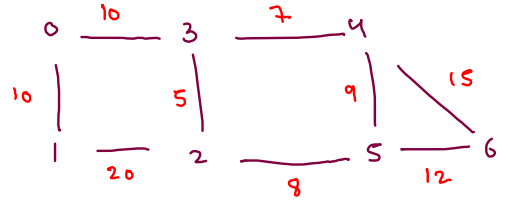
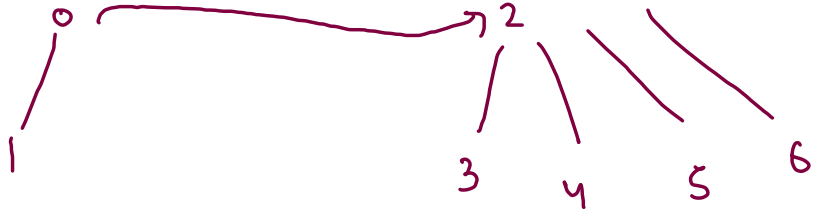
Kruskal Algorithm

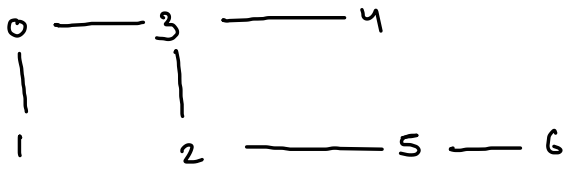
DSU





MST

(min spanning tree)



edges

2-3 @ 5 ✓

3-4 @ 7 ✓

2-5 @ 8 ✓

4-5 @ 9 ✗

0-1 @ 10 ✓

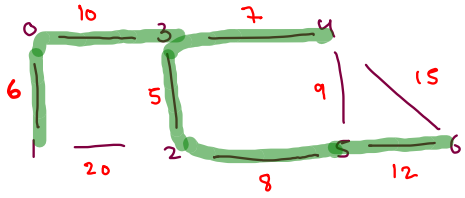
0-3 @ 10 ✓

5-6 @ 12 ✓

4-6 @ 15 ✗

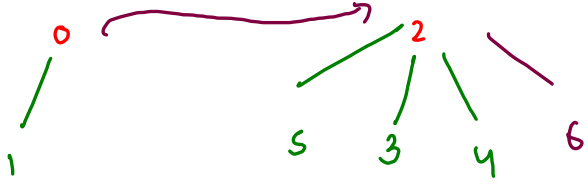
1-2 @ 20 ✗

```java
//sort the array of edges
Pair[]arr = new Pair[pipes.length];

for(int i=0; i < pipes.length;i++) {
    arr[i] = new Pair(pipes[i][0],pipes[i][1],pipes[i][2]);
}

Arrays.sort(arr);
```




```java
//apply dsu
parent = new int[n];
rank = new int[n];

for(int i=0; i < n;i++) {
    parent[i] = i;
    rank[i] = 0;
}

int cost = 0;
for(int i=0; i < arr.length;i++) {
    int u = arr[i].u;
    int v = arr[i].v;
    int wt = arr[i].wt;


    int pu = find(u);
    int pv = find(v);

    if(pu != pv) {
        cost += wt;

        if(rank[pu] < rank[pv]) {
            parent[pu] = pv;
        }
        else if(rank[pu] > rank[pv]) {
            parent[pv] = pu;
        }
        else {
            parent[pu] = pv;
            rank[pv]++;
        }
    }
}
```

Cost = 5 + 6 + 7 + 8 + 10 + 12



edges

2 - 3 @ 5 ✓

0 - 1 @ 6 ✓

3 - 4 @ 7 ✓

2 - 5 @ 8 ✓

4 - 5 @ 9 ✗

0 - 3 @ 10 ✗

5 - 6 @ 12 ✓

4 - 6 @ 15 ✗

1 - 2 @ 20 ✗

Sentence Similarity II

```
["great","acting","skills"]
["fine","drama","talent"]
```

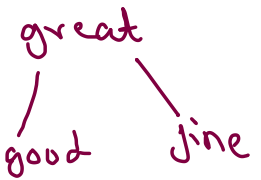
[["great","good"],["fine","good"],["drama","acting"],
["skills","talent"]]

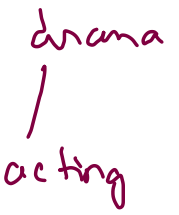
great acting skills

fine drama talent

parent : hm <String, String>

rank : hm <String, Integer>





[["great","good"],
["fine","good"],
["drama","acting"],
["skills","talent"]]

Parent

great -> great

good -> great

fine -> great

drama -> acting

acting -> acting

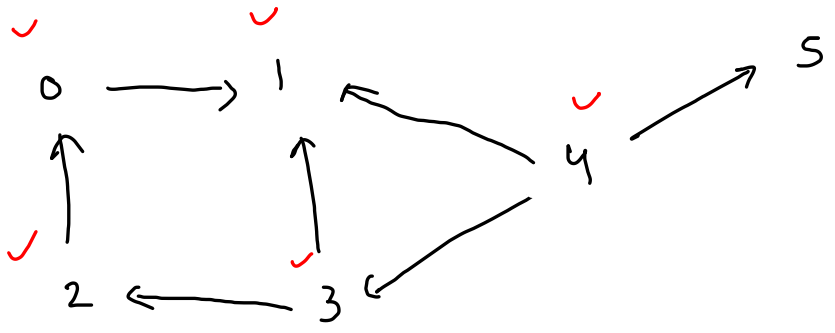
skills -> talent

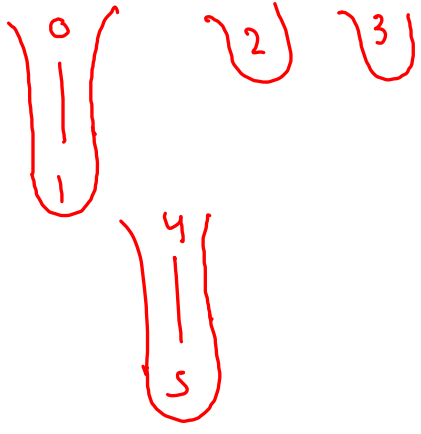
talent -> talent

topological sort → DAG (directed acyclic graph)

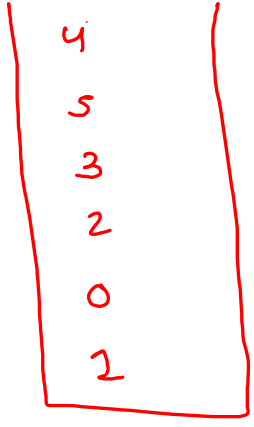
it is a permutation of vertices such that

∀ u→v edges, u should before v.





DFS



ts: 4 5 3 2 0 1

order of work : reverse (ts)

[u→v
u is dependent on v]

Kahn' Algo : (i) if acyclic -> you will get topological sort

what

(ii) if cyclic -> it will get stuck

reason – cycle detetion

how : (a) create an indegree array and those vertices which have 0 indegree add them in queue.

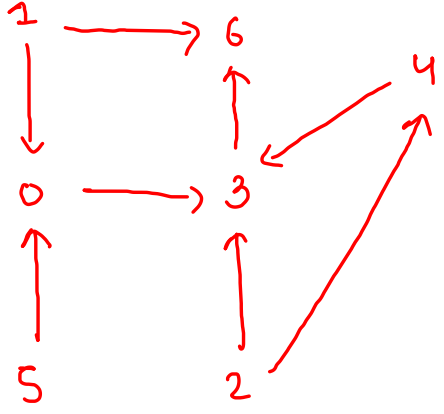
(b) process the queue

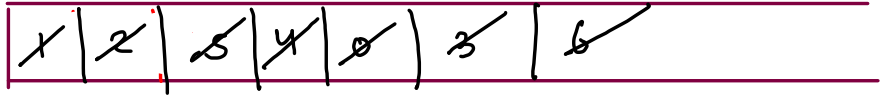
how: (a) create an indegree array and those vertices which have 0 indegree add them in queue

(b) process the queue

indegree:

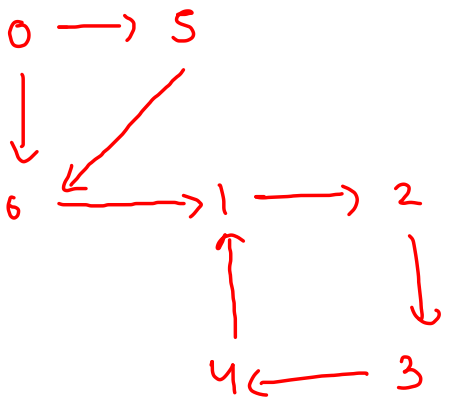
0	1	2	3	4	5	6
~~2~~ 1 0	0	0	~~3~~ 2 1 0	~~1~~ 0	0	0 ~~2~~ 1

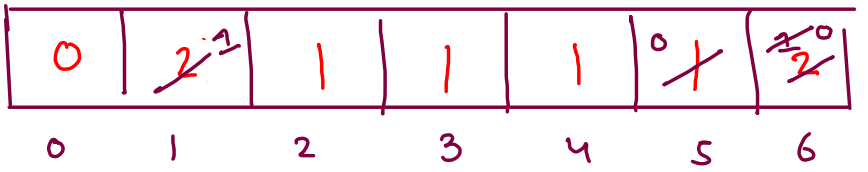




1 2 5 4 0 3 6

cycle detection





ts : 0 5 6