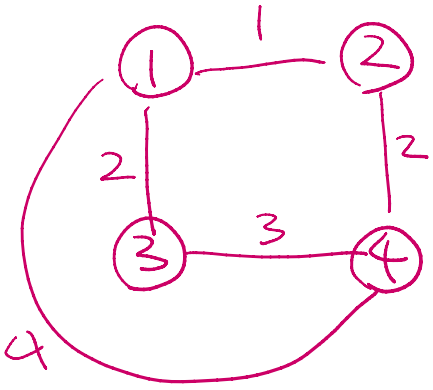
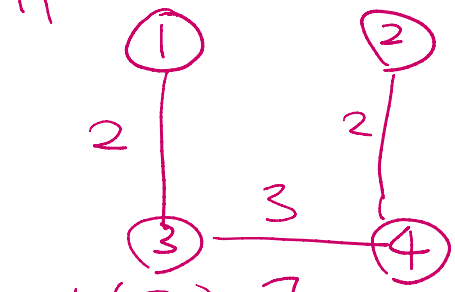
08-03

Monday, August 3, 2020 6:30 PM

MST [Minimum Spaning Tree]

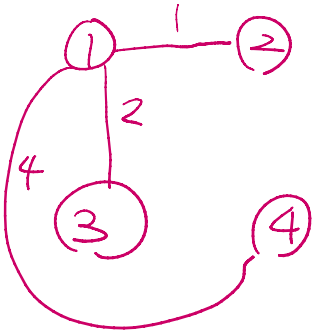


T_1



cost(T_1)=7

T_2



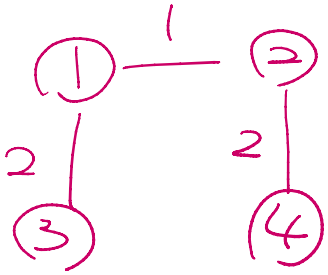
cost(T_2)=7

$G=(V, E)$

- undireted
- connected

$T=(V, E')$

T_3



cost(T_3)=5

T_3 is MST

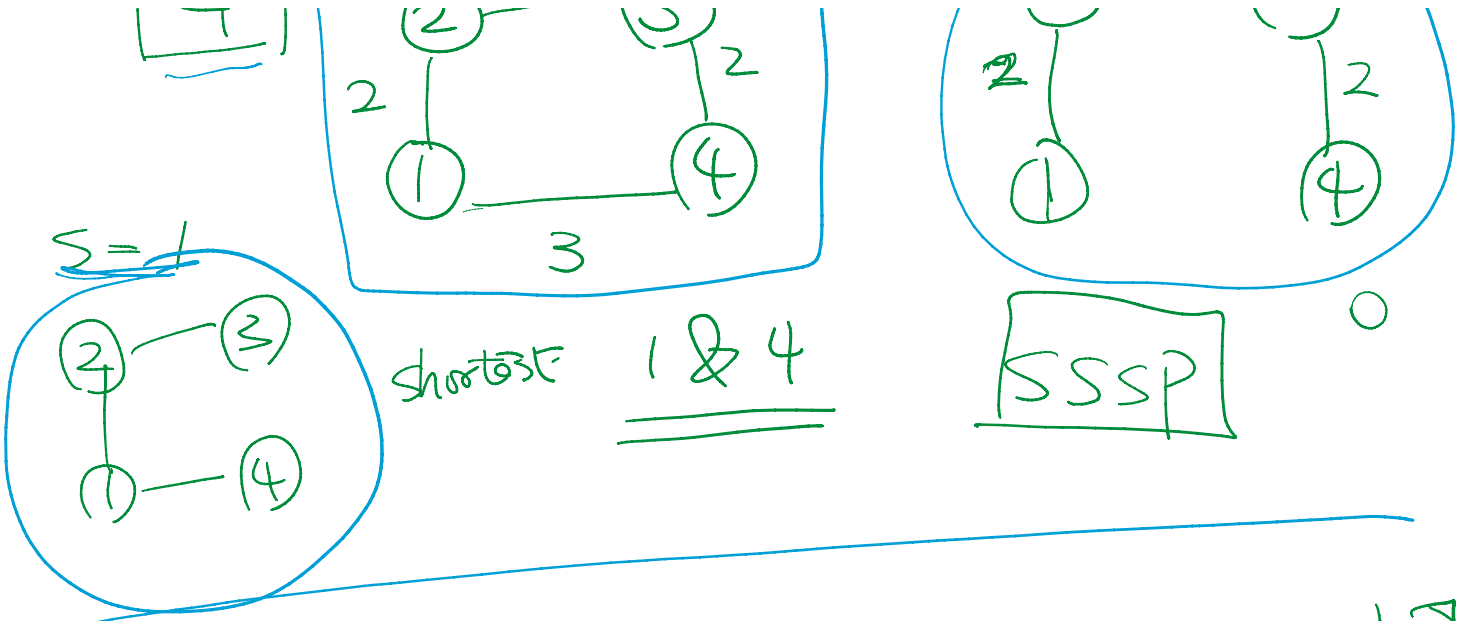
---

application

TSP

sensor network

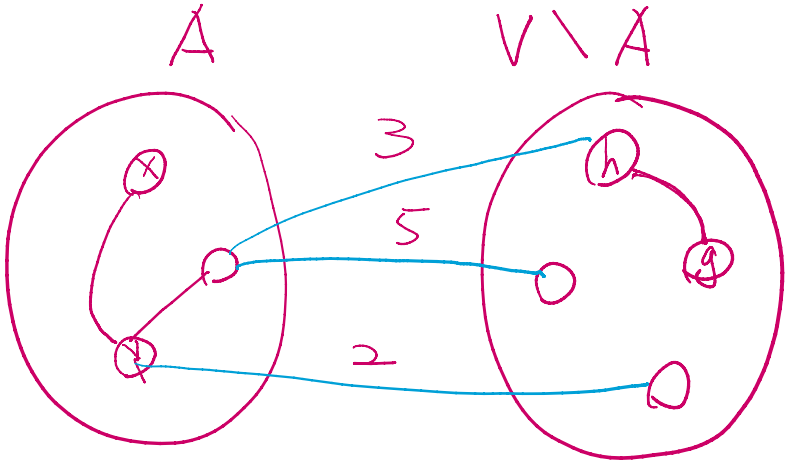
n cities



MST
- Prim's algo [similar to Dijkstra's algo]
- Kruskal's algo [Disjoin Set Union & Find]

Cut of a graph $G = (V, E)$

partition vertices into two set



e cross this cut $(A, V \setminus A)$

cross this cut $(A, V \setminus A)$

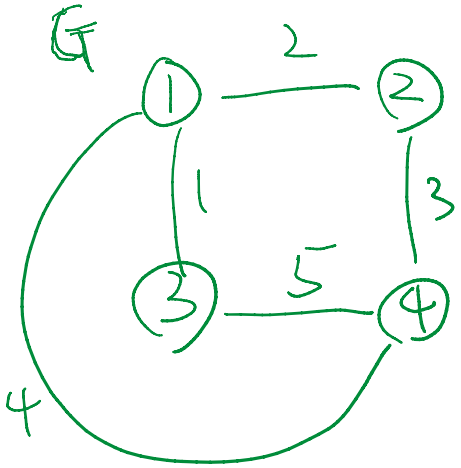
MST — one of three edges has to be in MST

at least

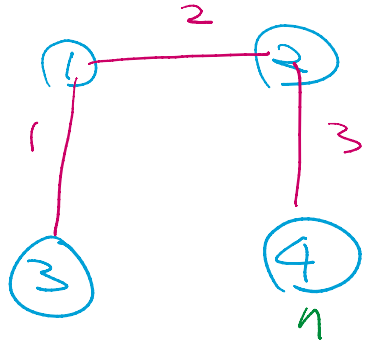
safe edge

Prim's algo

- iterative algo / greedy
- start w/ set of 1 vertex



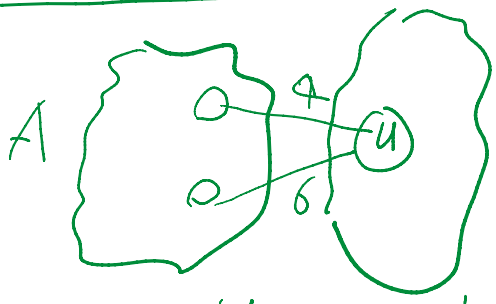
start from 3



T

dist[u] = 4

$N = \{\ \}$



while $(N \neq V)$ {

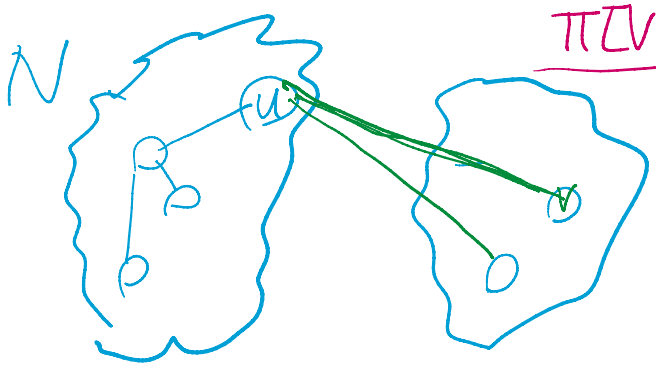
① pick vertex $u \in V \setminus N$ that has the smallest dist[ ] value.

② $N = N \cup \{u\}$ $\quad T = T \cup \{(u, \pi[u])\}$

③ for (v in G.neighbors(u))

MST

③ for (v in G.neighbors(u))
if (dist[v] > cost[u,v])
dist[v] = cost[u,v]
π[v] = u
}



running time ??

$O(n^2)$ bool array

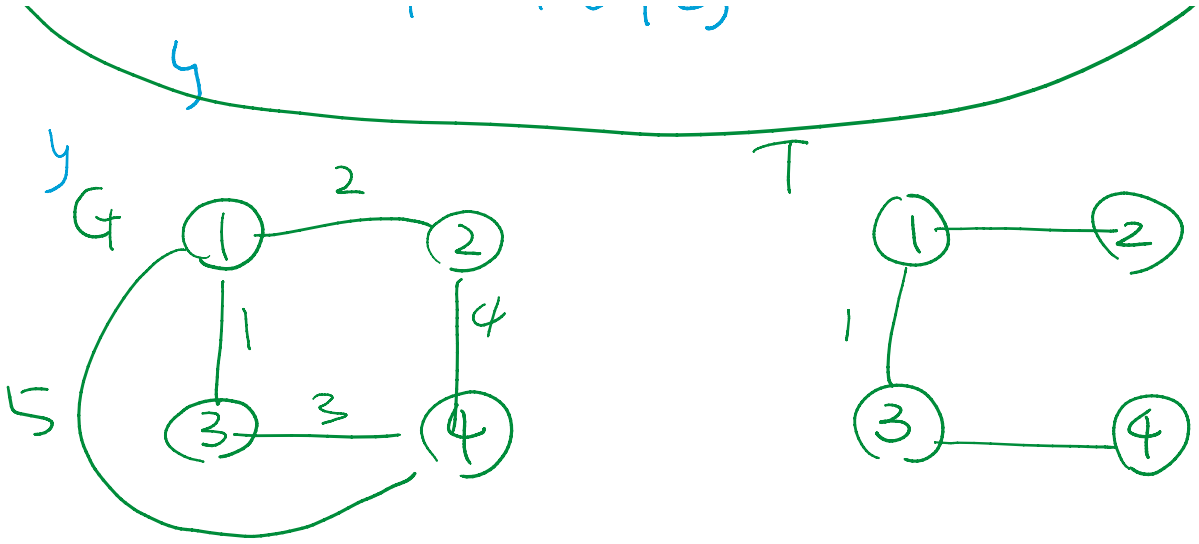
$O(m \lg n)$ priority queue / heap

BFS vs Dijkstra

- uniform edge cost
- general version bfs

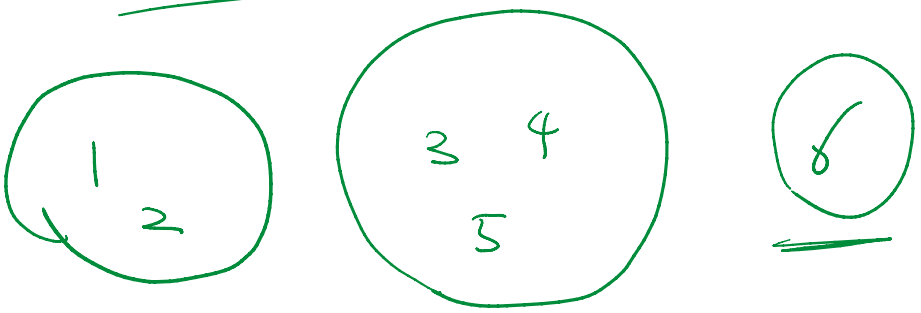
Kruskal's algo

T = { }
for (e in sort(E)) { m
if ((T ∪ {e} close not contain cycle)) {
T = T ∪ {e}
}



Disjoint Set n item

- union
- find



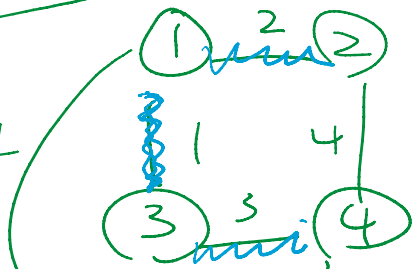
find (item) ⇒ which set

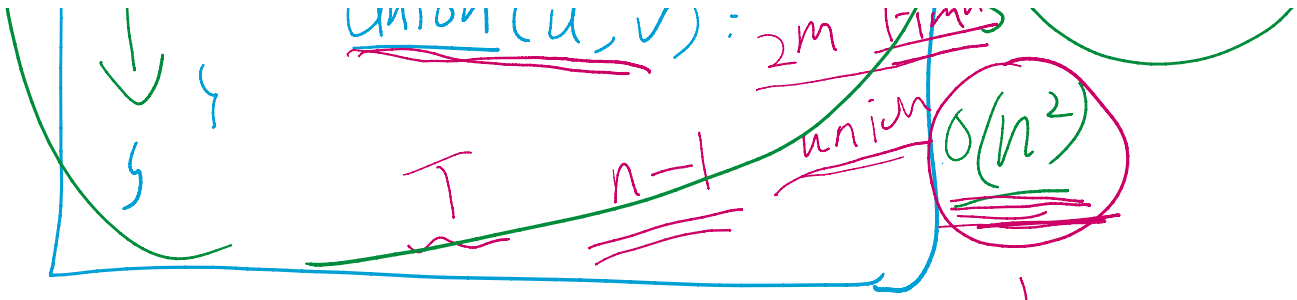
union (set A, set B) ⇒ merge A & B into one set

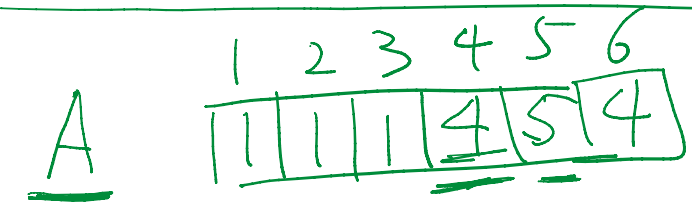
```
for( (u,v) in sort(E)){
    if (Find(u) ≠ Find(v)){
        T = T U {(u,v)}
        Union(u,v): 2m Finds
}
```

m

$O(m \lg m) = O(m \lg n)$

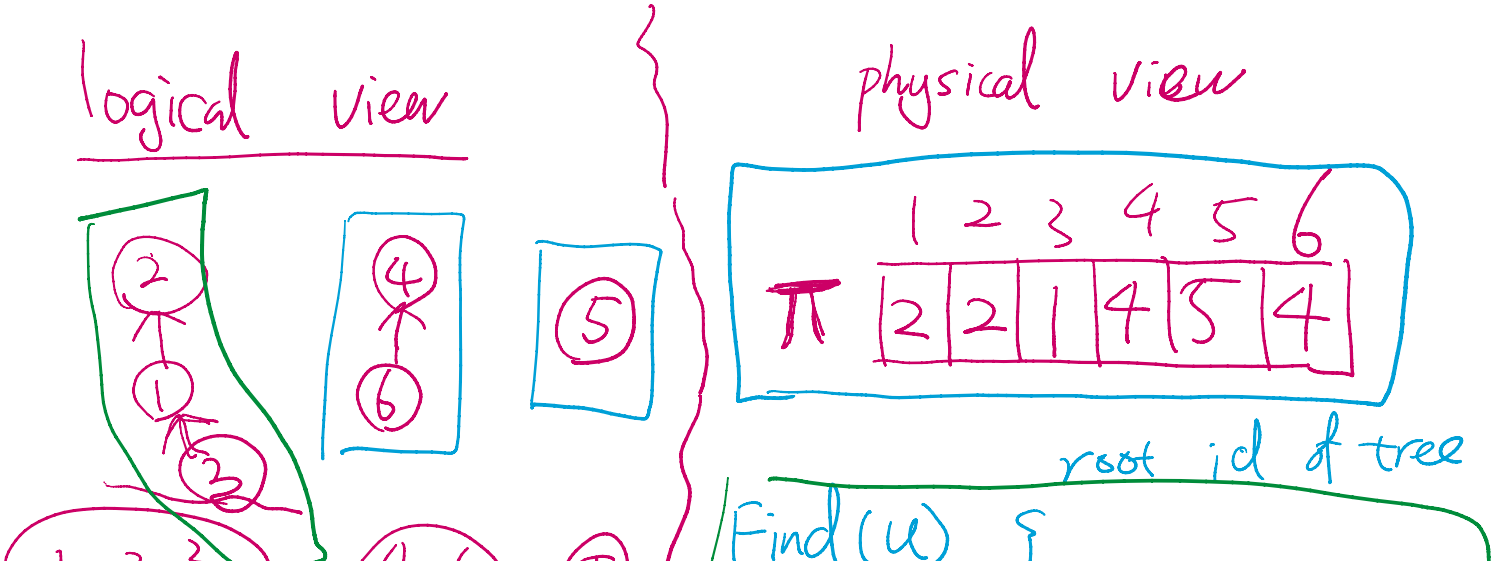






Find(u) : return A[u];

Union(u, v): for (i = 1, ..., n) {
if (A[i] == A[v]) {
A[i] = A[u]
}
}



1,2,3 (4,6) (5)

```
Find(u) {
    if (π[u]==u)
        return u;
    return Find(π[u]);
}
```

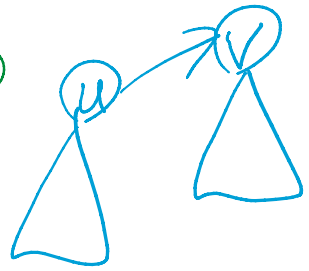
$O(n)$

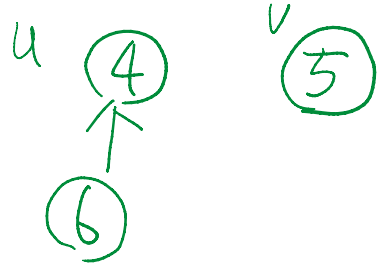
union(4,5)

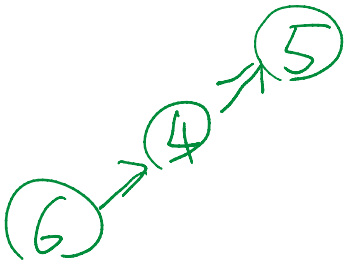
```
union(u, v) {
    π[u] = v;
}
```

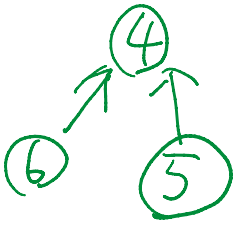
u, v are roots

$O(1)$









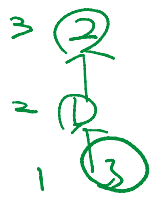
size

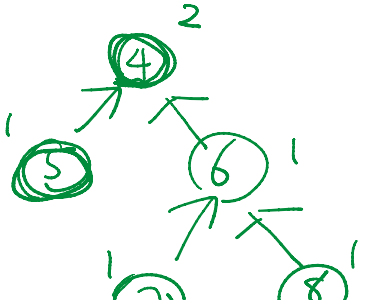
Union by rank

π

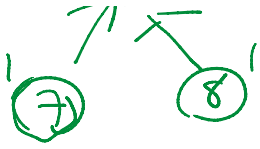
r

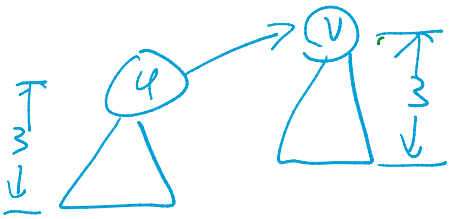
```
union(u, v) {
    if (r[u] < r[v]) {
        π[u] = v;
```





```
π[u] = v;
} else if (r[u] > r[v]) {
    π[v] = u;
} else {
    π[u] = v;
    r[v] = r[v]+1;
}
}
```





- sort(E) — O(m lg n)
- 2m Find, n-1 union — O(m lg n)

O(m lg n)

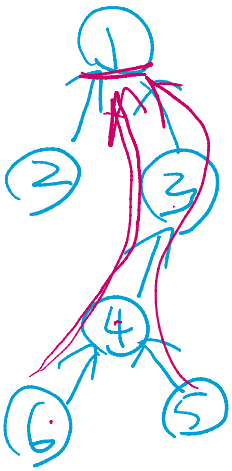
Find O(1)

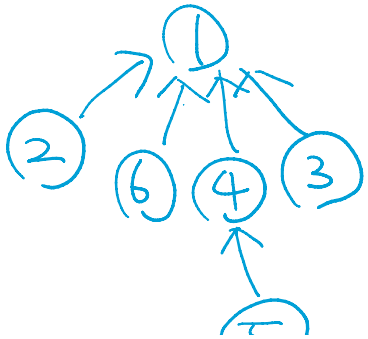
O(1)

O(lg n)

Path compression.

Find(6) Find(5) Find(6) ...






```
Find(u) {
    if (π[u] == u) {
        return u;
    }
    π[u] = Find(π[u])
```

(5)

return π[u];

}

worst case

still O(lgn)

amortized O(1) avg. O(1)

$O(\log^* n)$

O(1)

$\log n$

$\log_2^* n$

$\log^* 2 = 1$

$\log^* 4 = 2$

$\log^* 16 = 3$

$\log^* 65536 = 4$

$\log^* 2^{65536} = 5$

n

log

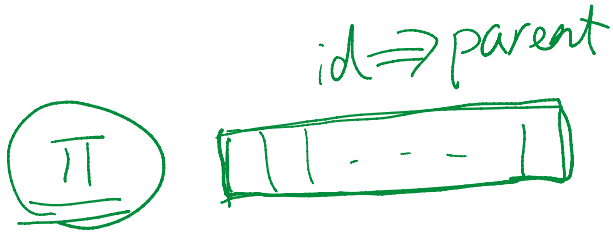
log 8

# of atom in the universe

10^{80}

- Sort(E) $O(m \lg n)$
- 2m Find $O(m \cdot \log^* n)$
- n-1

- $\leq m$ Find $O(m \cdot \log^* n)$
  $n-1$ union



Quiz 1 + Final

Graph

2 question