گرافها- قسمت دوم

سیدمهدی وحیدی پور

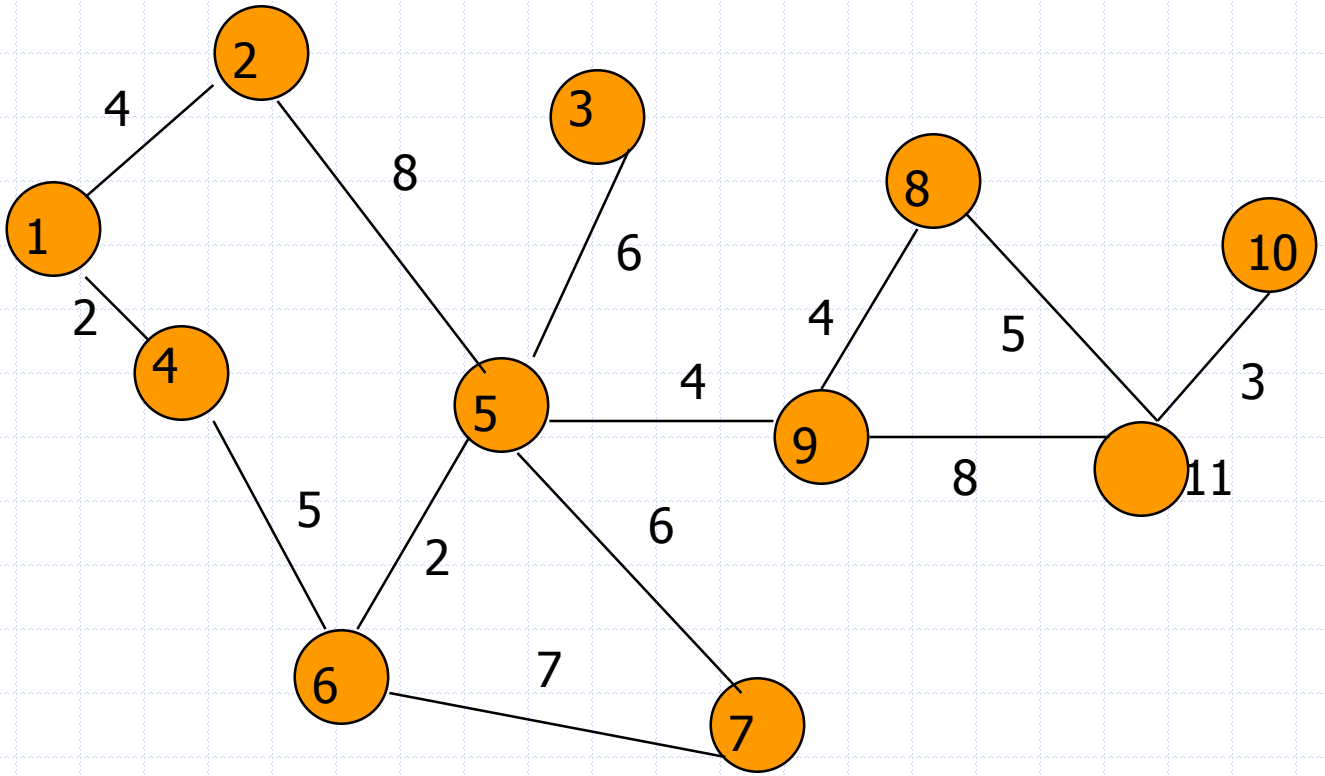
با تشکر از آقای دکتر سلیمی

دانشگاه کاشان دانشکده برق و کامپیوتر

گراف ها

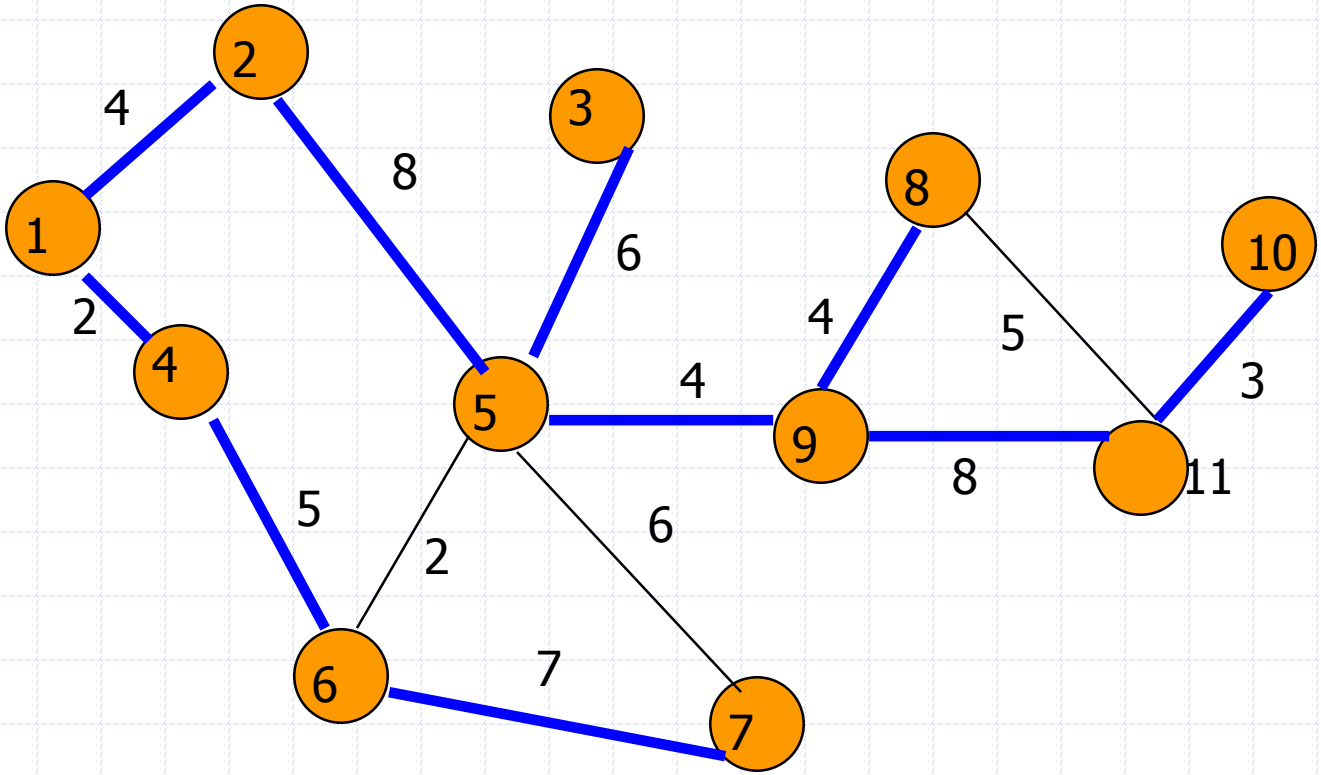
- نوع داده مجرد گراف
  - صورت های مختلف بازنمایی گراف
- عملیات روی گراف ها
  - جستجوی روی گرافها
    - جستجوی عمقی
    - جستجوی عرضی
  - مولفه های همبند
  - درختهای پوشا
  - مولفه های همبند دوطرفه
- درختهای پوشا با کمترین هزینه
- کوتاه ترین مسیر
  - بستار متعدی

درختهای پوشا با کمترین هزینه



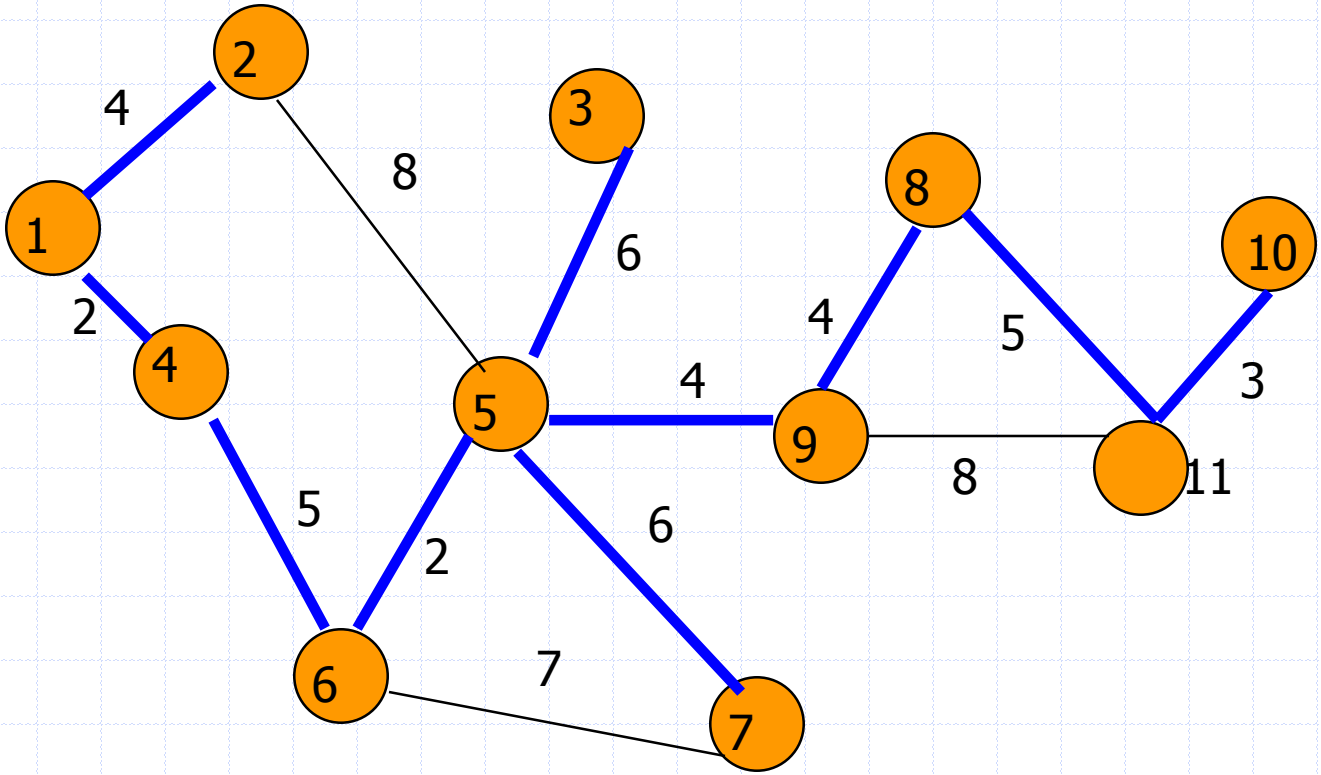
- هزینه درخت پوشای یک گراف بدون جهت وزن دار، مجموع هزینه (وزن) های یال های درخت پوشا است.

درختهای پوشا با کمترین هزینه



هزینه درخت پوشا 51.

درختهای پوشا با کمترین هزینه



هزینه درخت پوشا 41.

درختهای پوشا با کمترین هزینه

- درخت پوشای با کمترین هزینه، درخت پوشایی است که کمترین هزینه را دارد.
- سه الگوریتم مختلف برای به دست آوردن درخت پوشا با کمترین هزینه از یک گراف بدون جهت وجود دارد.
  - الگوریتم Kruskal
  - الگوریتم Prim
  - الگوریتم Sollin
- همه این الگوریتم ها از روش طراحی حریصانه (greedy method) استفاده می کنند.

درختهای پوشا با کمترین هزینه

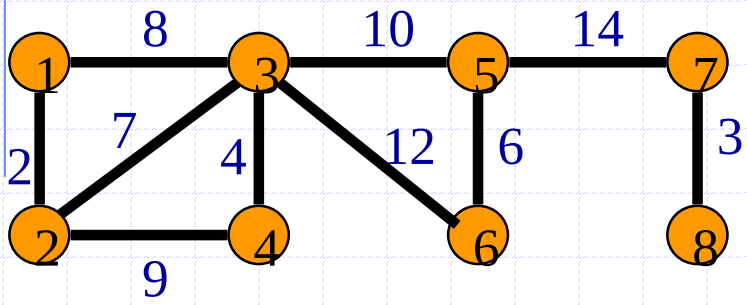
- راهکار حریصانه
  - در هر مرحله بهترین تصمیم را بر اساس اطلاعات موجود در آن لحظه اتخاذ می کنیم. معمولا اتخاذ تصمیم در هر مرحله بر اساس کمترین هزینه یا بیشترین سود استوار است.
    - برای ایجاد درخت های پوشا با کمترین هزینه از معیار کمترین هزینه استفاده می کنیم.
  - در مراحل بعدی نمی‌توانیم تصمیم های قبل را عوض کنیم بنابراین باید مطمئن شویم این تصمیم منجر به راه حل معتبر می شود.
  - یک راه حل معتبر بر اساس قیدهای بیان شده در مساله تعیین می شود.
- راه حل ما باید در شرایط زیر صدق کند:
  - تنها باید از یالهای گراف استفاده کند.
  - باید دقیقا از n-1 یال استفاده کند.
  - از یالهایی که دور ایجاد می کنند نمی توان استفاده کرد.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

- الگوریتم Kruskal
  - درخت پوشا با کمترین هزینه را با اضافه کردن یک یال در هر مرحله می سازد.
  - یال ها برای اضافه شدن به T به ترتیب غیر نزولی هزینه شان انتخاب می شوند.
  - یک یال به T اضافه می شود مشروط به اینکه با یالهایی که قبلا در T بوده اند دور تشکیل ندهد.
  - از آنجا که گراف G همبند است و $n > 0$ راس دارد دقیقا $n-1$ یال برای اضافه شدن به T انتخاب می شود.

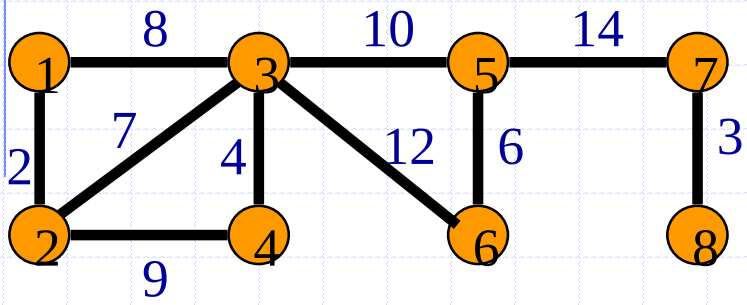
- **قضیه** : اگر G یک گراف همبند بدون جهت باشد آنگاه الگوریتم Kruskal یک درخت پوشا با کمترین هزینه را تولید می کند.
- Time complexity: $O(e \log e)$

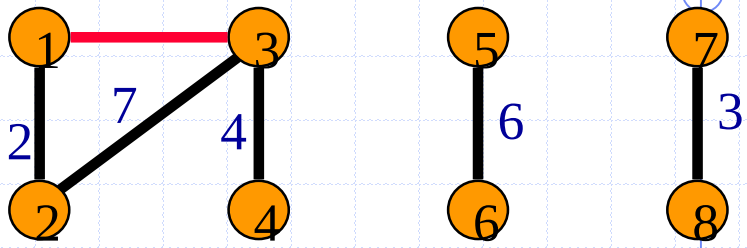
درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)



- با یک جنگل که هیچ یالی ندارد شروع کنید
- لبه ها را به ترتیب صعودی وزن آنها انتخاب کنید.
- لبه (1,2) انتخاب شده و به جنگل اضافه می شود.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)





در مرحله بعد لبه (7,8) انتخاب شده و اضافه می شود.

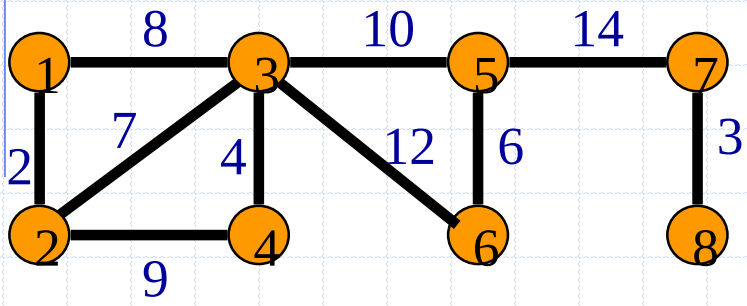
در مرحله بعد لبه (3,4) انتخاب شده و اضافه می شود.

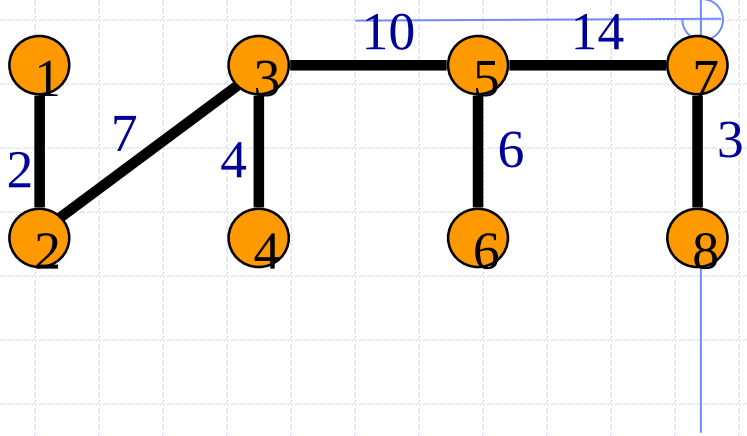
در مرحله بعد لبه (5,6) انتخاب شده و اضافه می شود.

در مرحله بعد لبه (2,3) انتخاب شده و اضافه می شود.

در مرحله بعد لبه (1,3) انتخاب شده و به دلیل آنکه دور ایجاد می کند حذف می شود.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)





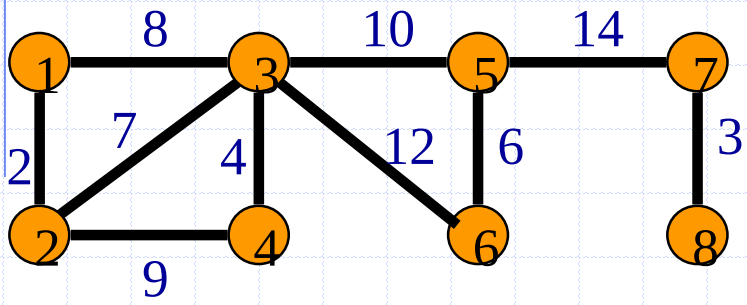
در مرحله بعد لبه (2,4) انتخاب شده و به دلیل آنکه دور ایجاد می کند حذف می شود.

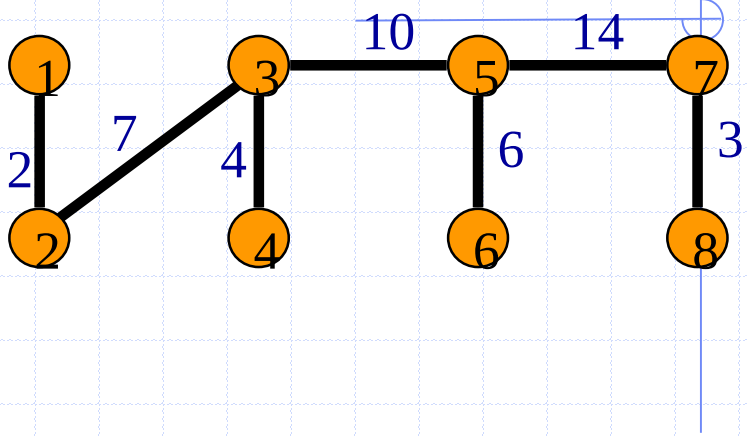
در مرحله بعد لبه (3,5) انتخاب شده و اضافه می شود.

در مرحله بعد لبه (3,6) انتخاب شده و به دلیل آنکه دور ایجاد می کند حذف می شود.

در مرحله بعد لبه (5,7) انتخاب شده و اضافه می شود.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)



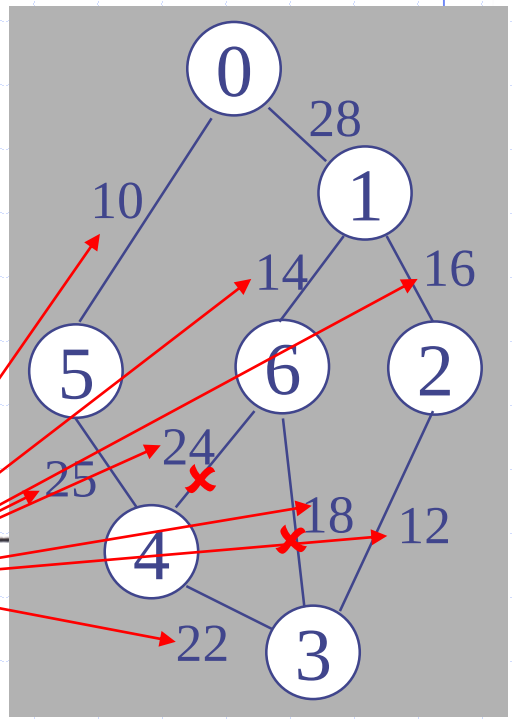


- n-1 یال انتخاب شد و هیچ دوری ایجاد نگردید.
- بنابراین ما یک درخت پوشا به دست آوردیم.
- هزینه این درخت 46 است.
- در صورتی که هزینه یالهای مختلف درخت متفاوت باشد درخت پوشای با کمترین هزینه یکتا است.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

- شبه کد الگوریتم Kruskal

```
T = {};
while (T contains less than n-1 edges && E is not empty) {
    choose a least cost edge (v,w) from E;
    delete (v,w) from E;
    if ((v,w) does not create a cycle in T)
        add (v,w) to T;
    else
        discard (v,w);
}
if (T contains fewer than n-1 edges)
    printf("No spanning tree\n");
```



درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

ساختمان داده برای الگوریتم Kruskal

- مجموعه لبه ها E
- عملیات:
  - آیا E تهی است؟
  - انتخاب و حذف یال با کمترین هزینه
- می توان یالها را به ترتیب غیر نزولی در زمان O(e log e) مرتب کرد
- می توان از یک min heap برای نگهداری یالها استفاده کرد.
  - ساختن اولیه heap: O(e)
  - تعیین و حذف یال با کمترین هزینه O(log e)

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

ساختمان داده برای الگوریتم Kruskal

- مجموعه یالهای انتخاب شده T
- عملیات:
  - آیا T، n-1 یال دارد؟
  - آیا اضافه کردن یال (V,W) به T دور ایجاد می کند ؟
  - یک یال به T اضافه کنید.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

ساختمان داده برای الگوریتم Kruskal

- نگهداری مجموعه یالهای انتخاب شده T در یک آرایه
  - آیا T، n-1 یال دارد؟

    تعداد یالها را در ارایه بررسی می کنیم: O(1)

  - آیا اضافه کردن یال (V,W) به T دور ایجاد می کند ؟

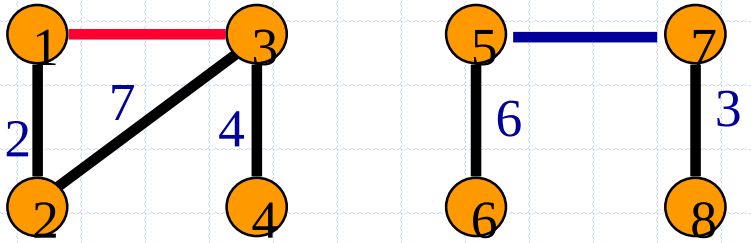
    آسان نیست

  - یک یال به T اضافه کنید.

    آن را به انتهای آرایه اضافه می کنیم: O(1)

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

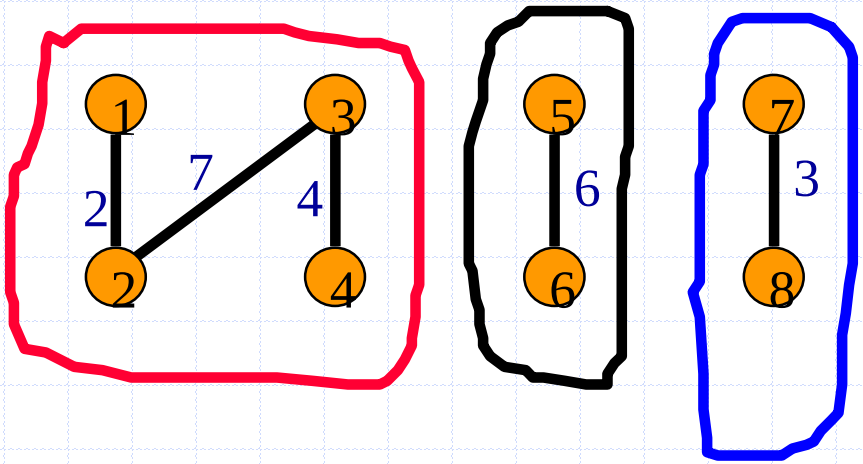
ساختمان داده برای الگوریتم Kruskal



آیا اضافه کردن یال (v, w) به T دور ایجاد می کند؟

- اجزای T در هر لحظه درخت هستند.
- وقتی v و w در یک جزء باشند اضافه کردن یال (v,w) دور ایجاد می کند.
- وقتی که v و w در دو جزء جدا باشند اضافه کردن یال (v,w) دور ایجاد نمی کند.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)



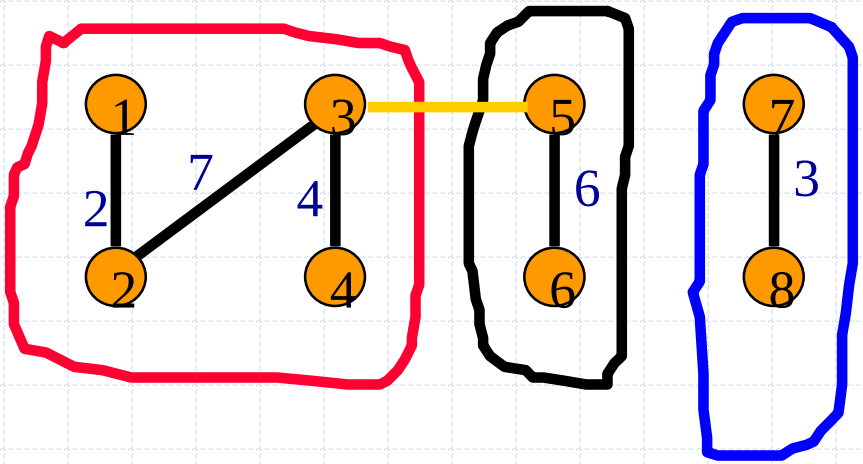
ساختمان داده برای
الگوریتم Kruskal

- هر جزء T با رئوس درون آن مشخص می شود.
- هر جزء را با مجموعه رئوس آن نشان می دهیم

{1, 2, 3, 4}, {5, 6}, {7, 8}

- دو راس در یک جزء هستند اگر درون یک مجموعه باشند.

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)



ساختمان داده برای الگوریتم Kruskal

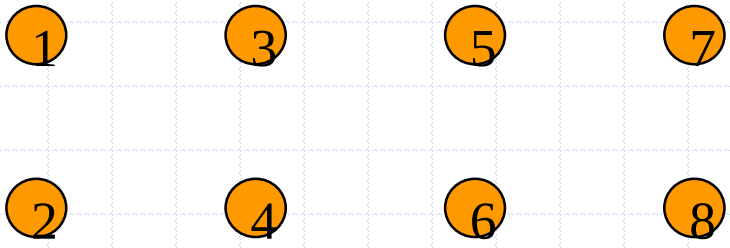
- هنگامی که یال (v,w) به T اضافه می شود، دو جزئی که شامل v و w بوده اند با هم ترکیب می شوند تا یک جزء را تشکیل دهند.
- چنانچه مجموعه ها برای بازنمایی اجزای T به کار روند، مجموعه های شامل v و w را با هم اجتماع می گیریم.

{1, 2, 3, 4} + {5, 6} => {1, 2, 3, 4, 5, 6}

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

ساختمان داده برای الگوریتم Kruskal

- در ابتدا T تهی است.



- مجموعه ها در ابتدا به صورت:
  - {1} {2} {3} {4} {5} {6} {7} {8}
- آیا اضافه کردن یال (v,w) به T دور ایجاد می کند ؟ اگر دور ایجاد نمی کند آن را اضافه کنید.

```
s1 = Find(v); s2 = Find(w);
if (s1 != s2) Union(s1, s2);
```

درختهای پوشا با کمترین هزینه (الگوریتم Kruskal)

ساختمان داده برای الگوریتم Kruskal

- استفاده از توابع سریع برای پیاده سازی مجموعه ها
- مقدار دهی اولیه:

O(n)

- حداکثر 2e عمل find و n-1 عمل اجتماع:

O(n+e) خیلی نزدیک به

- عملیات روی min heap برای انتخاب یالها به ترتیب غیر نزولی

O(e log e)

- پیچیدگی زمانی الگوریتم Kruskal :

O(n + e log e)

درختهای پوشا با کمترین هزینه (الگوریتم Prim)

- الگوریتم Prim
  - درخت پوشا با کمترین هزینه را با اضافه کردن یک یال در هر مرحله می سازد.
  - یال ها برای اضافه شدن به T به ترتیب غیر نزولی هزینه شان انتخاب می شوند.
  - در تمام مراحل الگوریتم مجموعه یالهای انتخاب شده یک درخت را تشکیل می دهند.
  - **توجه** در الگوریتم Kruskal مجموعه یالهای انتخاب شده در هر مرحله یک جنگل را می سازند.

درختهای پوشا با کمترین هزینه (الگوریتم Prim)

- الگوریتم Prim
  - در ابتدا T تنها شامل یک راس است.
  - یال (u,v) با حداقل هزینه را به گونه ای می یابیم که با اضافه شدن آن به T ، T درخت باقی بماند.

    یال (u,v) به گو نه ای است که تنها یکی از رئوس u و یا v در T باشد.
  - اضافه کردن یال را ادامه می دهیم تا T شامل n-1 یال شود.

- الگوریتم Prim برای هر گراف بدون جهت همبند درخت پوشا با کمترین هزینه را پیدا می کند .