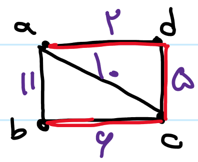
جلسه بیست هفتم

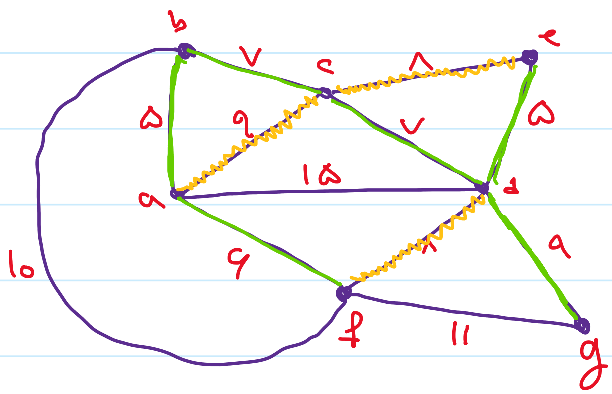
13:03 Tuesday, December 28, 2021

موضوع: درخت پوشای کمینه. الگوریتم کراسکال

۲+۵+۶ = ۱۳



الگوریتم کروسکال:
با $T=\phi$ شروع کن. در هر مرحله، یال با وزن کمینه که اضافه کردن آن با یال های قبلی انتخاب شده ایجاد دور نمی کند را به T اضافه کن.
n-1



T = {ed, ab, af, cd, bc, dg}

مجموع وزن: ۳۹

بعد از n-1 مرحله: $|T| = n-1$ ، یال های T دور ندارند ⇐ T یک ST است

اثبات درستی الگوریتم کراسکال:

برهان خلف:

کراسکال

$\boxed{OPT \neq ALG}$

فرض*: وزن هیچ دو یالی برابر نیست.

مرتب شده برحسب وزن →

OPT : $e_1 \quad e_2 \quad e_3 \quad \ldots \quad e_{n-1}$

ALG : $e'_1 \quad e'_2 \quad e'_3 \quad \ldots \quad e'_{n-1}$

→ مرتب شده برحسب وزن

چون AlG $\neq$ OPT است به ازای حداقل یک i، $e'_i \neq e_i$.

∴ فرض کنید i، اولین اندیسی است که $e'_i \neq e_i$

مثلاً اگر i=۳

$e_1 = e'_1$

$e_2 = e'_2$

$e_3 \neq e'_3$

$$\omega(e_i) \overset{?}{>} \omega(e'_i)$$

← چون کراسکال در هر مرحله یال با وزن کمینه را انتخاب می‌کند.

$e_1 = e'_1$

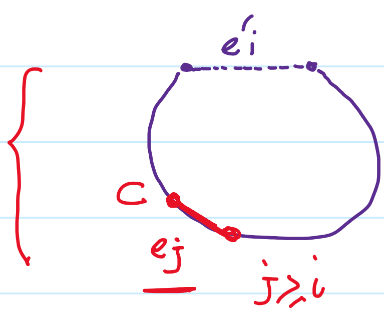
$e_2 = e'_2$

$\vdots$

$e_{i-1} = e'_{i-1}$

$e_i \neq e'_i$ و $\omega(e_i) > \omega(e'_i)$

e'_i در OPT نبوده



← e'_i را به OPT اضافه کنید!

← در OPT یک دور ایجاد می‌شود.

← در دور C یک یال e_j وجود دارد که $i < j$ است.

← e_j را حذف می‌کنیم. وزن درخت جدید به اندازه $\omega(e_j) - \omega(e'_i)$ کمتر از درخت قبلی است.

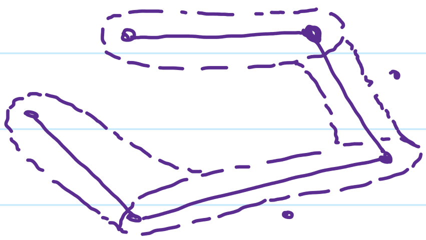
* کمینه بودن OPT

زمان اجرای الگوریتم کروسکال :

* مرتب کردن یال ها : $O(m \log m)$

* مرتب کردن یال ها : $O(m\log m)$

* اضافه کردن یک یال به T ایجاد دور نمی کند، اگر و تنها اگر دو راس آن در دو مولفه همبندی مختلف باشند.

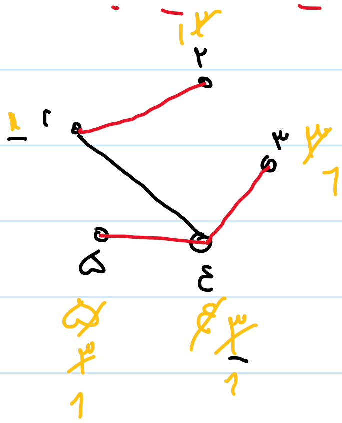


$O(m\log m + m + n^2)$

* به ازای هر یال، مولفه همبندی ۲ سر آن را پیدا کن ← Find

- اگر یکی بودند : اضافه کردن آن ایجاد دوری کند.
- " " نبودند : یال را به T اضافه می کنیم و سپس مولفه های مربوط به ۲ سر آن یال را ادغام می کنیم. ← union

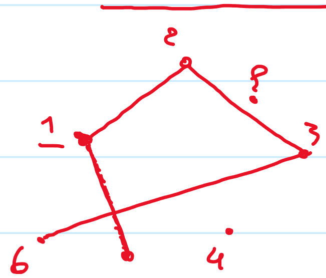
* راه حل ۱) به هر راس یک برچسب نسبت دهیم به طوری که برچسب راس های حاضر در یک مولفه یکسان باشند.

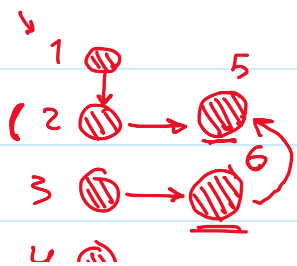


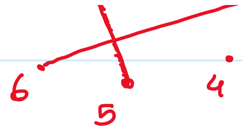
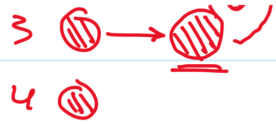
Find() : $O(1)$

Union() : $O(n)$

$O(m\log m + n^2)$





m ← $O(n)$: find( )

$O(m\log m + mn + n)$ n ← $O(1)$: Union( )

$O(mn)$

داده ساختار disjoint set :

m ×
$$\left\{\begin{array}{l} \text{find} : O(\log^* n) \\ \text{Union} : O(\log^* n) \end{array}\right\}$$ سرشکن

$\log^* 2 = 1$ $\log^* 2^2 = 2$

$\log^* 2^{2^2} = 3$ $\log^* \underbrace{2^{2^{2^2}}}_{65536} = 4$

$\log^* ? = 5$

$\downarrow$

2^{65536}

$O(m\log m + m\log^* n + n\log^* n)$

۷ * اگر بتوانیم یال ها را با مرتب سازی خطی (مثلاً counting sort) مرتب کنیم :

$O(m + m\log^* n + n\log^* n) = \underline{O(m\log^* n) \simeq O(m)}$

$\underline{\alpha(n)}$ ← رشد کندتر از $\log^*$

$$\longrightarrow \begin{cases} \text{Find} \rightarrow \alpha(n) \\ \text{union} \rightarrow \alpha(n) \end{cases}$$