درس طراحی الگوریتم
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی
کتاب مرجع: طراحی الگوریتم، جان کلاینبرگ، اوا تاردش

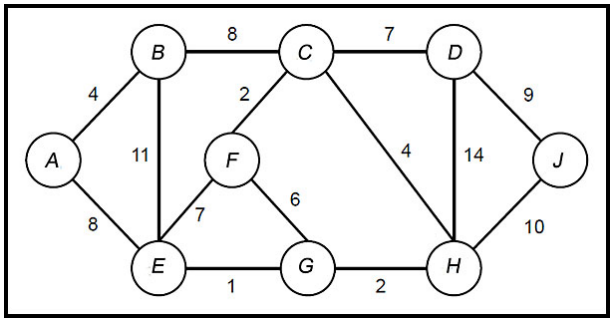
الگوریتم های حریصانه

۱ minimum spanning tree مسئله درخت فراگیر کمینه

گراف ساده و همبند $G = (V, E)$ داده شده است. یک تابع هزینه $c : E \to \mathbb{R}$ هم داده شده که هزینه هر یال را مشخص میکند. دقت کنید اینجا هزینه یال میتواند منفی هم باشد. میخواهیم مجموعه ای از یالها $T \subseteq E$ را انتخاب کنیم بطوریکه یالهای انتخابی تشکیل یک درخت را بدهند و مجموع هزینه یالهای انتخابی مینیمم باشد. یعنی درختی فراگیر و کم هزینه تر از آن وجود نداشته باشد.

تعریف: به مجموعه ای از یالها که تشکیل یک درخت بدهد و همه رئوس گراف را در بر بگیرد یک درخت فراگیر گفته میشود.

مسئله به عبارت دیگر این است که در گراف داده شده میخواهیم یک درخت فراگیر با کمترین هزینه پیدا کنیم.



یکی از کاربردهای مسئله درخت فراگیر کمینه در طراحی شبکه های کامپیوتری است. اینجا میخواهیم مجموعه ای از سرورها servers را با یک شبکه کم هزینه به هم متصل کنیم بطوریکه بین هر دو سرور حداقل یک مسیر وجود داشته باشد. لینکهای ارتباطی که همان یالها هستند هر کدام هزینه برقراری و نصب خود را دارد. دقت کنید اینجا هزینه ها مثبت هستند. روشن است که راه کم هزینه این است که یالها تشکیل یک درخت فراگیر را بدهند. علاوه بر این درختی میخواهیم که در مجموع یالهایش هزینه کمتری بخواهد.

۲ الگوریتم کراسکال

الگوریتم زیر برای مسئله درخت فراگیر کمینه توسط جوزف کراسکال ریاضیدان آمریکایی پیشنهاد شده است.

۱. یالهای گراف را به ترتیب وزن بطور صعودی مرتب کن. فرض $e_1, \ldots, e_m$ لیست یالهای مرتب شده باشد.

۲. درخت فراگیر کمینه T در ابتدای کار تهی است.

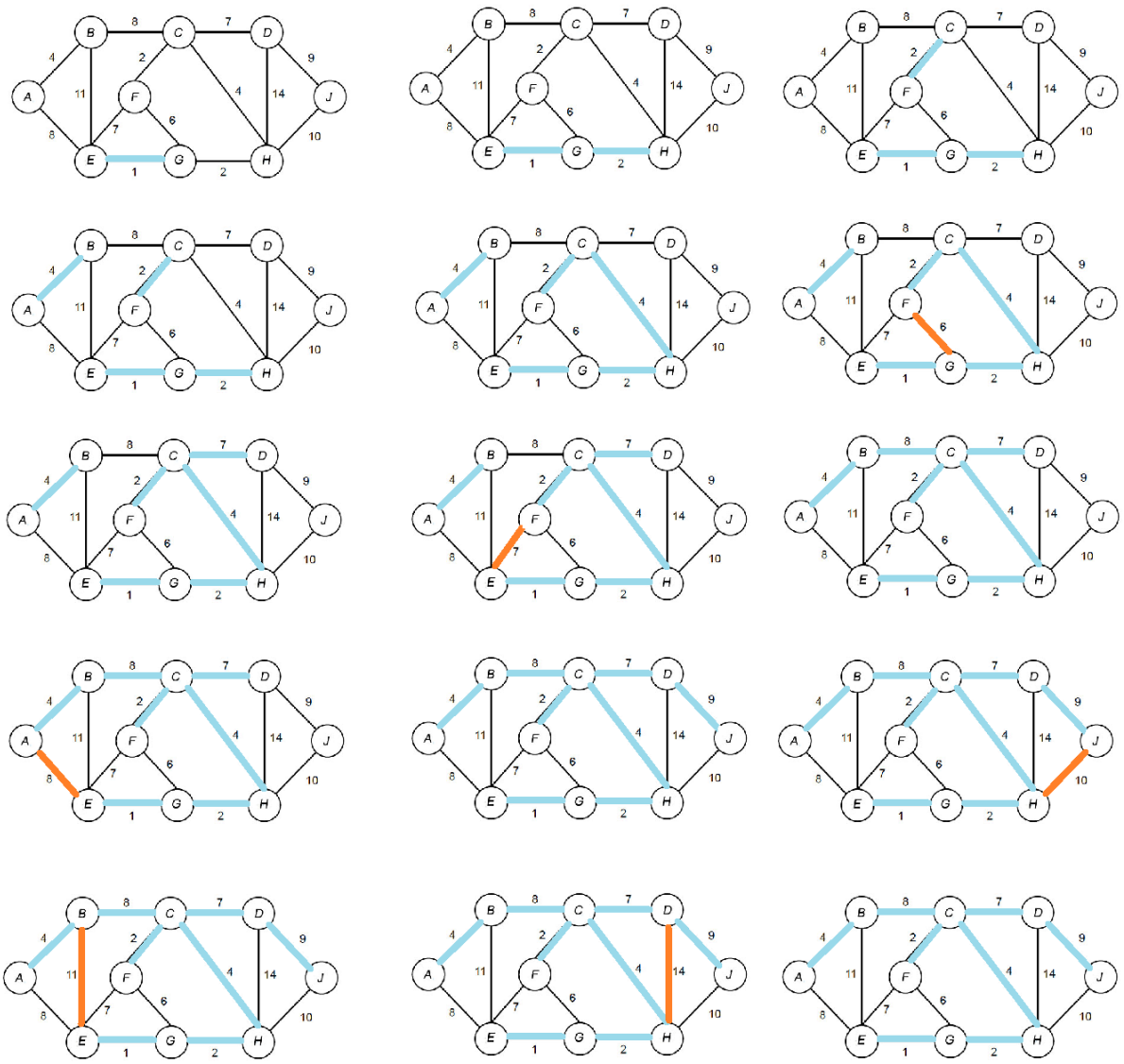
۳. برای هر i از 1 تا m کارهای زیر را انجام بده

- اگر اضافه کردن e_i به T دوری ایجاد میکند از e_i صرف نظر کن در غیر اینصورت $T \leftarrow T \cup \{e_i\}$

۴. در انتهای کار T جواب الگوریتم است.

مراحل اجرای الگوریتم کراسکال برای گراف شکل صفحه قبل.

sorted list of edges: (EG, GH, CF, AB, CH, FG, CD, EF, BC, AE, DJ, JH, BE, DH)



۳ الگوریتم پریم

الگوریتم پریم برای مسئله درخت فراگیر کمینه توسط رابرت پریم ریاضیدان آمریکایی پیشنهاد شده است. طرز کار این الگوریتم خیلی شبیه الگوریتم دایکسترا است. ابتدا یک راس دلخواه $s \in V$ انتخاب میشود و مجموعه $S = \{s\}$ مقداردهی میشود. الگوریتم در هر مرحله یک راس $x \in V \setminus S$ را به جمع رئوس S اضافه میکند. راسی انتخاب میشود که با کمترین هزینه به S وصل شود. یالی که راس مورد نظر را به S وصل میکند به درخت در حال رشد T که در ابتدا تهی است اضافه میشود. کار ادامه پیدا میکند تا زمانیکه S همه گراف را در بر بگیرد.

۱. راس دلخواه $s \in V$ را انتخاب کن.

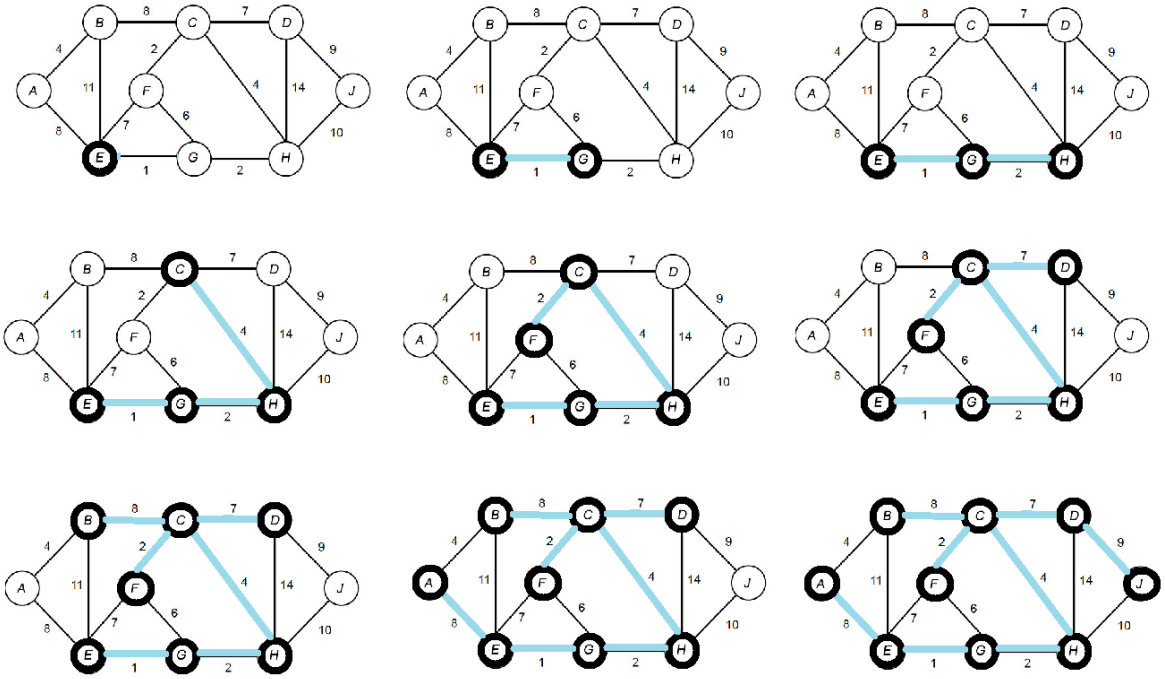
۲. قرار بده $S = \{s\}$.

۳. $T \leftarrow \emptyset$

۴. تا زمانیکه $S \neq V$ کارهای زیر را انجام بده:

- $(y, x) \leftarrow \text{argmin}_{y \in S, x \in V \setminus S}\{c(y, x)\}$ به عبارت دیگر، یال (y, x) را پیدا کن که یک سر در S و سر دیگر در $V \setminus S$ داشته باشد و کمترین وزن را داشته باشد.
- یال انتخابی (y, x) در قدم قبلی را به T اضافه کن. همچنین راس x را به S اضافه کن.

یک نمونه از اجرای الگوریتم پریم در شکل زیر نشان داده شده است.



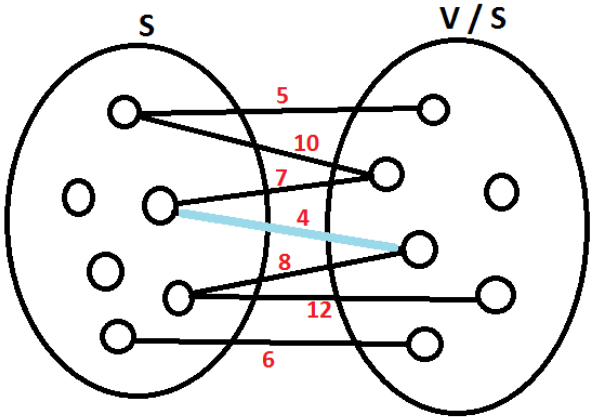
۴ الگوریتم برعکس کراسکال

در این روش برای پیدا کردن درخت فراگیر کمینه، ابتدا یالها را از وزن بزرگ به کوچک مرتب می‌کنیم (برعکس روش کراسکال). حال یالها را به ترتیب بررسی می‌کنیم. یال e_i را حذف می‌کنیم به شرطی که حذف آن گراف کنونی را غیرهمبند نکند. اگر حذف یال باعث شود که گراف غیر همبند شود آن را نگه می‌داریم. یالهایی که در انتها باقی می‌مانند، همانند یالهای درخت فراگیر کمینه هستند.

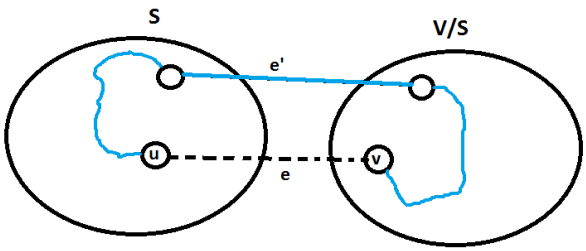
۵ دو لم اساسی در مورد درخت فراگیر کمینه

در این قسمت ابتدا فرض می‌گیرم که یالهای گراف وزن تکراری ندارند. به عبارت دیگر وزنها همه متمایز هستند. در بخش بعدی این پیشفرض را حذف می‌کنیم و نشان می‌دهیم که بدون این پیشفرض هم الگوریتم کراسکال و پریم و الگوریتم برعکس کراسکال درخت بهینه را پیدا می‌کنند. برای اثبات درستی الگوریتمها دو لم اساسی در مورد درختهای فراگیر کمینه را بیان می‌کنیم و آن را اثبات می‌کنیم.

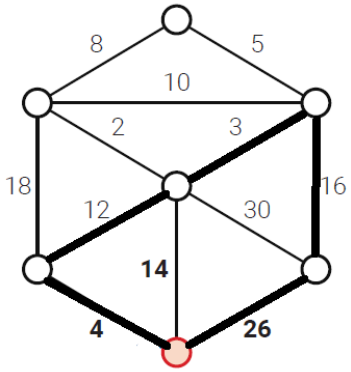
لم: (۱) گراف ساده و همبند $G = (V, E)$ همراه با تابع وزن $c : E \to \mathbb{R}$ را در نظر بگیرید. فرض کنید که وزن یالها متمایز باشند. اگر مجموعه رئوس را به دو قسمت S و $V \setminus S$ افراز کنیم، آنگاه کم وزنترین یال که یک سر در S و سر دیگر در $V \setminus S$ دارد حتما جزو درخت فراگیر کمینه است.



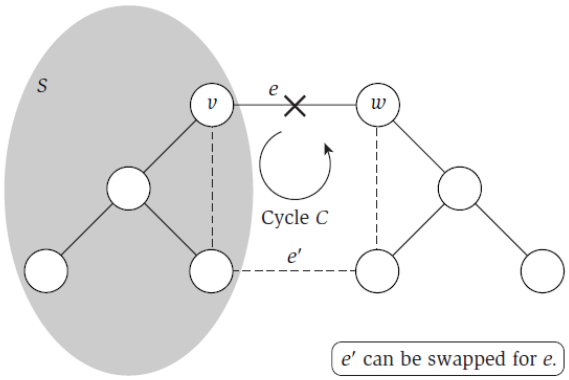
اثبات: اثبات با برهان خلف. فرض کنید T یک درخت فراگیر کمینه برای G است در حالیکه یک برش $(S, V \setminus S)$ از گراف وجود دارد بطوریکه $e = (u, v)$ سبکترین یال در برش است ولی در T نیست.
فرض کنید $u \in S$ و $v \in V \setminus S$. چون T یک درخت فراگیر است، پس حتما مسیری از u به v در T وجود دارد. این مسیر جایی برای اولین بار از S خارج میشود. اسم یالی در مسیر مورد نظر که برای اولین بار از S خارج میشود را e' میگذاریم. روشن است که e' هم مانند e در برش $(S, V \setminus S)$ قرار دارد.
توجه کنید که با حذف e' از T، درخت به دو مولفه همبند تقسیم میشود که u در یک مولفه و v در مولفه دیگر قرار میگیرد.
از طرف دیگر، اگر e' را با e عوض کنیم یک درخت فراگیر دیگر بدست می آید که نام آن را T' میگذاریم. چون با فرض برهان خلف e سبکترین یال برش بود و وزنها متمایز هستند، پس حتما $c(e) < c(e')$. لذا وزن درخت T' باید سبکتر از وزن T باشد. این با کمینه بودن T در تناقض است. □



لم: (۲) گراف ساده و همبند $G = (V, E)$ همراه با تابع وزن $c : E \to \mathbb{R}$ را در نظر بگیرید. فرض کنید که وزن یالها متمایز باشند. اگر C یک دور در گراف G باشد و یال $e \in C$ سنگین ترین یال در دور C باشد آنگاه هیچ درخت فراگیر کمینه‌ای نمی‌تواند شامل یال e باشد.



اثبات: اثبات با برهان خلف. فرض کنید T یک درخت فراگیر کمینه برای G است در حالیکه یال سنگین e در دور C عضوی از درخت T است. ادعا می‌کنیم می‌توانیم یال e را با یالی دیگر عوض کنیم و درخت فراگیر بهتری پیدا کنیم. فرض کنید $e = (v, w)$. اگر یال e را از T حذف کنیم دو مولفه همبند بدست می‌آید. فرض کنید S مجموعه رئوسی باشد که جزو مولفه‌ای است که شامل v است (مانند شکل زیر.)



حال دور C را در نظر بگیرید. یال e' در دور C وجود دارد که از مجموعه S خارج می‌شود و وارد مجموعه V/S می‌شود. این یال حتما جزو درخت T نیست (چون وجود آن در T باعث ایجاد دور می‌شود.) با فرض لم، می

دانیم که وزن یال e' از یال e کمتر است. پس با کنار گذاشتن e و اضافه کردن e' یک درخت فراگیر با وزن کمتر پیدا می‌کنیم که با فرض متناقض است. □

۱.۵ اثبات درستی الگوریتم کراسکال و پریم

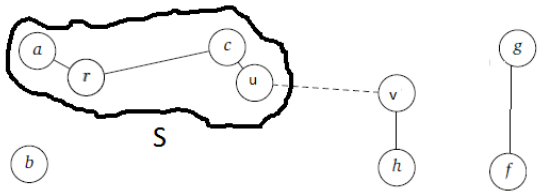
حال با استفاده از لم (۱) اثبات می‌کنیم که دو الگوریتم کراسکال و پریم یک درخت فراگیر کمینه را پیدا می‌کنند.

الگوریتم پریم در واقع یک نتیجه مستقیم از لم بالاست. در هر مرحله از الگوریتم یالی اضافه میشود که سبکترین یال برش مربوطه است و لذا پیرو لم بالا این یال حتما جزوی از درخت فراگیر کمینه است. چون الگوریتم در انتها یک درخت فراگیر را تولید میکند پس خروجی یک درخت فراگیر کمینه است.

برای اثبات درستی کراسکال، با استفاده از استقرا، نشان می‌دهیم همه انتخابهایی که الگوریتم انجام می‌دهد درست است. یعنی یالهایی که الگوریتم کراسکال انتخاب می‌کند حتما جزو درخت فراگیر کمینه هستند.

استقرا روی یالهایی که الگوریتم انتخاب میکند تعریف میشود. اولین یالی که انتخاب میکند حتما جزو درخت فراگیر کمینه است (پایه استقرا). فرض کنید اولین یال $e = (x, y)$ باشد. مجموعه $S = \{x\}$ را تعریف میکنیم. قطعا e سبکترین یال در برش $(S, V \setminus S)$ است چون e سبکترین یال گراف است.

حال فرض کنید که در یک مقطع از الگوریتم تعدادی یال انتخاب شده است و در مرحله جدید الگوریتم یال جدید $e = (u, v)$ را به T اضافه میکند. با فرض استقرا همه یالهایی که قبلا انتخاب شده اند باید جزوی از درخت فراگیر کمینه باشند. مجموعه S به اینصورت تعریف میکنیم. S شامل همه رئوسی از طریق یالهای T به u دسترسی دارند. این رئوس در واقع مولفه همبندی است که شامل راس u است.



میخواهیم ادعا کنیم e سبکترین یال برش $(S, V \setminus S)$ است. اگر e سبکترین یال نباشد حتما باید یالی در این برش قبل از یال e توسط الگوریتم پردازش شده باشد. این یال عضوی از مجموعه S را به عضوی از $V \setminus S$ وصل می‌کند. اما مجموعه S بصورت یک مولفه همبندی است و هیچ عضوی از آن به خارج ارتباطی ندارد. این متناقض با فرض استقرا است. در نتیجه e باید سبکترین یال برش باشد و لذا بنا به لم قبل که اثبات کردیم حتما جزوی از درخت فراگیر کمینه است.

در این انتها باید نشان دهیم خروجی الگوریتم کراسکال، یعنی T یک درخت فراگیر است. روشن است که T دور ندارد. از طرف دیگر T بیشتر از یک مولفه ندارد چون اگر اینطور باشد، یالهایی که بین مولفه ها هستند الگوریتم از آنها صرف نظر کرده چون ایجاد دور می‌کنند اما یالی که دو درخت را به هم وصل میکند ایجاد دور نمیکند. این نشان میدهد که T حتما یک مولفه دارد و لذا یک درخت فراگیر است.

۲.۵ اثبات درستی الگوریتم عکس کراسکال

برای اثبات درسی الگوریتم عکس کراسکال می‌توانیم از لم (۲) استفاده کنیم. جزئیات این بحث را به عنوان تمرین به دانشجو واگذار می‌کنیم.

۶ حذف پیشفرض متمایز بودن وزن یالها

اگر در گراف ورودی وزن یالی تکرار شده باشد، باز هم الگوریتمهای پریم و کراسکال یک درخت فراگیر کمینه را تولید می‌کنند. برای اثبات این مطلب از یک ترفند استفاده می‌کنیم. قبل از اینکه این ترفند را شرح دهیم، متذکر می‌شویم که این ترفند در عمل پیاده سازی نمیشود و صرفا برای اثبات درستی الگوریتم است.

فرض کنید به به همه یالهای گراف یک عدد مثبت خیلی کوچک را اضافه کنیم بطوریکه وزن جدید یالها متمایز شوند. به عبارت دیگر به یال i ام وزن بسیار کوچک ϵ_i را اضافه میکنیم.

$$c'(e_i) = c(e_i) + \epsilon_i$$

گراف جدید که یالهایش ثابت مانده و فقط وزن یالها تغییر کرده را G' می نامیم. می‌توانیم مقادیر ϵ را آنقدر کوچک انتخاب کنیم که درخت فراگیر کمینه G' از لحاظ مجموعه یالی تغییری نکند (برای این منظور، مقدار $\sum_{i=1}^{m} \epsilon_i$ حداکثر چقدر باید باشد؟) یعنی اگر T یک درخت فراگیر کمینه برای G باشد، آنگاه T یک درخت فراگیر کمینه برای G' نیز باشد و بالعکس.

چون وزنهای G' متمایز هستند پس الگوریتم کراسکال و پریم درخت فراگیر کمینه را حتما بدرستی پیدا میکنند. پس در واقع درخت فراگیر کمینه G را پیدا میکنند.

به الگوریتم کراسکال توجه کنید. این الگوریتم ابتدا یالها را برحسب وزن بطور صعودی مرتب می‌کند. یالهایی که وزن یکسان دارند پشت سر هم قرار می‌گیرند با یک ترتیب نامشخص. بعد از مرتب سازی یالها، دیگر الگوریتم به وزن یالها کاری ندارد. دقت کنید اضافه کردن اعداد کوچک ϵ به وزن یالها، مانند این است ترتیب خاصی برای یالهایی که وزن یکسان دارند اعمال کنیم. ترتیب بقیه یالها، با توجه به کوچک بودن ϵ تغییری نمی‌کند. لذا با توجه به بحثهایی که کردیم، درخت فراگیری که الگوریتم کراسکال تولید می‌کند همان درخت کمینه است.