گراف ها

گراف ها

❖ هر گراف G شامل دو مجموعه V وE است :

V : مجموعه محدود و غیرتهی از رئوس است

E : مجموعه ای محدود و احتمالا تهی از لبه ها می باشد.

V(G) وE(G) : مجموعه رئوس و لبه های گراف G را نمایش می دهند.

برای نمایش گراف هم می توانیم بنویسیم

G=(V ،E)

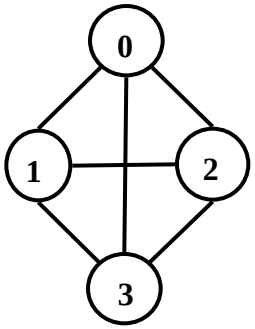
گراف ها

❖ در یک گراف بدون جهت، زوج رئوس ، زوج مرتب نیستند ، بنابراین زوج های (v_0, v_1) و (v_1, v_0) با هم یکسانند.

❖ در یک گراف جهت دار هر لبه با زوج مرتب $< v_0, v_1 >$ نمایش داده می شود. v_0 ابتدای لبه و v_1 انتها هستند. بنابراین $< v_0, v_1 >$ و $< v_1, v_0 >$ دو لبه متفاوت را نمایش می دهند.

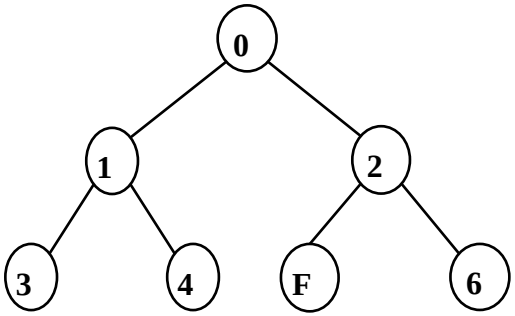
گراف ها

- برای گراف بدون جهت ، لبه ها به صورت خطوط یا منحنی نمایش داده می شوند.
- برای گراف های جهت دار لبه ها به صورت فلش هایی که از انتها به ابتدا رسم شده است ، ارایه می گردند.

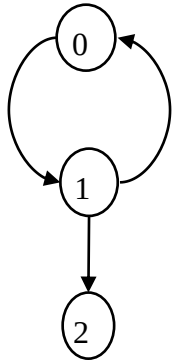


G_1

بدون جهت



G_2



G_3

جهت دار

محدودیت گراف ها

محدودیت های زیر بر روی گراف ها اعمال می شود :

- یک گراف فاقد لبه ای از یک راس مانند i ، به خودش می باشد. این مطلب بدین معنی است که لبه (v_i , v_i) غیر معتبر می باشد.
- یک گراف فاقد رویداد چندگانه از یک لبه می باشد.

- ***گراف کامل :*** گراف کامل گرافی است که دارای حداکثر تعداد لبه باشد.

گراف ها

❖ برای یک گراف بدون جهت با n راس ، حداکثر تعداد لبه ها ، تعداد متمایز و غیرمرتب زوج های (v_i , v_j) ، $i \neq j$ می باشد. این تعداد برابر است با :

$$n(n-1) / 2$$

❖ اگر گراف جهت داری با n گره وجود داشته باشد، بیشترین تعداد لبه های آن برابر است با :

$$n(n-1)$$

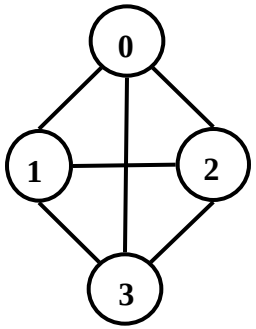
گراف ها

❖ اگر (v_0, v_1) یک لبه در گراف بدون جهت باشد، می گوییم رئوس v_0 و v_1 دو راس مجاور و لبه (v_i, v_j) یک لبه متلاقی روی v_0 و v_1 است .

❖ یک *زیر گراف* G ، گرافی است مانند G' به نحوی که $V(G') \subseteq V(G)$ و $E(G') \subseteq E(G)$ باشد.

گراف ها

- طول یک مسیر تعداد لبه های موجود در آن است.
- مسیر ساده ، مسیری است که همه رئوس آن احتمالا به جز اولی و آخری مجزا باشند.
- حلقه یا سیکل ، یک مسیرساده است که اولین و آخرین راس آن یکی باشد.



G_1

برای مثال 0،2،1،0 یک حلقه در G_1 است.

گراف ها

❖ در گراف بدون جهت مانند G ، دو راس v_0 و v_1 را متصل می گویند، اگر مسیری در G از v_0 به v_1 وجود داشته باشد.

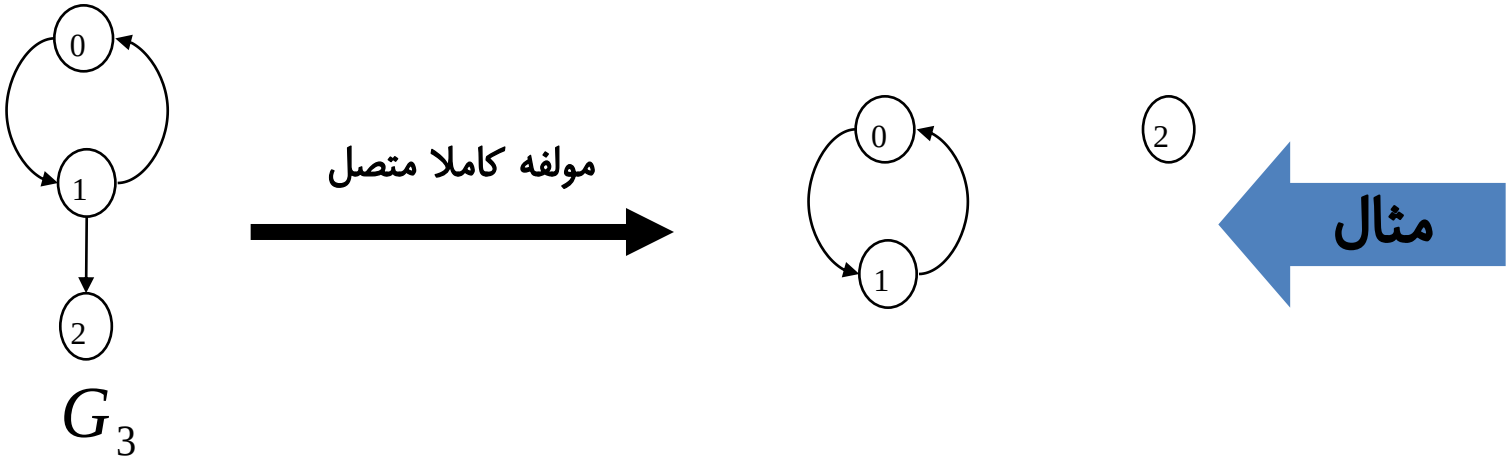
❖ یک گراف بدون جهت را متصل می نامیم اگر برای هر زوج راس v_i, v_j در V(G) مسیری از v_i به v_j در G وجود داشته باشد.،

❖ یک مولفه اتصال یا به طور ساده تر یک مولفه ، در گراف بدون جهت ، بزرگترین زیرگراف متصل آن است.

گراف ها

❖ یک گراف جهت دار کاملا متصل نامیده می شود ، اگر برای هر زوج از رئوس v_i, v_j در V(G) ، مسیری جهت دار از v_i به v_j و همچنین از v_j به v_i وجود داشته باشد.

❖ یک مولفه کاملا متصل ، بزرگترین زیرگرافی است که کاملا متصل باشد.



گراف ها

❖ درجه یک راس تعداد لبه های متلاقی با آن راس است .

اگر در گراف G با n راس ، d_i درجه راس i و e تعداد لبه ها باشد ، به آسانی می توان دید که تعداد لبه ها برابر است با :

$$e = (\sum_{0}^{n-1} d_i) / 2$$

نمایش گراف ها

نمایش گراف ها به سه صورت است :

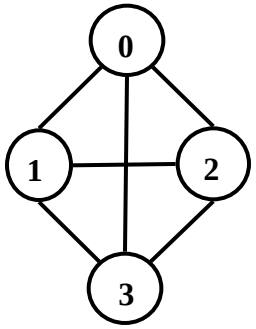
- ماتریس مجاورتی
- لیست های مجاورتی
- لیست های چندگانه

ماتریس مجاورتی

❖ فرض کنید G=(V,E) یک گراف با n راس باشد ، $n \geq 1$، ماتریس مجاورتی گراف G یک آرایه دوبعدی n × n به نام adj_mat می باشد. اگر لبه (v_i, v_j) (برای گراف جهت دار $<v_i, v_j>$) در E(G) باشد ، آنگاه adj_mat[i] [j]= 0 خواهد بود.

❖ ماتریس مجاورتی برای یک گراف بدون جهت متقارن است زیرا لبه (v_i, v_j) در E(G) خواهد بود ، اگر و تنها اگر لبه(v_i, v_j)نیز در E(G) باشد

ماتریس مجاورتی



G_1

ماتریس مجاورتی

	0	1	2	3
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0

G_1

ماتریس مجاورتی

❖ برای گراف بدون جهت ، درجه هر راس مانند i مجموع عناصر سطری آن است :

$$\sum_{j=0}^{n-1} adj\_mat[i][j]$$

❖ برای یک گراف جهت دار ، مجموع سطری ، درجه خارجه و مجموع ستونی ، درجه وارده خواهد بود.

لیست مجاورتی

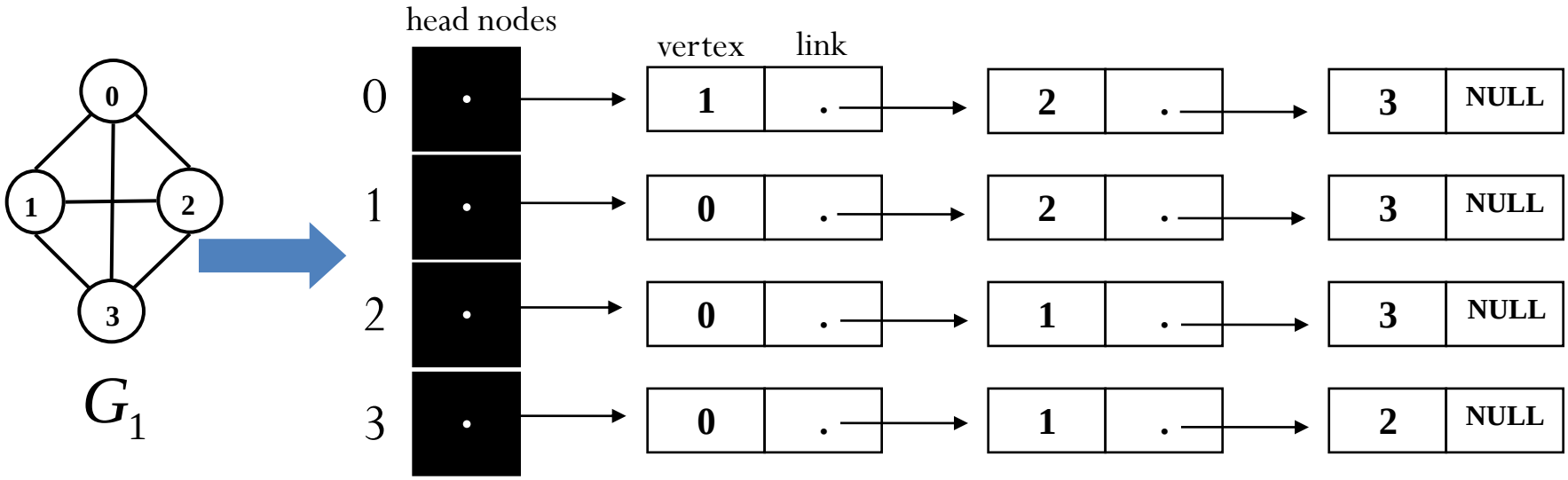
❖ با این نمایش ، n سطر ماتریس مجاورتی در n لیست پیوندی قرار می گیرد. برای هرراس از گراف G ، یک لیست وجود دارد.

هر گره حداقل دو فیلد دارد : راس و اتصال

در هر لیست مشخصی مانند i ، گره های لیست حاوی رئوس مجاور از راس i می باشند.

در یک گراف بدون جهت با n راس و e لبه ، n گره head و 2e گره لیست دارد. هر گره لیست دو فیلد لازم دارد.

لیست مجاورتی



لیست مجاورتی

- درجه هر راس در یک گراف بدون جهت را می توان به سادگی با شمارش تعداد گره های آن در لیست مجاورتی تعیین کرد.
- اگر تعداد رئوس گراف G برابر با n باشد، تعداد کل لبه ها، در زمان $O(n + e)$ تعیین می شود.

اعمال ابتدایی گراف

❖ با توجه به گراف بدون جهت G=(V,E) و راس v از V(G) می خواهیم رئوسی از G را که از v قابل دسترسی هستند، به دست آوریم(یعنی همه رئوس متصل به v) . برای این کار دو روش وجود دارد :

❁ جستجوی عمقی : روش عمقی تا حدودی شبیه پیمایش preorder یک درخت است.

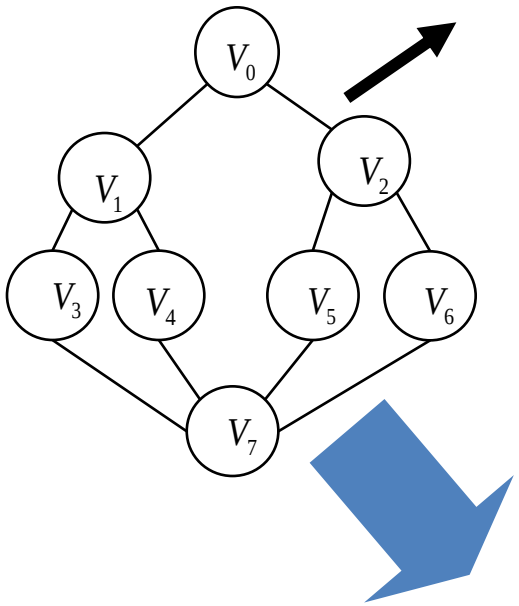
❁ جستجوی ردیفی : این روش تا حدودی پیمایش ترتیب سطحی را مجسم می کند.

در پیمایش های عمقی و ردیفی ، فرض می کنیم که برای نمایش گراف ها از لیست مجاورتی پیوندی استفاده شده است.

جستجوی عمقی

❖ در آغاز راس v را ملاقات می کنیم. بعد راسی مانند w را که قبلا ملاقات نشده و مجاور به v است را انتخاب کرده و روش جستجوی عمقی را با w دنبال می کنیم. موقعیت جاری راس v در لیست مجاورتی با قرار دادن آن در یک پشته صورت می گیرد. در نهایت ، جستجو به راسی مانند u خواهد رسید که فاقد هر گونه راس غیرملاقات شده در لیست مجاورتی باشد. در این مرحله راسی از پشته انتخاب و و حذف شده و فرآیند فوق به همین صورت ادامه پیدا می کند. بر اساس این روش ، رئوس ملاقات شده، خارج شده و رئوس ملاقات نشده ، داخل پشته قرار می گیرند. جستجو زمانی پایان می پذیرد که پشته تهی باشد.

جستجوی عمقی

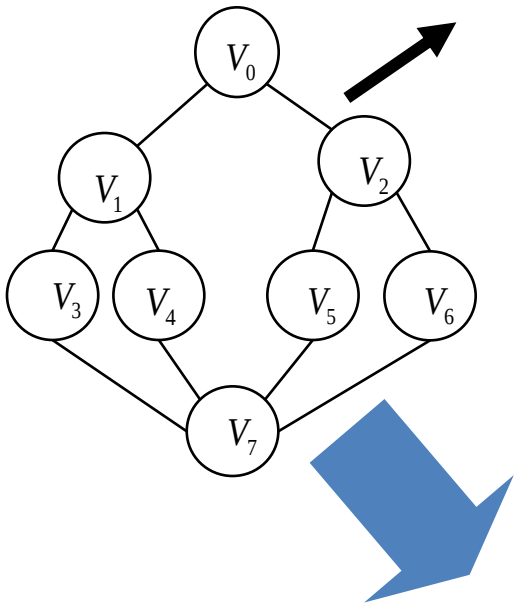


- اگر روش جستجوی عمقی را از راس v_0 آغاز کنیم ، رئوس گراف G به ترتیب **v0, v1, v3, v7, v4, v5, v2, v6** ملاقات می شوند.

جستجوی ردیفی

❖ پیمایش را با راس v شروع نموده ، پس از ملاقات راس مزبور ، آنرا علامت گذاری می کنیم. اول هر یک از رئوس مجاور به راس v را در لیست مجاورتی ملاقات می کنیم. زمانی که همه رئوس مجاور با راس v را ملاقات کردیم ، تمام رئوس ملاقات نشده که مجاور با اولین راس مجاور با v در لیست مجاورتی است را ملاقات می کنیم. برای انجام این کار مادامیکه هر راس ملاقات می شود ، آنرا در یک صف قرار می دهیم. هنگامی که لیست مجاورتی تمام شد ، راسی را از صف حذف و با تست هر یک از رئوس در لیست مجاورتی این فرآیند را ادامه می دهیم. رئوس ملاقات نشده ، ملاقات و سپس در صف قرار می گیرند. رئوس ملاقات شده نادیده گرفته می شوند. جستجو هنگامی که صف تهی گردد ، خاتمه می یابد.

جستجوی ردیفی



v0, v1, v2, v3, v4, v5, v6, v7

درخت های پوشا

❖ چنانچه گراف G متصل باشد ، پیمایش های جستجوی عمقی یا جستجوی ردیفی ، تمام رئوس گراف G را ملاقات می کنند.

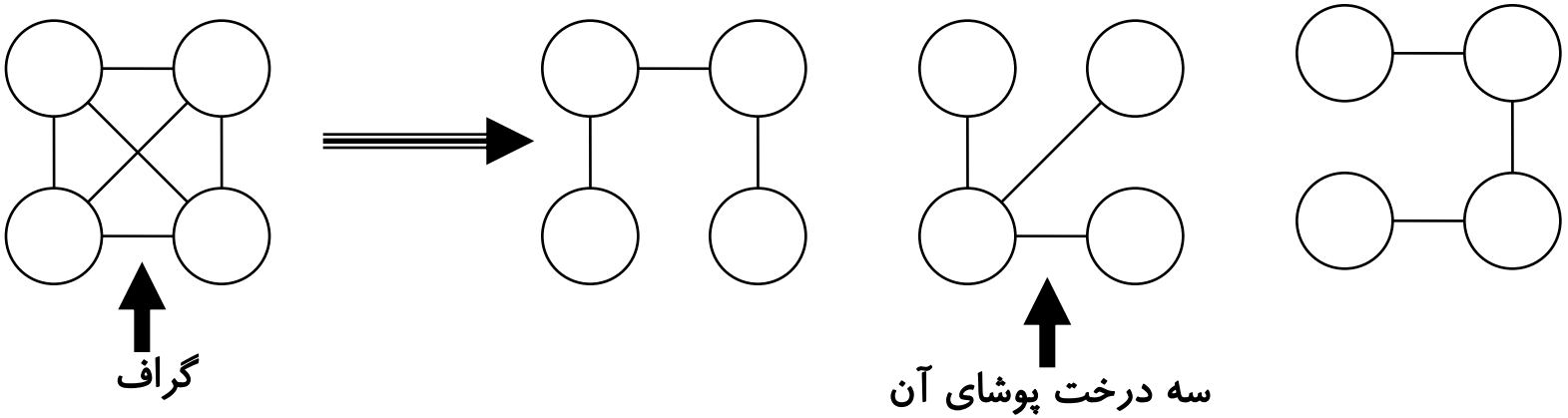
❖ در این حالت با اعمال هر یک از پیمایش ها لبه های گراف G به دو قسمت تقسیم می شوند :

- T(برای لبه های درخت) : مجموعه لبه های به کار رفته یا پیموده شده در جستجو می باشد.
- N (برای لبه های غیر درخت) : مجموعه لبه های باقی مانده می باشد.

درخت های پوشا

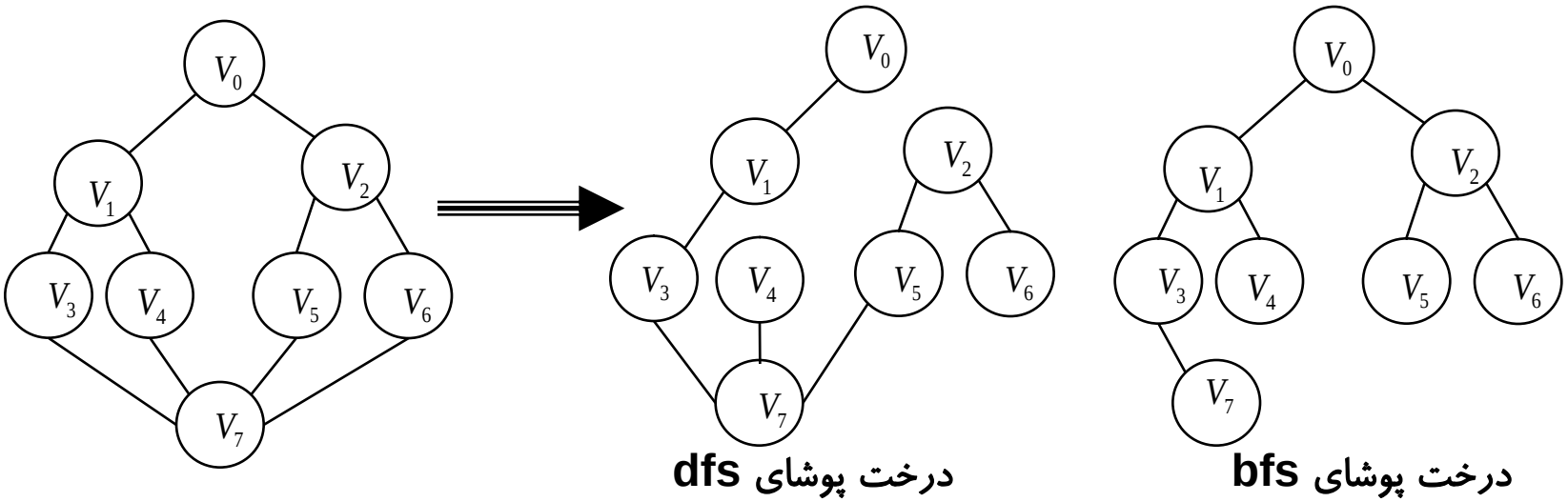
❖ لبه های T ، یک درخت که شامل تمام رئوس گراف G می باشد را تشکیل می دهند .

تعریف درخت پوشا : درختی که تعدادی از لبه ها و تمام رئوس G را در بر دارد ، درخت پوشا نامیده می شود :



درخت های پوشا

- درخت پوشایی که از فراخوانی dfs به دست می آید را درخت پوشای عمقی می نامند.
- چنانچه از روش bfs استفاده شود ، درخت پوشای حاصل را درخت پوشای ردیفی می نامند.



درخت پوشای **dfs** درخت پوشای **bfs**

درخت های پوشا

❖ مقصود از زیر گراف حداقل ، یعنی زیرگرافی که تعداد لبه هایش کمترین باشد.

➢ هر گراف متصل با nراس ، بایستی حداقل n-1 لبه داشته باشد و همه گراف ها متصل با n-1 لبه ، درخت هستند.

➢ درخت پوشا دارای n-1 لبه می باشد.

➢ ایجاد زیرگراف های حداقل ، کاربردهای متعددی در طراحی شبکه های ارتباطی دارد.

مثال : اگر رئوس گراف G نماینده شهرها و لبه ها معرف جاده های ارتباطی بین شهرها بلشد ، آنگاه حداقل تعداد خطوط مورد نیاز برای اتصال n شهر به یکدیگر n-1 خواهد بود.

درخت های پوشا با حداقل هزینه

❖ هزینه یک درخت پوشای یک گراف جهت دار دارای وزن ، مجموع هزینه های(وزن های) لبه ها در درخت پوشا می باشد.

❖ درخت پوشای حداقل هزینه ، درخت پوشایی است که دارای کمترین هزینه باشد.

❖ برای به دست آوردن درخت پوشای حداقل هزینه یک گراف جهت دارمتصل می توان از سه الگوریتم متفاوت استفاده نمود :

الگوریتم راشال ، الگوریتم پریم ، الگوریتم سولین

هر سه روش از یک طراحی الگوریتمی به نام خط مشی greedy استفاده می کنند.

درخت های پوشا با حداقل هزینه

❖ برای درخت های پوشا از ملاک کمترین هزینه استفاده می شود. روش ما باید دارای شرایط زیر باشد :

1) باید فقط از لبه های داخل گراف استفاده کنیم.

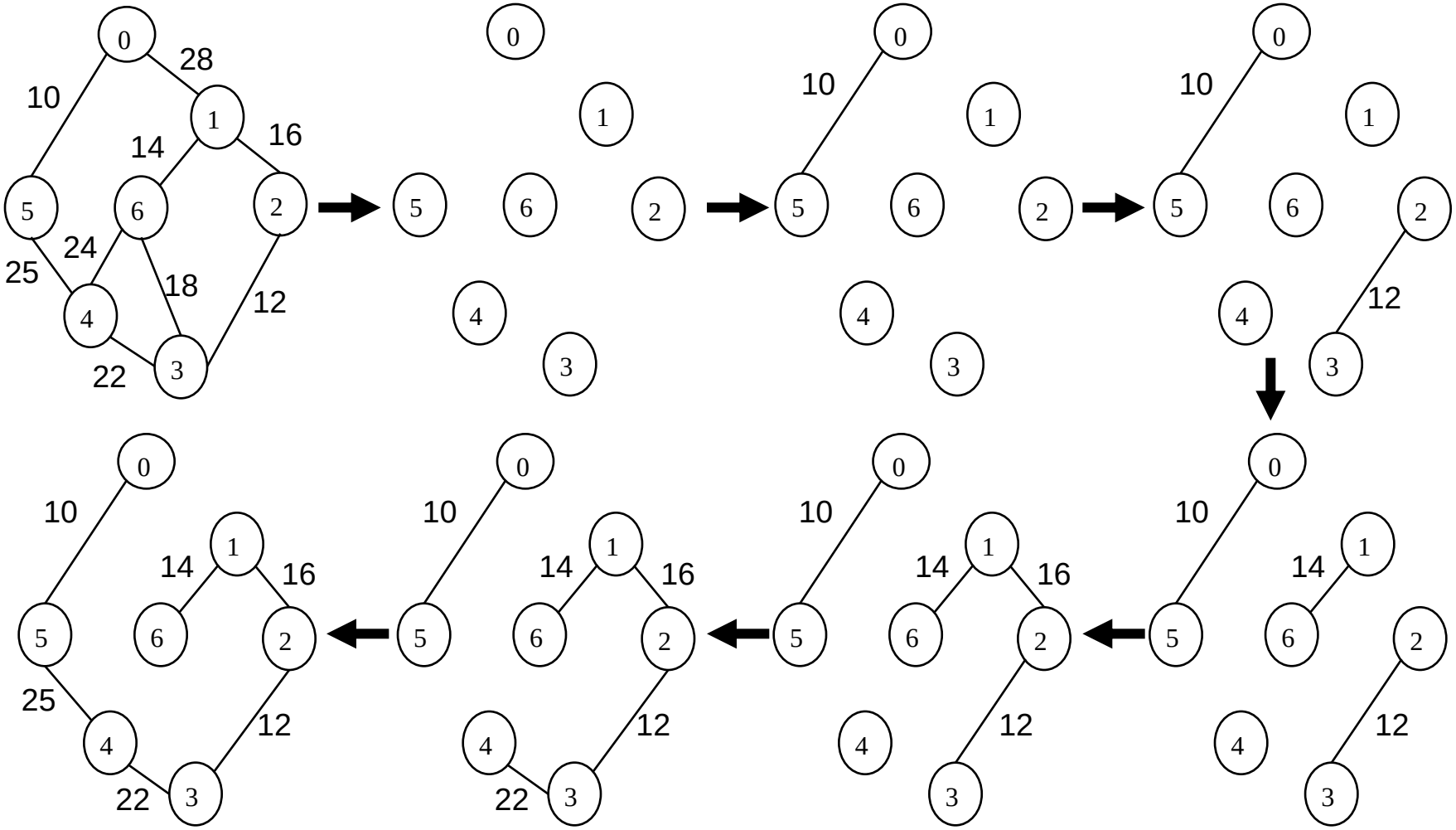
2) باید دقیقا از n-1 لبه استفاده کنیم.

3) نباید از لبه هایی که ایجاد یک حلقه می کنند ، استفاده کنیم.

الگوریتم راشال

❖ در این روش ، درخت پوشای با کمترین هزینه T ، لبه به لبه ساخته می شود. لبه های مورد استفاده در T ، به ترتیب صعودی وزن ها می باشد. یک لبه در T خواهد بود، اگر با لبه های قبل که در T بوده اند ، تشکیل یک حلقه ندهد چون G متصل است و دارای $n > 0$ راس است ، دقیقا $n - 1$ لبه برای T انتخاب می شود.

الگوریتم راشال



الگوریتم پریم

❖ الگوریتم پریم مانند الگوریتم راشال ، در هر زمان یک لبه از درخت پوشای حداقل هزینه را می سازد. هر چند در هر مرحله الگوریتم ، مجموعه لبه ها انتخاب شده یک درخت را تشکیل می دهند . در مقابل ، مجموعه لبه های انتخاب شده در الگوریتم راشال در هر مرحله یک جنگل را تشکیل می دهند. الگوریتم پریم با یک درخت مانند T ، که تنها شامل یک راس است ، شروع می کند. این می تواند هر یک از رئوس در گراف اصلی باشد. سپس یک لبه با کمترین هزینه مانند (u,v) به T اضافه می شود به نحوی که $T \bigcup \{(u,v)\}$ نیز خود یک درخت می باشد. این عمل را تا زمانی که T شامل $n-1$ لبه باشد ، ادامه می دهیم.

الگوریتم پریم