طراحی و تحلیل الگوریتم ها

مدرس: سعدون عزیزی

s.azizi@uok.ac.ir

گروه مهندسی کامپیوتر

نیم سال دوم ۹۷-۹۶

الگوریتم‌های گراف

- الگوریتم‌های اولیه گراف
- **درخت‌های پوشای کمینه**
- کوتاهترین مسیرهای با مبدأ یکسان
- کوتاهترین مسیرها بین هر دو رأس

مقدمه

- در طراحی مدارهای الکترونیکی، معمولاً لازم است که پتانسیل پین‌های مولفه‌های مختلفی را با سیم‌بندی آنها به یکدیگر، یکی کنیم
- برای اتصال مجموعه‌ای از n پین، می‌توان از n-1 سیم استفاده کرد که هر کدام از آنها دو پین را به یکدیگر متصل می‌کنند
- از تمامی راه‌های ممکن برای انجام این کار، معمولاً روشی که از کمترین مقدار سیم استفاده می‌کند، مطلوب طراحان است
- می‌توان این مسئله را به کمک یک گراف بدون جهت همبند مدل کرد و مسئله درخت پوشای کمینه را برای آن پیدا کرد

تعریف

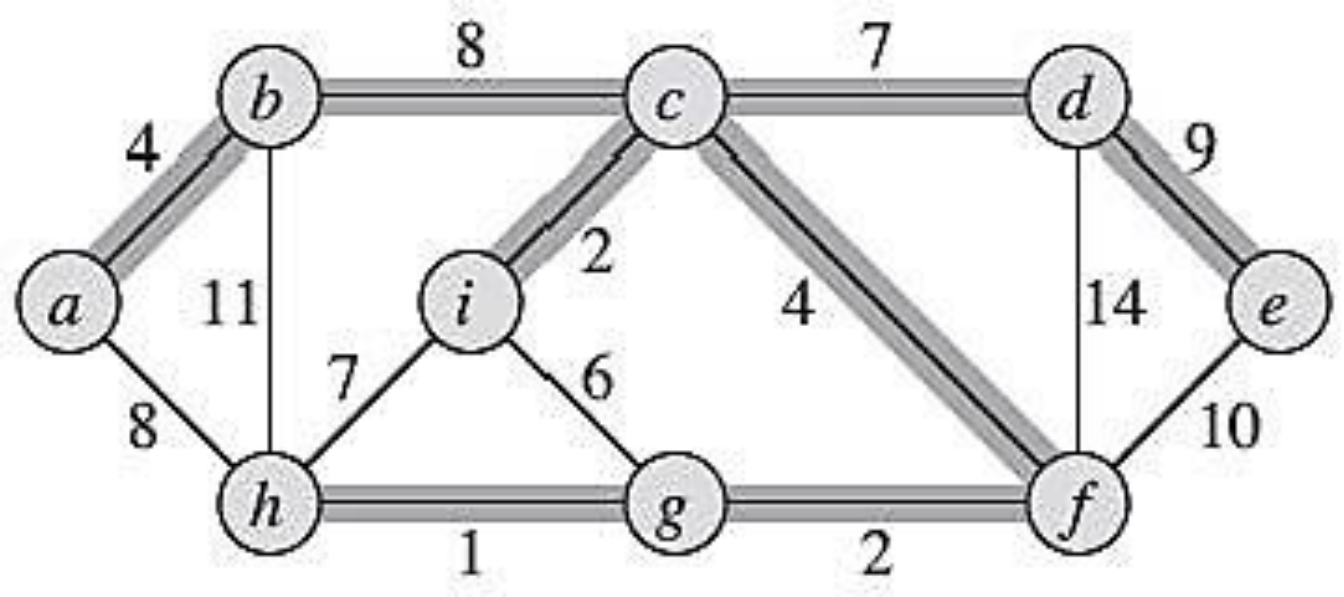
تعریف

فرض کنید که یک گراف بدون جهت و همبند $G=(V,E)$ به ما داده شده است که در آن به ازای هر یال $(u,v)\in E$ یک وزن $w(u,v)$ را داریم که هزینه آن یال را مشخص می‌کند. می‌خواهیم یک زیرمجموعه‌ی بدون دور $T\subseteq E$ را پیدا کنیم به طوری که تمام رأس‌های گراف G را بهم متصل کند و وزن کل آن کمینه شود. یعنی کمینه کردن رابطه زیر:

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

- مسئله‌ی یافتن T به مسئله‌ی درخت پوشای کمینه (Minimum Spanning Tree - MST) معروف است.

مثال



- در این بخش، دو الگوریتم برای حل مسئله‌ی درخت پوشای کمینه را بررسی می‌کنیم: الگوریتم کروسکال و الگوریتم پریم؛ که هر دو از تکنیک حریصانه برای حل مسئله استفاده می‌کنند.

استراتژی حریصانه کلی

- استراتژی حریصانه کلی: رشد دادن درخت پوشای کمینه با اضافه کردن یک یال در هر مرحله
- الگوریتم کلی یک مجموعه‌ی A از یال‌ها را نگه می‌دارد، که ثابت حلقه‌ی زیر را حفظ کند:

  قبل از هر بار تکرار، A یک زیرمجموعه از یک درخت پوشای کمینه است.

- در هر مرحله، یک یال مانند (u,v) را پیدا می‌کنیم که می‌تواند بدون نقض این ثابت حلقه به A اضافه شود، به طوری که $A \cup \{(u,v)\}$ هم زیرمجموعه‌ای از یک درخت پوشای کمینه باشد؛ به چنین یالی یک یال ایمن (safe edge) گفته می‌شود.

استراتژی حریصانه کلی (ادامه)

```
GENERIC-MST(G, w)
1  A = ∅
2  while A does not form a spanning tree
3      find an edge (u, v) that is safe for A
4      A = A ∪ {(u, v)}
5  return A
```

از ثابت حلقه به صورت زیر استفاده می‌کنیم:

- آغاز: در ابتدا چون A تهی است، پس به صورت بدیهی ثابت حلقه را حفظ می‌کند.
- ادامه: حلقه در خطوط ۲-۴ ثابت حلقه را با اضافه کردن فقط یال‌های ایمن حفظ می‌کند.
- پایان: تمام یال‌هایی که به A اضافه می‌شوند در یک درخت پوشای کمینه هستند. بنابراین، مجموعه‌ی A که در خط ۵ بازگردانده می‌شود باید یک درخت پوشای کمینه باشد

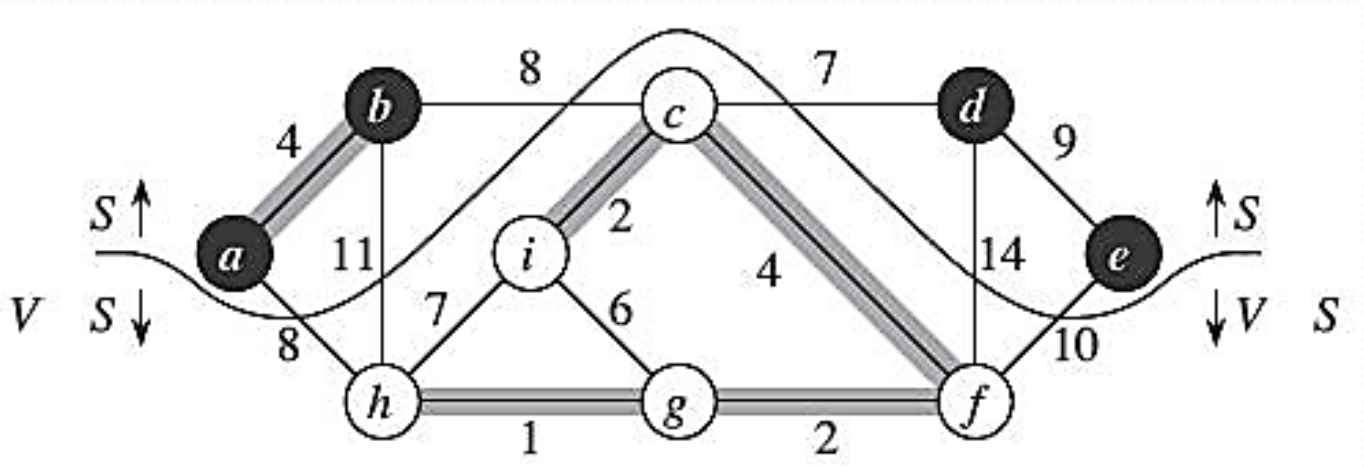
چند تعریف دیگر

- یک برش $(S,V-S)$ از یک گراف بدون جهت $G=(V,E)$، یک تقسیم‌بندی روی مجموعه رئوس G است که آن را به دو زیرمجموعه S و $V-S$ افراز می‌کند
- می‌گوییم یال $(u,v)\in E$ از برش $(S,V-S)$ عبور می‌کند اگر یکی از رأس‌های آن در S و دیگری در $V-S$ باشد
- می‌گوییم یک برش به مجموعه یال‌های A احترام می‌گذارد اگر هیچ یالی از A برش را قطع نکند
- یک یال، یال سبک عبورکننده از یک برش است اگر وزن آن میان تمام یال‌های عبورکننده از برش، کمینه باشد

مثال

* راس‌های مجموعه S با رنگ تیره و راس‌های مجموعه V-S با رنگ روشن مشخص شده‌اند

* یال‌های سایه زده شده متعلق به مجموعه A هستند



قضیه

قضیه

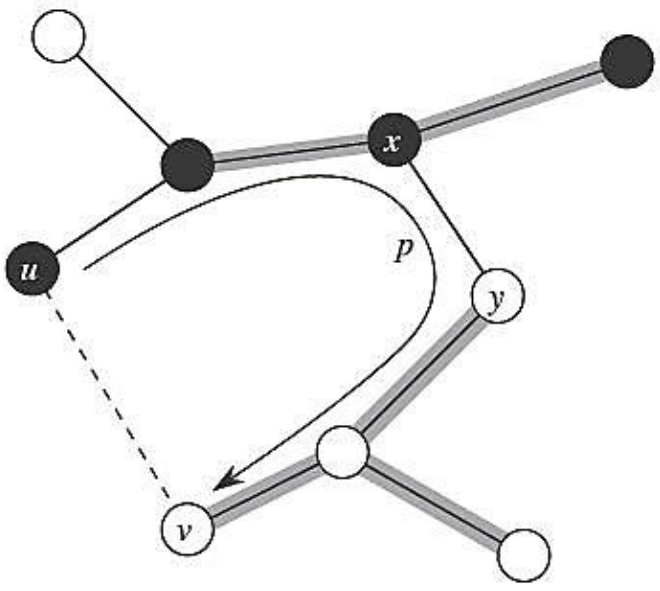
فرض کنید که $G=(V,E)$ یک گراف بدون جهت و همبند باشد که در آن به ازای هر یال $(u,v)\in E$ یک وزن $w(u,v)$ را داریم که هزینه آن یال را مشخص می‌کند. همچنین فرض کنید A یک زیرمجموعه از E باشد که یک درخت پوشای کمینه در E آن را در بر می‌گیرد و فرض کنید که $(S,V-S)$ یک برش دلخواه باشد که به A احترام می‌گذارد و (u,v) یک یال سبک عبورکننده از برش $(S,V-S)$ باشد. در این صورت، یال (u,v) برای برش $(S,V-S)$ یک یال ایمن است.

اثبات:

- فرض کنید T یک درخت پوشای کمینه باشد که A را در بر می‌گیرد؛ همچنین فرض کنید که T شامل یال (u,v) نباشد، چرا که اگر این طور باشد، اثبات تمام شده است.

قضیه (ادامه)

- حال، ما یک درخت پوشای کمینه دیگر مانند T' می‌سازیم که شامل $A \cup \{(u,v)\}$ است (اگر بتوانیم چنین کاری را انجام بدهیم، آنگاه یال (u,v) برای A ایمن خواهد بود و اثبات انجام خواهد شد)
- یال (u,v) به همراه یال‌های روی مسیر ساده‌ی p از u به v در T، یک دور را تشکیل می‌دهد



قضیه (ادامه)

- چون u و v در دو طرف مختلف برش (S,V-S) قرار دارند، حداقل یک یال در T روی مسیر p قرار دارد که از برش (S,V-S) عبور می‌کند. فرض کنید (x,y) چنین یالی باشد. یال (x,y) در A نیست، چون برش به A احترام می‌گذارد.
- از آنجایی که (x,y) روی مسیر یکتایی از u به v در T است، حذف یال (x,y) درخت T را به دو مؤلفه می‌شکند. اضافه کردن (u,v) دوباره این دو مؤلفه را به یکدیگر متصل می‌کند و یک درخت پوشای جدید $T'=T-\{(x,y)\}\cup\{(u,v)\}$ را می‌سازد.
- چون (u,v) یک یال سبک عبورکننده از (S,V-S) است و یال (x,y) هم از این برش عبور می‌کند، $w(u,v)\leq w(x,y)$. بنابراین:

$$w(T')=w(T)-w(x,y)+w(u,v)\leq w(T)$$

قضیه (ادامه)

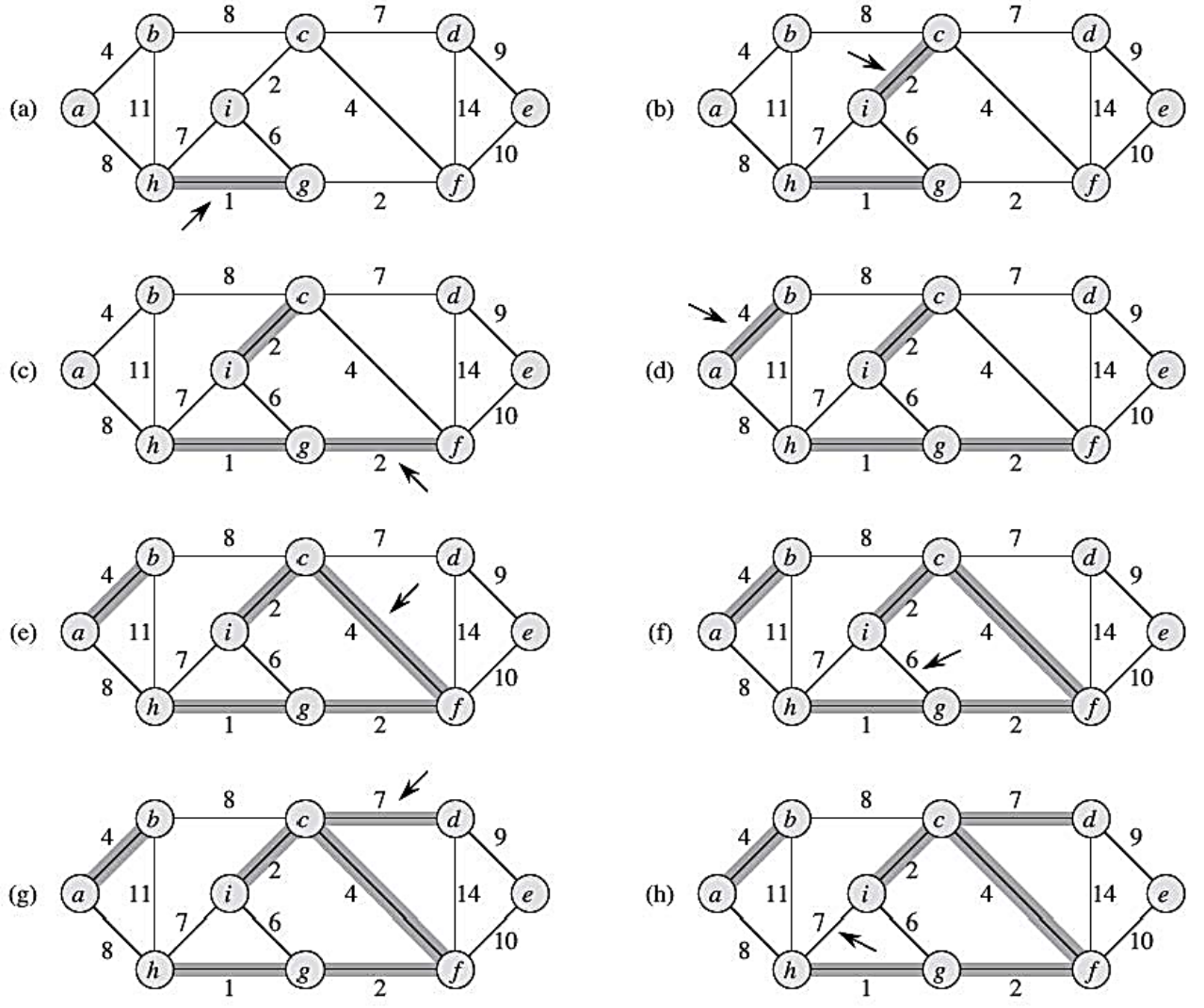
- ولی T یک درخت پوشای کمینه است، بنابراین، $w(T)\leq w(T')$. پس T' هم باید یک درخت پوشای کمینه باشد.
- کافی است نشان دهیم که (u,v) واقعاً یک یال ایمن برای A است. داریم $A\subseteq T'$، چرا که $A\subseteq T$ و $(x,y)\notin A$؛ بنابراین $A\cup\{(u,v)\}\subseteq T'$.
- در نتیجه، از آنجایی که T' یک درخت پوشای کمینه است، (u,v) برای A ایمن است.

الگوریتم کروسکال: شبه کد

- الگوریتم کروسکال (Kruskal) در هر مرحله، یال ایمن (u,v) را با یافتن یک یال با وزن کمینه از میان تمام یال‌هایی که دو درخت مختلف را در جنگل به هم متصل می‌کنند، تعیین می‌نماید.

```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```

الگوریتم کروسکال: مثال



الگوریتم کروسکال: زمان اجرا

- زمان اجرای الگوریتم کروسکال از مرتبه $O(E \lg E)$ است. (چرا؟)
- زمان اجرای این الگوریتم را می‌توان به صورت $O(E \lg V)$ هم بازنویسی کرد.

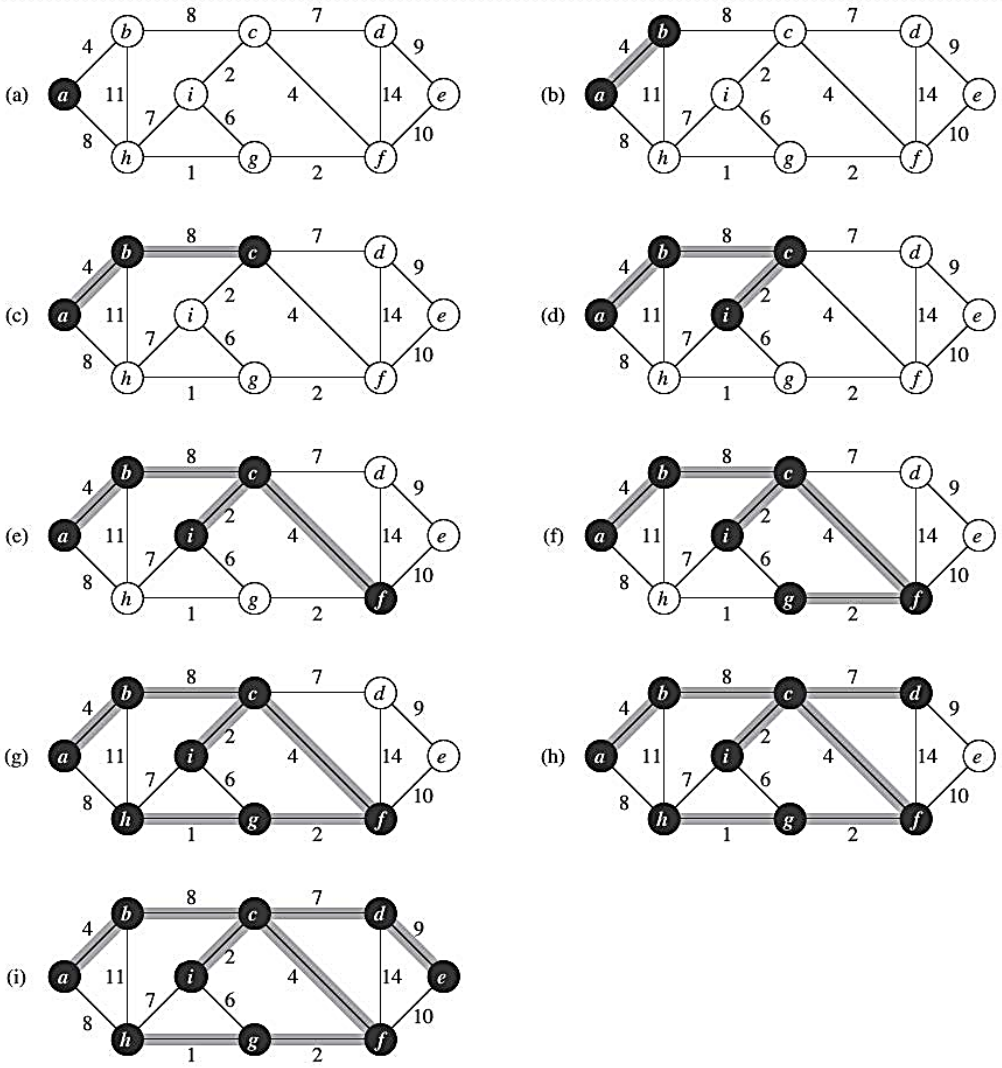
الگوریتم پریم

- الگوریتم پریم این خصوصیت را دارد که یال‌های مجموعه‌ی A همیشه تشکیل یک درخت می‌دهند.
- این درخت از یک رأس ریشه‌ی دلخواه r آغاز می‌شود و به رشد کردن ادامه می‌دهد تا زمانی که تمام رأس‌های V را بپوشاند.
- در هر مرحله، یک یال سبک به درخت A اضافه می‌شود که یکی از رأس‌های مجموعه A را به یک رأس ایزوله وصل می‌کند.
- حین اجرای الگوریتم، تمام رأس‌هایی که در درخت نیستند در یک صف اولویت کمینه‌ی Q قرار دارند، که در آن اولویت بر حسب فیلد key تعیین می‌شود.
- برای هر رأس مانند v، فیلد v.key برابر وزن کمینه یالی است که v را به یک راس درخت متصل می‌کند. اگر چنین یالی وجود نداشته باشد آنگاه v.key=∞
- فیلد v.π پدر v را در درخت مشخص می‌کند.

الگوریتم پریم: شبه کد

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

الگوریتم پریم: مثال



الگوریتم پریم: زمان اجرا

- زمان اجرای الگوریتم پریم برابر است با $O(E \lg V)$