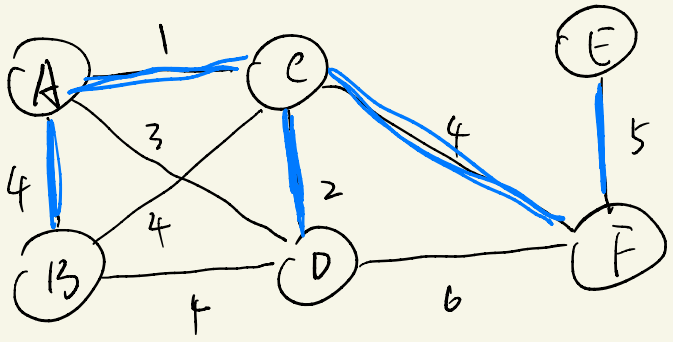
Lec 6

Minimum Spanning Tree



Connect the Graph with minimum cost.

Prop. Must be a tree

Input: Undirected Graph $G = (V, E)$, edge weight we

Output: Tree $T = (V, E')$,

$E' \subseteq E$ minimizes $wt(T) = \sum_{e \in E'} w(e)$.

Kruskal: Greedy strategy.

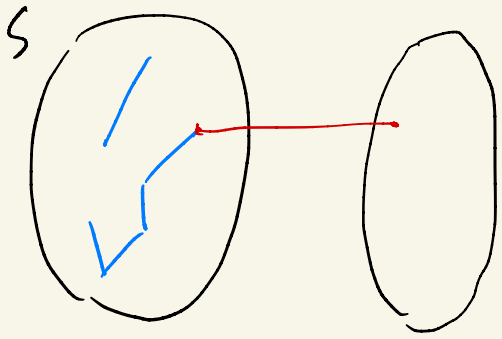
" Pick the lightest edge that doesn't produce a cycle".

Proof of correctness

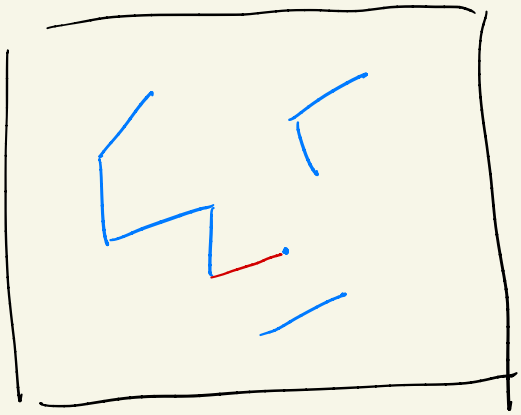
prop. X part of MST of $G(V, E)$. $S \subseteq V$.

$E(S, V-S) \cap X = \phi$. $e = \underset{e' \in E(S, V-S)}{\arg\min} \; w(e')$

$\Rightarrow$ e is in a MST.



Correctness of Krustal.



Directly apply the cut property.

Implementation

Kruskal $(G, \{w_e\}_{e \in E(G)})$.

for $u \in V$: makeset (u).

$X = \emptyset$.

Sort edges in E by $\{w_e\}$.

for $e = \{u, v\} \in E$ in increasing order of $\{w_e\}$

if find $(u) \neq$ find (v) — check whether u, v are in the same connected component.

add e to X.

union (u, v).

The operations of datastructure.

makeset (x).	initialize a singleton set $\{x\}$.
find (x)	Return the representative element of the set containing x.
union (x, y)	merge the set Containing x and y.

Union-Find Set 并查集.

" Each set is a tree. representative is the root ".

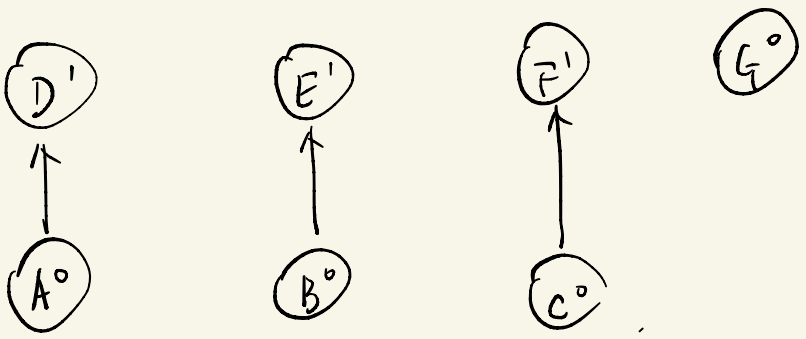
```
makeset(x)
    π(x) = x.
    rank(x) = 0.
```

```
find(x)
    while x ≠ π(x)
        x = π(x)
    return x
```

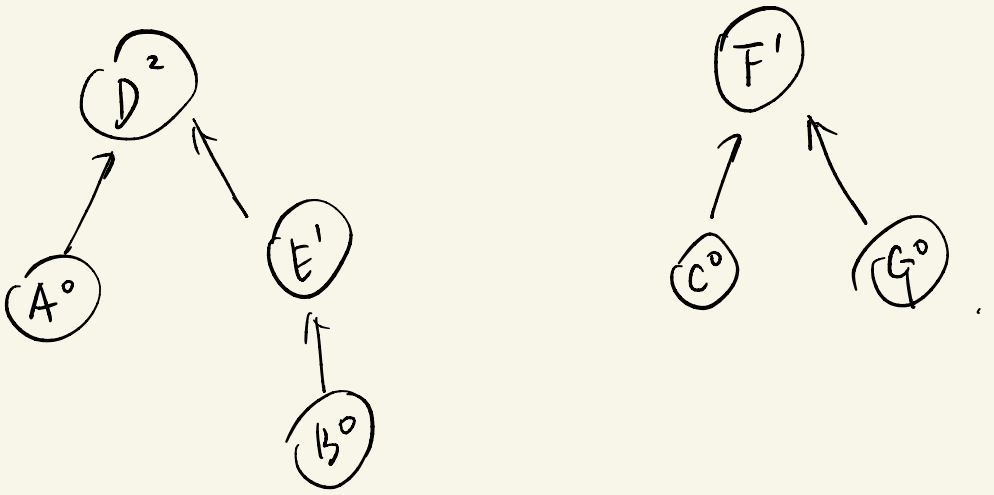
```
union(x, y)
    r_x = find(x)
    r_y = find(y).
    if r_x = r_y : return
    if rank(r_x) > rank(r_y)
        π(r_y) = r_x.
    else  π(r_x) = r_y.
        if rank(r_x) = rank(r_y) : rank(r_y) ++ ;
```

Example

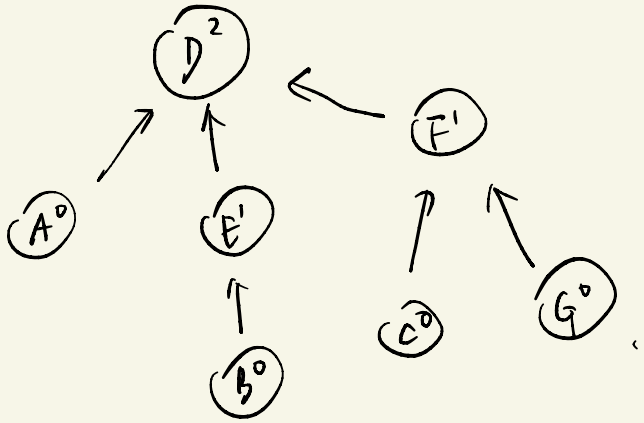
union (A, D). union (B, E). union (C, F).



union (C, G). union (E, A).



Union (B, G).



Properties of the Union-Find set.

P1>. root node of rank k has at least 2^k descendents

Proof by induction

P2>. $\text{rank}(\pi(x)) > \text{rank}(x)$. $\forall x$.

P3>. The number of nodes of rank $k \leq n/2^k$.

Cor. Union, Find take time $O(\log n)$.

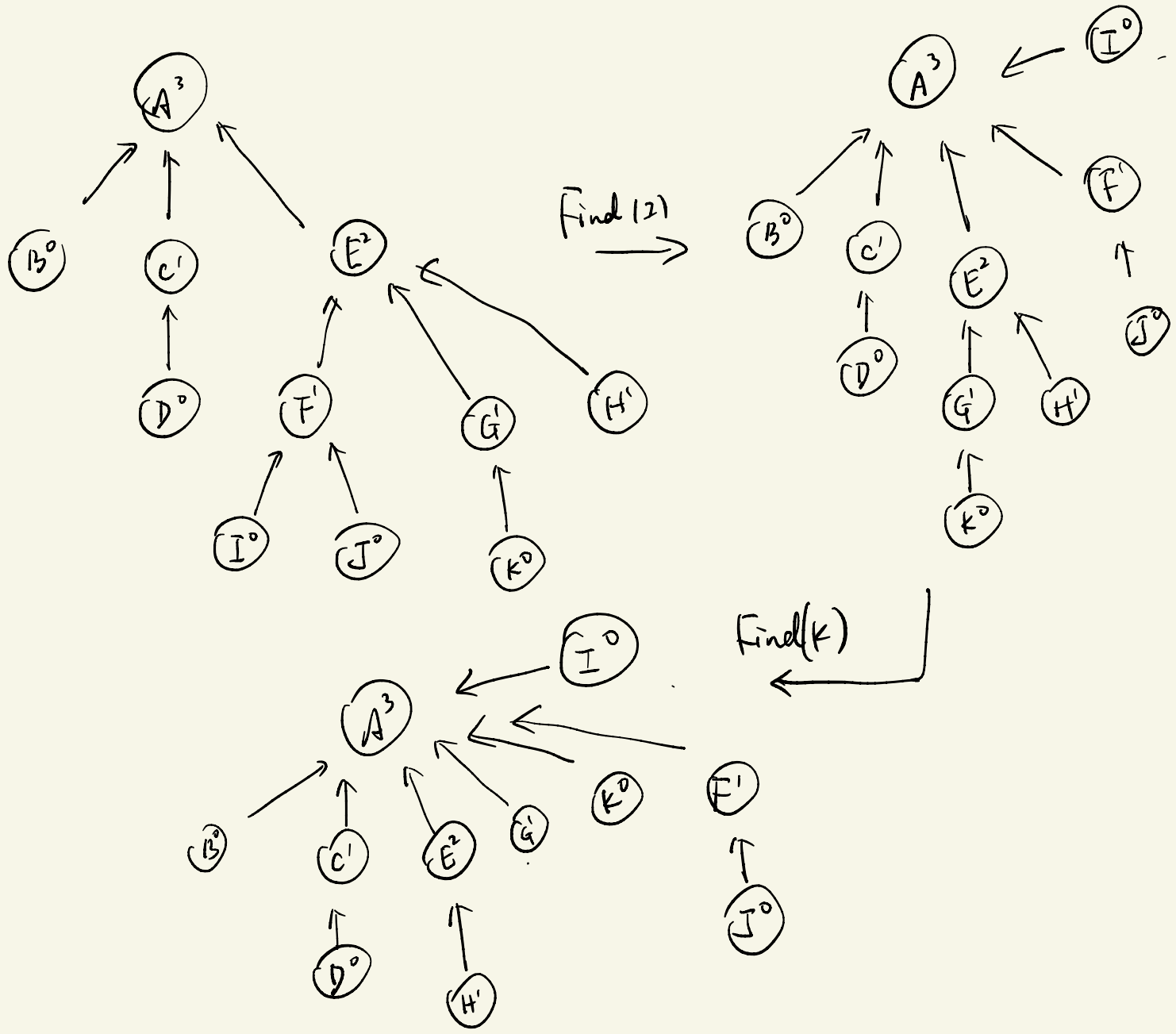
Complexity of Kruskal

Sort : $O(|E| \log |V|)$

Union & Find : $O(|E| \log |V|)$.

Path Compression

```
find(x)
    if x ≠ π(x) : π(x) = find(π(x))
    return π(x)
```



Analysis of Union-Find Set with Path Compression

[R. Tarjan 1975]

m operations $\Theta(m\alpha(m))$.

→ Very slowly growing function.

$\alpha(m)$: inverse Ackermann function e.g $\alpha(98761) = 5$

[J. Hopcroft, J. Ullman 1973].

m operations $O(m \log^* n)$.

# of times of take log until ≤ 1.

$$\log^*\left(2^{2^{2^2}}\right) = \log\left(2^{65536}\right) = 5.$$

proof * Group numbers.

$\{1\}, \{2\}, \{3,4\}, \{5,6,\dots,16\}, \{17,18,\dots,2^{16}=65536\}, \{65537,\dots,2^{65536}\}$.

$\{k+1, k+2, \dots, 2^k\}$

# of Groups is $\log^* n$.

Charging argument

* We give each node some amount of money.
* The nodes use these money to "pay" for the operation time.

Each node receives its money when it's no longer a root.

The amount of the money u received is 2^k
if $\text{rank}(u) \in \{k+1, \dots, 2^k\}$.

* Only need to pay once for each node.
* P3 $\Rightarrow$ # of nodes with rank $> k$

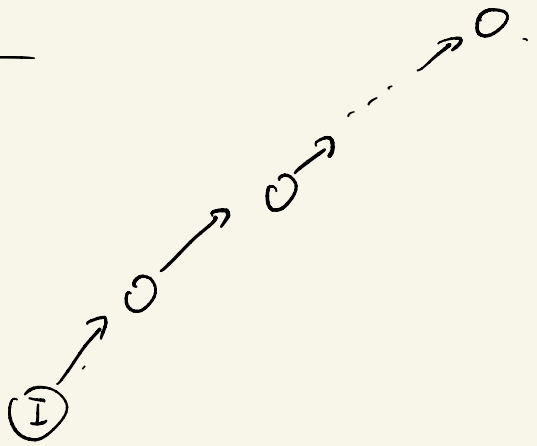
$$\le \frac{n}{2^{k+1}} + \frac{n}{2^{k+2}} + \cdots \le n/2^k.$$

The money paid for this interval is $\le n$.

The total pay is $\le n \log^* n$.

Cost of find(x)

Find(I):



Three types of node visited:

① parent(z) is the root : cost 1.

② rank(parent(z)) in a higher group : $\leq \log^* n$

③ rank(parent(z)) in the same group. charge \$1.

The cost of the operation.

$\leq \log^* n + 1 +$ [Money charged].

*). No node is made bankrupt.

We z is charged, rank(parent(z)) ++.

It pays at most 2^k.

Once rank(parent) is in higher interval, it remains so.