوریتم PCT
ورودی: درخت $T = (V,E)$.
خروجی: مسیر مرکزی درخت.

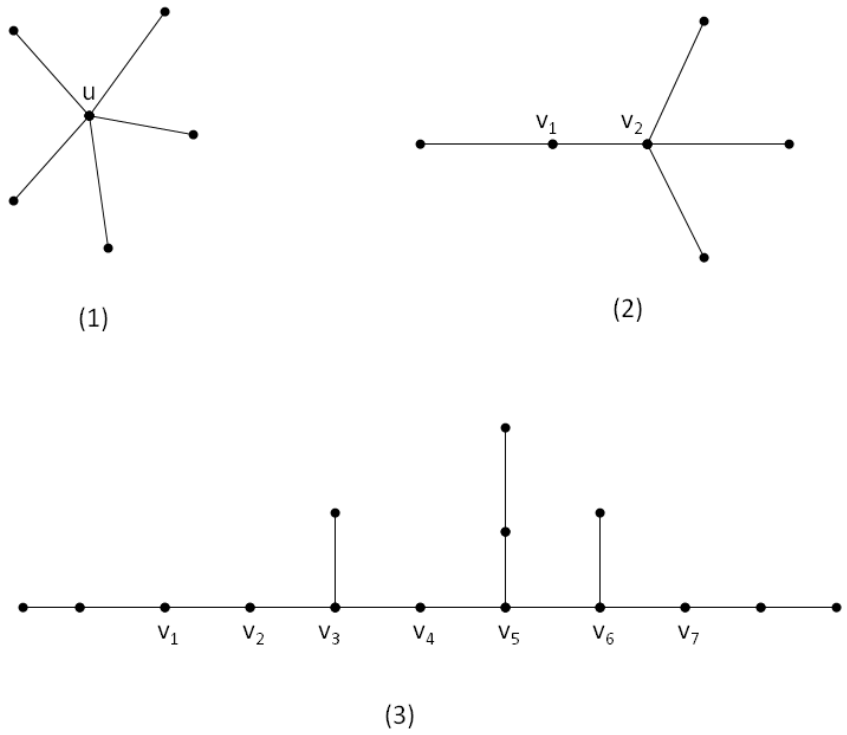
گام ۱ : اگر مرکز درخت T برابر با $P =< u,v >$ است قرار دهید $k = ۲$ و $P = v_۱, v_۲ = u,v$ و به گام ۳ بروید .

گام ۲ : اگر مرکز مطلق درخت T رأس $\{u\}$ است و سه یا تعداد بیشتری از عناصر $T \setminus u$ شامل رأس‌هایی هستند که فاصله آن‌ها از u برابر $e(u;T)$ است قرار دهید $P = u$ و به گام ۵ بروید . اگر دقیقاً دو تا از عناصر T شامل رأس‌هایی باشند که فاصله آن‌ها از رأس u برابر $e(u;T)$ باشد و برای مثال اگر v و w رأس‌هایی از این عناصر باشند که با رأس u همسایه نیز هستند ، قرار دهید $k = ۲$ و $P = v_۱, v_۲, v_۳ = v,u,w$ و به گام ۳ بروید .

گام ۳ : اگر $e_P(v_k)$=$e_P(v_1)$=$e_P(v_i)$ برای $2 \leq i \leq k-1$ به گام ۵ بروید . برای $h=1$ یا $h=k$ اگر $T_h - v_h$ شامل دو یا تعداد بیشتری از عناصری است که شامل رأس‌هایی هستند که فاصله آن‌ها از v_h برابر $e_P(v_h)$ است به گام ۵ بروید .

گام ۴ : برای $h=1$ و $h=k$ اگر دقیقاً یکی از عناصر $T_h - v_h$ شامل رأسی است که فاصله آن از v_h برابر $e_P(v_h)$ می‌باشد و w_1 و w_2 رأس‌هایی هستند که با v_1 و v_h همسایه هستند مقدار k را به ۲ افزایش داده و قرار دهید $P =< v_1, \ldots, v_k >=< w_1, \ldots, w_2 >$ و $P(w_1, w_2)$ و به گام ۳ بروید .

گام ۵ : متوقف شوید ، مسیر مرکزی درخت P است .



شکل ۸.۲: مسیر مرکزی

مثال ۱۱. سه درخت نشان داده شده در شکل ۸.۲ را در نظر بگیرید . مسیر مرکزی هر کدام را پیدا کنید.

درخت (۱) دارای مرکز مطلق $\{u\}$است ، لذا بر طبق گام (۲) الگوریتم بالا چون ۵ تا از عناصر $T-u$ شامل رأس‌هایی است که فاصله آن‌ها از u برابر ۱ $e(u;T)=$ می‌باشد پس به گام ۵ رفته و مسیر مرکزی آن برابر $P=u$ خواهد بود . درخت ۲ دارای مرکز مطلق $\{v_۱,v_۲\}$ است پس بااستفاده از گام ۱ و ۳ وسپس گام ۵ ، $T_۲-v_۲$ شامل بیش از یک عنصر است که رأسی را دارد که فاصله آن از $v_۲$ برابر ۱ $e_P(v_۲)=$ است . لذا مسیر مرکزی درخت $P=v_۱,v_۲$ است . درخت (۳) دارای مرکز مطلق $v_۴$ است لذا با استفاده از گام ۴ و ۳ مسیر مرکزی درخت (۳) مسیر $P=\{v_۱,v_۲,v_۳,v_۴,v_۵,v_۶,v_۷\}$ است.

۴.۲ مسیر میانه مرکزی

۱.۴.۲ مقدمه

دو معیار اصلی که اغلب در تئوری مکانیابی شبکه برای تعیین محل بهینه یک وسیله مورد استفاده قرار می‌گیرد عبارتند از: ماکزیمال فاصله بین وسیله تا مشتری‌ها و میانگین فاصله بین وسیله و مشتری‌های مورد نظر [۱، ۲].

نوع دیگری از مسائل مکانیابی که به آن پرداخته شده، پیدا کردن محل بهینه یک مسیر با استفاده از ترکیب معیارهای ماکزیمم فاصله و میانگین فاصله مشتری‌ها از سرویس‌دهنده‌ها است. چنین مسیری،مسیر میانه مرکزی نام دارد.در واقع مسیر میانه مرکزی مسیری است که در آن هم (مینیمم ماکزیمم فاصله[۱]) بین مشتری‌ها تا سرویس دهنده مینیمم است و هم (مینیمم مجموع فاصله[۲]) بین مشتری‌ها تا سرویس‌دهنده مینیمم است [۶، ۷]. در واقع این مسئله ترکیبی از مسائل مسیر میانه و مسیر مرکزی است. به عنوان مثالی از چنین مسائلی می‌توان به طراحی مسیر خطهای ریل راه آهن، مسیر راه‌هایی هوایی، مسیر خطهای لوله آب و فاضلاب اشاره کرد. که بطور مثال در طراحی مسیر حرکت خطهای ریل راه‌آهن بدنبال مسیری هستیم که تا حد امکان هم بیشترین فاصله مسافران تا آن کمترین مقدار ممکن باشد

[۱] $Minmax$ [۲] $Minsum$

و هم میانگین فاصله مسافران تا آن کمترین مقدار ممکن باشد. و در مقابل در طراحی مسیر حرکت لوله‌های فاضلاب بدنبال مسیری هستیم که تا حد امکان هم کم‌ترین فاصله مسافران تا آن بیشترین مقدار ممکن باشد و هم میانگین فاصله مسافران تا آن بیشترین مقدار ممکن باشد.

مسئله پیدا کردن مکان بهینه یک مسیر در یک شبکه درختی با استفاده از ترکیب معیارهای ماکزیمم فاصله و میانگین فاصله مشتری‌ها از سرویس‌دهنده‌ها نخست توسط (هدتنیمی و اسلیتر[۱]) بررسی شد و آن‌ها توانستند الگوریتمی خطی برای آن ارائه دهند [۸، ۳]. همچنین در مقالات دیگری به این مسئله پرداخته شد و الگوریتم‌های متعددی برای آن ارائه شد [۸، ۱۰، ۵]. در سال ۱۹۹۹ سه الگوریتم با مرتبه‌های زمانی مختلف توسط (برمن و اورباخ[۲]) ارائه شد و آن‌ها ثابت کردند، اگر تعداد رأس‌های درخت n باشد، تعداد نمایش‌های مسیرهای بهینه $n-1$ است. همچنین نشان دادند که با وجود این‌که تعداد مسیرهای بهینه $o(n^2)$ است، مرتبه زمانی پیدا کردن این مسیرها $o(nlogn)$ است[۸].

۲.۴.۲ معرفی مسئله

گفتیم که مسیر میانه مرکزی مسیری است که در آن هم ماکزیمم فاصله بین مشتری‌ها تا سرویس دهنده مینیمم است و هم میانگین فاصله بین مشتری‌ها تا سرویس‌دهنده مینیمم است لذا تابع هدف این مسئله بصورت زیر خواهد بود:

$$C(P) = \min_{P} \{Max(P) + Mean(P)\} \qquad (۱۵.۲)$$

که $Max(P)$ و$Mean(P)$ بصورت زیر تعریف می‌شوند:

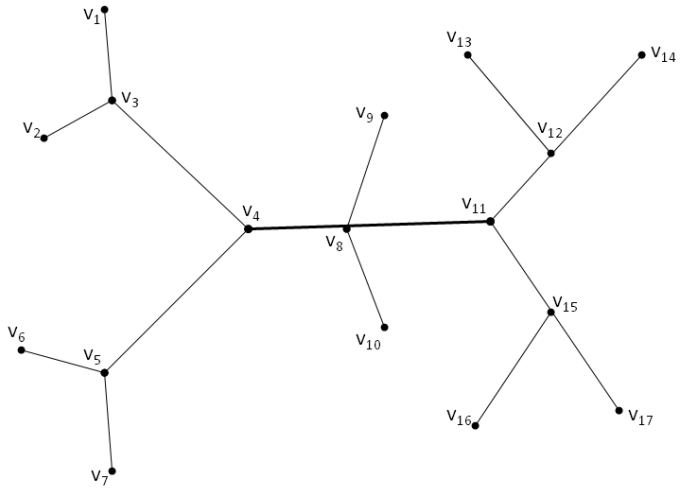
$$F(P) = Max(P) = \max_{v \in V} d(v, P) \qquad (۱۶.۲)$$

$$G(P) = Mean(P) = \sum_{v \in V} w(v) d(v, P) \qquad (۱۷.۲)$$

در ابتدا تعاریفی را بیان می‌کنیم که برای بررسی این مسئله ضروری است.

[۱] *Hedetniemi et al and Slater* [۲] *Berman and Averbakh*

تعریف ۱۳.۲. فرض کنید $T = (V, E)$ یک شبکه درختی باشد که V مجموعه رئوس آن با $|V| = n$ و E مجموعه یال‌های آن باشد، بطوری‌که طول تمام یال‌ها غیر صفر و وزن‌های نامنفی $w(v)$ متناظر با رأس‌های v باشد. اکنون برای هر رأس دلخواه v مجموعه ای پیوسته از $T - v$ را با v-شاخه [۱] نشان می‌دهیم. برای هر رأس دیگر مانند q مجموعه $B(v, q)$ رابه عنوان یک v- شاخه منحصر بفرد در نظر می‌گیریم که شامل q است و $N(v, q)$ را مجموعه رأس‌های آن تعریف می‌کنیم.



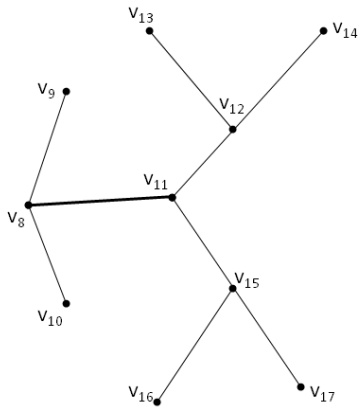
شکل ۹.۲: مثال یافتن v-شاخه در یک درخت

مثال ۱۲. برای درک بهتر تعاریف قبل به شکل ۹.۲ توجه کنید . در این شکل درختی با ۱۷ رأس و ۱۶ یال وجود دارد. رأس $v_۴$ را در نظر بگیرید. $v_۴$-شاخه مجموعه‌ای بصورت زیر است :

$$T - v_۴ = \{v_۱, v_۲, v_۳, v_۵, v_۶, v_۷, v_۸, v_۹, v_{۱۰}, v_{۱۱}, v_{۱۲}, v_{۱۳}, v_{۱۴}, v_{۱۵}, v_{۱۶}\}$$

حال به عنوان کاندیدا برای رأس $v_۸$ ، $B(v_۴, v_۸)$ زیر درختی است شامل رأس $v_۸$ که رأس $v_۴$ را در بر ندارد. شکل ۱۰.۲ را ببینید. $N(v_۴, v_۸)$ مجموعه رأس‌های این زیر درخت است که در زیر آمده است:

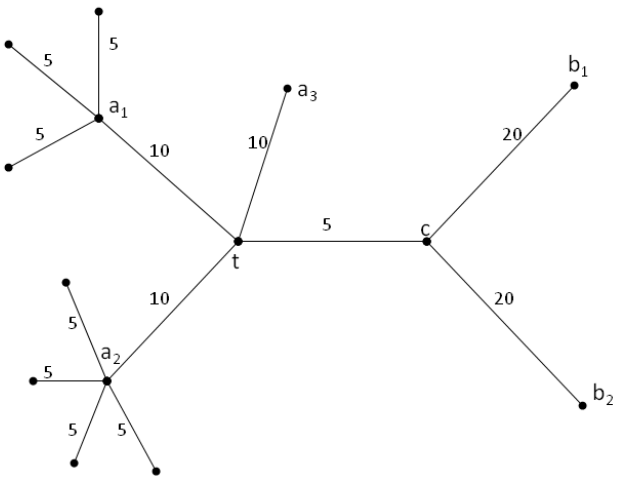
$$N(v_۴, v_۸) = \{v_۸, v_۹, v_{۱۰}, v_{۱۱}, v_{۱۲}, v_{۱۳}, v_{۱۴}, v_{۱۵}, v_{۱۶}, v_{۱۷}\}$$



شکل ۱۰.۲: $B(v_۴, v_۸)$

تعریف ۱۴.۲. (رأس بحرانی[۱]) برای مسیر دلخواه P رأس $a \in P$ یک رأس بحرانی است، اگر رأسی مانند $b \in P$ به قسمی وجود داشته باشد که :

$$d(b,a) = d(b,P) = F(P) \qquad (۱۸.۲)$$



شکل ۱۱.۲: مثال تعیین رأس بحرانی برای یک مسیر

مثال ۱۳. رأس t در درخت شکل ۱۱.۲ یک رأس بحرانی برای مسیر $P(a_۱, a_۲)$ است زیرا:

[۱] $V - Branch$ $Critical - Vertex$

$$d(t, b_1) = d(P(a_1, a_2), b_1) = Max(P) = ۲۵$$

تعریف ۱۵.۲. (مسیر میانه مرکزی[۱]) مسیر $P' \in \Phi$ یک مسیر پرتو بهینه [۲] یا یک مسیر میانه مرکزی است ، اگر مسیر دیگری مانند $P \in \Phi$ وجود نداشته باشد به قسمی‌که :

$$F(P) \leqslant F(P') \qquad (۱۹.۲)$$

$$G(P) \leqslant G(P') \qquad (۲۰.۲)$$

یا حدأقل یکی از نامساوی‌ها اکید باشد .

تعریف ۱۶.۲. فرض کنید Π مجموعه‌ای از تمام مسیرهای پرتو بهینه باشد، برای هر مجموعه Q از این مسیرها،مجموعه $\phi(Q)$ را به‌صورت زیر تعریف می‌کنیم:

$$\phi(Q) = \{(F, G) \in R^2; \exists P \in Q; F = Max(P), G = Mean(P)\} \qquad (۲۱.۲)$$

در واقع مجموعه Φ مجموعه تمام مسیرهایی است که در هر دو معیار ماکزیمم فاصله و مجموع فاصله صدق می‌کنند.

قضیه ۲.۲. اگر مجموعه‌های V_1 و V_2 را بصورت زیر تعریف کنیم:

$V_1 = \{v \in V | deg(v) \geq ۳\}$ و $V_2 = \{v \in V | deg(v) \leq ۲\}$ واضح است که تنها گره‌های V_1 می‌توانند بحرانی باشند و برای هر $P \in \phi$ ، $Max(P) \geq ۰$. برای اثبات به مرجع [۲] مراجعه کنید.

درادامه چند لم را بیان خواهیم کرد که خصوصیات مسیر میانه مرکزی را نشان می‌دهد .

لم ۳. اگر $P \in \Pi$ و t یک رأس بحرانی از P باشد آنگاه $c \in P$ (c مرکز درخت است) یا $P \cap B(t, c) = \phi$.

اثبات.اگر $c \notin P$ آنگاه مسیر P فقط یک رأس بحرانی دارد که نزدیک‌ترین رأس از P به c است ، لذا اگر $c \notin P$ و $P \cap B(t, c) \neq \emptyset$ ، t نمی‌تواند یک رأس بحرانی باشد که تناقض با فرض است.

[۱] $Path - Medi - Center$ [۲] $Pareto - Optimal$

لم ۴. فرض کنید $P \in \Pi$ و t یک راس بحرانی از P باشد و $c \notin P$، در این‌صورت رأس‌های $u \in NEI(t) \setminus P(t,c) \setminus u_۱$ به‌قسمی وجود دارند که به ازاء $u_۱, u_۲ \in NEI(t) \setminus P(t,c)$ داریم:

$$Q_۵(t,u_۱) \geqslant Q_۵(t,u_۲) \geqslant Q_۵(t,u) \qquad (۲۲.۲)$$

$$F(P) = \frac{diam}{۲} + d(t,c) \qquad (۲۳.۲)$$

$$G(P) = R(t) - Q_۵(t,u_۱) - Q_۵(t,u_۲) \qquad (۲۴.۲)$$

اثبات.اگر مسیر P در مفروضات لم صدق کند بر طبق لم ۳ داریم $P \cap B(t,c) = \emptyset$ است. بنابراین $F(P) = \frac{diam}{۲} + d(t,c)$. اکنون واضح است که مسیر بهینه P باید به عنوان بیشترین فاصله میانی ممکن از t به منبع ذخیره توسعه یابد. بنابراین داریم :

$$G(P) = R(t) - Q_۵(t,u_۲) - Q_۵(t,u_۱)$$

مثال ۱۴. درختی را در شکل ۱۱.۲ در نظر بگیرید، فرض کنید t یک رأس بحرانی از مسیری مانند P باشد، بطوری‌که $c \notin P$ در نتیجه داریم $F(P) = ۲۵$ و $Q_۵(t,a_۲) = ۵۵$ ، $Q_۵(t,a_۱) = ۴۵$ ، $Q_۵(t,a_۳) = ۱۰$ ، $R(t) = ۱۹۰$ پس $G(P) = ۱۹۰ - ۵۵ - ۴۵ = ۹۰$ در نتیجه یک مسیر پرتو بهینه باید از t به توی $B(t,a_۱)$ و $B(t,a_۲)$ توسعه یابد، اما چون t یک رأس بحرانی است نمی‌تواند به توی $B(t,c)$ توسعه یابد.

لم ۵. اگر t یک رأس بحرانی از مسیر P باشد و $c \in P$ آن‌گاه $t \in PC$.

اثبات.چون t یک رأس بحرانی است و $c \in P$ پس برای هر مسیر دیگری مانند P' که $c \in P'$ و $t \notin P$ داریم $F(P') \geq F(P)$ و هم‌چنین می‌دانیم که $c \in PC$ حال اگر $t \notin PC$ خواهیم داشت $F(PC) \geq F(P)$. که این با توجه به تعریف PC غیر ممکن است، بنابراین $t \in PC$.

لم ۶. فرض کنید $P \in \Pi$ ، $c \in P$ ، $PC = c$ آن‌گاه c یک رأس بحرانی از P است و رأس‌هایی مانند $u_۱, u_۲ \in NEI(c)$ موجود هستند به‌قسمی‌که برای هر $u \in NEI(c) \setminus u_۱$ که $Q_۵(c, u_۱) \geq Q_۵(c, u_۲) \geq Q_۵(c, u)$ خواهیم داشت :

$$F(P) = \frac{diam}{۲} \qquad (۲۵.۲)$$

$$G(P) = R(c) - Q_۵(c, u_۱) - Q_۵(c, u_۲) \qquad (۲۶.۲)$$

مثال ۱۵. برای مثال در درخت شکل ۱۱.۲ داریم $PC = c$ لذا بر طبق لم ۶ خواهیم داشت:

$$Q_۵(c, b_۱) = Q_۵(c, b_۲) \text{ ، } Q_۵(c, t) = ۱۱۰ \text{ ، } R(c) = ۲۳۰$$

بنابراین برای هر مسیر پرتو بهینه P که شامل مرکز مطلق درخت یعنی c است داریم :

$$G(P) = ۱۰۰ \text{ ، } F(P) = ۲۰$$

به وضوح اگر $P(c_۱, c_۲) = PC \neq c$ داریم $c_۱ \neq c$ و $c_۲ \neq c$.

لم ۷. فرض کنید $P \in \Pi$ و $P(c_۱, c_۲) = PC \subset P$ و $P \neq c$ پس داریم :

$$F(P) = F(PC), G(P) = G(PC) - A_۱ - A_۲ \qquad (۲۷.۲)$$

که $A_۱$ و $A_۲$ به‌صورت زیر تعریف می‌شوند :

$$A_i = \begin{cases} ۰ & deg(c_i) = ۱ \\ max\{Q_۵(c_i, u) \mid u \in NEI(c_i) \setminus PC\} & deg(c_i) \geq ۲ \end{cases} \qquad (۲۸.۲)$$

لم ۸. فرض کنید $P \in \Pi$ ، $c \in \Pi$ ، $PC = P(c_۱, c_۲) \neq c$ حال اگر قرار دهیم $c' = NEI(c) \cap P(c_۱, c)$ و $c'' = NEI(c) \cap P(c, c_۲)$ آنگاه $P(c', c'') \subset P$ و یا c یک رأس بحرانی است و برای $u_۱, u_۲ \in NEI(c)$ و $u \in NEI(c) \setminus u_۱$ که $Q_۵(u_۱) \geq Q_۵(u_۲) \geq Q_۵(u)$ روابط زیر برقرار است :

$$F(P) = \frac{diam}{۲}, G(P) = R(c) - Q_۵(c, u_۱) - Q_۵(c, u_۲) \qquad (۲۹.۲)$$

لم ۹. فرض کنید $P \in \Pi$ ، $PC = P(c_1, c_2) \neq c$ و رأس‌های $p_1, p_2 \in PC \cap P$ به قسمی موجود باشند که $c \in P(p_1, p_2)$ ، $p_1 \in P(c_1, p_2)$ ، $p_1 \neq c_1 \neq c$ ، $p_2 \neq c$. هم‌چنین فرض کنید $Q_3(p_1, c_1)\{Q_3(p_2, v)|v \in NEI(p_2) \setminus P(p_1, p_2)\}$ (توجه کنید اگر $p_2 \neq c_2$ آن‌گاه $max\{Q_3(p_2, v)|v \in NEI(p_2) \setminus P(p_1, p_2)\} = Q_3(p_2, c_2)$) حال اگر قرار دهیم $p' = NEI(p_1) \cap P(c_1, p_1)$ آن‌گاه هر $p'_1 \in P$ یا p_1 یک رأس بحرانی برای P است و داریم:

$$F(P) = Q_3(p_1, p'_1), \tag{۳۰.۲}$$

$$G(P) = G(P(p_1, p_2)) - A_1 - A_2 \tag{۳۱.۲}$$

که A_1 و A_2 بصورت زیر تعریف می‌شوند :

$$A_1 = max\{Q_5(p_1, u)|u \in NEI(p_1) \setminus PC\} \tag{۳۲.۲}$$

$$A_2 = \begin{cases} 0 & deg(p_2) = 1 \\ max\{Q_5(p_2, u)|u \in NEI(p_2) \setminus P(p_1, p_2)\} & deg(p_2) \geq 2 \end{cases} \tag{۳۳.۲}$$

اکنون قادریم با استفاده از قضایا و لم‌های فوق الگوریتم‌هایی را برای بدست آوردن مسیر میانه مرکزی یک درخت ارائه دهیم.

۳.۴.۲ الگوریتم $Berman$ برای یافتن مسیر میانه مرکزی درخت

الگوریتم $Berman$
ورودی: درخت $T = (V, E)$.
خروجی: مسیر میانه مرکزی درخت.

گام ۱ : با استفاده از الگوریتم‌های ۲.۳.۲ ، ۴.۳.۲ بیان شد مرکز مطلق درخت (c) مسیر مرکزی (PC)، قطر درخت و $MAXD(PC)$ را به‌دست آورید .

گام ۲ : برای تمام یال‌های $(a, b) \in E$ و $v \in V$ مقادیر $Q_i(a, b); i = 1, \ldots 5$ و $R(v)$ را محاسبه کنید .

گام ۳ : برای هر $t \in V_2$ و $t \neq c$ مراحل زیر را به ترتیب انحام دهید :

(a) رأس‌های $u_1, u_2 \in NEI(t) \setminus P(t,c)$ را به‌قسمی پیدا کنید که برای هر $u \in NEI(t) \setminus P(t,c) \setminus u_1$ داشته باشیم : $Q_5(t,u_1) \geq Q_5(t,u_2) \geq Q_5(t,u)$

(b) نقاط f و g را با استفاده از رابطه زیر پیدا کنید . $g = \frac{diam}{2} + d(t,c)$ و $f = R(t) - Q_5(t,u_1) - Q_5(t,u_2)$.

(c) نقطه (g,f) و رأس متناظر آن یعنی t را در W ثبت کنید.

در این گام برطبق لم (۲) ، W مجموعه‌ای از مسیرهای پرتو بهینه‌ای است که شامل c نیست.

گام ۴ : در این گام ازمسیر کمکی $\overline{P} = P(p_1,p_2)$ استفاده می‌شود . در ابتدای گام قرار دهید $p_1 = p_2 = c$.

مرحله(a) رأس‌های $u_1, u_2 \in NEI(c)$ را به‌قسمی پیدا کنید که برای هر $u \in NEI(c) \setminus u_1$ داشته باشیم : $Q_5(c,u_1) \geq Q_5(c,u_2) \geq Q_5(c,u)$

نمونه(۱) اگر $PC = c$ قرار دهید : $g = \frac{diam}{2}$ ، $f = R(c) - Q_5(c,u_1) - Q_5(c,u_2)$.و نقطه (g,f) را در W ثبت کرده و قرار دهید $p_1 = p_2 = c$ ومتوقف شوید . برطبق لم (۴) در این نمونه (g,f) نمایشی از همه مسیرهای پرتو بهینه شامل c است .

نمونه(۲) اگر$PC = P(c_1,c_2)$ ؛ از لم (۶) استفاده می‌کنیم . فرض کنید $c' = NEI(c) \cap P(c_1,c)$ و $c'' = NEI(c) \cap P(c,c_2)$.

اگر $\{c',c''\} = \{u_1,u_2\}$ قراردهید $p_1 = c'$ ، $p_2 = c''$ ، $\overline{p} = P(p_1,p_2)$

$MEAND(\overline{p}) = R(c) - Q_1(c,c') \cdot d(c,c') - Q_1(c,c'') \cdot d(c,c'')$ به مرحله (۲) بروید .

اگر $\{c',c''\} \neq \{u_1,u_2\}$ سپس قراردهید :

$g = \frac{diam}{2}$ ، $f = R(c) - Q_5(c,u_1) - Q_5(c,u_2)$ نقطه (g,f) را در W ثبت کنید

. قرار دهید $P_۱ = P_۲ = C$ حال از ابتدای گام قرار دهید $p_۱ = c^{'}$ ، $p_۲ = c^{''}$ ، $\overline{P} = P(p_۱, p_۲)$

برویم به $MEAND(\overline{P}) = R(c) - Q_۱(c, c^{'}) \cdot d(c, c^{'}) - Q_۱(c, c^{''}) \cdot d(c, c^{''})$. مرحله (۲)

مرحله(k) مرحله $k, k = ۲, ۳ \ldots$ ، در این مرحله $\overline{P} = P(p_۱, p_۲) \subset PC = P(c_۱, c_۲)$ ، $c \in \overline{P}$ ، $p_۲ \neq c$ ، $p_۱ \neq c$ ،و مقدار $MEAND(\overline{P})$ را می‌دانیم .

نمونه(۱) اگر $\overline{P} = PC(p_۱ = c_۱, p_۲ = c_۲)$ ازلم (۵) استفاده کنید . مقادیر زیر را پیدا کنید :

$$A_i = \begin{cases} ۰ & deg(c_i) = ۱ \\ max\{Q_۵(c_i, u) \mid u \in NEI(c_i) \setminus PC\} & deg(c_i) \geq ۲ \end{cases} \quad i = ۱, ۲ \qquad (۳۴.۲)$$

قرار دهید $g = MAXD(PC)$ و $f = MEAND(\overline{P}) - A_۱ - A_۲$. (g, f) را در W ثبت کرده و قرار دهید $(p_۱ = c_۱, p_۲ = c_۲)$. متوقف شوید .

نمونه(۲) اگر $\overline{P} \neq PC$ و $p_۱ \neq c_۱$ و $Q_۳(p_۱, c_۱) \geq \{Q_۳(p_۲, v) | v \in NEI(p_۲ \setminus P(p_۱, p_۲)\}$. از لم (۷) استفاده کنید .

(۱) اگر $deg(p_۱) \geq ۳$ ۳ بروید گام ۲ به .

اگر $deg(p_۱) \geq ۳$ ۳ . مقادیر زیر را پیدا کنید :

$$A_۱ = max\{Q_۵(p_۱, u) | u \in NEI(p_۱) \setminus PC\} \qquad (۳۵.۲)$$

$$A_۲ = \begin{cases} ۰ & deg(p_۲) = ۱ \\ max\{Q_۵(p_۲, u) \mid u \in NEI(p_۲)(p_۱, p_۲)\} & deg(p_۲) \geq ۲ \end{cases} \qquad (۳۶.۲)$$

وهم‌چنین رأس $u_۱ \in NEI(p_۱) \setminus PC$ را به قسمی بیابید که $Q_۵(p_۱, u_۱) = A_۱$.

قرار دهید $g = Q_۳(p_۱, p'_۱)$ و $f = MEAND(\overline{P}) - A_۱ - A_۲$ و نقطه (g, f) را در W ثبت کنید .

(۲) به ابتدا برگردید و قرار دهید:

$$MEAND(\overline{P}) = MEAND(\overline{P}) - Q_۱(p_۱, p'_۱) \cdot d(p_۱, p'_۱) \qquad (۳۷.۲)$$

$$p_۱ = p'_۱, \overline{P} = P(p'_۱, p_۲) \qquad (۳۸.۲)$$

به گام $k + ۱$ بروید.

نمونه(۳) اگر $\overline{P} \neq PC$ و $p_۲ \neq c_۲$ ، $Q_۳(p_۲, c_۲) \geq max\{Q_۳(p_۱, v)|v(p_۱)\backslash P(p_۱, p_۲)\}$. تمام عملیات در این نمونه شبیه نمونه (۲) است ، با این تفاوت که گره‌های با اندیس ۱ باید باگره‌های با اندیس ۲ جایگزین شوند . رخ دادن سایر نمونه‌ها غیر ممکن است .

گام‌های الگوریتم کامل است .

۴.۴.۲ الگوریتم $Averbakh$ برای یافتن مسیر میانه مرکزی درخت

اکنون الگوریتم دیگری را ارائه خواهیم داد که مشابه الگوریتم بالا مسیر میانه مرکزی درخت را پیدا می‌کند .

الگوریتم $Averbakh$

ورودی: درخت $T = (V, E)$.
خروجی: مسیر میانه مرکزی درخت.

در شروع الگوریتم فرض می‌کنیم نقطه $(g, f) \in W$ داده شده است .(داده‌های به‌دست آمده در الگوریتم (۱) مورد نیاز است) همچنین از مسیر کمکی $\tilde{P} = \tilde{P}(q_۱, q_۲)$ استفاده می‌کنیم که در ابتدا تعریف نشده است و معرف یک بخش از مسیر مورد تقاضا است .

گام ۱ : با توجه به گام ۳ الگوریتم (۱) اگر نقطه (g, f) متعلق به W است و رأس $t \in v_۲$ متناظر با رأس ثبت شده است ، آن‌گاه رأس‌های $u_۱, u_۲ \in NEI(t) \setminus P(t, c)$را طوری پیدا کنید که برای هر رأس $u \in NEI(t) \setminus P(t, c) \setminus u_۱$ داشته باشیم : $Q_۵(t, u_۱) \geq Q_۵(t, u_۲) \geq Q_۵(t, u)$ سپس قرار دهید $q_۱ = u_۱$ و $q_۲ = u_۲$ و $\tilde{P} = P(q_۱, q_۲)$ و به گام ۲ بروید .

فرض کنید در گام ۴ الگوریتم ۱ نقطه $(g, f) \in W$ و $p_۱, p_۲ \in PC$ متناظر با رأس‌های ثبت شده $p_۱ \in P(c_۱, c)$ و $p_۲ \in P(c, c_۲)$ باشند آن‌گاه :

(a) اگر $PC = c$ آن‌گاه رأس‌های $u_۱, u_۲ \in NEI(c)$ را طوری پیدا کنید که برای هر رأس $u \in NEI(c) \setminus u_۱$ داشته باشیم : $Q_۵(c, u_۱) \geq Q_۵(c, u_۲) \geq Q_۵(c, u)$ سپس قرار دهید $q_۱ = u_۱$ و $q_۲ = u_۲$ و $\tilde{P} = P(u_۱, u_۲)$ و به گام ۲ بروید .

(b) اگر $PC \neq c$ و $p_۱ = p_۲ = c$ ($(g, f) \in W$) رأس‌های $u_۱, u_۲ \in NEI(c)$ را به قسمی پیدا کنید که برای هر رأس $u \in NEI(c) \setminus u_۱$ داشته باشیم :
$Q_۵(c, u_۱) \geq Q_۵(c, u_۲) \geq Q_۵(c, u)$ و حدأقل یکی از رأس‌های $u_۱$ و $u_۲$ متعلق به PC نباشد . سپس قرار دهید : $q_۱ = u_۱$ ، $q_۲ = u_۲$ ، $\tilde{P} = P(u_۱, u_۲)$ و به گام ۲ بروید .

(c) اگر $PC \neq c$ و $\{p_۱, p_۲\} = \{c_۱, c_۲\}$ سپس قرار دهید : $q_۱ = c_۱$ ، $q_۲ = c_۲$ و $\tilde{P} = PC$ و به گام ۲ بروید .

(d) اگر $PC \neq c$ ، $p_۱ \neq c_۱$ ، $Q_۳(p_۱, c_۱) \geq max\{Q_۳(p_۲, v) | v \in NEI(p_۲) \setminus P(p_۱, p_۲)\}$ رأس $u_۱ \in NEI(p_۱) \setminus PC$ را به‌قسمی پیدا کنید که برای هر رأس $u \in NEI(p_۱) \setminus PC$ داشته باشیم : $Q_۵(p_۱, u_۱) \geq Q_۵(p_۱, u)$. اکنون قرار دهید $q_۱ = u_۱$ ، $q_۲ = p_۲$ ، $\tilde{P} = P(u_۱, p_۲)$ و به گام ۲ بروید .

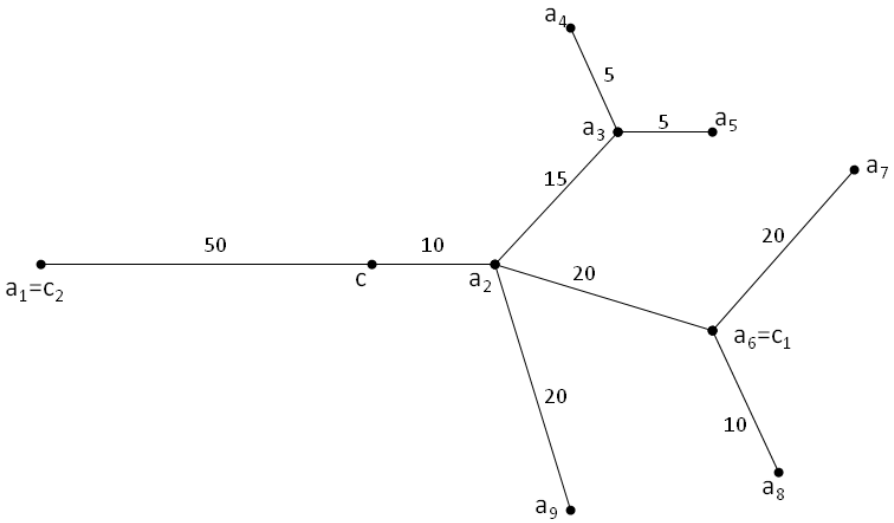
(e) اگر $PC \neq c$ ، $p_۲ \neq c_۲$ ، $Q_۳(p_۲, c_۲) > max\{Q_۳(p_۱, v) | v \in NEI(p_۱) \setminus P(p_۱, p_۲)\}$ رأس $u_۲ \in NEI(p_۲) \setminus PC$ را به‌قسمی پیدا کنید که برای هر رأس

$u \in NEI(p_2) \setminus PC$ داشته باشیم : $Q_5(p_2, u_2) \geq Q_5(p_2, u)$. اکنون قرار دهید $q_1 = p_1$ ، $q_2 = u_2$ ، $\tilde{P} = P(p_1, u_2)$ و به گام ۲ بروید .

گام $k, k = 2, 3, \ldots$: در گام $k-1$ مسیر $\tilde{P} = P(q_1, q_2)$ بدست می‌آید که $q_1 \neq q_2$. اگر $deg(q_1) \neq 1$ رأس $u_1 \in NEI(q_1 \setminus \tilde{P}$ را به‌قسمی بیابید که برای هر رأس $u \in NEI(q_1) \setminus \tilde{P}$ داشته باشیم $Q_5(q_1, u_1) \geq Q_5(q_1, u)$. اکنون قرار دهید $q_1 = u_1$ ، $q_2 = q_2$ و به گام $k+1$ بروید .

اگر $deg(q_1) = 1$ ، $deg(q_2) \neq 1$ ، رأس $u_2 \in NEI(q_2) \setminus \tilde{P}$ را طوری پیدا کنید که برای رأس $u \in NEI(q_2) \setminus \tilde{P}$ داشته باشیم $Q_5(q_2, u_2) \geq Q_5(q_2, u)$. اکنون قرار دهید $q_1 = q_1$ ، $q_2 = u_2$ و $\tilde{P} = P(q_1, u_2)$ و به گام $k+1$ بروید .

اگر $deg(q_1) = 1$ ، $deg(q_2) = 1$ متوقف شوید . مسیر $\tilde{P}$ همان مسیر مورد نظر است.



شکل ۱۲.۲ : مسیر میانه مرکزی

مثال ۱۶. درخت داده شده در شکل ۱۲.۲ را در نظر بگیرید ، با به‌کاربردن دو الگوریتم

۳.۴.۲ و ۴.۴.۲ مسیر میانه مرکزی درخت را پیدا کنید . وزن تمام رأس‌ها را یک در نظر بگیرید .

گام ۱ : در این گام $PC = P(a_۶, a_۱)$ یعنی داریم $c_۱ = a_۶, c_۲ = a_۱$ از طرفی $F(PC) = ۲۰, diam = ۱۰۰$

گام ۲ : چون درخت ساده است این گام را حذف می‌کنیم . اما محاسبه داده‌ها در گام‌های ۳, ۴ لازم است.

گام ۳ : چون $V_۲ = \{a_۲, a_۳, a_۶\}$ لذا ما باید فقط محاسبات را روی این سه گره انجام دهیم . به عنوان کاندیدا برای $t = a_۲$ داریم $NEI(t) \setminus P(t, c) = \{a_۳, a_۶, a_۹\}$ و $Q_۵(a_۲, a_۳) = ۵۰, Q_۵(a_۲, a_۶) = ۸۰, Q_۵(a_۲, a_۹) = ۲۰$

بنابراین $u_۱ = a_۶, u_۲ = a_۳$ (یعنی مسیرهای پرتو بهینه با رأس بحرانی $a_۲$ که شامل c نیست باید از $a_۲$ به توی $a_۳, a_۶$ گسترش یابد) اکنون داریم $R(a_۲) = ۲۳۵$ بنابراین $f = ۲۳۵ - ۸۰ - ۵۰ = ۱۰۵$ ، $g = ۶۰$ ونقطه $(۶۰, ۱۰۵)$ را در W ثبت می‌کنیم . پس نقطه $(۶۰, ۱۰۵)$ یک $\varphi - Representation$ از همه مسیرهای بهینه پرتو شامل رأس بحرانی $a_۲$ که شامل c نیست .(برای مثال مسیر $P(a_۷, a_۵)$) به همین ترتیب برای $t = a_۳$ ، $(u_۱, u_۲) = (a_۴, a_۵)$ و نقطه $(g, f) = (۷۵, ۲۸۵)$ را در W ثبت می‌کنیم . بطور مشابه $t = a_۶$ نقطه $(g, f) = (۸۰, ۲۸۵)$ را در W ثبت می‌کند .

گام ۴ : در ابتدای گام $p_۱ = p_۲ = c$.

مرحله(۱) چون $u_۱ = a_۲, u_۲ = a_۱$ پس نمونه دو را داریم $c' = a_۲, c'' = a_۱$ بنابراین $PC = P(c_۱, c_۲) = P(a_۶, a_۱) \neq c$ و هر مسیر پرتو بهینه شامل c باید به توی $a_۱$ و $a_۲$ گسترش یابد . به ابتدا باز گردید و قرار دهید $\bar{P} = P(a_۲, a_۱), p_۲ = a_۱, p_۱ = a_۲, Mean(\bar{P}) = ۱۶۵$ وبه مرحله ۲ بروید .

مرحله(۲) در این مرحله $p_۱ = a_{۲۱}, p_۲ = a_۱ = c_۲$ لذا نمونه دو را داریم.چون

$NEI(a_۲) \cap P(a_۶, a_۲) = a_۶$ لذا $p'_۱ = a_۶$. از طرفی چون $deg(a_۲) \geq ۳$ لذا باید مقادیر زیر را پیدا کنیم :

$$A_۱ = max\{Q_۵(a_۲, u) | u \in \{a_۳, a_۹\}\} = ۵۰$$

$$A_۲ = ۰ \text{ (چون } deg(a_۲) = ۱)$$

بنابراین هر مسیر پرتو بهینه که شامل $\bar{P} = P(a_۲, a_۱)$ است وشامل $a_۶$ نیست باید به توی $a_۳$ گسترش یابد . حال قرار می‌دهیم $g = Q_۳(a_۲, a_۶) = ۴۰$ و $f = Mean(\bar{P}) - A_۱ - A_۲ = ۱۱۵$ و نقطه $(۴۰, ۱۱۵)$ و رأس‌های متناظر آن یعنی $a_۲, a_۱$ را در W ثبت می‌کنیم . (نقطه $(۴۰, ۱۱۵)$ یک φ از همه مسیرهای پرتو بهینه است که شامل $P(a_۲, a_۱)$ است و شامل $a_۶$ نیست) اکنون قرار می‌دهیم:

$Mean(\bar{P}) = Mean(\bar{P}) - Q_۱(a_۲, a_۶) \cdot d(a_۲, a_۶) = ۱۶۵ - ۳۸۲۰ = ۱۰۵$ و $P_۱ = a_۶, \bar{P} = P(A_۶, A_۱)$ و به گام ۳ می‌رویم.

مرحله(۳) در این مرحله نمونه ۱ را داریم ، پس مقادیر زیر را پیدا می‌کنیم.

$$A_۱ = max\{Q_۵(a_۶, u) | u \in \{a_۷, a_۸\}\} = ۲۰$$

$$A_۲ = ۰ \text{ (چون } deg(c_۲) = ۱)$$

قرار می‌دهیم $g = Max(PC) = ۲۰$ و $f = Mean(\bar{W}) - A_۱ - A_۲ = ۸۵$ سپس نقطه $(۲۰, ۸۵)$ و رأس‌های متناظر آن یعنی $a_۶, a_۱$ را در W ثبت می‌کنیم . (نقطه $(۲۰, ۸۵)$ یک φ از همه مسیرهای پرتو بهینه شامل PC است ، متوقف شوید اکنون مجموعه W با ۵ نقطه بصورت زیر به‌دست می‌آید .

$$W = \{(۶۰, ۱۰۵), (۷۵, ۲۸۵), (۸۰, ۲۸۵), (۴۰, ۱۱۵), (۲۰, ۸۵)\}$$

در نتیجه $\phi(\Pi) = \{(۲۰, ۸۵)\}$ و نقطه $(۲۰, ۸۵)$ معرف خصوصیت تمام ممسیرهای پرتو بهینه است. در واقع هر مسیری با این خصوصیت یک جواب بهینه برای مسئله مورد نظر است. اکنون با استفاده از الگوریتم ۴.۴.۲ چنین مسیری را پیدا می‌کنیم.

گام ۱ : در ابتدای گام اول الگوریتم ۴.۴.۲ نقطه $(۲۰, ۸۵)$ نقطه‌ای از مجموعه W

بدست آمده از گام ۴ الگوریتم ۳.۴.۲ است. لذا قرار می‌دهیم $q_۱ = a_۶, q_۲ = a_۱$ $\tilde{P} = PC = P(a_۶, a_۱)$ و به گام ۲ می‌رویم.

گام ۲ : در این گام چون درجه رأس $a_۶$ بیشتر از یک است لذا باید بدنبال رأس u بگردیم که $Q_۵(a_۶, u) \geq Q_۵(a_۶, u)$ و این رأس، رأس $a_۷$ است. در نتیجه قرار می‌دهیم $q_۱ = a_۷, P = P(a_۷, a_۱)$ و به گام ۳ می‌رویم.

گام ۳ : در این گام چون درجه $a_۷$ و $a_۱$ هردو یک است، لذا متوقف می‌شویم و مسیر $P(a_۷, a_۱)$ را به عنوان مسیر میانه مرکزی درخت معرفی می‌کنیم.

فصل ۳

هسته درختی با k برگ

۱.۳ مقدمه و تاریخچه

تئوری مکان‌یابی قدیمی به مکان‌یابی وسیله‌هایی بر روی یک شبکه محدود می‌شد که تنها می‌توانستند بصورت نقاط یکتایی باشند. در حالی‌که در بسیاری از کاربردهای واقعی ممکن است وسیله‌هایی که مکان‌یابی می‌شوند بسیار بزرگ باشند یعنی به فرم یک مسیر یا یک درخت باشند. در حالتی‌که مسیر مورد نظر بصورت یک درخت است مسئله هسته درختی با k برگ[۱] مطرح می‌شود، در واقع این مسئله تعمیمی از مسئله هسته است و عبارت است از پیدا کردن زیردرختی با دقیقاً k برگ که مجموع فاصله وزنی رئوس شبکه تا آن مینیمم است. این مسئله برای اولین بار توسط پنگ ات آل[۲] معرفی شد و او یک الگوریتم با مرتبه زمانی $O(kn)$ برای یافتن آن ارائه داد [۱۱].
مسئله هسته درختی بعداً توسط شئورا و یونو[۳] بررسی شد و آن‌ها توانستند یک الگوریتم خطی را برای آن ارئه دهند [۱۲]. مسئله هسته درختی در ساختن شبکه های ارتباطی با سرعت بالا بسیار کاربرد دارد.

ساختار کلی این فصل بصورت زیر است: در بخش اول برخی تعاریف و نکات لازم را بیان می‌کنیم. در بخش دوم یک الگوریتم با مرتبه زمانی $O(kn)$ برای یافتن هسته درختی به ازاء مقادیر ثابت k ارائه می‌دهیم و برای آن یک مثال کاربردی بیان می‌کنیم.

[۱] $k-tree\ core$ [۲] $Peng\ et\ al$ [۳] $Shioura\ and\ Uno$

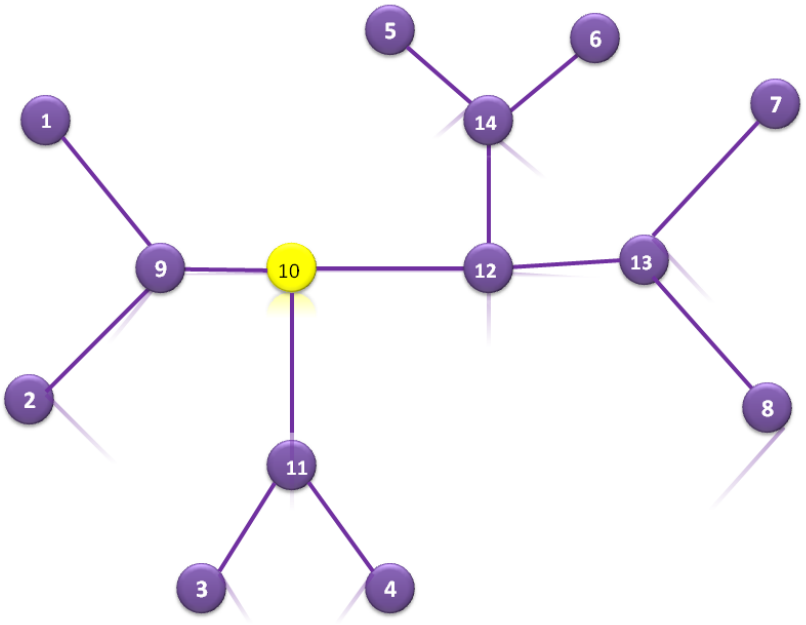
در بخش سوم الگوریتم دیگری با مرتبه زمانی $O(nlgn)$ برای یافتن هسته درختی به ازاء مقادیر متغیر و بزرگ k ارائه می‌دهیم.

۲.۳ فرمول‌بندی مسئله

در بخش قبل گفتیم که هسته درختی، زیردرختی با دقیقاً k برگ است که مجموع فاصله وزنی رئوس شبکه تا آن مینیمم است. در این بخش قصد داریم این مسئله را فرمول‌بندی کنیم که قبل از آن برخی از تعاریف مورد نیاز را بیان می‌کنیم.

تعریف ۱.۳. فرض کنید T یک درخت ریشه‌دار شده باشد، در این‌صورت $V(T)$ معرف مجموعه رأس‌های درخت T و $|T|$ معرف تعداد رأس‌های درخت است.

تعریف ۲.۳. (برگ[۱]) برگ در درخت T رأسی با درجه ۱ است و والد[۲] آن رأس بالای آن نزدیک به ریشه است.



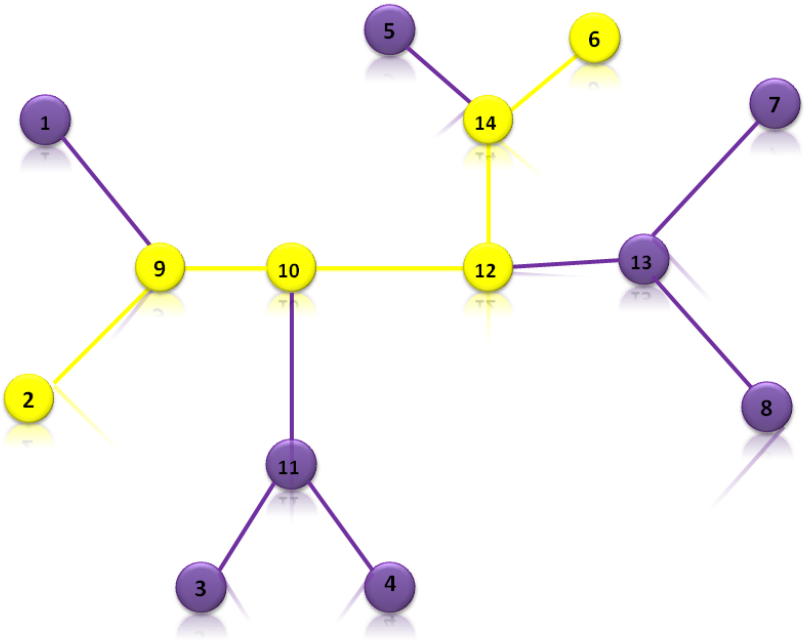
شکل ۱.۳ : فاصله از رأس

[۱] $Leaf$ [۲] $Parent$

تعریف ۳.۳. به ازای هر رأس $v \in V(T)$ فاصله از v را با $d(v) = \sum_{u \in V(T)} d(u, v)$ نشان می‌دهیم که $d(u, v)$ فاصله دو رأس دلخواه است.

مثال ۱۷. در درخت ۱.۳ فاصله از رأس $v_{۱۰}$ را بدست آورید.

$$d(v_{۱۰}) = ۱ + ۱ + ۱ + ۲ + ۲ + ۲ + ۲ + ۲ + ۲ + ۳ + ۳ + ۳ = ۳ = ۲۷$$



شکل ۲.۳ : فاصله از مسیر

تعریف ۴.۳. به ازای هر مسیر دلخواه P فاصله از مسیر را به‌صورت زیر تعریف می‌کنیم:

$$d(P) = \sum_{u \in V(T)} d(u, P) \qquad (۱.۳)$$

که $d(u, P) = min_{v \in P} d(u, v)$ است.

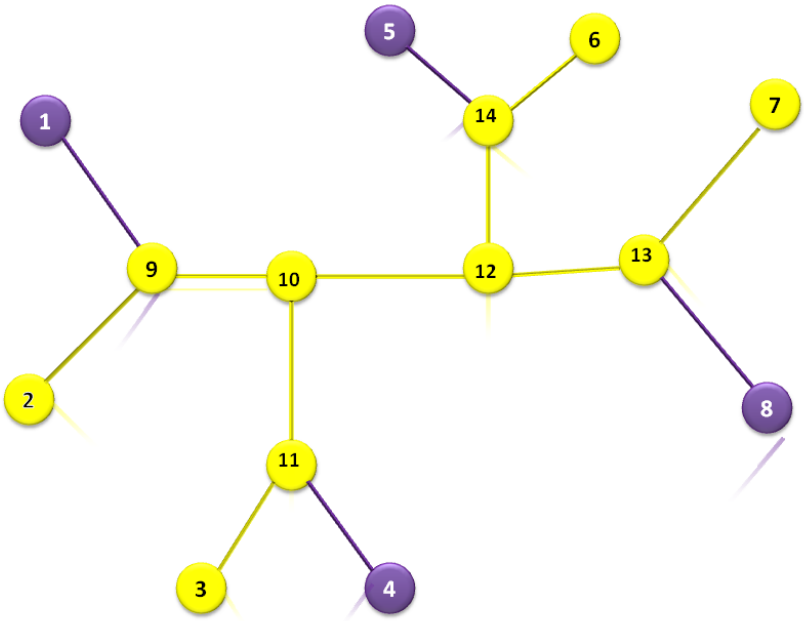
مثال ۱۸. در درخت ۲.۳ فاصله از مسیر $P_{v_۲, v_۶}$ را بدست آورید.

$$d(P_{v_۲, v_۶}) = ۱ + ۱ + ۱ + ۱ + ۲ + ۲ + ۲ + ۲ = ۱۲$$

تعریف ۵.۳. به ازاء هر زیر درخت T' از درخت T فاصله از T' را بصورت زیر تعریف می‌کنیم:

$$d(T') = \sum_{v \in V(T)} d(v, T') \qquad (۲.۳)$$

که $d(v, T') = \min_{u \in V(T')} d(u, v)$ می‌باشد.



شکل ۳.۳: فاصله از زیر درخت

مثال ۱۹. در درخت ۳.۳ فاصله از زیر درخت $T' = \{۲, ۹, ۱۰, ۱۱, ۳, ۱۲, ۱۴, ۶, ۱۳, ۷\}$ را بدست آورید.

$$d(T') = ۱ + ۱ + ۱ + ۱ = ۴$$

تعریف ۶.۳. هسته درختی با k برگ، زیر درختی از درخت T است که دقیقاً k برگ دارد و فاصله از آن مینیمم مقدار است و آن را به اختصار با KTC نشان می‌دهیم. تابع هدف این مسئله بصورت زیر است:

$$F(T') = \min_{T' \in T} \sum_{v \in V(T)} d(v, T') \qquad (۳.۳)$$

تعریف ۷.۳. اگر $P_{v,l}$ مسیری از رأس v تا یک برگ مانند l باشد، آن‌گاه فاصله ذخیره شده به‌وسیله این مسیر را به‌صورت زیر تعریف می‌کنیم:

$$DSAV(P_{v,l}) = d(v) - d(P_{v,l}) \qquad (۴.۳)$$

که $DSAV(P_{v,l})$ بهترین مقدار برای انتخاب مسیر گسترش‌یافته از رأس v است.

تعریف ۸.۳. فرض کنید $P_{v,l}$ مسیری به‌صورت $< v_۰, v_۱, \ldots, v_r >$ باشد، بطوری‌که $v = v_۰$ و $l = v_r$. دراین‌صورت زیر درخت آویزان شده روی v_i را با $Tr(v_i)$ نشان می‌دهیم. در واقع $Tr(v_i)$ زیر درخت تولید شده به‌وسیله مجموعه رأس‌های متصل شده به v_i به‌وسیله مسیرهای است که تنها شامل v_i و آن رأس‌هایی است که در مسیر $P_{v,l}$ نیستند. از این‌رو داریم:

$$DSAV(P_{v,l}) = \sum_{i=۱}^{r} i|Tr(v_i)| \qquad (۵.۳)$$

که $Tr(v_i)$ زیر درخت‌های آویزان شده روی رأس v_i است که در این زیر درخت‌ها وزن رئوس و طول یال‌ها یک است.

مثال ۲۰. در درخت ۲.۳ فاصله ذخیره شده به‌وسیله مسیر $P_{v_۲,v_۶}$ را با استفاده از تعاریف ۷.۳ و ۸.۳ بدست آورید.

$DSAV(P_{v_۲,v_۶}) = d(v_۲) - d(P_{v_۲,v_۶}) = ۴۷ - ۱۲ = ۳۵$

$DSAV(P_{v_۲,v_۶}) = \sum_{i=۱}^{r} i|Tr(v_i)| = ۱ \times ۲ + ۲ \times ۴ + ۳ \times ۴ + ۴ \times ۲ + ۵ \times ۱ = ۳۵$

تعریف ۹.۳. فرض کنید $< v_۰, v_۱, \ldots, v_r >$ هسته درخت T باشد، $Tr(v_i); i = \ldots, r$ را زیر درخت های آویزان شده روی هسته می‌نامیم که متصل به رأس v_i هستند. همچنین باید توجه داشت که $Tr(v_i)$ دوبه‌دو مجزا هستند و $\cup_{i=۱}^{r} V(Tr(v_i)) = V(T)$.

اکنون قادریم با استفاده از تعاریف بالا الگوریتم هسته درختی را که بر پایه الگوریتم هسته بنا شده است و از یک استراتژی قوی برای گسترش هسته به یک هسته درختی استفاده می‌کند ارائه دهیم.

۱.۲.۳ الگوریتم بدست آوردن KTC به ازاء مقادیر ثابت K

این بخش شامل دو قسمت است. در بخش اول با استفاده از تعاریف بخش قبل الگوریتم هسته درختی را که بر پایه الگوریتم هسته بنا شده است و از یک استراتژی قوی برای گسترش هسته به یک هسته درختی استفاده می‌کند بیان کنیم. این مسئله در سال ۱۹۹۲ برای اولین بار توسط پنگ ات آل[۱] معرفی شد و او یک الگوریتم با مرتبه زمانی $O(kn)$ برای یافتن آن ارائه داد [۱۱]. نحوه عملکرد این الگوریتم به این ترتیب است که ابتدا هسته درخت را پیدا می‌کند سپس اقدام به اضافه کردن یال به آن می‌کند تا زمانی‌که تعداد برگ‌های درخت به k برسد. در بخش دوم به پیاده سازی الگوریتم فوق بر روی یک مثال کاربردی می‌پردازیم.

الگوریتمKTC به ازاء مقادیر ثابت k
ورودی: درخت T ، $k < e$
خروجی: KTC

گام۱) با استفاده از الگوریتم‌های هسته که در فصل ۲ بیان شد، هسته $< v_0, v_1, \ldots, v_r >$ درخت T را پیدا می‌کنیم.

گام۲) فرض کنید $Tr(v_i); i = 1, \ldots, r$ زیر درخت های آویزان شده روی هسته به‌وسیله رأس v_i باشند. در این صورت قرار می‌دهیم:

$P := k - treecore$ و $CANSET := \{v_0, v_1, \ldots, v_r\}$ و $j := k - 2$ که مجموعه $CANSET$ شامل رأس‌های هسته است.

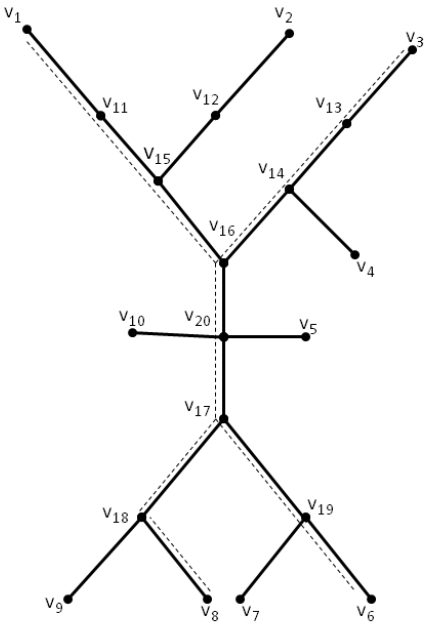
به ازای هر رأس $v \in CANSET$ قرار دهید: $DSAV(v) := \max_{l \in Tr(v)} DSAV(P_{v,l})$

[۱] *Peng et al*

گام۳) تا زمانی‌که $j > ۰$ است مراحل زیر را انجام دهید:

فرض کنید v_q رأسی باشد که به ازاء آن $DSAV(v_q) := \max_{v \in CANSET} DSAV(v)$ و $P_q =< v_{q۰}, \ldots, v_{qs} >$ مسیری در $Tr(v_q)$ باشد که مقدار $DSAV(v_q)$ را بدست می‌دهد $(v_{q۰} = v_q)$ و هم‌چنین فرض کنید $Tr(v_{qi}); i = i, \ldots, r$ زیر درخت‌های آویزان شده روی مسیر P_q باشند. در این صورت یال $e_{q۰,q۱}$ را از درخت $Tr(v_q)$ حذف کرده و قرار دهید:

$KTC := KTC \cup P_q$ ، $CANSET := CANSET \cup \{v_{q۱}, \ldots, v_{qs}\}$ و به ازاء $i = ۱, \ldots, s$ قرار دهید: $DSAV(v_i) := \max_{l \in Tr(v_i)} DSAV(P_{v_i,l})$ و $j := j - ۱$ و KTC را معرفی کنید.



شکل ۴.۳: هسته درختی

مثال ۲۱. در درخت شکل ۴.۳ هسته درختی با ۴ برگ را پیدا کنید(وزن رئوس و طول یال‌ها را یک در نظر بگیرید.)

ابتدا با استفاده از الگوریتم هسته که در بخش ۲ معرفی کردیم، هسته درخت را پیدا می‌کنیم: $P =< v_۱, v_{۱۱}, v_{۱۵}, v_{۱۶}, v_{۲۰}, v_{۱۷}, v_{۱۸}, v_۸ >$ است. اکنون مجموعه $CANSET$ را پیدا

می‌کنیم $CANSET := \{v_۱, v_{۱۱}, v_{۱۵}, v_{۱۶}, v_{۲۰}, v_{۱۷}, v_{۱۸}, v_۸\}$ سپس ماکزیمم فاصله‌های ذخیره شده را برای زیر درخت‌های آویزان شده روی رأس‌های مسیر C را بدست می‌آوریم که درجدول ۱.۳ نشان داده شده است.

جدول ۱.۳: مقادیر $DSAV$ برای رئوس درخت

v_i	$v_۱$	$v_{۱۱}$	$v_{۱۵}$	$v_{۱۶}$	$v_{۲۰}$	$v_{۱۷}$	$v_{۱۸}$	$v_۸$
$DSAV$	۰	۰	۳	۷	۱	۴	۱	۰

به توجه به جدول ۱.۳ می‌بینیم که مسیر $P' =< v_۳, v_{۱۳}, v_{۱۴}, v_{۱۶} >$ باید به درخت اضافه شود و همچنین رأس‌های $v_۳, v_{۱۳}, v_{۱۴}$ به مجموعه $CANSET$ اضافه می‌شوند. اکنون با توجه به جدول ۲.۳ واضح است که مسیر $< v_۶, v_{۱۹}, v_{۱۷} >$ در تکرار دوم باید به درخت اضافه شود و در این مرحله چون $k = ۴$ الگوریتم به پایان رسیده و هسته درختی به‌صورتی که در شکل نشان داده شده بدست می‌آید.

جدول ۲.۳: مقادیر $DSAV$ برای رئوس درخت

v_i	$v_۱$	$v_{۱۱}$	$v_{۱۵}$	$v_{۱۶}$	$v_{۲۰}$	$v_{۱۷}$	$v_{۱۸}$	$v_۸$	$v_۳$	$v_{۱۳}$	$v_{۱۴}$
$DSAV$	۰	۰	۳	۰	۱	۴	۱	۰	۰	۰	۱

۲.۲.۳ روابط بین هسته و هسته درختی

در ادامه به معرفی چند لم می‌پردازیم که ارتباط بین هسته و هسته درختی را روشن می‌کند.

لم ۱۰. در یک درخت T هر KTC با یک هسته اشتراک دارد.

اثبات. فرض کنید C یک هسته با برگ‌های $l_۱$ و $l_۲$ و KTC یک هسته درختی باشد. به برهان خلف فرض کنید C و KTC در هیچ رأسی مشترک نباشند. در نتیجه یک مسیر یکتا مانند $P_{u,v}$ از C به KTC به قسمی وجود دارد که $u \in C$ و $v \in KTC$ است. اکنون فرض کنید $l_۳$ برگی در KTC باشد. در نتیجه چون C هسته درخت T است $DSAV(P_{v,l_۲}) \geq$

$DSAV(P_{u,l_3})$ است از طرفی داریم: $DSAV(P_{u,l_3}) \geq DSAV(P_{v,l_3})$ لذا بنا به دو نامساوی فوق داریم: $DSAV(P_{v,l_2}) \geq DSAV(P_{u,l_3}) \geq DSAV(P_{v,l_3})$ اکنون با اضافه کردن مسیر P_{v,l_2} به KTC و حذف کردن بخش مناسبی از مسیر P_{v,l_3} زیر درختی با مقدار $d(T')$ کمتری از $d(T)$ را بدست آوریم که این موضوع تناقض با این موضوع دارد که KTC یک هسته درختی است.

□

لم ۱۱. در یک درخت T هر KTC شامل یک هسته است.

اثبات. فرض کنید C هسته‌ای باشد که مشمول در یک KTC نیست. در این صورت دو حالت زیر اتفاق می‌افتد.

(۱) مسیر C و زیر درخت KTC فقط در یک رأس یکتا u مشترک باشند. در این حالت ادعا می‌کنیم که به ازاء $k+2$ برگ $l \in C \cup KTC$ باید تمامی مقادیر $DSAV(P_{u,l})$ برابر باشند. اگر چنین نباشد، مثلاً مسیری مانند $P_{u,l_.}$ در KTC وجود داشته باشد به قسمی که دارای فاصله ذخیره شده کمتری از یک مسیری مانند P_{u,l_1} در C است. درنتیجه می‌توانیم برای ایجاد یک زیر درخت جدید T' با k برگ با این خصوصیت که $d(T') < d(KTC)$ مسیر $P_{u,l_.}$ را با مسیر P_{u,l_1} جایگزین کنیم. در نتیجه مسیر متصل کننده هر دو برگ در میان k برگ KTC و گذشته از رأس u خود تشکیل یک هسته را می‌دهد.

(۲) مسیر C و زیردرخت KTC در یک مسیر مانند $P_{u,v}$ مشترک باشند. در این حالت حداقل یکی از دو رأس u و v غیر برگی هستند. فرض کنید u غیر برگی و v یک برگ باشد. در نتیجه با برهانی شبیه به حالت قبل ادعا می‌کنیم که برای تمام مسیرهای $P_{u,l}$ که l برگی در C یا KC و $P_{u,l} \cap P_{u,v} = u$ است، مقادیر $DSAV(P_{u,l})$ برابر هستند. اکنون فرض کنیم که S مجموعه ای از برگ‌های $l \in KTC$ است به قسمی که $P_{u,l} \cap P_{u,v} = u$ و این موضوع نتیجه می‌دهد که مسیر $P_{v,l}$ که $l \in S$ است هسته می‌باشد. در حالتی که u, v هر

دو غیر برگی هستند تنها کافیست روند اثبات فوق را برای رأس v نیز اجرا کنیم. با این فرض که S' مجموعه‌ای از برگ‌های $l \in KTC$ باشد به قسمی‌که $P_{v,l} \cap P_{u,v} = v$ است. و بلافاصله می‌توانیم نتجه بگیریم که مسیر $P_{l,l'}$ که $l \in S$ و $l' \in S'$ است، هسته درخت است.

□

قضیه ۱.۳. الگوریتم ۱.۲.۳ به درستی یک KTC از درخت T را پیدا می‌کند.

اثبات. با توجه به لم ۱۱ می‌دانیم که یک KTC شامل یک هسته است. همچنین می‌توان یک هسته را با اضافه کردن یک یال مشخص به یک KTC تبدیل کرد.
(۱) در هر تکرار از الگوریتم داریم: $d(KTC \cup P_q) \leq d(KTC \cup P_{q'})$ که $P_{q'}$ یک مسیر آغازی از رأس $q' \in KTC$ است بطوری‌که $P_{q'} \cap KTC = q'$. این مسئله باتوجه به روند کار الگوریتم قابل درک است. به عنوان نتیجه، اگر یک KTC شامل یک هسته مانند C باشد و الگوریتم از هسته C شروع شود به راحتی می‌توان یک KTC را تولید کرد.
(۲) یک هسته در تعدادی هسته درختی قرار دارد. به منظور اثبات این مطلب فرض کنید C' یک هسته و T' زیر درخت متناظر با آن باشد . همچنین فرض کنید C هسته‌ای باشد که درون یک هسته درختی مانند KTC قرار دارد. در نتیجه چون $C \cap C' \neq \phi$ است $T'' = KTC \cap T' \neq \phi$ از طرفی با توجه به این‌که T' بر روی KTC انتخاب شده است نتیجه می‌گیریم:

$$\sum_{v \in T} d(v, T') \leq \sum_{v} d(v, KTC)$$

که این موضوع نتیجه می‌دهد که T' یک KTC است.

اکنون با ترکیب کردن حالت‌های ۱ و ۲ خواهیم دید که در i امین تکرار ، الگوریتم مورد نظر یک $(i+۲) - treecore$ از درخت T را تولید می‌کند. در نتیجه الگوریتم ۱.۲.۳ یک KTC را پیدا می‌کند.

نکته ۳. در یک درخت T یک $k-treecore$ باید شامل یک $(k-۱)-treecore$ باشد.

قضیه ۲.۳. مرتبه زمانی الگوریتم ۱.۲.۳ برابر $O(kn)$ است.

اثبات. می‌دانیم که الگوریتم پیدا کردن هسته یک درخت، دارای مرتبه زمانی $O(n)$ است. همچنین به ازاء هر رأس $v \in CANSET$ عملیات محاسبه $DSAV(v)$ دارای مرتبه زمانی $O(|Tr(v)|)$ می‌باشد. از طرفی چون $\sum_{v \in CANSET} |Tr(v)| = n$ در نتیجه مرتبه محاسباتی $DSAV(v)$ برابر $O(n)$ است و در نهایت به آسانی می‌توان دید که هر تکرار از عملیات حلقه تکرار برنامه اصلی، پیچیدگی $O(n)$ را دارد. در نتیجه مرتبه زمانی الگوریتم برابر با $O(kn)$ است. □

۳.۲.۳ الگوریتم بدست آوردن KTC به ازاء مقادیر بزرگ K

در بخش قبل با یک الگوریتم کارا برای بدست آوردن KTC آشنا شدیم. و دیدیم که اگر k یک مقدار ثابت باشد مرتبه زمانی الگوریتم برابر با $O(kn)$ است. درحالی‌که اگر k ثابت نباشد، مثلاً $k = n^{\alpha}$ که $۰ < \alpha < ۱$ آن‌گاه پیچیدگی الگوریتم برابر با $O(n^{۱+\alpha})$ می‌باشد. اکنون می‌خواهیم الگوریتمی موثر برای حالتی‌که k بسیار بزرگ است ارائه بدهیم. مرتبه زمانی این الگوریتم برابر با $O(klogn)$ است.

قبل از معرفی الگوریتم فوق برخی از تعاریف و اصطلاحات مورد نیاز را بیان می‌کنیم.

تعریف ۱۰.۳. به ازاء برگ $l \in T$ سر شاخه منتهی به l را با $head(l)$ نشان می‌دهیم، $head(l)$ جد برگ l است که درجه پدر آن یعنی $p(head(l))$ بیشتر از دو است.

تعریف ۱۱.۳. فاصله ذخیره شده به وسیله شاخه منتهی به برگ l را با $ds(l)$ نشان می‌دهیم و بصورت زیر تعریف می‌کنیم:

$$ds(l) = DSAV(P_{p(head(l)),l})$$

مثال ۲۲. در درخت ۴.۳ برای رأس $v_۱$ مقادیر $head(v_۱)$ و $ds(v_۱)$ را پیدا کنید.

$head(v_۱) = v_{۱۱}, head(v_{۱۱}) = v_{۱۱} \Longrightarrow head(v_۱) = v_{۱۱}$

$ds(v_۱) = DSAV(P_{p(head(v_۱)),v_۱}) = DSAV(P_{p(v_{۱۱}),v_۱}) = DSAV(P_{v_{۱۵},v_۱}) = ۳$

تعریف ۱۲.۳. $|T(head(l))|$ اندازه زیر درخت ریشه‌دار شده در $head(l)$ است.

اکنون می‌توانیم با استفاده از تعاریف فوق الگوریتم KTC را بصورت زیر فرمول بندی کنیم. دوباره فرض می‌کنیم تنها حالت مناسب در الگوریتم حالت $k < e$ است.

الگوریتمKTC به ازاء مقادیر بزرگ K
ورودی: درخت T ، $k < e$
خروجی: KTC

عملیات $COMPUT(v)$

قرار دهید:

$$|T(p(v))| := |T(v)| + |T(p(v))|$$

$$leaf(p(v)) := leaf(v)$$

$$v := p(v)$$

$$p(v) := p(p(v))$$

$$ds(leaf(v)) := ds(leaf(v)) + |T(v)|$$

تا زمانی‌که $deg(p(v)) > ۲$ قرار دهید $head(leaf(v)) := v$

برنامه اصلی

گام۱) شروع: به ازاء هر رأس $v \in T$ اگر v برگ است قرار دهید:

$$leaf(v) := v$$

$$ds(v) := |T(v)| = ۱$$

در غیر اینصورت قرار دهید: $leaf(v) = \phi$

به ازاء هر برگ l انجام دهید:

اگر ۲ $= deg(p(l))$ عملیات $COMPUT(l)$ را انجام دهید.

گام۲) حذف مینیمم شاخه: عملیات زیر را تکرار کنید:

$v := DEL(Q);$

$|T(p(head(v)))| := |T(p(head(v)))| + |T(head(v))|$

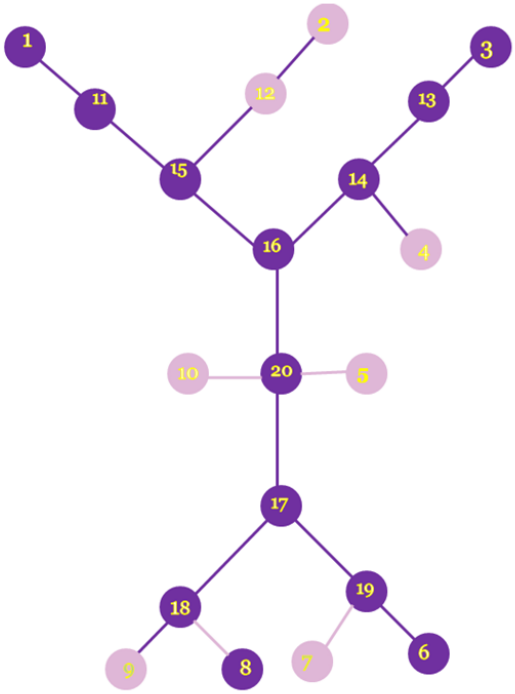
$deg(p(head(v))) := deg(p(head(v))) - ۱$

یال $< head(v), p(head(v)) >$ را از درخت T حذف کنید.و قرار دهید$r := r - ۱$

اگر ۲ $= deg(p(head(v)))$ و تنها یکی از دو همسایه اش ، مثلا u در رابطه $leaf(u) \neq \phi$ صدق کند آنگاه عملیات $COMPUT(u)$ و $UPDAT(Q, leaf(Q))$ را انجام دهید.

اگر ۰ $= r$ توقف کنید.

مثال ۲۳. در درخت ۵.۳ هسته درختی با ۴ برگ را پیدا کنید.



شکل ۵.۳: هسته درختی با ۴ برگ

گام۱ :ابتدا برای تمام برگ‌های درخت، با استفاده از مقادیر ds آن‌ها یک صف اولویت‌دار تشکیل می‌دهیم که در جدول ۳.۳ نشان داده شده است.

جدول ۳.۳: مقادیر ds برای برگ‌های درخت

v_i	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
ds	۳	۳	۳	۱	۱	۱	۱	۱	۱	۱

گام۲ :در این گام استفاده از عملیات $COMPUT(u)$ مقادیر $head(l)$ را برای تمام برگ‌های درخت پیدا می‌کنیم. نتیجه در جدول ۴.۳ نشان داده شده است.

جدول ۴.۳: مقادیر $head(l)$ برای برگ‌های درخت

v_i	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
$head(l)$	v_{11}	v_{12}	v_{13}	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}

گام۳:در این گام مقدار r برابر است با $r = m - k = ۱۰ - ۴ = ۶$ لذا عملیات حذف مینیمم شاخه را تا زمانی‌که $r \neq ۰$ ادامه می‌دهیم.

طی اجرای عملیات حذف مینیمم شاخه، رأس‌های $\{v_4, v_5, v_{10}, v_7, v_9\}$ از درخت حذف می‌شوند. در نتیجه مقادیر ds رأس‌های $\{v_3, v_6, v_8\}$ بصورتی‌که در جدول ۵ نشان داده شده بهنگام می‌شوند.

جدول ۵.۳: مقادیر ds برای رئوس درخت

v_i	v_1	v_2	v_3	v_8	v_6
ds	۳	۳	۷	۴	۴

اکنون با توجه به مقادیر ds در جدول ۵.۳ واضح است که یکی از دو رأس v_1 یا v_2 باید حذف شوند. در این مرحله چون $k = ۴$ لذا الگوریتم خاتمه یافته و هسته درختی بصورتی‌که در شکل نشان داده شده بدست می‌آید.

فصل ۴

هسته درختی با k برگ و طول مقید l

۱.۴ مقدمه و تاریخچه

مسئله هسته درختی عبارت است از پیدا کردن زیر درختی با دقیقاً k برگ و طول مقید l به‌گونه‌ای مجموع فاصله رئوس درخت تا آن مینیمم باشد. این مسئله در سال ۲۰۰۲ توسط بکر [۱] بررسی شد و او توانست دو الگوریتم برای یافتن هسته درختی ارائه دهد [۱۵]. الگوریتم اول برای حالتی است که درخت بی‌وزن باشد و مرتبه زمانی این الگوریتم $O(n^۲)$ است. الگوریتم دوم زمانی کاربرد دارد که درخت وزن‌دار باشد، در این حالت مرتبه محاسباتی الگوریتم بکر $O(n^۲logn)$ است. مسئله هسته درختی در شبکه‌های ارتباطی با سرعت بالا مانند شبکه‌های اینترنتی بسیار کاربرد دارد.

۲.۴ مدل‌بندی مسئله

قبل از فرمول بندی مسئله هسته درختی برخی از تعاریف مورد نیاز این مسئله را بیان می‌کنیم.

تعریف ۱.۴. در یک درخت $T = (V, E)$ با تعداد رئوس $|V| = n$ طول هر یال $(i, j) \in T$ را با $a(i, j)$ و وزن هر رأس $v \in V$ را با $W(v)$ نشان می‌دهیم.

تعریف ۲.۴. به ازاء دو رأس دلخواه $u, v \in V$ مسیر یکتایی که این دو رأس را به هم متصل

[۱] $Beker$

می‌کند را با $P_{u,v}$ نشان می‌دهیم. در نتیجه فاصله بین این دو رأس یعنی $d(u,v)$ برابر با طول مسیر $P_{u,v}$ است.

تعریف ۳.۴. فرض کنید P مسیر دلخواهی در درخت T باشد ، در این‌صورت طول این مسیر را با $L(P) = \sum_{(i,j)\in P} a(i,j)$ نشان می‌دهیم.

تعریف ۴.۴. (قطر[۱]) یک درخت، مسیری است که بیش‌ترین طول ممکن را دارد و آن‌را با d_T نشان می‌دهیم.

تعریف ۵.۴. فرض کنید $T' = (V', E')$ زیر درختی از درخت T باشد. در این‌صورت فاصله تمام رأس‌های درخت تا این زیر درخت را با $d(T') = \sum_{v\notin V'} d(v, T')$ نشان می‌دهیم، که $d(T')$ می‌نیمم فاصله $v \notin V'$ تا یک رأس در درخت T' است. همچنین $d(T')$ را با $DISTSUM(T')$ نشان می‌دهیم.

تعریف ۶.۴. یک $(k,l) - Core$ زیر درختی از درخت T است با حداکثر k برگ و قطر حداکثر l بطوری‌که مقدار $DISTSUM$ آن کمترین مقدار ممکن است.

تعریف ۷.۴. به ازاء یک رأس دلخواه $c \in V$ درخت ریشه‌دار شده در این رأس را با T^c نشان می‌دهیم.

تعریف ۸.۴. یک $KTC(c)$ زیر درختی است با حداکثر k برگ و طول حداکثر l که شامل c است و مقدار $DISTSUM$ آن کمترین مقدار ممکن است.

تعریف ۹.۴. زیر درختی از درخت T^c که شامل رأس v و تمام فرزندان آن است را با T^c_v نشان می‌دهیم.

تعریف ۱۰.۴. مجموع وزنی رئوس در درخت‌های T^c و T^c_v را به ترتیب با $sum(v)$ و $sum_c(v)$ نشان می‌دهیم.

[۱] $Diameter$

تعریف ۱۱.۴. فاصله یک رأس v از تمام رأس‌های دیگر در درخت T_v^c را با $d_v(c)$ نشان می‌دهیم.

تعریف ۱۲.۴. برای یک مسیر $P_{v,u}$ و یک مسیر $P_{u,x}$ فاصله ذخیره شده مسیر $P_{u,x}$ نسبت به مسیر $P_{v,u}$ را با $sav(P_{v,u}, P_{u,x}) = d(P_{v,u}) - d(P_{u,x})$ نشان می‌دهیم.

اگر مسیر $P_{v,u}$ تنها شامل یک رأس v باشد به‌جای $sav(P_{v,u}, P_{u,x})$ قرار می‌دهیم $sav(v, P_{v,x})$ که این sav در خاصیت زیر صدق می‌کند:

به ازاء دو مسیر $P_{v,u}$ و $P_{u,x}$ به ترتیب در زیر درخت‌های T_v^c ، T_v^c با $u \in T_v^c$ و یال‌های مجزاء، فاصله ذخیره شده مسیر$P_{v,x}$ بصورت زیر محاسبه می‌شود:

$$sav(v, P_{v,x}) = sav(v, P_{v,u}) + sav(u, P_{u,x})$$

تعریف ۱۳.۴. به ازاء یک رأس دلخواه $v \in V$ و زیردرخت ریشه‌دار شده T_v^c در v مقادیر $sum_c(v)$ و $d_v(c)$ را بصورت‌های زیر تعریف می‌کنیم:

$$sum_c(v) = \begin{cases} w(v) & \text{اگر } v \text{ یک برگ از } T^c \text{ است} \\ w(v) + \sum_{u \in child(v)} sum_c(u) & \text{در غیر این‌صورت} \end{cases} \quad (۱.۴)$$

$$d_c(v) = \begin{cases} ۰ & \text{اگر } v \text{ یک برگ از } T^c \text{ است} \\ \sum_{u \in child(v)} [d_c(v) + a(v,u) sum_c(u)] & \text{در غیر این‌صورت} \end{cases} \quad (۲.۴)$$

تعریف ۱۴.۴. به ازاء یک رأس $u \in T_v^c$ و $u \neq v$، $f(u)$ را پدر رأس u در نظر می‌گیریم. و مقدار$sav(v, P_{v,u})$ را بصورت زیر تعریف می‌کنیم:

$$sav(v, P_{v,u}) = sav(v, P_{v,f(u)}) + a(f(u), u) sum_c(u) \quad (۳.۴)$$

اگر $v = f(u)$ آن‌گاه $sav(v, P_{v,f(u)}) = sav(v, v) = ۰$

در این بخش برخی تعاریف و نکات لازم برای معرفی مسئله $(k,l)-core$ را بیان می‌کنیم.
به ازاء یک درخت ریشه‌دار شده T^c ابتدا $k-treecore(c)$ را پیدا می‌کنیم.
به اینصورت که بعد از ریشه‌دار کردن درخت در یک رأس دلخواه مانند c،برای ساختن مسیرهای را پیدا می‌کنیم که از c شروع شده باشند و تا یک برگ ادامه داشته باشند. همچنین فرض کنید $v_1,\ldots,v_m$ فرزندان رأس c باشند. علاوه براین فرض کنید L^1 و L^2 دو مسیر باشند که از C عبور می‌کنند و دو فرزند مختلف آن یعنی v_i، v_j ؛ $i\neq j$ دو برگ در T^c باشند.که به‌ترتیب دارای ماکزیمم فاصله ذخیره شده و دومین ماکزیمم فاصله ذخیره شده هستند.اکنون فرض کنید مسیر P_c مسیر بدست آمده از اجتماع این دو L^1 و L^2 مسیر باشد کهP_c را هسته درخت T^c که از رأس c عبور می‌کند می‌نامیم.

تعریف ۱۵.۴. (هسته موضعاً ریشه‌دار شده[1]) به ازاء یک زیر درخت S از درخت T^c و یک رأس $v\notin S$، فرض کنید u رأسی از درخت T^c باشد که همسایه v است و در رابطه مقابل صدق می‌کند $d(u,S)=d(v,S)-1$
هسته موضعاً ریشه‌دار شده v نسبت به S را بصورت زیر تعریف می‌کنیم.
مسیری‌است مانند $P_{u,x}$ گذشته از رأس v که ماکزیمم $sav(u,P_{u,x})$ را دارد. این هسته را با $LRC(v,S)$ نشان می‌دهیم. البته باید به این نکته توجه داشت که یک رأس $v\notin S$ ممکن است بیش‌تر از یک هسته موضعاً ریشه‌دار شده داشته باشد. مجموعه $LRC(v,S)$ را مجموعه هسته‌های موضعاً ریشه‌دار شده ماکزیمال نسبت به c می‌نامیم، که در $O(n)$ مرتبه بدست می‌آید.

در نتیجه برای پیدا کردن یک $k-treecore(c)$ کافی است نخست مسیر P_c را پیدا کنیم و سپس مجموعه LS را نسبت به مسیر P_c بیابیم و در نهایت $k-2$ عضو از مجموعه LS با ماکزیمم فاصله ذخیره شده را به مسیر P_c اضافه کنیم.

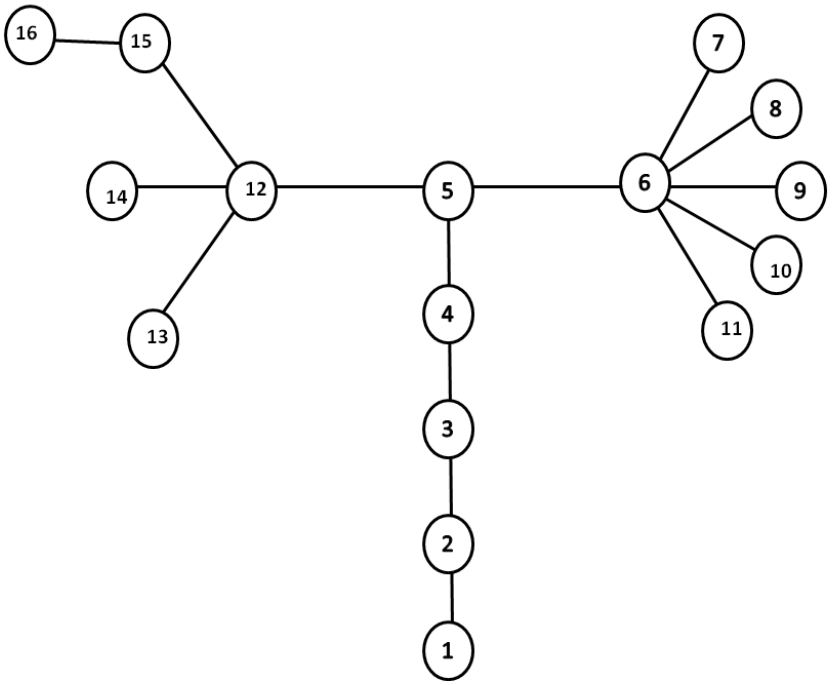
[1] *Locally Rooted Core*

لم ۱۲. برای هر KTC مانند S یک $(k+۱)-treecore$ مانند S' به‌قسمی وجود دارد که $S \subset S'$ است.

نکته ۴. برای هر KTC مانند S فرض کنید P یک هسته موضعاً ریشه‌دار شده نسبت به S باشد که ماکزیمم فاصله ذخیره شده را درمیان تمام رأس‌های v همسایه با S دارد. در این‌صورت $S \cup P$ نیز یک $(k+۱)-treecore$ است.

۱.۲.۴ الگوریتم $Beker$ برای پیدا کردن $(k,l)-treecore$ در یک درخت بی‌وزن

در این بخش الگوریتمی با مرتبه زمانی $O(n)$ را برای پیدا کردن یک $(k,l)-treecore$ در یک درخت غیر وزن‌دار را بیان می‌کنیم که قبل از آن بیان یک سری نکات و تعاریف ضروری است. اگرچه این الگوریتم بر پایه لم ۱۲ بنا شده است، اما با یک مثال نشان می‌دهیم که اگر KTC را با یک $(k,l)-treecore$ با قطر محدود به l جایگزین کنیم این لم برقرار نخواهد بود.



شکل ۱.۴: مثال یافتن $(۲,۴)-core$ در یک درخت

مثال ۲۴. درخت ۱.۴ را در نظر بگیرید و $(۲,۴) - core$ را در این درخت پیدا کنید.
تنها مسیری که مینیمم مقدار تابع هدف را دارد مسیر $P = \{۲,۳,۴,۵,۶\}$ با مقدار تابع هدف $d(P) = ۱۶$ است. با شروع از این مسیر برای پیدا کردن $(۳,۴) - core$ تنها مسیر ممکن مسیر $\{۱۲,۶,۵,۴,۳,۲\}$ با تابع هدف $d(T') = ۱۱$ است. زیر درختی که مینیمم مقدار تابع هدف را دارد، درخت $\{۱۵,۱۲,۵,۶,۷,۴,۳\}$ با مقدار $d(T') = ۱۰$ است که این زیر درخت شامل $(۲,۴) - core$ نیست.

- این مثال نشان می‌دهد که برای پیدا کردن $(k,l) - core$ استفاده از الگوریتم ۱.۲.۳ امکان‌پذیر نیست.

تعریف ۱۶.۴. نقطه میانی مسیر P با طول l، رأس(یالی) است که با حذف آن، مسیر P به دو مسیر با طول $\frac{l}{۲}(\frac{l-۱}{۲})$ تقسیم می‌شود. نقطه میانی مسیر را با c نشان می‌دهیم.
هندلر[۱] ثابت کرده است که به ازاء یک درخت دلخواه T مرکز درخت، همان رأس میانی بلندترین مسیر درخت است [؟].

قبل از ارائه الگوریتم هسته درختی با طول l و حداکثر k برگ لازم است به توضیح عملیات هرس کردن[۲] در درخت بپردازیم که در ادامه از آن بهره خواهیم گرفت.

۲.۲.۴ عملیات هرس کردن در درخت‌های بی‌وزن

ابتدا درخت T را در مرکز آن یعنی رأس c ریشه‌دار می‌کنیم و آن‌را T^c می‌نامیم. در درخت جدید مسیرهایی که از c شروع می‌شوند را به منظور بدست آوردن درختی که در آن فاصله تمام رأس‌ها تا مرکز درخت حداکثر $\frac{l}{۲}$ است قطع می‌کنیم. این درخت جدید را با $\widehat{T^c}$ نشان می‌دهیم.
در درخت $\widehat{T^c}$ توابع وزن را بصورت زیر تعریف می‌کنیم:

[۱] $Handler$ [۲] $Prune$

$$W^{'}(v) = \begin{cases} ۱ & \text{اگر } v \text{ یک برگ از } \widehat{T}^c \text{ نیست} \\ sum_c(v) & \text{اگر } v \text{ یک برگ از } \widehat{T}^c \text{ است} \end{cases} \quad (۴.۴)$$

الگوریتم $Beker$ برای پیدا کردن $(k,l)-core$

ورودی: درخت غیر وزن‌دار T
خروجی:یک $(k,l)-core$ مانند S^*

گام ۱ : قرار دهید $d^* = \infty$ و $S^* = \phi$

گام ۲ : به ازاء هر نقطه میانی ممکن مانند c در V یا E مراحل زیر را انجام دهید:

مرحله ۱) عملیات هرس کردن را روی درخت T انجام داده و درخت جدید $\widehat{T}^c$ را بدست آورید.

مرحله ۲) وزن تمام رئوس را با استفاده از رابطه‌ی ۴.۴ در درخت $\widehat{T}^c$ بدست آورید.

مرحله ۳) به ازاء هر رأس $v \neq c$ مجموع وزن تمام رئوس در درخت $\widehat{T}^c_v$ را بدست آورید.

مرحله ۴) به ازاء رأس c مقدار $d(c)$ را پیدا کنید.

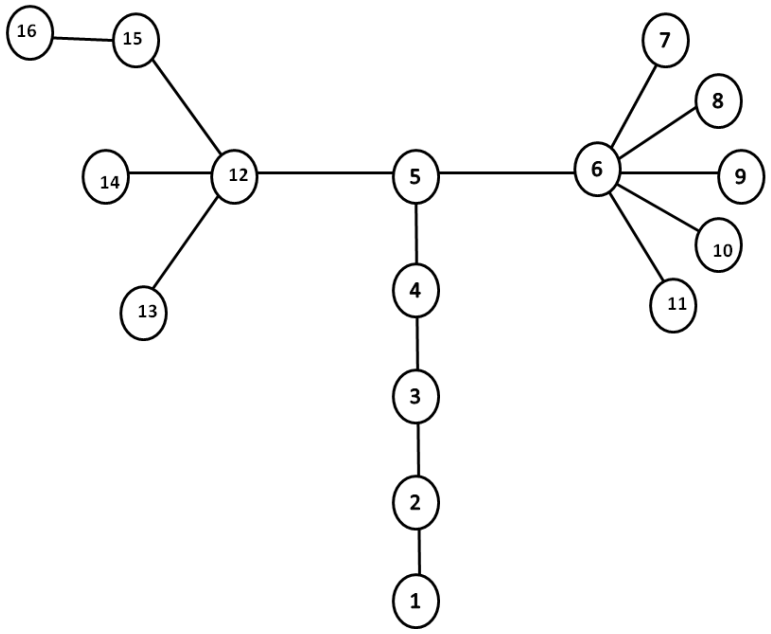
مرحله ۵) مسیر P_c را مشخص کنید.

مرحله ۶) با توجه به مسیر P_c مجموعه LS را پیدا کنید.

گام ۳ : اگر تعداد برگ‌های درخت $\widehat{T}^c$کمتر یا مساوی k است تمام عناصر در مجموعه LS را انتخاب کرده و قرار می‌دهیم $S_c = \widehat{T}^c$ با مقدار d_c.
و اگر درخت $\widehat{T}^c$ بیشتر از k برگ دارد، $k-۲$ عضو از مجموعه LS با ماکزیمم فاصله ذخیره شده را پیدا کرده و مجموعه S_c را به عنوان $KTC(c)$ بدست آمده از اجتماع P_c و $k-۲$ عضو از مجموعه LS با مقدار d_c به عنوان تابع هدف آن قرار می‌دهیم.

گام ۴ : اگر $d_c < d^*$ آن‌گاه قرار دهید: $d^* = d_c$ ، $S^* = S_c$

گام ۵ : درخت S^* را به عنوان $(k,l) - core$ معرفی کنید.



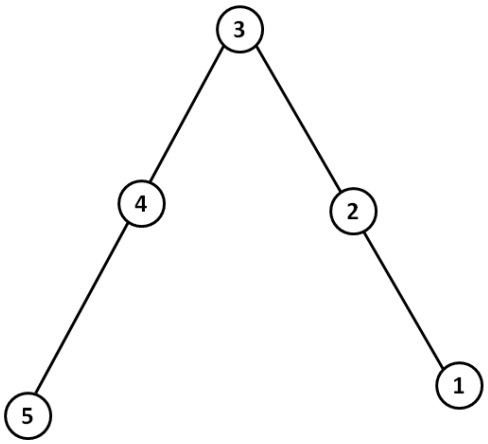
شکل ۲.۴ : مثال یافتن $(۳,۴) - core$ در یک درخت

مثال ۲۵. درخت نشان داده شده در شکل ۲.۴ را در نظر بگیرید و با استفاده از الگوریتم ۱.۲.۴ $(۳,۴) - core$ را پیدا کنید.

ابتدا تمام رأس‌های میانی ممکن درخت را با توجه به مقدار $l = ۴$ بدست می‌آوریم، به این صورت که تمام مسیرهای با طول ۴ را پیدا کرده سپس نقطه میانی این مسیر نقطه میانی درخت است. به عنوان مثال مسیر $\{v_۱, v_۲, v_۳, v_۴, v_۵\}$ را با طول ۴ در نظر می‌گیریم و نقطه میانی این مسیر رأس $v_۳$ است. برای سایر مسیرها نیز با این طریق رأس‌های میانی آن‌ها را پیدا می‌کنیم. در نتیجه رأس‌های درخت عبارتند از $v_۳, v_۴, v_۵, v_{۱۲}$ سپس برای تک تک این رأس‌ها گام‌های زیر را انجام می‌دهیم.

گام ۱: به ازاء رأس مرکزی $c = v_۳$ مراحل زیر را انجام می‌دهیم.

(۱) ابتدا عملیات هرس کردن را روی درخت انجام می‌دهیم و درخت $\widehat{T}^۳$ را بصورت شکل



شکل ۳.۴: درخت هرس شده $\widehat{T}^{۳}$

۳.۴ بدست می‌آوریم. وزن رئوس در درخت جدید در جدول ۱.۴ نشان داده شده است.

جدول ۱.۴: وزن رئوس درخت در گام ۱

v_i	$v_۱$	$v_۲$	$v_۴$	$v_۵$
w_i	۱	۱	۱	۱۲

(۲) به ازاء تمام رئوس $v \neq v_۳$ مجموع وزنی رئوس را در درخت $\widehat{T}^{۳}_v$ بدست می‌وریم. این مقادیر در جدول ۲.۴ نشان داده شده است.

جدول ۲.۴: مجموع وزنی رئوس در درخت $\widehat{T}^{۳}_v$

v_i	$v_۱$	$v_۲$	$v_۴$	$v_۵$
$\widehat{T}^{۳}_i$	۱	۲	۱۳	۱۲

(۳) مقدار $d(۳)$ که مجموع فاصله تمام رئوس درخت تا رأس $v_۳$ است، برابر با $d(۳) = ۶$ است.

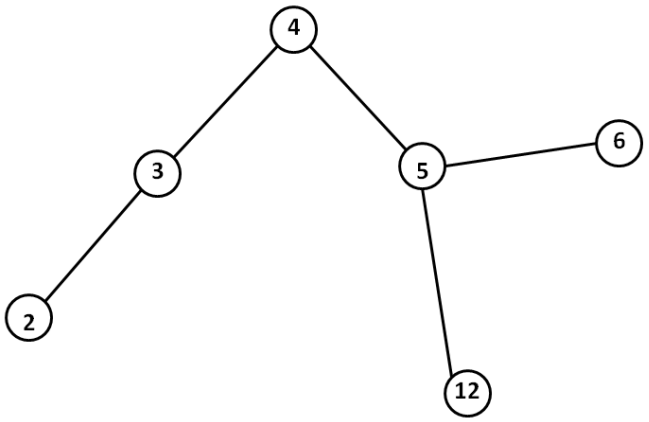
(۴) در این مرحله باید مسیر $P_۳$ را پیدا کنیم. برای این منظور ابتدا باید دو مسیر $l_۱$ و $l_۲$ را بیابیم، که به ترتیب عبارتند از مسیرهایی که از $v_۳$ شروع شده و با بیشترین فاصله ذخیره

شده و دومین بیشترین فاصله ذخیره شده. در نتیجه $P_۳ = l_۱ \cup l_۲$. در جدول ۳.۴ مسیرها و مقادیر فاصله ذخیره شده آن‌ها نشان داده شده است.

جدول ۳.۴: $sav(۳, P_{۳,i})$

v_i	$v_۱$	$v_۲$	$v_۴$	$v_۵$
$sav(۳, P_{۳,i})$	۳	۲	۱۳	۲۵

لذا با توجه به جدول فوق $l_۱ = \{۳, ۵\}$ و $l_۲ = \{۳, ۱\}$ در نتیجه $P_۳ = \{۱, ۲, ۳, ۴, ۵\}$ در نتیجه در این گام $S_c = \widehat{T}^۳$ با مقدار $d_۳ = ۲۱$



شکل ۴.۴: درخت هرس شده $\widehat{T}^۴$

گام ۲: به ازاء رأس مرکزی $c = v_۴$ مراحل زیر را انجام می‌دهیم.

(۱) ابتدا عملیات شاخه زدن را روی درخت انجام می‌دهیم و درخت $\widehat{T}^۴$ را بصورت شکل ۴.۴ بدست می‌آوریم. وزن رئوس در درخت جدید در جدول ۴.۴ نشان داده شده است.

جدول ۴.۴: وزن رئوس درخت در گام ۲

v_i	$v_۲$	$v_۳$	$v_۵$	$v_۶$	$v_{۱۲}$
w_i	۲	۱	۱	۶	۵

(۲) به ازاء تمام رئوس $v \neq v_۴$ مجموع وزنی رئوس را در درخت $\widehat{T}_v^۴$ بدست می‌وریم. این مقادیر در جدول ۵.۴ نشان داده شده است.

جدول ۵.۴: مجموع وزنی رئوس در درخت $\widehat{T}_v^۴$

v_i	$v_۲$	$v_۳$	$v_۵$	$v_۶$	$v_{۱۲}$
$\widehat{T}_i^۴$	۲	۳	۱۲	۶	۵

(۳) مقدار $d(۴)$ که مجموع فاصله تمام رئوس درخت تا رأس $v_۴$ است، برابر با $d(۴) = ۸$ است.

(۴) در این مرحله باید مسیر $P_۴$ را پیدا کنیم. برای این منظور ابتدا باید دو مسیر $l_۱$ و $l_۲$ را بیابیم، که به ترتیب عبارتند از مسیرهایی که از $v_۴$ شروع شده و با بیشترین فاصله ذخیره شده و دومین بیشترین فاصله ذخیره شده. در نتیجه $P_۴ = l_۱ \cup l_۲$. در جدول ۶.۴ مسیرها و مقادیر فاصله ذخیره شده آن‌ها نشان داده شده است.

جدول ۶.۴: $sav(۴, P_{۴,i})$

v_i	$v_۲$	$v_۳$	$v_۵$	$v_۶$	$v_{۱۲}$
$sav(۴, P_{۴,i})$	۵	۳	۱۲	۱۸	۱۷

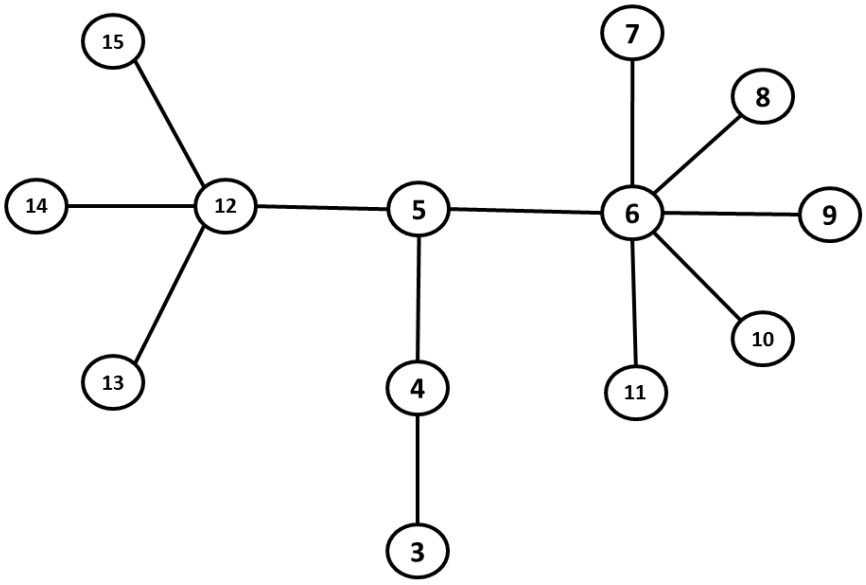
لذا با توجه به جدول فوق $l_۱ = \{۴, ۵, ۶\}$ و $l_۲ = \{۴, ۳, ۲\}$ در نتیجه $P_۴ = \{۲, ۳, ۴, ۵, ۶\}$ در نتیجه در این گام $S_c = P_۴ \cup (۵, ۱۲)$ با مقدار $d_۴ = ۱۱$

گام ۳: به ازاء رأس مرکزی $c = v_۵$ مراحل زیر را انجام می‌دهیم.

(۱) ابتدا عملیات شاخه زدن را روی درخت انجام می‌دهیم و درخت $\widehat{T}^۵$ را بصورت شکل ۵.۴ بدست می‌آوریم. وزن رئوس در درخت جدید در جدول ۷.۴ نشان داده شده است.

جدول ۷.۴: وزن رئوس درخت در گام ۳

v_i	$v_۳$	$v_۴$	$v_۶$	$v_۷$	$v_۸$	$v_۹$	$v_{۱۰}$	$v_{۱۱}$	$v_{۱۲}$	$v_{۱۳}$	$v_{۱۴}$	$v_{۱۵}$
w_i'	۳	۱	۱	۱	۱	۱	۱	۱	۱	۱	۱	۲



شکل ۵.۴: درخت هرس شده $\widehat{T}^{۵}$

(۲) به ازاء تمام رئوس $v \neq v_۵$ مجموع وزنی رئوس را در درخت $\widehat{T}^{۵}_{v}$ بدست می‌وریم. این مقادیر در جدول ۸.۴ نشان داده شده است.

جدول ۸.۴: مجموع وزنی رئوس در درخت $\widehat{T}^{۵}_{v}$

v_i	$v_۳$	$v_۴$	$v_۶$	$v_۷$	$v_۸$	$v_۹$	$v_{۱۰}$	$v_{۱۱}$	$v_{۱۲}$	$v_{۱۳}$	$v_{۱۴}$	$v_{۱۵}$
$\widehat{T}^{۵}_{i}$	۳	۴	۶	۱	۱	۱	۱	۱	۵	۱	۱	۲

(۳) مقدار $d(۵)$ که مجموع فاصله تمام رئوس درخت تا رأس $v_۵$ است، برابر با $d(۵) = ۸$

(۴) در این مرحله باید مسیر $P_۵$ را پیدا کنیم. برای این منظور ابتدا باید دو مسیر $l_۱$ و $l_۲$ را بیابیم، که به ترتیب عبارتند از مسیرهایی که از $v_۵$ شروع شده و با بیشترین فاصله ذخیره شده و دومین بیشترین فاصله ذخیره شده. در نتیجه $P_۵ = l_۱ \cup l_۲$. در جدول ۹.۴ مسیرها و مقادیر فاصله ذخیره شده آن‌ها نشان داده شده است.

جدول ۹.۴: $sav(۵, P_{۵,i})$

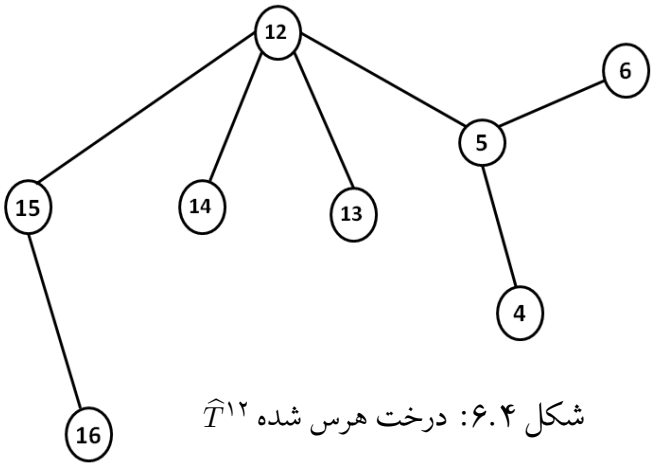
v_i	$v_۳$	$v_۴$	$v_۶$	$v_۷$	$v_۸$	$v_۹$	$v_{۱۰}$	$v_{۱۱}$	$v_{۱۲}$	$v_{۱۳}$	$v_{۱۴}$	$v_{۱۵}$
$sav(۵, P_{۵,i})$	۷	۴	۶	۵	۵	۵	۵	۵	۵	۶	۶	۸

لذا با توجه به جدول فوق $l_1 = \{۴, ۵, ۶\}$ و $l_2 = \{۴, ۳, ۲\}$ در نتیجه $P_۵ = \{۲, ۳, ۴, ۵, ۶\}$

در نتیجه در این گام $S_c = P_۵ \cup (۵, ۱۲)$ با مقدار $d_۵ = ۱۱$

گام ۴: به ازاء رأس مرکزی $c = v_{۱۲}$ مراحل زیر را انجام می‌دهیم.

(۱) ابتدا عملیات شاخه زدن را روی درخت انجام می‌دهیم و درخت $\widehat{T}^{۱۲}$ را بصورت شکل ۶.۴ بدست می‌آوریم. وزن رئوس در درخت جدید در جدول ۱۰.۴ نشان داده شده است.



شکل ۶.۴: درخت هرس شده $\widehat{T}^{۱۲}$

جدول ۱۰.۴: وزن رئوس درخت در گام ۴

v_i	$v_۴$	$v_۵$	$v_۶$	$v_{۱۳}$	$v_{۱۴}$	$v_{۱۵}$	$v_{۱۶}$
w_i'	۴	۱	۶	۱	۱	۱	۱

(۲) به ازاء تمام رئوس $v \neq v_{۱۲}$ مجموع وزنی رئوس را در درخت $\widehat{T}_v^{۱۲}$ بدست می‌وریم. این مقادیر در جدول ۱۱.۴ نشان داده شده است.

جدول ۱۱.۴: مجموع وزنی رئوس در درخت $\widehat{T}_v^{۱۲}$

v_i	$v_۴$	$v_۵$	$v_۶$	$v_{۱۳}$	$v_{۱۴}$	$v_{۱۵}$	$v_{۱۶}$
$\widehat{T}_i^{۱۲}$	۴	۱۱	۶	۱	۱	۲	۱

(۳) مقدار $d(۱۲)$ که مجموع فاصله تمام رئوس درخت تا رأس $v_{۱۲}$ است برابر $d(۱۲) = ۱۰$

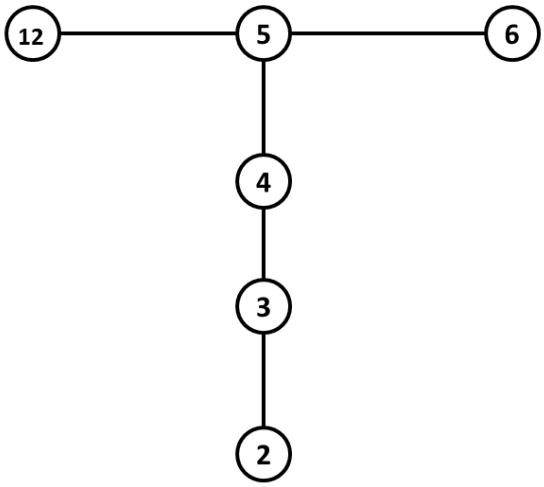
(۴) در این مرحله باید مسیر $P_{۱۲}$ را پیدا کنیم. برای این منظور ابتدا باید دو مسیر l_1 و l_2 را

بیابیم، که به ترتیب عبارتند از مسیرهایی که از $v_{۱۲}$ شروع شده و با بیشترین فاصله ذخیره شده و دومین بیشترین فاصله ذخیره شده. در نتیجه $P_{۱۲} = l_۱ \cup l_۲$ در جدول ۱۲.۴ مسیرها و مقادیر فاصله ذخیره شده آن‌ها نشان داده شده است.

جدول ۱۲.۴ : $sav(۱۲, P_{۱۲,i})$

v_i	$v_۴$	$v_۵$	$v_۶$	$v_{۱۳}$	$v_{۱۴}$	$v_{۱۵}$	$v_{۱۶}$
$sav(۱۲, P_{۱۲,i})$	۱۵	۱۱	۱۷	۱	۱	۲	۳

لذا با توجه به جدول فوق $l_۱ = \{۱۲, ۵, ۶\}$ و $l_۲ = \{۱۲, ۱۵, ۱۶\}$ در نتیجه $P_{۱۲} = \{۱۶, ۱۵, ۱۲, ۵, ۶\}$ در نتیجه در این گام $S_c = P_{۱۲} \cup (۵, ۴)$ با مقدار $d_{۱۲} = ۱۳$



شکل ۷.۴: $(۳, ۴) - core$

در نتیجه $(۳, ۴) - core$ درخت فوق بصورت درخت ۷.۴ می‌باشد با $d^* = ۱۱$.

۳.۲.۴ الگوریتم یافتن $(k, l) - core$ در یک درخت وزن‌دار

در این بخش به معرفی الگوریتمی با مرتبه‌ی زمانی $O(n^۳)$ برای پیدا کردن $(k, l) - core$ در یک درخت وزن‌دار می پردازیم. درخت وزن‌دار به درختی گفته می‌شود که در آن هریک از رأس‌ها وزن متفاوت و هریک از یال‌ها طول متفاوتی نسبت به دیگری دارند. چون در

این‌حالت l متناظر با تعداد یال‌های یک مسیر نیست در نتیجه نمی‌توانیم از استراتژی بخش قبل برای پیدا کردن همه رأس‌های میانی مسیر قطری استفاده کنیم. در حقیقت در این‌حالت نقطه میانی یک مسیر،یک رأس یا یک یال نیست بلکه ممکن است نقطه‌ی میانی یک یال باشد. قبل از ارائه گام‌های الگوریتم، برخی نکات و تعاریف مورد نیاز این الگوریتم را بیان می‌کنیم.

نکته ۵. به ازاء درخت $T = (V, E)$ و رأس دلخواه $v \in V$ دو حالت زیر را داریم:

(۱) $(k, l) - core$ شامل رأس v است.

(۲) $(k, l) - core$ کاملاً در یکی از زیر درخت‌های بدست‌آمده در اثر حذف v از T قرار دارد. با حذف رأس v و یال‌های وابسته به آن، درخت T به دو یا چند زیر درخت تفکیک می‌شود.

تعریف ۱۷.۴. رأس مرکزی[۱] درخت وزن‌دار T، رأسی است که ماکزیمم تعداد رئوس زیردرخت‌های بدست آمده در اثر حذف آن می‌نیمم است [۱].

مثال ۲۶.

تعریف ۱۸.۴. درخت T^v را در نظر بگیرید. به ازاء رأس $u \neq v$ و $u \in T^v \backslash \{v\}$ طول مسیر $P_{v,u}$ را بصورت زیر تعریف می‌کنیم:

$$L(P_{v,u}) = L(P_{v,f(u)}) + a(f(u), u); L(P_{v,v}) = ۰ \qquad (۵.۴)$$

تعریف ۱۹.۴. به ازاء مسیر P و رأس دلخواه v ،فرض کنید B مجموعه تمام فرزندان v باشد. در این‌صورت T_b^v زیر درخت ریشه دار شده در $b \in B$ و $T_{\bar{b}}^v$ را زیر درخت ریشه دار شده در $\bar{b} \in B$ می‌نامیم که روی مسیر P قرار دارد.

[۱] $Center\ Vertex$

۴.۲.۴ عملیات هرس کردن در درخت‌های وزن‌دار

به ازاء درخت T^v عملیات هرس کردن که را شامل ۲ مرحله است، بصورت زیر تعریف می‌کنیم:

گام ۱ : به ازاء مسیر $P \in T_b^v; b \neq \bar{b}$ اگر $L(P) \leq min[l - L(P), L(P)]$ مسیر P را از درخت حذف کنید.

گام ۲ : به ازاء مسیر $P = P_{vu}$ که $P \in T_{\bar{b}}^v; b = \bar{b}$ مسیرهایی مانند P_{xw} که $x \in P, x \neq v$ و $w \in T_{\bar{b}}^v \setminus P$ اگر $L(P_{xw}) > min[L(P) - L(P_{vx}); l - L(P_{xu})]$ مسیر P_{xw} را از درخت حذف کنید.

در درخت جدید $\widehat{T}_v$ وزن جدید رأس‌ها بصورت زیر خواهد بود:

$$w^{'}(u) = \begin{cases} w(u) & \text{اگر } u \text{ یک برگ از } \widehat{T}^v \text{ نیست} \\ sum_v(u) & \text{اگر } u \text{ یک برگ از } \widehat{T}^v \text{ است} \end{cases} \qquad (۶.۴)$$

در درخت جدید طول طولانی‌ترین مسیر حداکثر l می‌باشد و مسیر $P = P_{vu}$ طولانی‌ترین مسیر از v تا یک برگ است. در درخت $\widehat{T}^v$ فاصله ذخیره شده تمام مسیرهای شروع شده از v تا یک رأس $u \neq v$ را محاسبه می‌کنیم. اکنون اگر درجه رأس v حداقل ۲ باشد، مسیری با بیشترین فاصله ذخیره شده در $\widehat{T}_b^v; b \neq \bar{b}$ را با $P^{'}$نشان داده و این مسیر را به مسیر P اضافه می‌کنیم و مسیر جدید را با $P^{''}$ نشان می‌دهیم. در نتیجه طول مسیر $P'' = P \cup P^{'}$ حداکثر l است.

الگوریتم $(k,l) - core$

ورودی: درخت وزن‌دار T

خروجی:یک $(k,l) - core$ مانند S^* با $DISTSUM(S^*) = d^*$

گام ۱ : قرار دهید $d^* = \infty$ و $S^* = \phi$ و عملیات $SUBTREE(T)$ را انجام دهید.

گام ۲ : عملیات $SUBTREE(T')$:

ورودی: درخت وزن‌دار $T' = (V', E')$ که مقدار $DISTSUM$ آن d^* است.

خروجی: اگر $DISTSUM(T') \leq d^*$ آن‌گاه زیردرخت T' را جایگزین S^* کنید.

مرحله (۱) اگر T' تنها شامل یک رأس است آن‌گاه قرار دهید $S' = T'$ و مقدار $DISTSUM$ آن $d(S')$ باشد.

در غیر این‌صورت رأس مرکزی v از T' را پیدا کنید و قرار دهید:

$$S' := BEST - TREE(T', v)$$

مرحله (۲) اگر $d(S') < d^*$ آن‌گاه قرار دهید $d^* := d(S')$ و $S^* := S'$

مرحله (۳) برای هر زیر درخت T^i بدست آمده در اثر حذف v از T' عملیات $SUBTREE(T^i)$ را انجام دهید.

گام ۳ : عملیات $BEST - TREE(T', v)$**:**

ورودی: یک زیر درخت $T' = (V', E')$ با n' رأس که در رأس مرکزی v ریشه‌دار شده است.

خروجی: بهترین زیر درخت S' شامل رأس v با حداکثر k برگ و قطر حداکثر l که مقدار $DISTSUM$ آن $d(S')$ می‌باشد.

مرحله (۱) مسیرهایی مانند P با شروع از v و با طول حداکثر l را پیدا کنید.

مرحله (۲) برای هر مسیر P اقدامات زیر را انجام دهید:

(a) عملیات شاخه زدن T'^v را انجام داده و درخت $\widehat{T}'^v$را بدست آورید.

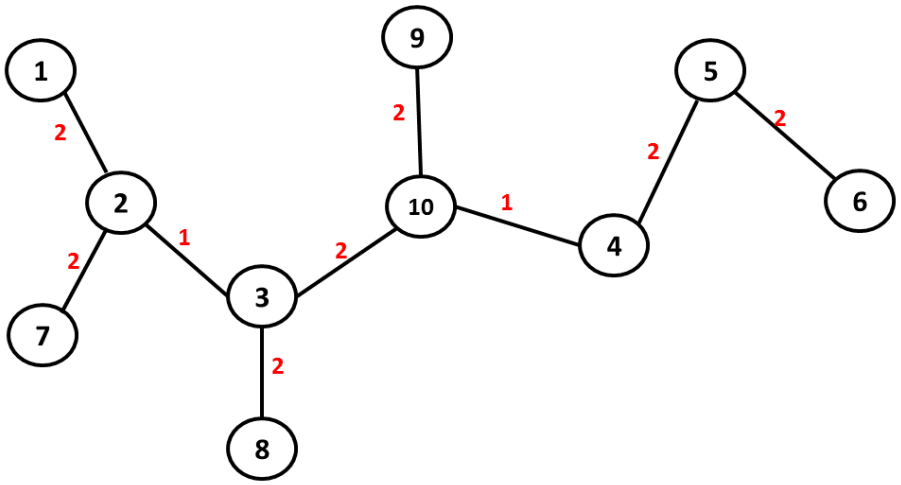
(b) در درخت $\widehat{T'}^v$ فاصله ذخیره شده تمام مسیرهای شروع شده از v تا یک رأس $u \neq v$ را بدست آورید.

(b) اگر $deg(v) \geq$ ۲ آن‌گاه مسیر P' را پیدا کنید و قرار دهید: $P'' = P \cup P'$

(c) در غیر این‌صورت قرار دهید $P'' = P$ با توجه به مسیر P'' مجموعه LS را پیدا کنید.

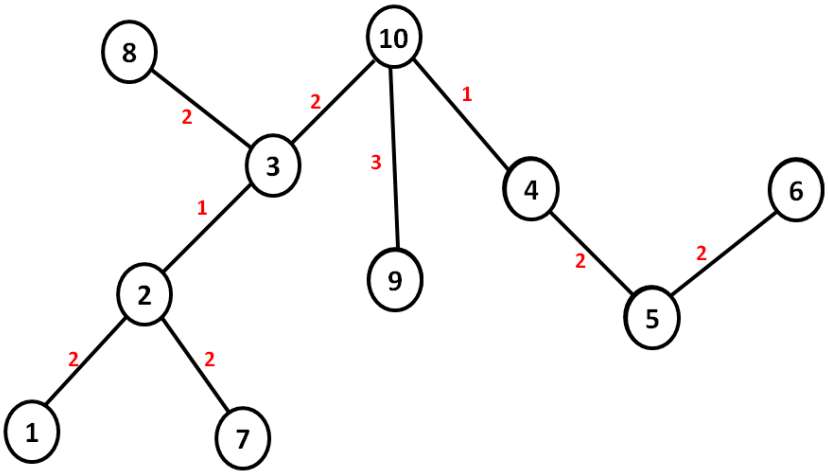
مرحله (۳) اگر درخت $\widehat{T'}^v$ بیشتر از k برگ دارد، $k -$ ۲ عضو از مجموعه LS با ماکزیمم فاصله ذخیره شده را برای اضافه کردن به P'' انتخاب کنید. $k - treecore(c)$ بدست آمده از اجتماع P_c و $k -$ ۲ عضو از مجموعه LS با مقدار d_c به عنوان $DISTSUM$ آن.و فرض کنید S' بهترین زیر درخت شامل v باشد که مقدار $DISTSUM$ آن $d(S')$ است.

مرحله (۴) اگر تعداد برگ‌های درخت $\widehat{T'}^v$ کمتر یا مساوی k است آن‌گاه تمام عناصر در مجموعه LS را انتخاب کنید و قرار دهید $S' = \widehat{T'}^v$ که مقدار $DISTSUM$ آن $d(S')$ است.



شکل ۸.۴: مثال یافتن $core - (۴,۵)$ در یک درخت وزن‌دار

مثال ۲۷. در درخت شکل ۸.۴ هسته $core - (۴,۵)$ را پیدا کنید. وزن رئوس در جدول ۱۳.۴ داده شده است.



شکل ۹.۴: درخت ریشه‌دار شده در رأس $v_{۱۰}$

جدول ۱۳.۴: وزن رئوس درخت مثال ۲۷

v_i	$v_۱$	$v_۲$	$v_۳$	$v_۴$	$v_۵$	$v_۶$	$v_۷$	$v_۸$	$v_۹$	$v_{۱۰}$
$W(v_i)$	۲	۱	۱	۲	۱	۳	۱	۲	۱	۳

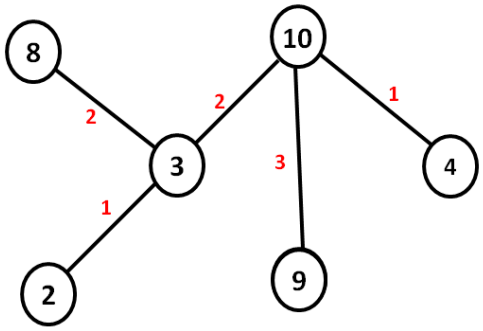
ابتدا قرار می‌دهیم $d^* = \infty$ و عملیات $SUBTREE(T)$ را انجام می‌دهیم.

در طی عملیات $SUBTREE(T)$ ابتدا باید رأس مرکزی درخت را پیدا کرده که رأس $v_{۱۰}$ است. سپس عملیات $BEST - TREE(T, v_{۱۰})$ را انجام می‌دهیم. در طی این عملیات ابتدا درخت T را در رأس $v_{۱۰}$ بصورت درخت ۹.۴ ریشه‌دار می‌کنیم. سپس مسیرهایی با طول حداکثر ۵ و شروع از $v_{۱۰}$ را پیدا می‌کنیم، این مسیرها عبارتند از:

$P_۱ = (۱۰, ۴)$ و $P_۲ = (۱۰, ۴, ۵)$ و $P_۳ = (۱۰, ۴, ۵, ۶)$ و $P_۴ = (۱۰, ۹)$

$P_۵ = (۱۰, ۳)$ و $P_۶ = (۱۰, ۳, ۸)$ و $P_۷ = (۱۰, ۳, ۲)$ و $P_۸ = (۱۰, ۳, ۲, ۱)$

گام۱: برای مسیر $P_۱ = (۱۰, ۴)$ عملیات شاخه زدن را روی درخت T انجام می‌دهیم. سپس فاصله ذخیره شده مسیرهای شروع از v تا یک رأس $u \neq ۱۰$ را پیدا می‌کنیم. این



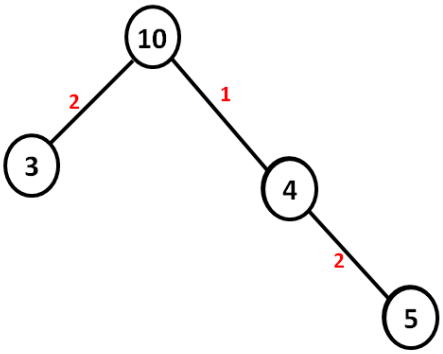
شکل ۱۰.۴: S'_1

مقادیر در جدول ۱۴.۴ زیر نشان داده شده است.

جدول ۱۴.۴: مقادیر $sav(10, P_{10,i})$ برای رئوس درخت

v_i	v_2	v_3	v_8	v_9
$sav(10, P_{10,i})$	۳۴	۲۶	۲۶	۳

در نتیجه $P' = (10, 3, 2)$ و $P'' = (2, 3, 10, 4)$ و مجموعه LS را پیدا کرده و در نتیجه درخت S'_1 با $d(S'_1) = 21$ بصورت درخت ۱۰.۴ بدست می‌آید.

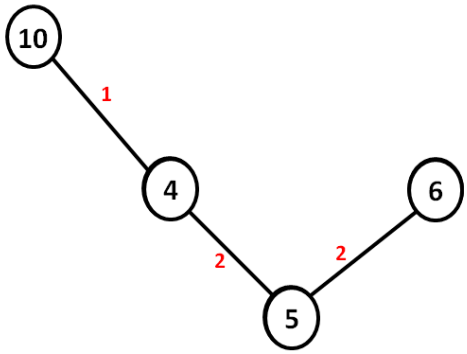


شکل ۱۱.۴: S'_2

گام۲: برای مسیر $P_2 = (10, 4, 5)$ عملیات شاخه زدن روی درخت T را انجام می‌دهیم.

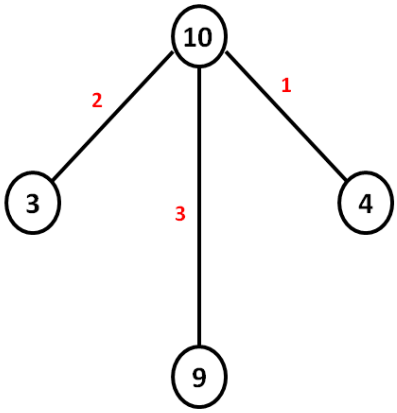
فاصله ذخیره شده مسیرهای شروع از v_{10} تا یک رأس $u \neq 10$ را پیدا می‌کنیم

$sav(10, P_{10,3}) = 26$ در نتیجه $P' = (10, 3)$ و $P'' = (3, 10, 4)$ و مجموعه LS را پیدا کرده و در نتیجه درخت S'_2 با $d(S'_2) = 24$ بصورت ۱۱.۴ بدست می‌آید.



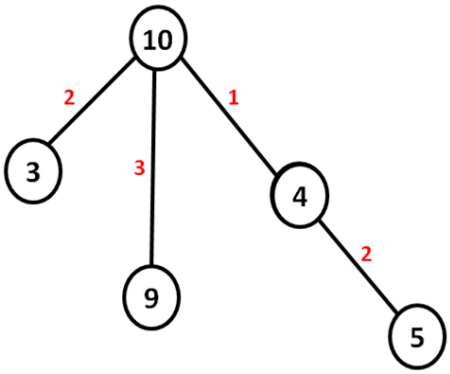
شکل ۱۲.۴: $S'_۳$

گام۳: برای مسیر $P_۳ = (۱۰, ۴, ۵, ۶)$ عملیات شاخه زدن را روی درخت T انجام می‌دهیم. در نتیجه درخت $S'_۳$ با $d(S'_۳) = ۳۲$ بصورت درخت ۱۲.۴ بدست می‌آید.



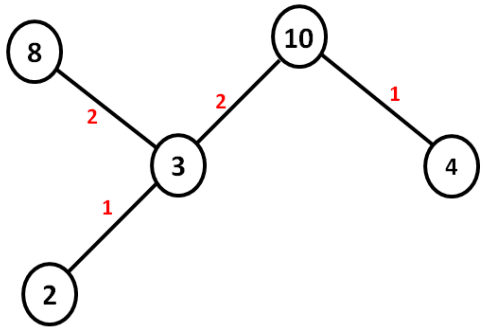
شکل ۱۳.۴: $S'_۴$

گام۴: برای مسیر $P_۴ = (۱۰, ۹)$ عملیات شاخه زدن را روی درخت T انجام می‌دهیم. در نتیجه درخت $S'_۴$ با $d(S'_۴) = ۳۳$ بصورت درخت ۱۳.۴ بدست می‌آید.



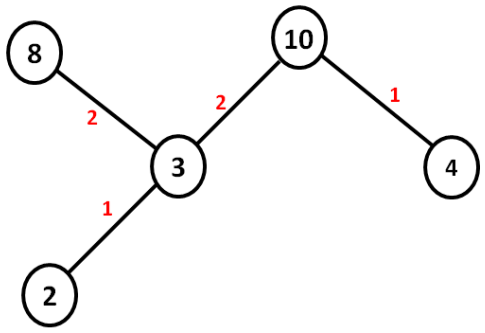
شکل ۱۴.۴: $S'_۵$

گام۵: برای مسیر $P_۵ = (۱۰, ۳)$ عملیات شاخه زدن را روی درخت T انجام می‌دهیم. در نتیجه درخت $S'_۵$ با $d(S'_۵) = ۲۱$ بصورت درخت ۱۴.۴ بدست می‌آید.



شکل ۱۵.۴: $S'_۶$

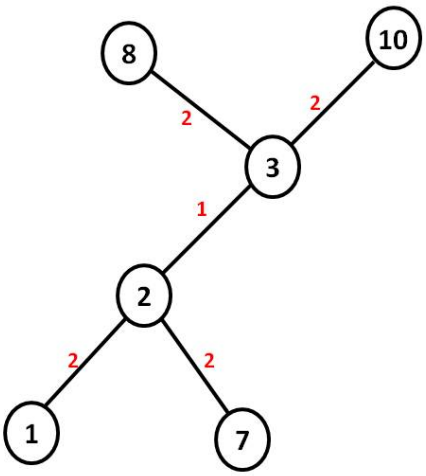
گام۶:برای مسیر $P_۶ = (۱۰, ۳, ۸)$ عملیات شاخه زدن را روی درخت T انجام می‌دهیم. در نتیجه درخت $S'_۶$ با $d(S'_۶) = ۲۴$ بصورت درخت ۱۵.۴ بدست می‌آید.



شکل ۱۶.۴ : $S'_{۷}$

گام۷:برای مسیر $P_{۷} = (۱۰, ۳, ۲)$ عملیات شاخه زدن را روی درخت T انجام می‌دهیم.

در نتیجه درخت $S'_{۷}$ با $d(S'_{۷}) = ۲۴$ بصورت درخت ۱۶.۴ بدست می‌آید.



شکل ۱۷.۴ : $S'_{۸}$

گام۸:برای مسیر $P_{۸} = (۱۰, ۴, ۵, ۶)$ عملیات شاخه زدن را روی درخت T انجام می‌دهیم.

در نتیجه درخت $S'_{۸}$ با $d(S'_{۸}) = ۲۰$ بصورت درخت ۱۷.۴ بدست می‌آید.

- در نتیجه با بررسی گام‌های ۱ تا ۸ هسته درختی عبارت است از $S^* = S'_{۸}$ با مقدار تابع هدف $d^* = ۲۰$.

نتیجه‌گیری و پیشنهادات

در این پایان‌نامه ۵ نوع مسئله مکان‌یابی در درخت‌هایی با طول یال‌ها و وزن‌های متفاوت را بررسی کردیم. در فصل دوم بعد از بیان مقدمات و تعاریف لازم، مسائل مسیر میانه، مسیر مرکزی و مسیر میانه مرکزی را بیان کردیم و برای هر کدام الگوریتم‌هایی ارائه دادیم. مسیر میانه که عبارت است از پیدا کردن مسیری که مجموع فاصله وزنی رئوس شبکه تا آن مینیمم مقدار ممکن باشد در پیدا کردن مکان بهینه مسر حرکت قطار شهری و شبکه‌های حمل و نقل بسیار کاربرد دارد. بعد از آن مسئله مسیر مرکزی که عبارت است از پیدا کردن مسیری که بیشترین فاصله مشتری‌ها تا آن کمترین مقدار ممکن باشد را معرفی کردیم و با استفاده از الگوریتم مینیکا جواب بهینه آن را پیدا کردیم. سپس با ترکیب این دو مسئله، مسیر میانه مرکزی را معرفی کردیم. در این مسئله دو معیار مینیمم مجموع فاصله و مینیمم ماکزیمم فاصله را در بررسی کردیم. کاربرد این مسئله در تعیین محل مسیر راه‌های هوایی و مسیر حرکت لوله‌های آب و فاضلاب است.

در فصل سوم مسئله هسته درختی با k برگ که در واقع تعمیمی از مسئله هسته است و عبارت است از پیدا کردن زیر درختی با دقیقاً k برگ به‌گونه‌ای که مجموع فاصله وزنی رئوس درخت تا آن مینیمم مقدار ممکن است را بررسی کردیم و برای این مسئله نیز الگوریتم‌هایی برای حالت‌های که k ثابت یا متغیر است ارائه دادیم و با استفاده از آن‌ها جواب بهینه را پیدا کردیم.

در فصل چهارم هسته درختی با k برگ و طول مقید l را که همان مسئله هسته درختی است با این تفاوت که در این مسئله قطر درخت مقید است را معرفی کردیم و با یک مثال نشان

دادیم که الگوریتم هسته درختی با k برگ برای این مسئله کاربرد ندارد. مسئله هسته درختی در شبکه‌های ارتباطی با سرعت بالا مانند شبکه‌های اینترنتی بسیار پر کاربرد است.

از اقدامات بعدی که می‌توان در این خصوص انجام داد حالتی است که طول یال‌ها و وزن رئوس، متغیرهای احتمالی یا فازی باشند. همچنین می‌توان به‌جای پیدا کردن یک هسته در درخت دو هسته متفاوت را پیدا کرد. همچنین می‌توانیم مسائل فوق را برای حالتی که وزن رئوس مقادیر منفی هستند نیز بررسی کنیم که در این حالت نقاط مورد نظر نقاط مهلکی هستند مانند نیروگاه‌های هسته‌ای و پالایشگاه‌های نفت و گاز.

واژه‌نامه فارسی به انگلیسی

مکان‌یابی ..... *Location*

شبکه ..... *Gragh*

رأس ..... *Vertex*

یال ..... *Edge*

وزن ..... *Weight*

طول ..... *Lenght*

مسیر ..... *Path*

فاصله ..... *Distance*

دور ..... *Cycle*

همبند ..... *Connected*

درخت ..... *Tree*

برگ ..... *Leaf*

شاخه ..... *Branch*

درجه ..... *Degree*

همسایه ..... *Adjacent*

قطر ..... *Diameter*

مسیر مرکزی ..... *Path Center*

مرکز مطلق ..... *Absolut Center*

مسیر میانی ..... *Path Median*

مسأله می‌نیمم مجموع ..... *Minisum Problem*

رأس بحرانی ..... *Critical Vertex*

مسأله حداقلِ حداکثر ..... *Minimax Problem*

میانه ..... *Median*

مسیر میانه مرکزی ..... *Path Medi Center*

ریشه ..... *Roote*

درخت ریشه‌دار شده ..... *Rooted Tree*

فرزند ..... *Child*

والد ..... *Parent*

برش درختی ..... *Tree Cut*

بهنگام کردن ..... *Update*

لیست نقطه بعدی ..... *End Point List*

لیست رئوس مجاور ..... *Adjacent Point List*

همسایه داخلی ..... $In - Neighbor$

مجموعه بیرونی ..... *Outset*

شاخه داخلی ..... $In - Branch$

پیمایش پیشرو ..... *Forward Pass*

پیمایش پسرو ..... *Backward Pass*

هسته درختی ..... $Tree - Core$

هسته موضعاً ریشه‌دار شده ..... *Locally Rooted Tree*

هرس کردن ..... *Prune*

پیوست

مراجع

[1] R.I Beker, Induktive algorithm on finite tree, Quaest Math 13 (1990), 165-181.

[2] S.L.Hakimi, E.F. Schmeichel,and M.Labbe,On locating path or tree shaped facilities on network, Networks 23 (1993) , 543-555.

[3] G.Y. Handler, Minimax locatiopn of a facility in an undi rected tree graph, Trans Sci 7 (1973), 287-293.

[4] S.P eng, A.B. Stephens, and Y.Yesha, Algorithms for a core and k-tree core of a tree, J Alg 15 (1993), 143-159.

[5] A.Shioura and T.Uno,A linear time algorithm for finding a k-tree core, J Alg 23 (1997), 281-290.

[6] P.J. Slater,Locating central paths in a network, Trans Sci 16(1982), 1-18.

[7] E.Minika and N.H.Patel, On finding the core of a tree with a specified length, J Alg 4(1983), 345-352.

[8] C,A. Morgan and P.J. Slater, A linear algorithm for a core of a tree, J Alg 1 (1980), 247-258.

[9] S.Peng and W.Lo, Efficient algorithm for finding a core of a tree with a specified length, J Alg 20(1996), 445-458.

[10] R.I. Beker and Y.P erl, finding the two-core of a tree, Discare Appl. Math. 11, No. 2 (1985), 103-113.

[11] Gomory R., “Outline of anAlgorithm for Integer Solutions to Linear Programs.” Bulletin of the American Mathematical Society, 64, 1958, 275–278.

[12] Heipcke S.,“ Applications of Optimization with Xpress-MP”, Dash Optimization, Blisworth, U.K., 2002.

[13] Jeroslow R.,“Representability in mixed integer programming I: Characterization results,” Discrete Applied Mathemetics 17(1987)223-243.

[14] Lovas L. and Schrijver A., “Cones of matrices and set-functions and 0-1 optimization ,” SIM Journal on Optimization 1(2)(1991) 166-190.

[15] Meyer R.,“Integer and mixed-integer programming models:general properties,” Journal of Optimization Theory and Applications 16(1975)191-206.

[16] Saad Y.,“Iterative methods for sparse linear systems,”Yousef Saad(2000).

[17] Sherali H., Adams W.,“ Ahierarchy of relaxations and convex hull representations for mixed integer zero-one programming problems,” Technical Report, Virginia Tech(Blacksburg, VA, 1989).

[18] Sherali H., Adams W., “A hierarchy of relaxations between the continuos and convex hull representations for zero-one programming problems,” SIM Journal on Discrete Mathematics 3(3)(1990) 411-430.

[19] Williams H.p.,“An alternative explanation of disjunctive formulations,” European Journal of Operations Research 72(1994)200-203.

Hakim Sabzevari University An Outline of MSc. Thesis		
Surname: Abareshi	**Name:** Mahnaz	**Student no:** 9023133021
Supervisor: Dr. Mehdi Zaferanieh	**Advisor:** Dr.Mahmood.Amin toosi	
Faculty: Mathematics and Computer Science	**Department:** Applied of Mathematics	
Program: Applied Mathematics	**Field of study:** Operations Research	
Title of thesis: To obtain the optimal paths on a tree network		
Key words: Location, Path Center, Path Median, Path Medi Center , k-Tree Core. (k,l)-tree Core		
Abstract:		

Hakim Sabzsvari University
Faculty of Mathematics and Computer Science

Thesis Submitted in partial Fulfillment of the
Requirements for the Degree of master of Science
(M.Sc.)
in Applied Mathematics,(Operations Research)

Title

To obtain the optimal paths on a tree network

Supervisor
Dr.Mehdi.Zaferanieh

Advisor
Dr.Mahmood.Amin toosi

Author
Mahnaz.Abareshi

2013