فصل ششم: گراف

بخش ها

- آشنایی با گراف
- نمایش گراف
- جستجوی گراف
- درخت پوشا

مفهوم گراف

هر گراف G شامل دو مجموعه V و E است :

- V : مجموعه محدود و غیرتهی از رئوس است.
- E : مجموعه ای محدود و احتمالا غیرتهی از لبه ها می باشد.
- V(G) و E(G) : مجموعه رئوس و لبه های گراف G را نمایش می دهند.
- برای نمایش گراف هم می توانیم بنویسیم G=(V,E)

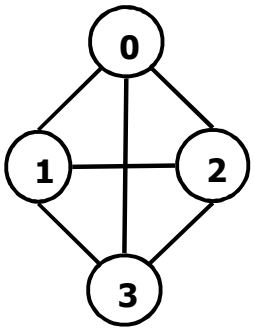
مفهوم گراف

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

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

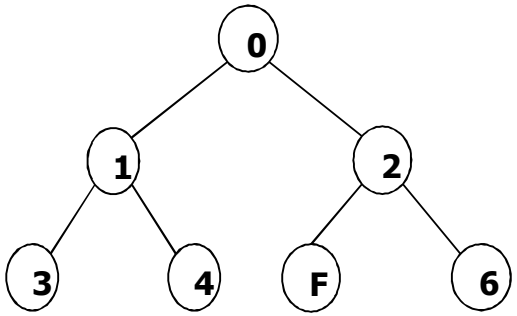
ترسیم گراف

- 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.

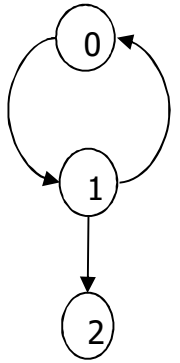


G_1

بدون جهت



G_2



G_3

جهت دار

اصطلاحات گراف

- 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.
- برای یک گراف بدون جهت با nراس ، حداکثر تعداد لبه ها ، تعداد متمایز و غیرمرتب زوج های (v_i,v_j)، $i \neq j$ می باشد. این تعداد برابر است با : $n(n-1) / 2$
- اگرگراف جهت داری با nگره وجود داشته باشد، بیشترین تعداد لبه های آن برابر است با :$n(n-1)$

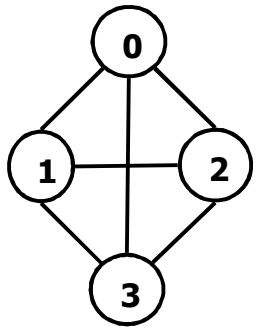
گراف کامل : گراف کامل گرافی است که دارای حداکثر تعداد لبه باشد.

اصطلاحات گراف

- اگر (v_0,v_1) یک لبه در گراف بدون جهت باشد ، می گوییم رئوس v_0و v_1 دو راس مجاور و لبه (v_i,v_j) یک لبه متلاقی روی v_iوv_j است.
- یک زیر گراف G، گرافی است مانند G’ به نحوی که $V(G')\subseteq V(G)$ و $E(G')\subseteq E(G)$ باشد.
- اگر از گره v_i با عبور از تعدادی یال و گره میانی به گره v_j در گراف برسیم، گوییم بین v_iوv_j مسیر با طول n وجود دارد که n تعداد یال هایی است که در مسیر پیموده می شوند.
- به عبارت دیگر طول یک مسیر تعداد لبه های موجود در آن است.

اصطلاحات گراف

- **مسیر ساده:** مسیری است که همه رئوس آن به جز اولی و آخری مجزا باشند.
- **حلقه**(Cycle): یک مسیرساده است که اولین و آخرین راس آن یکی باشد.
- برای مثال 0،2،1،0 یک حلقه در G_1 است.



G_1

اصطلاحات گراف

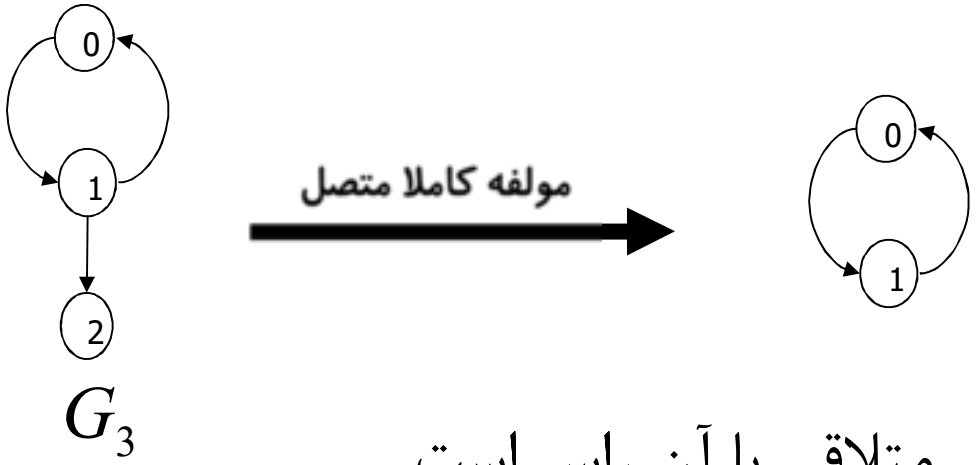
- در گراف بدون جهت مانند G ، دو راس v_0 و v_1 را **متصل** می گویند، اگر مسیری در G از v_0 به v_1 وجود داشته باشد.
- یک گراف بدون جهت را **متصل** می نامیم اگر برای هر زوج راس v_i و v_j در $V(G)$ ، مسیری از v_i به v_j در G وجود داشته باشد.
- یک **مولفه اتصال** یا به طور ساده تر یک مولفه، در گراف بدون جهت، بزرگترین زیرگراف متصل آن است.
- یک گراف جهت دار **کاملا متصل** نامیده می شود ، اگر برای هر زوج از رئوس v_i و 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.

اصطلاحات گراف

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

اصطلاحات گراف

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



G_3

- **درجه 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

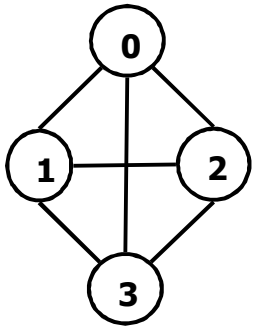
- نمایش گراف ها به سه صورت است :
  - ماتریس مجاورتی
  - لیست های مجاورتی
  - لیست های چندگانه

نمایش گراف با ماتریس مجاورتی

- فرض کنید $G=(V,E)$ یک گراف با n راس باشد که $n \geq 1$، ماتریس مجاورتی گراف G یک آرایه دوبعدی $n \times n$ 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]= 1 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_j,v_i) نیز در $E(G)$ باشد

نمایش گراف با ماتریس مجاورتی

- مثال



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 G_1

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

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

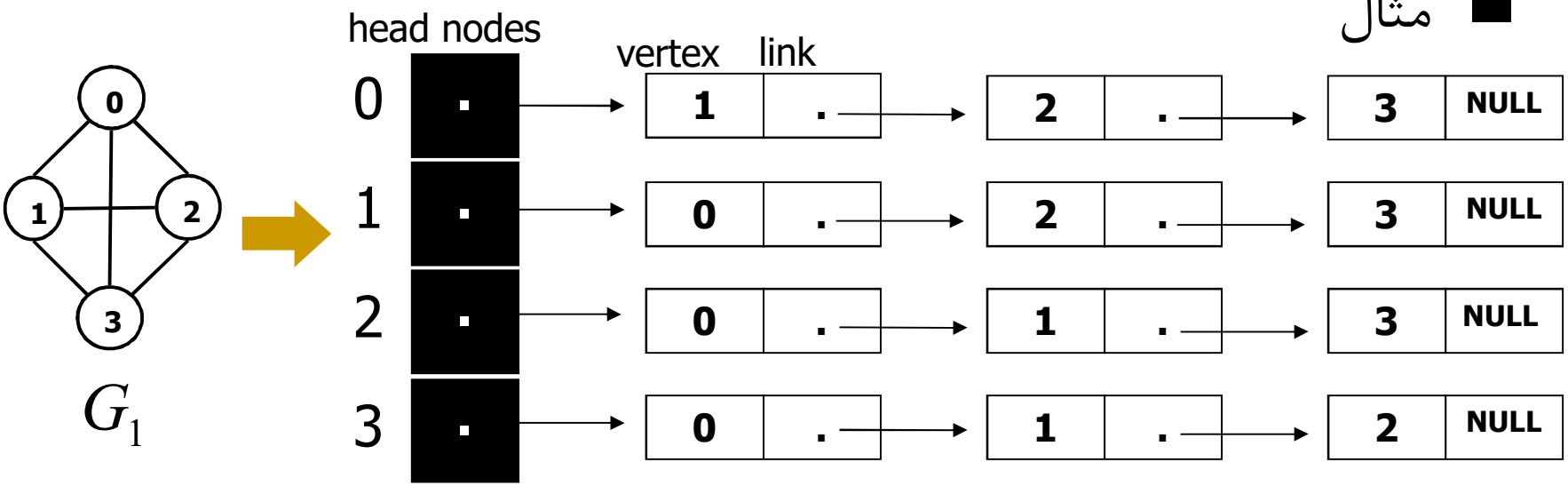
- برای یک گراف جهت دار ، مجموع سطری ، درجه خارجه و مجموع ستونی ، درجه وارده خواهد بود.

نمایش گراف با 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.

نمایش گراف با لیست مجاورتی

- مثال



- 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.

گراف با یال وزن دار (لبه های وزنی)

- گرافی که لبه هایش دارای وزن باشد ، **شبکه** نامیده می شود.
- در ماتریس مجاورتی ، کمیت ۱ که نشان دهنده وجود یک لبه است را با وزن یک لبه تعویض می کنیم.
- در لیست های مجاورتی و لیست های مجاورتی چندگانه ، فیلد weight به ساختار گره اضافه می شود.

عملیات ابتدایی بر روی گراف

- با توجه به گراف بدون جهت G(V,E) و راس v از V(G) میخواهیم 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 (سطحی)** : این روش تا حدودی پیمایش ترتیب سطحی درخت را مجسم می کند.

جستجوی عمقی Depth-First Search (DFS)

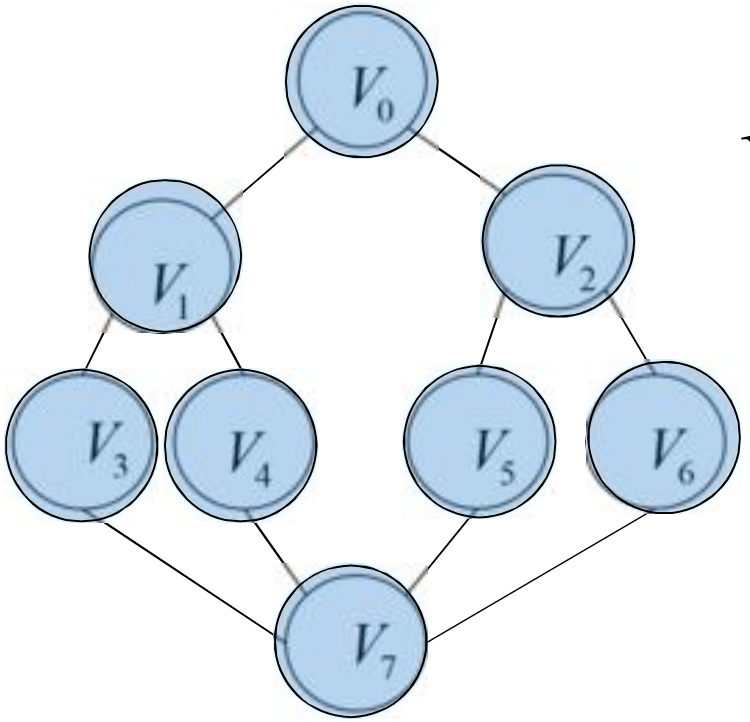
- در آغاز راس **V** را ملاقات می کنیم.
- بعد راسی مانند **W** را که قبلا ملاقات نشده و مجاور به **V** است را انتخاب کرده و روش جستجوی عمقی را با **W** دنبال می کنیم.
- موقعیت جاری راس **V** در لیست مجاورتی با قرار دادن آن در یک پشته صورت می گیرد.
- در نهایت ، جستجو به راسی مانند **U** خواهد رسید که فاقد 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 تا زمانی که پشته خالی نشده ادامه پیدا می کند.

جستجوی عمقی Depth-First Search (DFS)

- 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.
- اگر G توسط ماتریس مجاورتی نمایش داده شود ، زمان لازم برای تعیین همه رئوس مجاور به v ، $O(n)$ است. از آنجا که حداکثر n راس مشاهده می شود، کل زمان $O(n^2)$ خواهد شد.

جستجوی عمقی Depth-First Search (DFS)

- مثال: اگر روش جستجوی عمقی را از راس V_0 آغاز کنیم ، رئوس گراف G به ترتیب زیر ملاقات می شوند.



$$v_0, v_1, v_3, v_7, v_4, v_5, v_2, v_6$$

جستجوی سطحی Breadth-First Search (BFS)

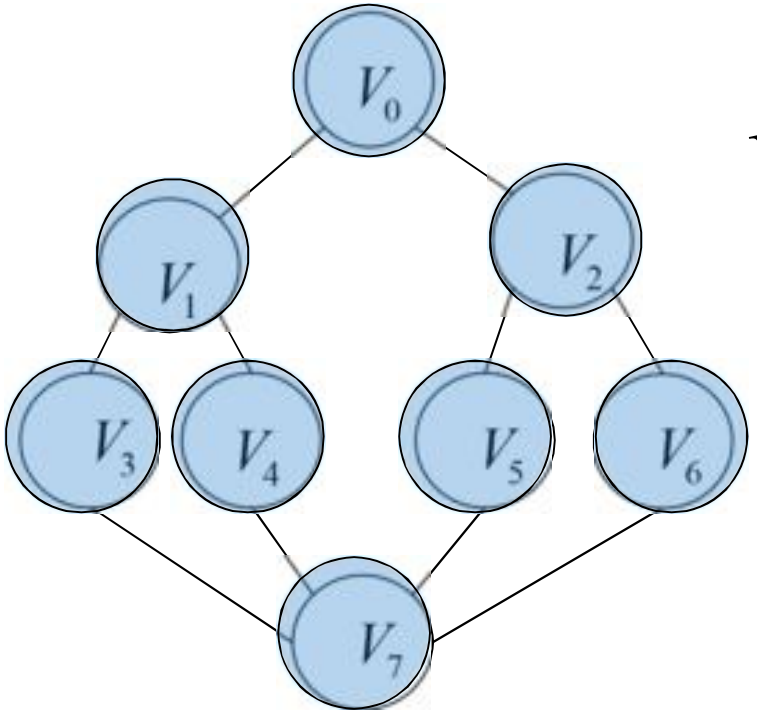
- پیمایش را با راس **v** شروع نموده، پس از ملاقات راس مزبور، آن را علامت گذاری می کنیم.
- اول هر یک از رئوس مجاور به راس **v** را در لیست مجاورتی ملاقات میکنیم.
- زمانی که همه رئوس مجاور با راس **v** را ملاقات کردیم ، تمام رئوس ملاقات نشده که مجاور با اولین راس مجاور با **v** در لیست مجاورتی است را ملاقات می کنیم.
- برای انجام این کار تا زمانی که هر راس ملاقات می شود ، آن را در یک صف قرار می دهیم.

جستجوی سطحی Breadth-First Search (BFS)

- 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 سطحی Breadth-First Search (BFS)

- مثال: اگر روش جستجوی سطحی را از راس V_0 آغاز کنیم ، رئوس گراف G به ترتیب زیر ملاقات می شوند.

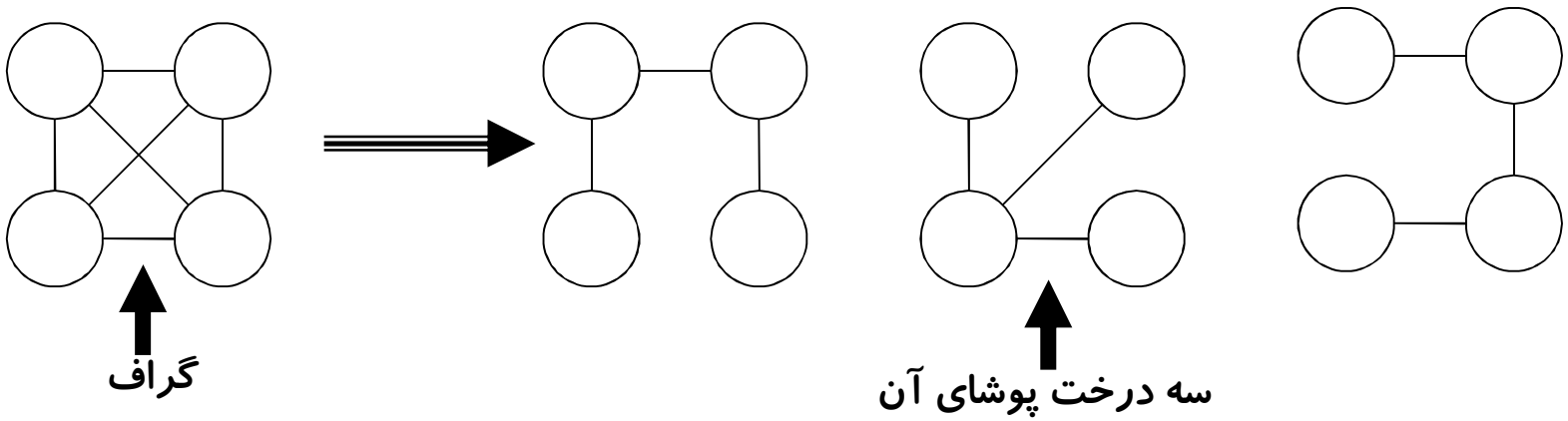


$$v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7$$

درخت پوشا

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

- مثال: سه درخت پوش از گراف داده شده



درخت پوشا

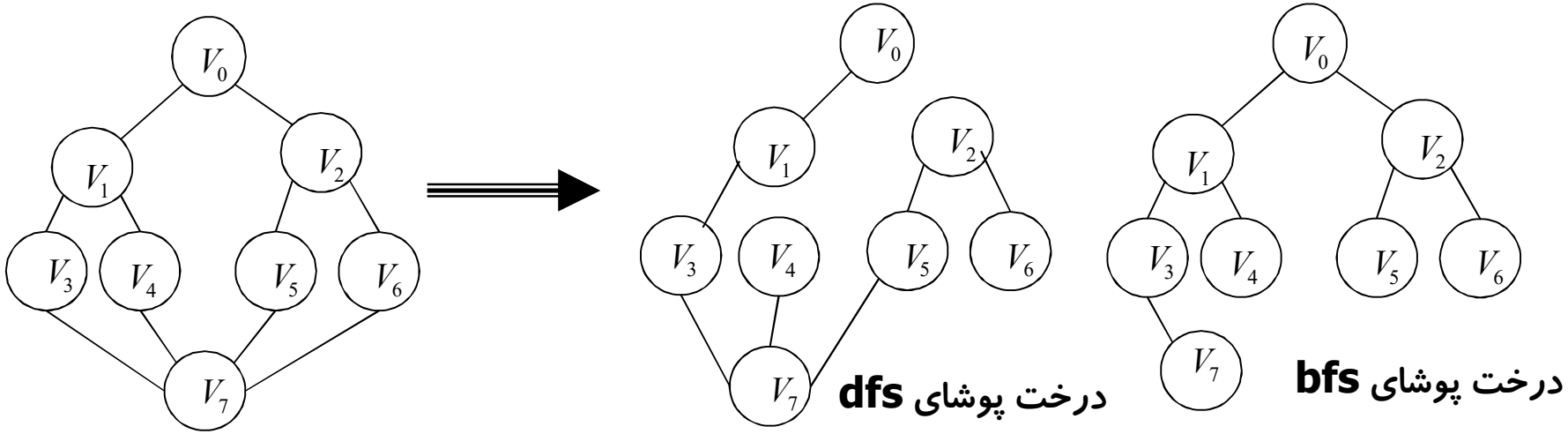
چنانچه گراف G متصل باشد ، پیمایش های جستجوی عمقی یا جستجوی ردیفی ، تمام رئوس گراف G را ملاقات می کنند.

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 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 عرضی (سطحی یا ردیفی) می نامند.



$v_0, v_1, v_3, v_7, v_4, v_5, v_2, v_6$ $v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7$

درخت پوشا

- مقصود از زیر گراف حداقل، یعنی زیرگرافی که تعداد لبه هایش کمترین باشد.
- هر گراف متصل با n راس ، بایستی حداقل n-1 لبه داشته باشد و همه گراف های 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.به طور مثال، اگر رئوس گراف G نماینده شهرها و لبه ها معرف جاده های ارتباطی بین شهرها باشد ، آنگاه حداقل تعداد خطوط مورد نیاز برای اتصال n شهر به یکدیگر n-1 خواهد بود.

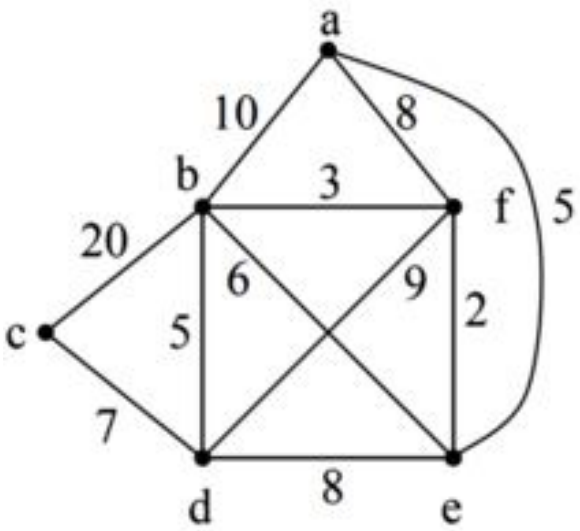
درخت پوشای کمینه(Minimum spanning tree)

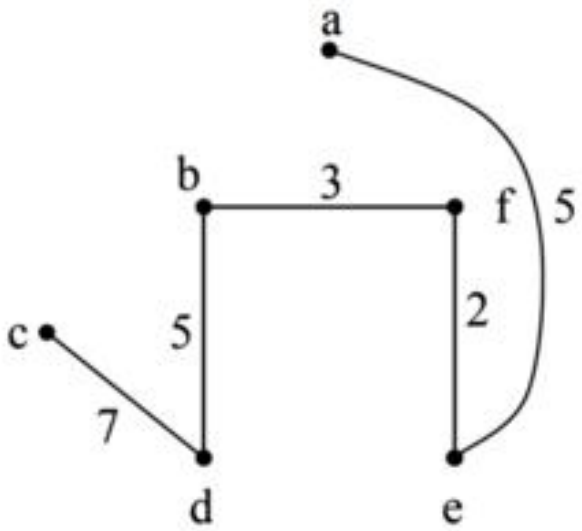
- در گراف های وزن دار ساخته می شود
- درختی است که اگر ارزش تمام یال های آن را جمع کنیم کوچکترین عدد ممکن حاصل گردد.
- روش های مختلفی برای ایجاد درخت پوشای بهینه از یک گراف وزن دار وجود دارد که از مهم ترین آنها دو روش زیرهستند:
  - الگوریتم کراسکال(kruskal)
  - الگوریتم پریم(prim)

درخت پوشای بهینه–الگوریتم کراسکال

- یالهای گراف را به ترتیب صعودی مرتب می کنیم . از اولین (کوچکترین) یال شروع کرده و هر یال را به گراف اضافه می کنیم به شرط اینکه دور در گراف ایجاد نگردد . این روال تا ایجاد درخت ادامه می دهیم.

✓ fe = 2
✓ bf = 3
✓ bd = 5 , ✓ ae = 5
× be = 6
✓ cd = 7
× de = 8 , af = 8
× df = 9
× ab = 10
× bc = 20





درخت پوشای بهینه–الگوریتم پریم

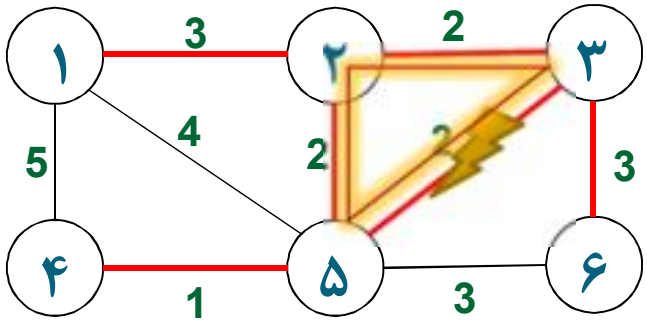
- از یک رأس شروع می کنیم و کمترین یال (یال با کمترین وزن) که از آن می گذرد را انتخاب می کنیم.
- در مرحله بعد یالی انتخاب می شود که کمترین وزن را در بین یالهایی که از دو گره موجود می گذرد داشته باشیم
- به همین ترتیب در مرحله بعد یالی انتخاب می گردد که کمترین وزن را در بین یالهایی که از سه گره موجود می گذرد داشته باشد.
- این روال را آنقدر تکرار می کنیم تا درخت پوشای بهینه حاصل شود.
- باید توجه کرد که یال انتخابی در هر مرحله در صورتی انتخاب می شود که در گراف دور ایجاد نکند.
- **تفاوت با روش کراسکال:** گراف حاصل در مراحل میانی در روش پریم همیشه متصل است ولی در الگوریتم کراسکال در آخرین مرحله قطعاً متصل است.

درخت پوشای بهینه–الگوریتم پریم

- از یک رأس شروع می کنیم و کمترین یال (یال با کمترین وزن) که از آن می گذرد را انتخاب می کنیم.
- در مرحله بعد یالی انتخاب می شود که کمترین وزن را در بین یالهایی که از دو گره موجود می گذرد داشته باشیم
- به همین ترتیب در مرحله بعد یالی انتخاب می گردد که کمترین وزن را در بین یالهایی که از سه گره موجود می گذرد داشته باشد.
- این روال را آنقدر تکرار می کنیم تا درخت پوشای بهینه حاصل شود.
- باید توجه کرد که یال انتخابی در هر مرحله در صورتی انتخاب می شود که در گراف دور ایجاد نکند.
- **تفاوت با روش کراسکال:** گراف حاصل در مراحل میانی در روش پریم همیشه متصل است ولی در الگوریتم کراسکال در آخرین مرحله قطعاً متصل است.

درخت پوشای بهینه

- مثال: درخت پوشای کمینه گراف زیر را با اعمال الگوریتم کراسکال به دست آورید.



درخت پوشای بهینه

- مثال: درخت پوشای کمینه گراف زیر را با اعمال الگوریتم پریم و شروع از گره شماره ۱ به دست آورید.