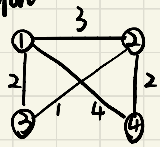
Minimum Spanning Tree (MST)

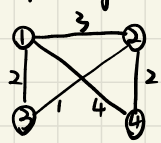
Aug 5

Input: connected undirected graph

Graph

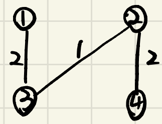


Spanning Tree1 2, .. n



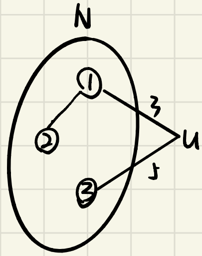
. . . .

Minimum Spanning Tree1

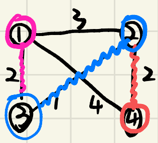


Prim Algorithm

dist[u] : distance from set N to u



dist[u] : 3



N={ } , dist

∞	∞	∞	∞
1	2	3	4

1). N={3}, dist

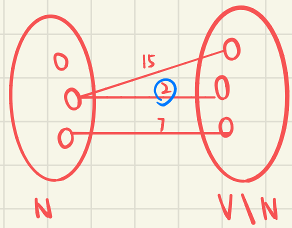
(3,1)	(3,2)		
2	1	{ }	∞
1	2	3	4

2). N={3,2} dist

			(2,4)
2	{	}	2
1	2	3	4

3). N={3,2,4} dist

(3,1)			
2	{		}
1	2	3	4



A cut in a graph is a partition of the vertices into two sets S and T.

Pseudocode

```
prims(G){          O(mlogn)   dist
    N={ }, T={ }              π
    while(|N| != |V|){
        ① pick vertex u ∈ V \ N with the smallest dist[]
        ② N = N ∪ {u} , T = {(u, π[u])} ∪ T
        ③ for( v in G.neighbor(u) ){
            if( Cuv < dist[v] ){
                dist[v] = Cuv ;
                π[v] = u ;
            }
        }
}
```

dist

∞	∞	∞	∞
1	2	3	4

π

-1	-1	-1	-1
1	2	3	4

dist

2 ~~∞~~	1 ~~∞~~	∞	∞
1	2	3	4

π

3 ~~-1~~	3 ~~-1~~	-1	-1
1	2	3	4

⋮

π

3	3	-1	2
1	2	3	4

pick edge (1,3) (2,3) (4,2)

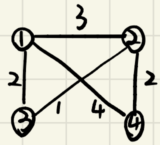
Kruskal's Algorithm

$n < m \le n^2$ $\quad$ $m \log m \le m \log n^2$

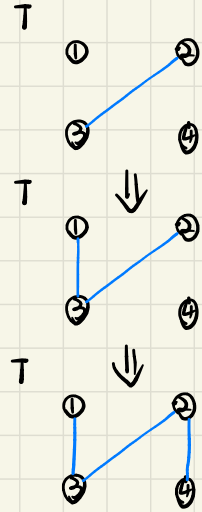
- Sort all edges by their weight $O(m \log n)$
- $T = \{\}$

```
for (edge e in sorted order) {
    if (T U {e} has no cycle) {
        T = T U {e};
    }
}
```

G



T



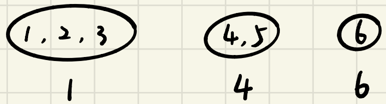
(2,3) = 1
(2,4) = 2
(1,3) = 2
(1,2) = 3
(1,4) = 4

$\Downarrow$ Implement Disjoint Set $\Rightarrow O(m)$

```
for (e = (u,v) in sorted order) {
    if ( Find(u) != Find(v) ) {
        T = T U {u};
        Union(u,v);
    }
}
```

Disjoint Set

- n items $\{1, 2, \cdots, n\}$
- group them into disjoint sets

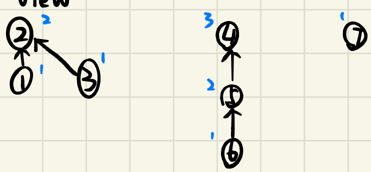


A					
1	1	1	4	4	6
1	2	3	4	5	6

Find(x) could be done in O(1)

Union(x, y)

Logical View



π						
2	2	2	4	4	5	7
1	2	3	4	5	6	7

rank						
1	2	1	3	2	1	1
1	2	3	4	5	6	7

root id is the set id

```
Find(x){            O(log n)
    if(π[u] == u){
        return u;
    }
    return find(π[u]);
}
```

\# of items $\geq 2^{rank-1}$

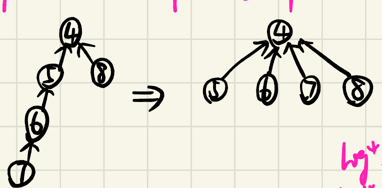
$\Rightarrow$ rank is bounded by log n

Optimization: Union by rank

```
union(u, v){   O(1)
    if(rank[u] < rank[v]){
        π[u] = v;
    } else if(rank[u] > rank[v]){
        π[v] = u;
    } else{
        π[u] = v;
        rank[v] += 1;
    }
}
```

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

Optimization: Path Compression



$\log^* 2 = 1$,

$\log^* 4 = 2$

$\log^* 16 = 3$.

$\log^* 65536 = 4$

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

Assume a sequence of m union & find operations

Overall running time $O(m \log^* n) \Rightarrow O(5m)$