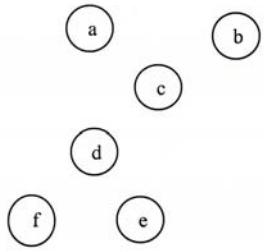
مثال:

a	b	c	d	e	f
۱	۱	۳	۳	۱	۱

c	c	d	d

شروع کار:

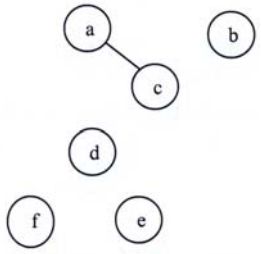


مرحله اول:

با حرکت از ابتدای رشته (که c می باشد) و با توجه به جدول، کوچکترین برگ را (a) انتخاب می‌کنیم حال از درجه c و برگ a یکی کم کرده و این کار را برای مراحل بعد نیز انجام می‌دهیم.

a	b	c	d	e	f
۱ ۰	۱	۳ ۲	۳	۱	۱

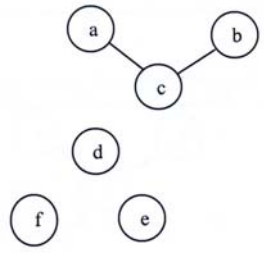
c	c	d	d



مرحله دوم:

a	b	c	d	e	f
۱ ۰	۱ ۰	۳ ۲ ۱	۳	۱	۱

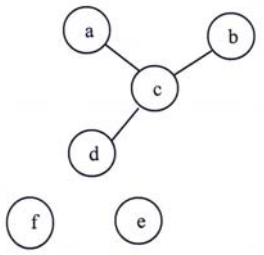
c	c	d	d



مرحله سوم:

a	b	c	d	e	f
۱ ۰	۱ ۰	۳ ۲ ۱ ۰	۳ ۲	۱	۱

c	c	d	d

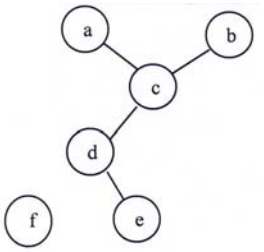


مرحله چهارم:

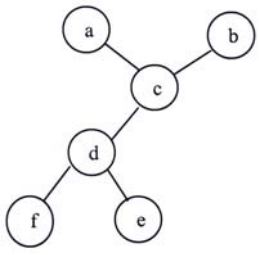
در این مرحله دو گره انتهایی را به یکدیگر وصل می‌کنیم.

a	b	c	d	e	f
۱ ۰	۱ ۰	۳ ۲ ۱ ۰	۳ ۲ ۱	۱ ۰	۱

c	c	d	d



مرحله آخر:



مسأله کوتاه‌ترین مسیر از مبداء واحد (Single Source Shortest Path)

در مسأله کوتاهترین مسیر یک گراف جهت‌دار وزن دار داده شده است. تابع وزنی $W : E \to R$ به هر یال یک مقدار حقیقی که وزن یال نیز نامیده می‌شود را نسبت می‌دهید.

وزن یک مسیر $P =< v_0, v_1, \ldots, v_k >$ عبارتست از:

$$W(P) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

کوتاهترین مسیر وزنی از u به v عبارتست از:

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{اگر مسیری از } u \text{ به } v \text{ وجود داشته باشد.} \\ \infty & \text{در غیر اینصورت} \end{cases}$$

کوتاهترین مسیر از u به v به صورت هر مسیری مانند p با وزن $W(p) = \delta(u,v)$ تعریف می‌شود.

نکته: زیرمسیرهای کوتاهترین مسیر خود کوتاهترین مسیر می‌باشند.

یال‌های با وزن منفی

چنانچه گراف $G = (V,E)$ شامل هیچ دور منفی که از s قابل دسترسی است، نباشد آنگاه کوتاهترین مسیر وزنی $\delta(s,v)$ حتی اگر مقدار آن منفی باشد خوش تعریف و بدون ابهام باقی می‌ماند اما در غیر از این حالت یعنی اگر از رأس S بتوان به یک دور با طول منفی رسید آنگاه $\delta(s,v)$ خوش تعریف نمی‌باشد و همیشه یک مسیر وزنی با مقدار کوچکتر را میتوان با آغاز از S و رسیدن به دور منفی و سپس پیمایش آن به دست آورد و در این حالت $\delta(s,v) = -\infty$ خواهد شد.

*برخی از الگوریتم‌ها مانند Dijkstra فرض را بر این بنا می‌گذارند که همه یال‌ها دارای وزن غیر منفی باشند.

* الگوریتم $Bellman-ford$ امکان می‌دهد که یال‌های گراف دارای وزن منفی نیز باشد. اما هیچ دور با طول منفی که از رأس مبداء قابل دسترسی باشد نباید در گراف وجود داشته باشد. اگر چنین دوری با طول منفی وجود داشته باشد، الگوریتم آن را آشکار و وجود آن را گزارش می‌دهد.

نکته: کوتاهترین مسیرها هرگز نباید شامل دور باشند. بنابراین بدون کاستن از کلیت مسأله فرض می‌کنیم که کوتاهترین مسیرها شامل هیچ دوری نیستند.

با توجه به اینکه هر مسیر غیر دوری در یک گراف $G = (V,E)$ شامل حداکثر $|V|$ رأس متمایز می‌باشد، همچنین شامل حداکثر $|V|-1$ یال خواهد بود، بنابراین ما توجه خود را به کوتاهترین مسیرهایی که حداکثر $|V|-1$ یال دارند معطوف می‌کنیم.

نمایش کوتاهترین مسیرها

فرض کنید $G = (V,E)$ گراف وزن‌دار و جهت‌دار با تابع وزنی $W : E \to R$ باشد به طوری که شامل هیچ دوری با طول منفی نباشد که بتواند از رأس مبدأ S قابل دسترسی باشد و بنابراین کوتاهترین مسیرها خوش تعریف هستند.

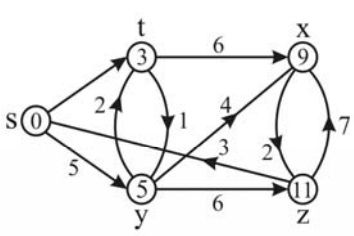
یک درخت کوتاهترین مسیرها (shortest path tree) در S، یک زیر گراف جهت‌دار، $G' = (V', E')$ است، که $V' \subseteq V$ و $E' \subseteq E$ به طوری که:

(۱) V' یک مجموعه از رئوس است که از S قابل دسترسی است. $(S \in V)$

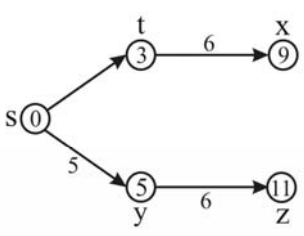
(۲) G' یک درخت ریشه دار(rooted tree) با ریشه S می‌باشد.

(۳) برای همه $V \in V'$، مسیر ساده منحصر بفرد از S به V در G'، یک کوتاهترین مسیر از S به V در گراف G خواهد بود.

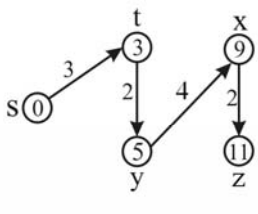
نکته: کوتاهترین مسیرها نه منحصر بفرد نمی‌باشند.



یک گراف وزن دار



یک درخت کوتاهترین مسیرها



یک درخت کوتاهترین مسیر دیگر

برای چاپ یک مسیر از الگوریتم زیر استفاده می‌شود:

```
PRINT-PATH(G,S,V)
    if v=s
      then print s
      else if π[v]=NIL
          then print "no path from"s "to" v "exists"
          else PRINT-PATH(G,S,π[v])
             print v
```

Relaxation

الگوریتم‌هایی که در این قسمت آمده‌اند از تکنیک relaxation استفاده می‌کنند. برای هر رأس، صفت $d[v]$ یک تخمین کوتاهترین مسیر می‌باشد. الگوریتم زیر مقداردهی‌های اولیه مناسب را در زمان $\theta(|V|)$ انجام می‌دهد.

فرایند relax کردن یال (u,v) یعنی اینکه آیا می‌توان مقدار $d[v]$ را با طول مسیری که مقدار آن با رابطه $d[u]+w(u,v)$ داده شده است، بهنگام و بهتر نماییم. هر الگوریتمی در این فصل، ابتدا رویه INITIALIZE-SINGLE SOURCE را فراخوانی و سپس یال‌ها را به طور تکراری relax می‌کند.

تفاوت الگوریتم‌هایی که در اینجا مطرح می‌شود در ترتیب و تعداد بارهایی است که یال‌ها را relax می‌کنند، است.

در الگوریتم کوتاه‌ترین مسیر از یک رأس واحد دایکسترا برای DAG، هر یال دقیقاً یک بار relax می‌شود.

در الگوریتم Bellman-ford هر یال چندین بار relax می‌شود.

برخی از خواص کوتاهترین مسیرها و عمل relaxation عبارتند از:

۱) نامساوی مثلثی (Triangle inequality)

$$\forall (u,v) \in E : \delta(s,v) \le (s,u) + w(u,v)$$

۲) خاصیت کران بالا (upper bound)

خاصیت $d[v]$:

$$\forall v \in V : \delta(s,v) \le d[v]$$

چنانچه یکبار $d[v]$ برابر $\delta(s,v)$ شود، دیگر تغییر نمی‌کند.

۳) خاصیت عدم وجود مسیر (no-path)

اگر هیچ مسیری از s به v وجود نداشته باشد آنگاه همیشه داریم:

$$d[v] = \delta(s,v) = \infty$$

۴) خاصیت همگرایی (convergence)

اگر مسیر $s \longrightarrow u \rightarrow v$ برای برخی u و vها در V، یک کوتاهترین مسیر بوده و اگر قبل از آغاز هرگونه relax کردن یال (u,v)، شرط $d[u] = \delta(s,w)$ برقرار باشد آنگاه $d[v] = \delta(s,v)$ بعد از آن همواره برقرار خواهد بود.

۵) خاصیت path-relaxation

اگر $P =< v_0, v_1, \ldots, v_k >$ کوتاهترین مسیر از $s = v_0$ به v_k باشد و یال‌ها P در ترتیب $(v_{k-1}, v_k), \ldots, (v_1, v_2), (v_0, v_1)$، relax شده باشند، آنگاه:

$$d[v_k] = \delta(s, v_k)$$

این خاصیت صرف‌نظر از هرگونه مرحله relaxation دیگری که انجام شود برقرار است حتی اگر آنها با relaxationهای یال‌های p مخلوط شوند.

۶) خاصیت Predecessor-subgraph

یکبار که برای همه vها در V، برابر $\delta(s,v)$ شد، آنگاه گراف پیشین یا $G_\pi = (V_\pi, E_\pi)$ یک درخت کوتاهترین مسیرها باریشه s خواهد بود.

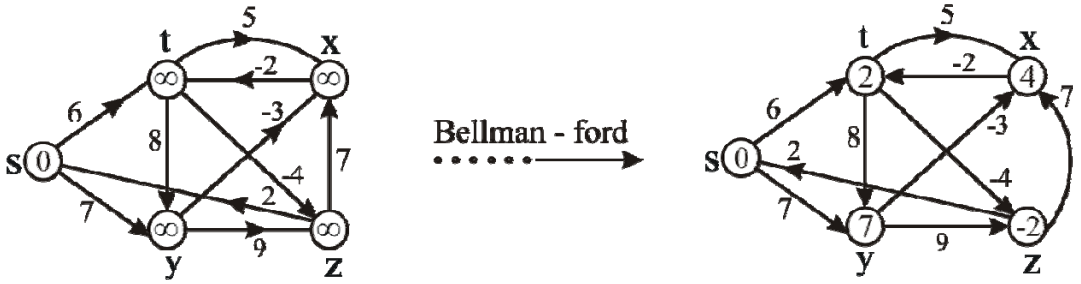
الگوریتم Bellman-Ford

گراف جهت‌دار وزن دار $G=(V,E)$ داده شده است. تابع وزنی W به صورت $W:E\to R$ و رأس مبدا S داده شده است. اگر گراف دارای دوری با طول منفی باشد که از مبداء S بتوان به آن رسید، الگوریتم مقدار false باز می گرداند و در این صورت هیچ راه حلی وجود ندارد.

اما اگر گراف دارای دوری با طول منفی نباشد که از s قابل دسترسی باشد، آنگاه الگوریتم True برمی‌گرداند و به طور پیوسته یال‌ها را relax میکند تا تخمین را برای هر رأس V کاهش دهد تا $d[v]=\delta(s,v)$ شود.

```
BELLMAN-FORD(G,W,S)
    INITIALIZE-SINGLE-SOURCE(G,S)
    for i ← ۱ to |V[G]|
        do for each edge (u,v) ∈ E[G]
            do RELAX (u,v,w)
    for each edge (u,v) ∈ E[G]
        do if d[v]>d[u]+w(u,v)
            then return FALSE
     Return TRUE
```

زمان اجرای این الگوریتم برابر $O(|V||E|)$ می‌باشد.



در مثال بالا، یال‌ها به ترتیب زیر relax می شوند:

$$(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$$

نکته: فرض کنید یک گراف جهت‌دار وزن دار با مبداء s و تابع وزنی w داده شده باشد و فرض کنید این گراف شامل هیچ دوری با وزن منفی که از s قابل دسترسی باشد نباشد. با $|V|-1$ بار تکرار حلقه‌های for اول و دوم الگوریتم فوق برای همه رئوسی که از s قابل دسترسی باشند خواهیم داشت:

$$d[v]=\delta(s,v)$$

نکته: فرض کنید گراف $G=(V,E)$ یک گراف وزن دار جهت‌دار با رأس مبدا و تابع وزنی W باشد در این صورت برای هر رأس v مسیری از s به v وجود دارد اگر و تنها اگر الگوریتم بلمن- فورد هنگامی که روی G اجرا می شود با $d[v]<\infty$ خاتمه یابد.

یافتن طول کوتاهترین مسیرها از یک مبداء واحد در DAG

مثال: گراف زیر را در نظر بگیرید: