

درخت پوشای (Spanning Tree)

اگر گراف همبند بوده ولی دور نداشته باشد، آن گراف، یک درخت است.

درخت و ماتریس درخت در رشته‌های مختلف مانند شیمی مهندسی برق و علم محاسبه کاربرد دارد. برای مثال:

✓ کیرشهف در سال ۱۸۴۷ میلادی هنگام حل دستگاههای معادلات خطی مربوط به شبکه‌های الکتریکی درخت‌ها را کشف و نظریه درخت‌ها را بارور کرد.

✓ کیلی در سال ۱۸۵۷ میلادی درخت‌ها را در ارتباط با شمارش ایزومرهای مختلف هیدروکربن‌ها کشف کرد.

در رشته ریاضیات و در زیرشاخه نظریه گراف، درخت پوشای درختی است که مجموعه‌ای از یال‌ها را شامل می‌شود در حالی که تمام رئوس را پوشش می‌دهد.

در واقع تمام رئوس گراف در درخت پوشای وجود دارند به شرطی که هیچ دوری ایجاد نشود و درخت همبند نیز باشد.

درخت پوشای گراف همبند G را می‌توان این‌گونه نیز تعریف کرد:

مجموعه‌ای حدّاًکثری از یال‌های G که هیچ دوری در آن وجود ندارد؛ یا مجموعه‌ای حدّاقلی از یال‌های G که همه رئوس را به یکدیگر متصل

می‌کند.

درخت پوشای زیرمجموعه‌ای از یک گراف است که همه رئوس آن با کمترین مقدار یال‌های ممکن پوشش یافته است.

از این رو یک درخت پوشای دور ندارد و هیچ رأس ناهمبندی در آن دیده نمی‌شود.

همه درخت‌های پوشای یک گراف، تعداد یکسانی از یال‌ها و رئوس را دارند.

یک درخت پوشای n راسی، $1-n$ یال دارد.

با حذف یک یال از درخت پوشای یک گراف غیر همبند تبدیل می‌شود؛ یعنی درخت پوشای دارای کمینه اتصال‌های ممکن است.

افزودن یک یال به درخت پوشای موجب ایجاد یک مدار یا طوقه می‌شود؛ یعنی درخت پوشای در حالت بیشینه غیر دوری (Maximally Acyclic) است.

درخت پوشای یکسان

درخت پوشایی که به طور تصادفی از بین همه درخت‌های پوشایی با احتمال برابر انتخاب شود را درخت پوشای یکسان می‌نامند.

این مدل به طور گسترده در احتمالات و ریاضی فیزیک تحقیق و بررسی می‌شود.

دور اساسی

اضافه نمودن حتّی یک یال به درخت پوشای باعث ایجاد دور می‌شود، چنین دوری را یک دور اساسی نامیده‌اند.

برای هر یالی، دور اساسی مجزّای وجود دارد. بنابراین یک ارتباط یک به یک بین دورهای اساسی و یال‌هایی که در درخت پوشای شرکت نمی‌کنند وجود دارد.

برای درخت همبندی با ۷ تا رأس، هر درخت پوشایی ۱-V یال خواهد داشت، در نتیجه هر گرافی با E یال، E-V-1 دور اساسی خواهد داشت.

برای هر درخت پوشای داده شده این دورها، پایه‌ای برای فضای دوری تشکیل می‌دهند. با حذف هر یک از یال‌ها در درخت پوشای، رئوس به دو دستهٔ مجزا تقسیم می‌شوند.

مجموعه برش اساسی این‌گونه تعریف شده‌است: مجموعه یال‌هایی که باید از گراف جدا شوند تا به تقسیم مشابهی برسیم.

به طور دقیق ۱-V مجموعه برش اساسی وجود دارد؛ هر یک برای یکی از یال‌های درخت پوشای. یال‌های ایجاد کننده دور که در درخت پوشای نیستند، فقط می‌توانند در مجموعه برش‌هایی از دیگر رئوس در دور حاضر شوند؛ و بر عکس.

جنگل پوشان (Spanning Forest)

جنگل پوشان، نوعی زیر درخت می‌باشد که مفهوم درخت پوشان را تعمیم داده است.

دو تعریف متقابل برای آن وجود دارد: جنگل پوشان زیر درختی است که شامل درخت پوشان در هر یک از بخش‌های همبند آن می‌باشد. به عبارت دیگر، زیر درخت بیشینه‌ای است که دور ندارد. این تعریف بیشتر در علوم کامپیوتر و بهینه سازی کاربرد دارد. تعریف دیگر که در نظریه گراف‌ها کاربرد دارد، این‌گونه است: جنگل پوشان هر زیر درختی است که هم جنگل است (دور ندارد) و هم پوشان (شامل تمام رئوس می‌شود).

شمارش درخت های پوشای

برای هر گراف G مقدار ثابت $t(G)$

(تعداد درخت های پوشای گراف G) از نظریه

ماتریس-درخت کیوشیف قابل محاسبه می باشد.

یک گراف ناهمبند؛ هیچ درخت پوشایی ندارد،

زیرا امکان پوشش همه رئوس آن میسر نیست.

هر گراف کاملاً همبند و غیر جهت دار دست کم یک

درخت پوشای دارد.

(مثلاً اگر یک گراف، خود، درخت باشد.)

تعداد درخت های پوشای یک گراف دوری n راسی،

برابر $n!$ است.

یک گراف کامل غیر جهت دار n راسی می تواند

بیشینه $(n-2)^n$ درخت پوشای داشته باشد.

(فرمول کایلی)

فرمول کایلی به وسیله نظریه

ماتریس-درخت کیوشیف یا کد پوفر،

قابل اثبات می باشد.

برای یک گراف کامل متشکل از دو بخش p و q

راسی G داریم:

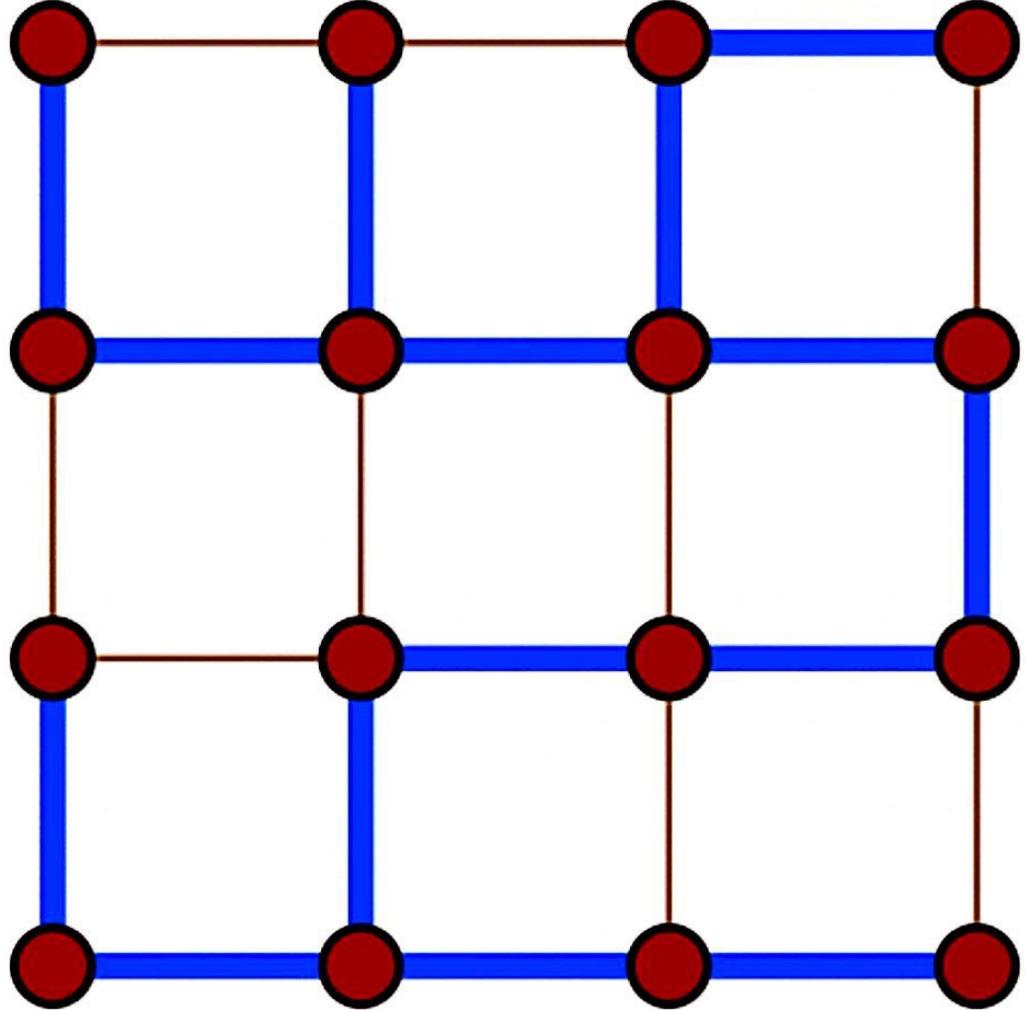
$$t(G) = p^q \cdot q^p$$

الگوریتم های درخت پوشان

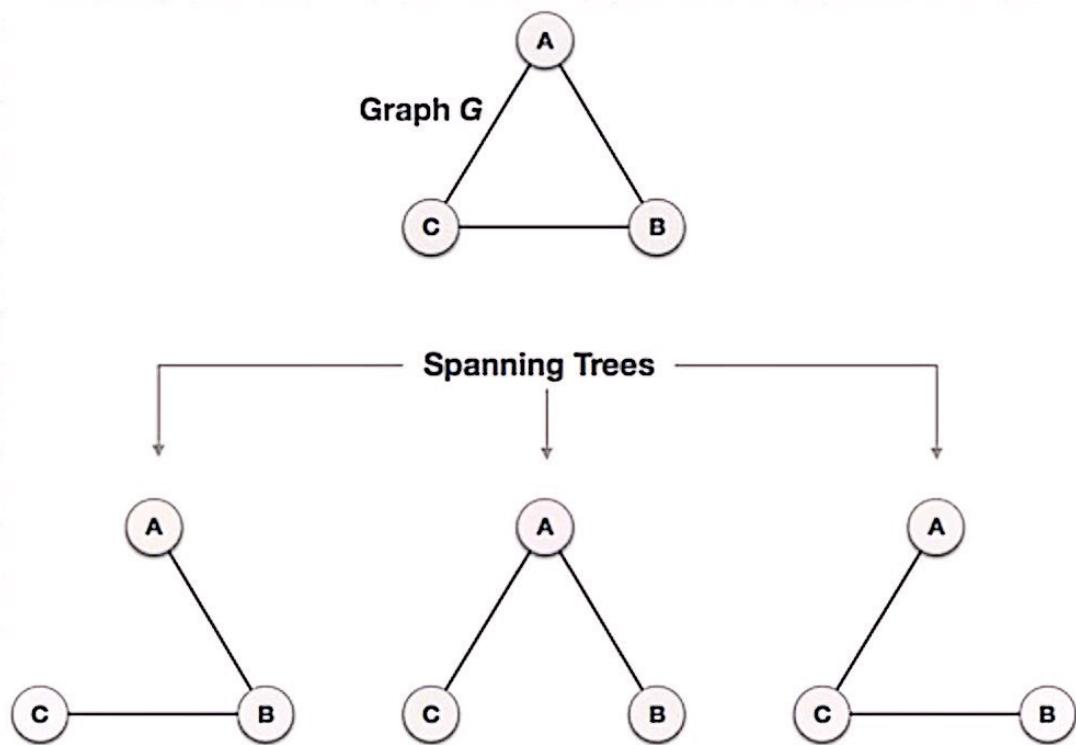
الگوریتم سنتی و متداول درخت پوشان بر مبنای جستجوی اول عمق (DFS) و جستجوی اول سطح (BFS) کار می‌کنند. الگوریتم‌های موازی از روش‌ها و رهیافت‌های دیگری به جز DFS و BFS استفاده می‌کنند. هالپرین و زوییک الگوریتم موازی تصادفی بهینه‌ای که زمان اجرایش $O(\log n)$ باشد بر روی EREW PRAM طراحی کرده‌اند.

الگوریتم شیلوچ-ویشکین که بر اساس تحقیقات یوسی شیلوچ و یوزی ویشکین طراحی شده است، پایه و اساس بسیاری از پیاده سازی‌های موازی می‌باشد.

الگوریتم بادر و کونگ نشان داده است که در گراف‌های گوناگون، بسیار سریع کار می‌کند. معمول‌ترین الگوریتم توزیع شده، پروتکل درخت پوشان می‌باشد که توسط دستگاه‌های لایه لینک در مدل OSI از شبکه‌های رایانه‌ای استفاده می‌شود. این دستگاه‌ها که عمدها روترها و سوئیچ‌ها هستند برای جلوگیری از توفان انتشار پیام‌های مسیریابی در شبکه‌های رایانه‌ای مورد استفاده قرار می‌گیرند؛ در واقع در هر مرحله دور موجود در شبکه را با غیرفعال کردن برخی از یال‌ها از بین می‌برند تا پیام‌های انتشاری در حلقه گیر نکنند و در شبکه اکو نشوند.



یالهای آبی رنگ، یک درخت پوشاده اند.



ساخت زیرگراف هایی از جنس درخت پوشاده اند.



در برخی از زیر رشته های نظریه گراف ها معمولاً پیدا کردن درخت پوشای کمینه در یک درخت وزن دار مورد بررسی قرار می گیرد.

مسائل بهینه سازی دیگری نیز در مورد درخت های پوشای بررسی شده اند از جمله موارد زیر:

- ✓ درخت پوشای بیشینه.

- ✓ درخت کمینه ای که شامل حداقل k رأس باشد.

- ✓ درخت پوشای کمینه ای که حداقل k یال به ازای هر رأس دارد.

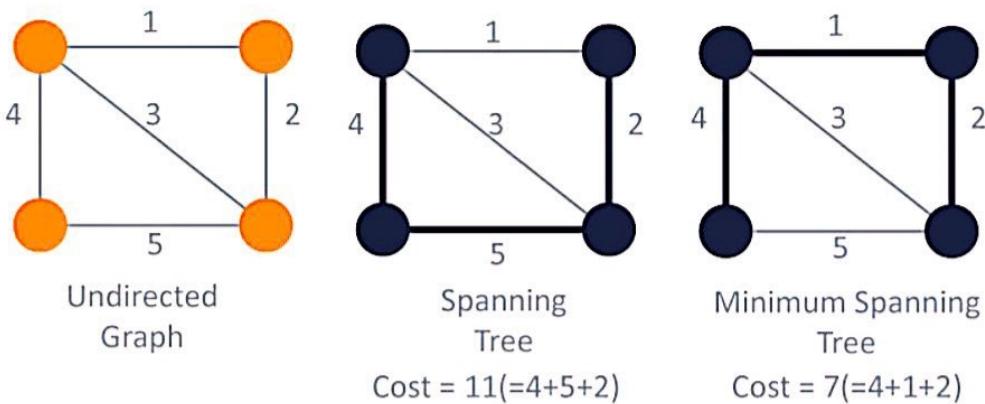
- ✓ درخت پوشایی که بیشترین تعداد برگ را دارد.

- ✓ درخت پوشایی با کمترین تعداد برگ (مرتبط به مسئله مسیر همیلتونی).

- ✓ درخت پوشایی با کمترین قطر.

درخت پوشای کمینه / مینیمم / بهینه (Minimum Spanning Tree)

درخت پوشای کمینه یک درخت پوشای مجموع هزینه (مجموع هزینه یال ها) باشد. مثال زیر را در نظر بگیرید.



در گراف G سمت چپ یک گراف وزن دار همبند غیر جهت است.

دو گراف دیگر زیر مجموعه ای از گراف G هستند که در شامل تمام راس ها می باشد و دوری در آن تشکیل نشده است.

درخت T اول دارای هزینه 11 و درخت دوم دارای هزینه 7 می باشد پس هر دو درخت، درخت پوشای هستند ولی درخت دوم به دلیل هزینه کمتر درخت پوشای کمینه است.

درخت پوشای مینیمم یا درخت فراگیر مینیمم در گراف های وزن دار ساخته می شود.

فرض شود گراف، یک گراف همبند باشد.

منظور از یک درخت پوشای از این گراف درختی است که شامل همه رئوس این گراف باشد ولی فقط

بعضی از یال‌های آنرا در برگیرد.
منظور از درخت پوشای مینیمم
(برای گراف همبند وزن دار)، درختی است که بین
درخت‌های پوشای آن گراف، مجموع وزن یال‌های
آن، کمترین مقدار ممکن باشد.

شمارش درخت‌های پوشای کمینه
اگر تمام یال‌ها وزن برابری داشته باشند،
تمام زیردرخت‌ها درخت پوشای کمینه هستند.
اگر وزن هر دو یال با هم متفاوت باشد، تنها یک
درخت پوشای کمینه خواهیم داشت؛ زیرا در چنین
گرافی، این الگوریتم یک درخت واحد تولید
می‌کند.

کم وزن ترین یال در برش گراف
اگر گراف مفروض G را به مجموعه های V و V' از رئوس افزایش کنیم، کم وزن ترین یال بین یال هایی که یک طرفشان در مجموعه V و طرف دیگرشان در مجموعه V' است، جز درخت پوشای کمینه می باشد.

اگر چند یال با کمترین وزن وجود داشت، باید دقیقاً یکی از آنها جزء درخت پوشای کمینه باشد.

- اثبات:

اگر یالی بین این دو مجموعه انتخاب نشود، گراف همبند نمی شود.
اگر دو یال یا بیشتر انتخاب شود با فرض همبندی در هر دو مجموعه دور ایجاد خواهد شد.
اگر یالی که کمترین وزن را ندارد انتخاب شود، می توان با عوض کردن آن با کم وزن ترین ویژگی های درخت را حفظ کرد و مجموع وزن ها را کاهش داد.
بنابراین کم وزن ترین انتخاب می شود.

کم وزن ترین یال گراف
اگر کم وزن ترین یال گراف یکتا باشد، در هر درخت پوشای کمینه ای وجود خواهد داشت.

پُر وزن ترین یال هر دور
اگر یالی در یک دور از تمام یال‌های موجود در آن دور وزنش بیشتر باشد، نمی‌تواند در درخت پوشای کمینه قرار بگیرد.

- اثبات به روش برهان خلف:

فرض خلف: فرض کنید این یال در یک درخت پوشای کمینه وجود دارد.

اگر آن را حذف کنیم، درخت به دو مؤلف تقسیم می‌شود و دور مذکور در گراف اصلی بدون این یال یک مسیر بین این دو مؤلفه می‌شود.

یالی از این مسیر که این مسیر را از این مؤلفه به آن مؤلفه منتقل می‌کند، جایگزین یال حذف شده به درخت اضافه کنید (ممکن است چند یال این کار را کنند که مهم نیست کدام را انتخاب می‌کنیم).

این یال کم‌وزن‌تر است؛ پس مجموع وزن‌های یال‌های درخت جدید کم‌تر است که با کمینه بودن درخت اول در تناقض است. پس فرض خلف باطل باطل، و حکم، ثابت می‌شود.

الگوریتم های یابنده درخت پوشای کمینه
برای به دست آوردن درخت پوشای بھینه یک گراف
جهت دار متصل می توان از الگوریتم های متفاوتی
استفاده نمود؛ از جمله:

✓ الگوریتم کروسال

✓ الگوریتم پریم

✓ الگوریتم بروکا (سولین)

✓ الگوریتم حذف معکوس

(Reverse Delete Algorithm)

✓ الگوریتم ژنتیک

(Particle Swarm Optimization) PSO ✓

✓ الگوریتم رقابت استعماری

✓ الگوریتم کرم شب تاب

کاربردهای درخت پوشای مینیمم

در مسائلی که هدف ایجاد شبکه‌ای است که برای ایجاد ارتباط بین هر دو عضو آن هزینه‌ای باید

بپردازیم و می‌خواهیم در نهایت در این شبکه بین

هر دو عضو ارتباط وجود داشته باشد، درخت

پوشای کمینه همان کم هزینه‌ترین شبکه است.

برای مثال فرض کنید در شهری می‌خواهیم طوری

شبکه ایجاد کنیم که بتوان از هر گره به هر گره

دیگری مسیر وجود داشته باشد و هزینه کابل کشی

بین هر دو گره را داریم.

برای پیدا کردن کم هزینه‌ترین شبکه بندی، باید

درخت پوشای کمینه را بیابیم.

بهبود طبقه بندی طیفی- مکانی
جنگل پوشای مینیمم (MSF)
با کاهش ابعاد تصاویر فراتیفی
برای سنجش از دور فراتیفی، الگوریتم جنگل
پوشای مینیمم (Minimum Spanning Forest)
مبتنی بر نشانه ها که یکی از دقیق ترین الگوریتم
ها در این زمینه است و تکنیک کاهش ابعاد معرفی
می شود.

در این روش تاثیر کاهش ابعاد تصاویر فراتیفی به
کمک الگوریتم ژنتیک در سه مرحله قبل و بعد از
انتخاب نشانه ها و به صورت همزمان بررسی می
گردد.

در این مطالعه نشانه ها از روی نقشه طبقه بندی
ماشین بردار پشتیبان (SVM) انتخاب شدند.
روش پیشنهادی بر روی سه تصویر فراتیفی
Indian Pines و Pavia، Telops گردید، نتایج آزمایشات بدست آمده برتری به
کارگیری الگوریتم ژنتیک را قبل از انتخاب نشانه ها
در تصاویر Pavia و Telops نشان می دهد.

در تصویر Indian Pines کاهش ابعاد در هر دو
مرحله قبل و بعد از انتخاب نشانه ها و به صورت
همزمان موجب افزایش دقّت طبقه بندی
می گردد.

پایان مطلب
همچنین بخوانید:
معرفی الگوریتم های درخت پوشای مینیمه

الگوریتم های درخت پوشای کمینه

الگوریتم کراسکال (Kruskal Algorithm) در نظریه گراف، الگوریتم کراسکال الگوریتمی برای یافتن یک زیرگراف فرآگیر همبند با کمترین وزن در یک گراف وزن دار است (در یک گراف وزن دار، به هر یال وزنی نسبت داده شده است).

این الگوریتم با گراف به صورت یک جنگل برخورد می کند که در آن هر گره یک درخت منفرد محسوب می شود.

یک درخت زمانی به درخت دیگر وصل می شود اگر و فقط اگر در میان همه گزینه های موجود، کمترین هزینه را داشته باشد و مشخصات درخت پوشای کمینه (MST) را نیز نقض نکند.

این الگوریتم برخلاف الگوریتم پریم لزوماً اجزایی که در حین اجرا جزء درخت پوشای کمینه تشخیص می دهد همبند نیستند و تنها تضمین می کند که در پایان این شرط برقرار است.

این الگوریتم یک الگوریتم حریصانه است پس انتخاب حریصانه انتخاب کوچکترین (کم هزینه ترین) وزن لبه است که باعث ایجاد چرخه ای در MST نشود.

به عنوان مثال فرض کنید یک شبکه راه آهن که تعدادی شهر را به یکدیگر متصل می کند در دست احداث است می خواهیم با داشتن هزینه C_{ij} مربوط به احداث خط مستقیم بین شهرهای v_i و

V_j شبکه را طوری طراحی کنیم که مجموع هزینه‌های ساخت به کمترین مقدار خود برسد. با در نظر گرفتن هر شهر به عنوان یک راس از گراف وزن دار با وزن‌های $(V_i, V_j) = w_{ij}$ مسئله به یافتن یک زیرگراف فراگیر همبند با کمترین وزن در یک گراف منجر می‌شود.

فرض کنید وزن‌ها نامنفی هستند بنابراین می‌توانیم تصور کنیم که زیرگراف فراگیر با کمترین وزن یک درخت فراگیر T از G است.

مراحل الگوریتم کراسکال

1. حذف همه یالهای طوقه و موازی
همه یالهای طوقه و یالهای موازی را از گراف
داده شده را حذف می‌کنیم. در مورد یالهای
موازی، آن یالی را نگه می‌داریم که کمترین هزینه را
دارد و بقیه را حذف می‌کنیم.

2. چیدمان همه یالها به ترتیب افزایش وزن
در این مرحله مجموعه‌ای از یالها و وزن‌هایشان
ایجاد می‌کنیم و آن‌ها را بر اساس ترتیب افزایش
وزن (هزینه) می‌چینیم.

3. یالی که کمترین وزن را دارد اضافه می‌کنیم
اینک برای افزودن یالها به گراف، کار خود را از
یالی آغاز می‌کنیم که کمترین وزن را دارد.
در تمام طول این فرایند بررسی می‌کنیم که
مشخصات پوشایش بودن همچنان برقرار باشد.
در موردی که با افزودن یک یال مشخصات درخت
پوشایش شود، نمی‌باشد این یال را وارد گراف
کنیم.

یالهای انتخاب شده تنها در آخر کار یک
مجموعه‌ی همبند تشکیل می‌دهند.

هر درخت فراگیر که با این الگوریتم ساخته می‌
شود، یک درخت بهینه است.

- اثبات به روش برهان خلف:

فرض خلف: فرض کنید درخت پوشای کمینه‌ی T وجود دارد که مجموع وزن یال‌های ساخته شده توسط آن کمتر از درخت تولید شده توسط الگوریتم کروسکال به نام G باشد. کم‌وزن‌ترین یالی در که عضو T است ولی عضو G نیست را انتخاب کنید.
این یال حتماً توسط الگوریتم کروسکال انتخاب می‌شود مگر اینکه با یال‌های قبلی دور بسازد و چون تمام یال‌های سبک‌تر از این یال در هر دو درخت وجود دارد به معنی آن است که در درخت T دور وجود دارد که تناقض است. پس فرض خلف باطل و حکم ثابت می‌شود.

شبه کد برای الگوریتم کراسکال

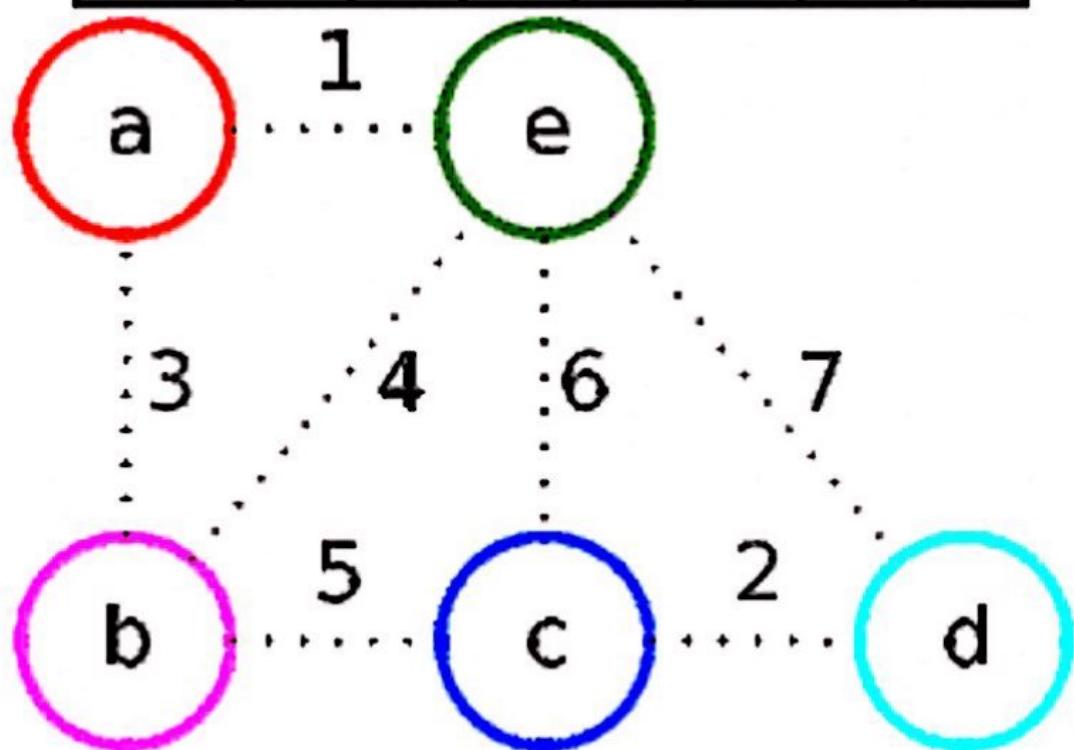
1. تمام یال‌ها را به ترتیب صعودی وزن مرتب کن.
2. برای هر راس v یک مجموعه بساز.
3. یال (u, v) را انتخاب کن.
4. اگر مجموعه‌ی u و v یکی نیستند، کارهای زیر را انجام بد:
 - مجموعه‌ی آنها را ادغام کن.
 - یال (u, v) را به عنوان یالی از درخت پوشای کمینه بردار.
5. اگر هنوز $n-1$ یال انتخاب نشده به شماره ۳ برو.
6. پایان

پیچیدگی الگوریتم کراسکال
مرتب کردن یال‌ها و بررسی یال‌ها از $O(m+m\log n)$
است که برابر $O(m\log n)$ است و هر بار اتصال و
مولفه از $O(n)$ است که چون این عمل $n-1$ بار
انجام می‌گردد، پیچیدگی الگوریتم،
 $(m\log n + n^2)$ می‌شود.

می‌توان از داده‌ساختار مجموعه‌های مجزا برای
مولفه‌های همبندی استفاده کرد، در این صورت
پیچیدگی الگوریتم به
 $O(m\log n+n) = O(m\log n)$ کاهش پیدا می‌کند.
گاهی چند یال دارای یک وزن هستند، در این حالت
ترتیب یال‌هایی که انتخاب می‌شوند مهم نیست.
درخت‌های پوشای حداقل مختلفی ممکن است
حاصل شود اما مجموع وزن آنها همیشه یکسان و
حداقل می‌شود. پیچیدگی زمانی الگوریتم $O(mn)$
می‌شود.

همچنین کارایی متوسط و پیچیدگی محاسباتی
بدترین حالت این الگوریتم، به ترتیب برابر
 $\Omega(m+n)$ و $O(m\log n)$ است.
 n تعداد رئوس و m ، تعداد یال‌های گراف G است.

Edge	ab	ae	bc	be	cd	ed	ec
Weight	3	1	5	4	2	7	6



الگوريتم كراسكال 

الگوریتم پریم (Prim's Algorithm)

الگوریتم پریم، الگوریتمی در نظریه گراف است که زیردرخت پوشای کمینه را برای یک گراف همبند وزن دار پیدا می‌کند یعنی زیرمجموعه‌ای از یال‌ها را در آن گراف می‌یابد که درختی را تشکیل می‌دهند که همه رئوس را شامل می‌شود در حالیکه مجموع وزن همه آن یال‌ها کمینه شده‌است.

این الگوریتم در تضاد با الگوریتم کراسکال است، چون با گره‌ها به عنوان یک درخت منفرد برخورد می‌کند و به افزودن گره‌ها به یک درخت پوشای گراف مفروض ادامه می‌دهد.

این الگوریتم در سال ۱۹۳۰ توسط ریاضیدانی به نام جارنیک داده شد و سپس در سال ۱۹۵۷ پریم، دانشمند علوم کامپیوتر آن را مستقل از جارنیک کشف کرد و در سال ۱۹۵۹ دایکسترا دوباره به آن دست یافت.

از این رو، این الگوریتم گاهی با نام الگوریتم DJP نیز شناخته می‌شود که برگرفته از اسمی دایکسترا، جارنیک و پریم است.

این الگوریتم برای پیدا کردن درخت پوشای کمینه یک گراف وزن دار است که مشابه الگوریتم دایکسترا برای درخت کوتاه ترین مسیر است.

مراحل الگوریتم پریم

1. حذف همه یالهای طوقه و موازی
همه یالهای طوقه و همچنین یالهای موازی را از
گراف مفروض حذف می‌کنیم.

در مورد یالهای موازی، یالهایی را حفظ می‌کنیم
که کمترین هزینه را دارند و بقیه را حذف می‌کنیم.

2. یک راس دلخواه را به عنوان ریشه انتخاب
کنید.

این گره به طور دلخواه انتخاب شده است و هر
راس دیگری به جای آن می‌توان انتخاب کرد.
در درخت پوشان، همه رئوس در یک گراف گنجانده
می‌شوند و از آن جا که گراف، همبند است، در این
صورت می‌بایست دست‌کم یک یال برای هر گره
باشد که آن را به بقیه درخت متصل سازد.

3. بررسی یالهای خروجی و انتخاب یالی که
کمترین هزینه را دارد.

یالی را انتخاب می‌کنیم که کمترین هزینه را دارد و
آن را در درخت می‌گنجانیم.

اینکه بار دیگر با آن به صورت یک گره برخورد
می‌کنیم و همه یالها را مجدداً بررسی می‌کنیم. با
این حال، مجدداً تنها یالی که کمترین هزینه را دارد
انتخاب می‌کنیم.

بنابراین می‌توانیم هر یک از آنها را که می‌خواهیم
به درخت اضافه کنیم.

از این رو یک درخت پوشای گنجاندن هر دو یال نشان می‌دهیم.

بدین ترتیب درمی‌یابیم که خروجی درخت پوشای گراف با استفاده از هر دو الگوریتم یکسان خواهد بود.

در الگوریتم پریم دو محدودیت در هر مرحله داریم یکی آن که جنگل ایجاد نشود و دیگری آنکه حلقه شکل نگیرد.

در الگوریتم پریم همیشه در طول ساخت درخت، مجموعه ما همبند بود.

شبه کد الگوریتم پریم

1. رئوس گراف را به دو مجموعه V' (شامل همه رئوس گراف به جز راس دلخواه v) و V (شامل v) افراز کن.

2. تا زمانی که V شامل تمام راس‌ها نشده کارهای زیر را انجام بدہ:

- بین تمام یال‌های بین مجموعه V و V' کم وزن‌ترین را انتخاب کن (مثلاً یال (u,v) که u در V است).

- راس v را از V' حذف، و به V اضافه کن.
- یال (u,v) را به عنوان یالی از درخت پوشای کمینه انتخاب کن.

3. پایان

پیچیدگی الگوریتم پریم

یک روش خوب برای بهینه کردن الگوریتم نگه داشتن کمترین فاصله‌ی هر راس تا راس‌های انتخابیست.

وقتی یک راس به مجموعه‌ی ما اضافه شد،

فاصله‌ی بقیه راس‌ها را به روز می‌کنیم.

در این صورت هر بار برای پیدا کردن راس نزدیک‌تر و به روز رسانی $O(n)$ عملیات انجام می‌دهیم و چون n بار این کار انجام می‌دهیم، پیچیدگی کل الگوریتم،

$O(n^2)$ می‌شود.

اگر فاصله‌های راس تا نزدیک‌ترین راس از بین راس‌هایی که در مجموعه قرار گرفته‌اند را در داده‌ساختاری مناسب ذخیره کنیم می‌توانیم پیچیدگی الگوریتم را بهتر کنیم. اگر این

داده‌ساختار هرم کمینه باشد، چون حذف و اضافه از

$O(\log n)$ است و به ازای یال حداقل یک بار

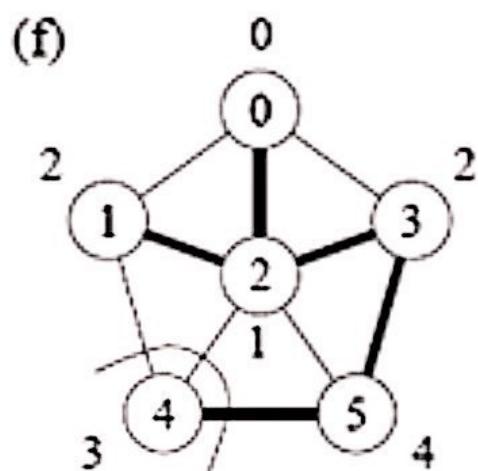
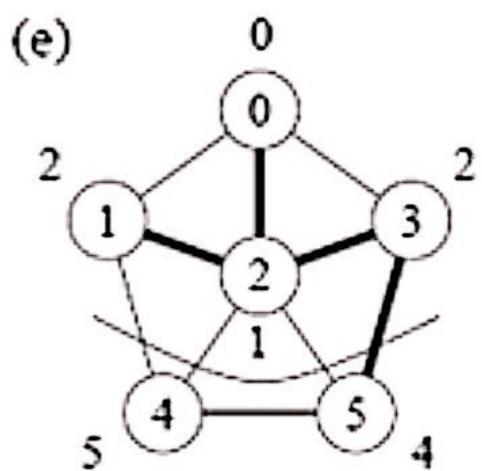
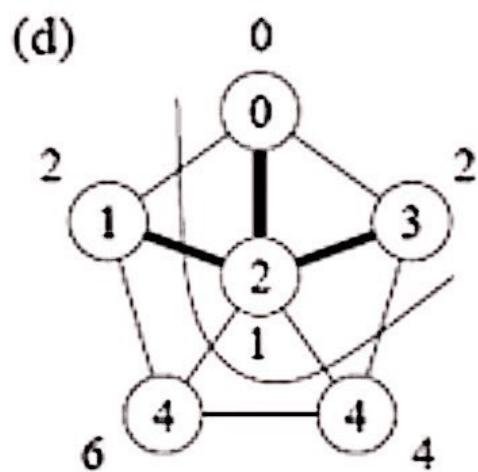
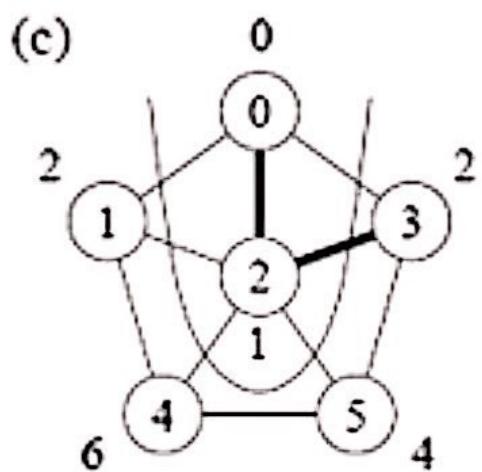
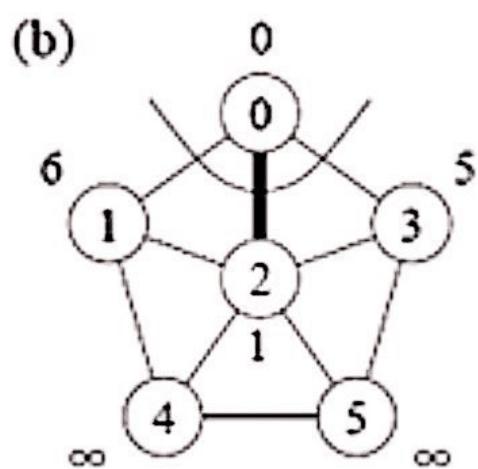
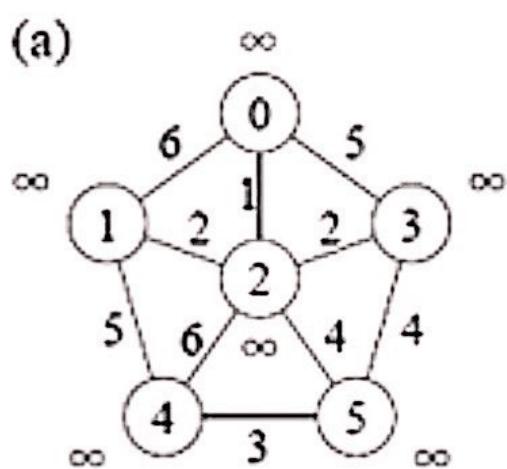
عملیات اضافه کردن رخ می‌دهد و چون قطعاً

تعداد عملیات حذف کمتر از تعداد عملیات اضافه کردن است پیچیدگی الگوریتم به

$O(m \log n + n)$ تغییر می‌کند که برای حالاتی که تعداد یال‌ها از

$|O(n^2) - O(\log n)|$ کمتر باشد پیچیدگی کل کاهش پیدا می‌کند. حال اگر از هرم فیبوناچی استفاده کنیم پیچیدگی به

کاہش پیدا می کند. $O(m+n)$
 n تعداد رئوس و m ، تعداد یال های گراف G است.



الگوریتم پریم.

 پیدا کردن درخت فراگیر مینیمال

الگوریتم بروکا (سولین)

الگوریتم بروکا برای پیدا کردن درخت پوشای مینه با وزن یال‌های مجزا در یک گراف است.

درخت پوشای مینه درختی است شامل تمام رأس‌های یک گراف، به‌طوری‌که مجموع وزن یال‌هایش کمترین باشد.

این اولین الگوریتمی بود که در سال ۱۹۲۶ برای پیدا کردن درخت پوشای مینه MSTs طراحی شد. آقای Otakar Boruvka از آن برای یافتن مسیریابی کارآمدترین شبکه برق استفاده کرده است.

الگوریتم‌ها و روش‌های زیادی برای پیدا کردن درخت پوشای حداقل وجود دارد.

الگوریتم Boruvka یک الگوریتم حریصانه است و مشابه الگوریتم کراسکال و الگوریتم پریم بوده و اساساً حد وسط بین دو این الگوریتم است.

همچنین این الگوریتم را می‌توان حالت موازی الگوریتم پریم دانست.

مراحل الگوریتم بروکا

در ابتدای یک مرحله لبه های انتخاب شده، همراه با تمام ۷ راس گراف مفروض، تشکیل یک درخت پوشای را می دهند.

در خلال یک مرحله یک لبه برای هر درخت انتخاب می شود که دارای حداقل هزینه بوده یعنی اینکه دقیقاً دارای یک راس در درخت می باشد.

از آنجا که دو درخت در جنگل می توانند یک لبه یکسان انتخاب کنند، لذا می توان کپی تکراری لبه ها را حذف کرد.

در ابتدای مرحله اول مجموعه لبه های انتخاب شده خالی است.

این الگوریتم هنگامی پایان می یابد که فقط یک درخت در انتهای یک مرحله باقی و یا هیچ لبه ای برای انتخاب باقی نمانده باشد.

بر خلاف الگوریتم های پریم و کروسکال، که در هر مرحله فقط یک یال به درخت اضافه می کردند، در الگوریتم سولین چندین لبه را برای درخت اضافه می کند.

پیچیدگی الگوریتم بروکا
در هر تکرار حلقه باید موارد زیر محاسبه شود:
در گام اول باید برای هر راس یال مورد نظر را پیدا
کرد که این کار بدون نیاز به مرتب کردن یال‌ها در
زمان $O(m)$ انجام می‌شود.

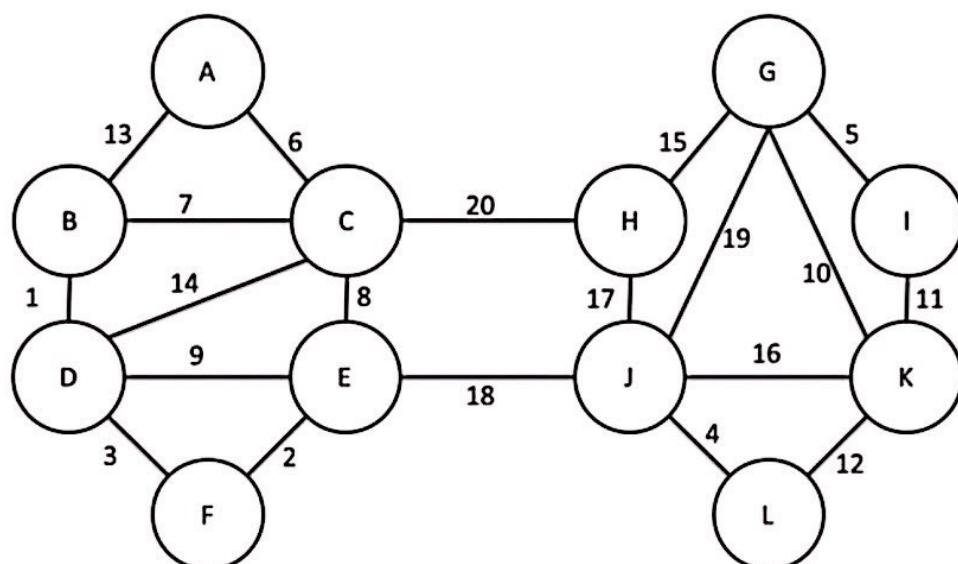
در گام بعد با پیدا شدن یال‌های مورد نظر باید
رأس‌ها را دوباره علامت‌گذاری کرد که این کار را نیز
می‌توان به کمک الگوریتم جستجوی عمق اول در
زمان $O(n)$ انجام داد.

در هر تکرار حداقل یک یال از اجزای متصل کم
می‌شود و بیشینه تکرار برابر $\log n$ است؛ پس
مرتبه کلی الگوریتم را با توجه به توضیحات
می‌توان $O(m \log n)$ در نظر گرفت.

سریع‌ترین الگوریتم در این زمینه را می‌توان با ترکیب الگوریتم پریم و الگوریتم بروکا به دست آورد. سریع‌ترین الگوریتم یافتن درخت پوشای کمینه تصادفی بر پایه الگوریتم بروکا است که در زمان $O(m)$ اجرا می‌شود.

بهترین الگوریتم قطعی شناخته شده برای یافتن درخت کمینه پوشای Bernard Chazelle، که توسط Bernard Chazelle ارائه شد نیز برپایه الگوریتم بروکا و با زمان اجرایی برابر $O(m\alpha(n))$ است.

این الگوریتم‌های قطعی و تصادفی مراحل الگوریتم بروکا را تلفیق می‌کنند به این صورت که تعداد مؤلفه‌هایی که برای اتصال باقی می‌ماند را با مراحل مختلفی که تعداد یال‌های بین جفت مؤلفه‌ها را کاهش می‌دهند، کم می‌کند. n تعداد رئوس و m ، تعداد یال‌های گراف G است.



الگوریتم بروکا



الگوریتم میانگین-خطی

این، یک الگوریتم تصادفی برای یافتن درخت پوشای مینه در یک گراف وزن دار بدون راس تنها است. این الگوریتم توسط David Karger، فیلیپ کلین و Robert Tarjan ابداع شده است.

این الگوریتم متکی بر شیوه الگوریتم بروکا است که برای یافت درخت پوشای کمینه در زمانی خطی است. این الگوریتم تشکیل شده از الگوریتم تقسیم و حل، الگوریتم حریصانه و الگوریتم های تصادفی برای رسیدن به زمان خطی مورد نظر.

کلید این الگوریتم، تقسیم تصادفی گراف به دو زیرگراف و تقسیم یال‌ها بین این زیرگراف هاست. الگوریتم به صورت بازگشتی جواب را برای یک زیرگراف می‌یابد و سپس آن را به کل گراف تعمیم می‌دهد.

کارایی این الگوریتم در صورتی تضمین می‌شود که نمونه‌گیری تصادفی باشد.

الگوريتم حذف معکوس

(Reverse-delete Algorithm)

در نظریه گراف الگوريتم حذف معکوس، الگوريتمی است که در یک گراف همبند با یال وزن دار، درخ پوشای کمینه را بدست می آورد.

اگر گراف ناهمبند باشد، این الگوريتم را برای هر مؤلفه همبندی می یابد که در این صورت مجموعه این درخت‌های پوشای کمینه را یک جنگل پوشای کمینه گویند.

این الگوريتم، حریصانه است که در هر لحظه بهترین انتخاب را انجام می‌دهد و برعکس الگوريتم کراسکال عمل می‌کند.

الگوريتم کراسکال با یک گراف خالی شروع می‌کند و یال‌ها را به آن اضافه می‌کند در حالیکه الگوريتم حذف معکوس با گراف اصلی شروع می‌کند و یال‌ها را از آن حذف می‌کند.

این الگوريتم این تضمین را می‌دهد که گراف همبند باقی بماند، چون تنها در صورتی یالی را حذف می‌کند که باعث ناهمبند شدن گراف نشود. هر یالی که حذف می‌شود قبل از حذف در دوری شرکت داشته است.

از آنجایی که الگوريتم از یال با بیشترین وزن شروع به حذف کردن می‌کند، آن یال بزرگ‌ترین یال در دور مربوط به خود است؛ پس بنابر تعریف درخت پوشای کمینه، یال حذف شده جزء درخت پوشای کمینه نخواهد بود.

مراحل الگوریتم حذف معکوس

1. با گراف G که لیستی از یال‌های E دارد، شروع کن.
2. E را به ترتیب نزولی وزن یال‌ها مرتب کن.
3. برای هر یال چک کن که آیا حذف این یال گراف را ناهمبند می‌کند یا نه.
4. اگر حذف کردن منجر به ناهمبندی گراف نمی‌شود، یال را حذف کن.

پیچیدگی الگوریتم حذف معکوس

الگوریتم در زمان $O(E \log E (\log (\log E))^3)$ انجام می‌شود که E تعداد یال‌ها و V تعداد گره‌های گراف است. این حد به صورت زیر محاسبه شده است:

مرتب سازی یال‌ها با استفاده از الگوریتم مرتب سازی مقایسه‌ای در زمان $O(E \log E)$ انجام می‌گیرد حلقه E بار تکرار می‌شود.

حذف کردن یال در $O(1)$ زمان انجام می‌گیرد. همبندی در زمان $O(\log V (\log \log V)^3)$ انجام می‌گیرد.

زمان اجرا را می‌توان $O(E \log V (\log (\log V))^3)$ در نظر گرفت زیرا E حداقل V^2 است.

دیگر الگوریتم ها بنا به دلیل رویکرد تکاملی در حل مسئله، مورد تشریح قرار نخواهند گرفت.

پایان

