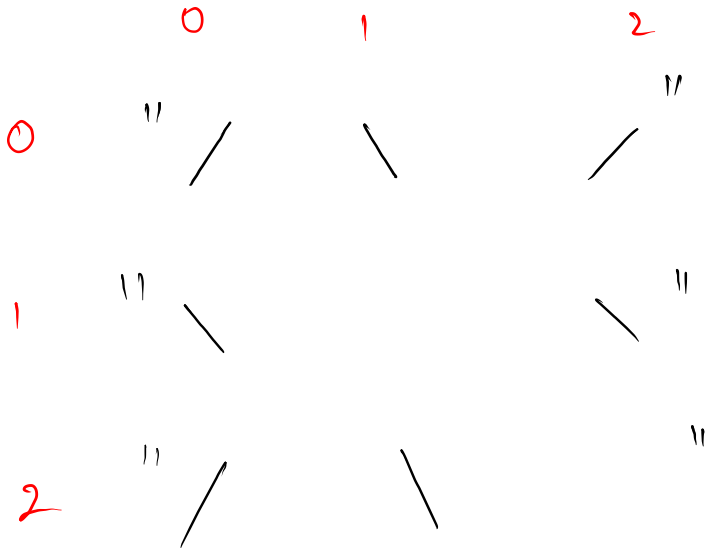
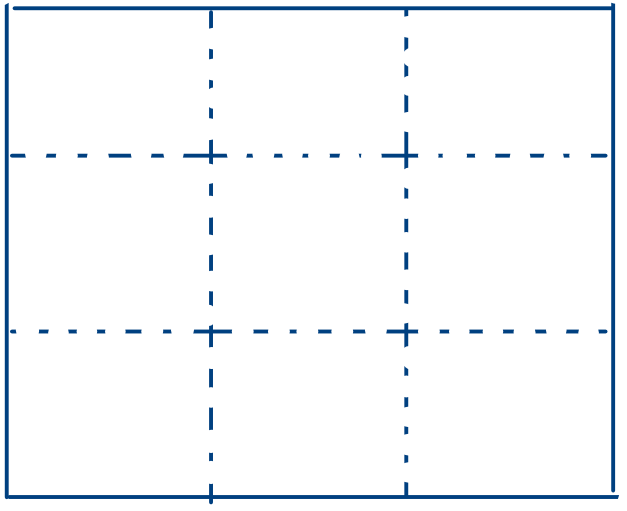
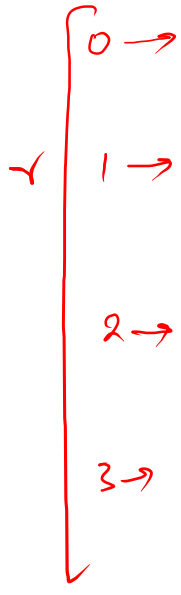
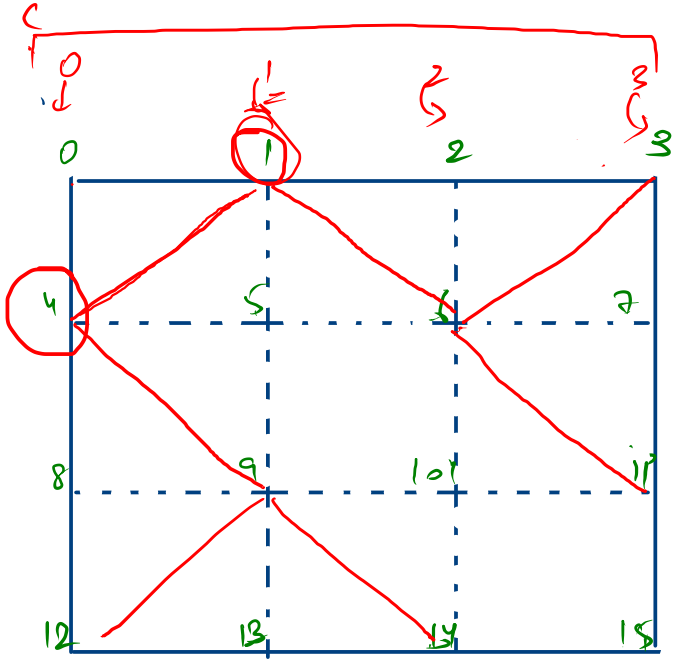
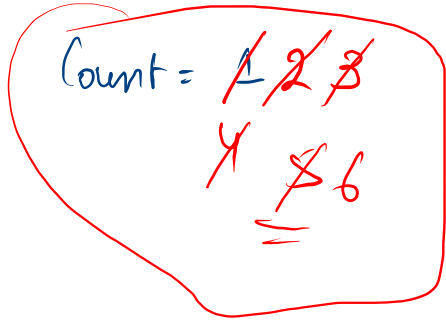
$n = 3$

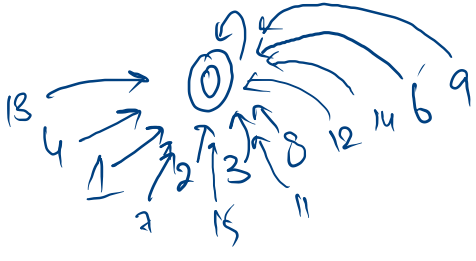












Input

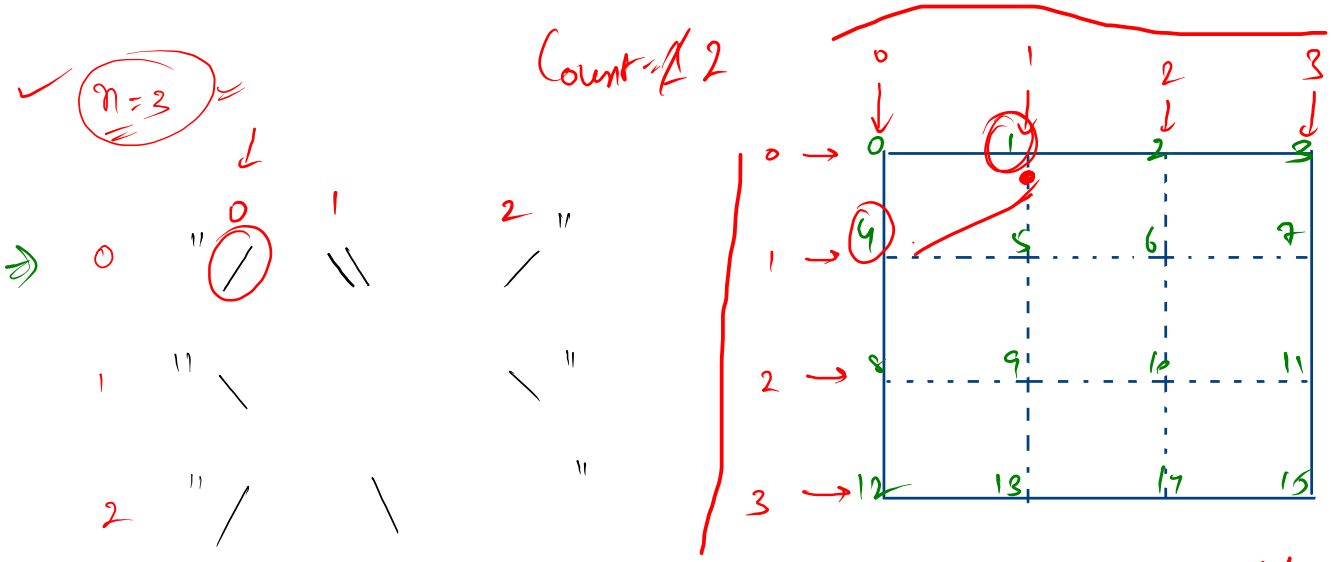
Point matrix

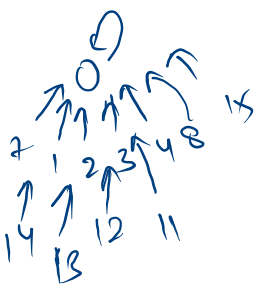
/ $(i,j) \longrightarrow (i, j+1)$ & $(i+1, j)$

\ $(i,j) \Longrightarrow (i,j)$ & $(i+1, j+1)$

$(2,1)$ & $(3,2)$

9 ← $(2,1)$ & $(3,2)$ → 14





Input

Point matrix

$(i,j) \rightarrow (i,j+1)$ & $(i+1,j)$

$(i,j) \Rightarrow (i,j)$ & $(i+1,j+1)$

```java
public int regionsBySlashes(String[] grid) {
    int n = grid.length;

    UnionFind uf = new UnionFind(n);
    for(int r = 0 ; r < n+1 ; r++){
        for(int c = 0 ; c < n+1 ; c++){
            if(r == 0 && c == 0){
                continue;
            }
            if(r == 0 || c == 0 || r == n || c == n){
                int point = (r*(n+1))+c;
                uf.union(0,point);
            }
        }
    }
    for(int i = 0 ; i < grid.length; i++){
        String rowInp = grid[i];
        for(int j = 0 ; j < rowInp.length() ; j++){
            System.out.println(uf.count);
            char ch = rowInp.charAt(j);
            if(ch == '/'){
                int p1 = (i*(n+1))+j+1;
                int p2 = ( (i+1)*(n+1) )+j;
                uf.union(p1,p2);
            }else if(ch == '\\'){
                int p1 = (i*(n+1))+j;
                int p2 = ( (i+1)*(n+1) )+j+1;
                uf.union(p1,p2);
            }
        }
    }
    return uf.count;
```

True

(great, good)

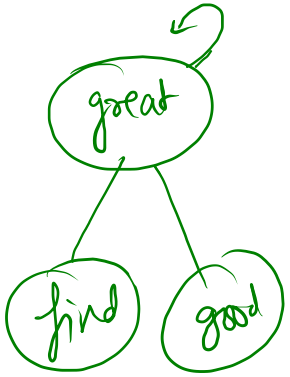
A = B, B = C, ⇒ A = C

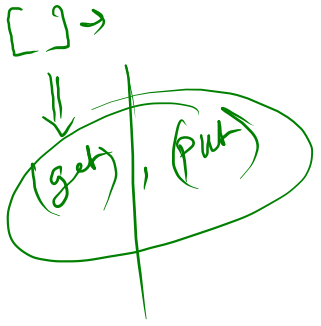
Input: sentence1 = ["great","acting","skills"], sentence2 = ["fine","drama","talent"], similarPairs = [["great","good"],["fine","good"],["drama","acting"], ["skills","talent"]]

[(great, good), (fine, good), (drama, acting), (skills, talent)]

Sentence 1 = ["great", "acting", "skills"]

Sentence 2 = ["fine", "drama", "talent"]





Par

<String>	<String>
great	great
good	great
find	great

Rank

<String>	<Integer>
great	1
good	0
find	0

a == b

a != b

false

a == b

c == d

b != d

b == c

DSU

Edge

Similarity

Equal

Relation

a* c b d

b , d

a a

false

Union → Find

Hashmap

Par

Rank

a b c d

0 1 2 3 4 5 6 7 8 9 10 11 12

26

2

char → idx

idx = ch - 'a'

idx → char

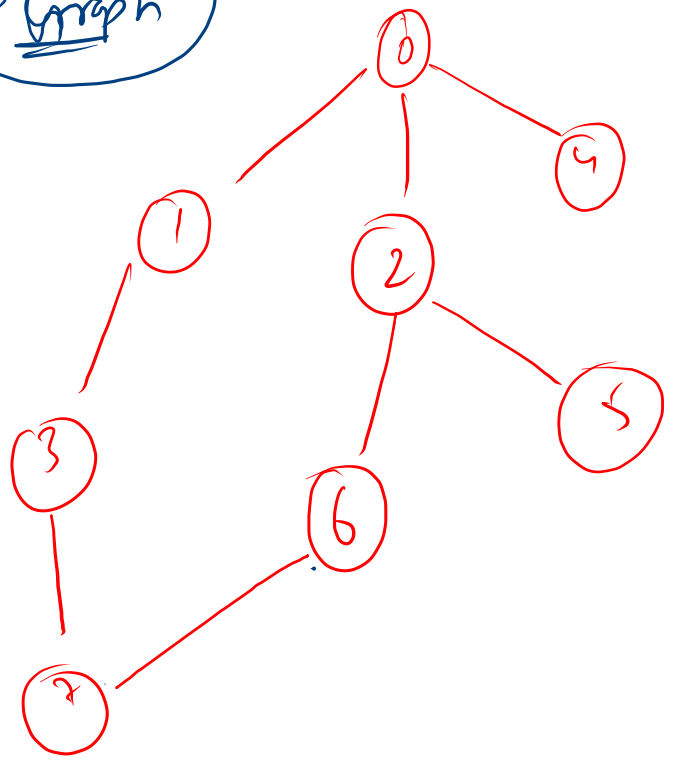
ch = (char)(idx + 'a')

n Nodes

↳ n-1

Edges

Trees ⇒ Graph



0-1

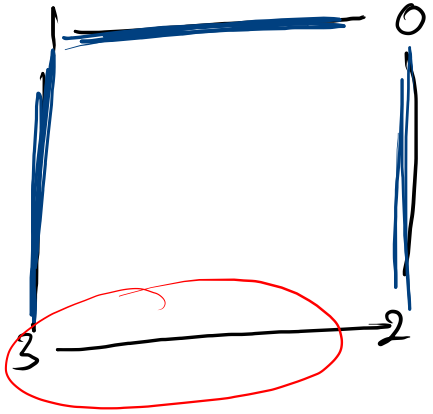
1-3

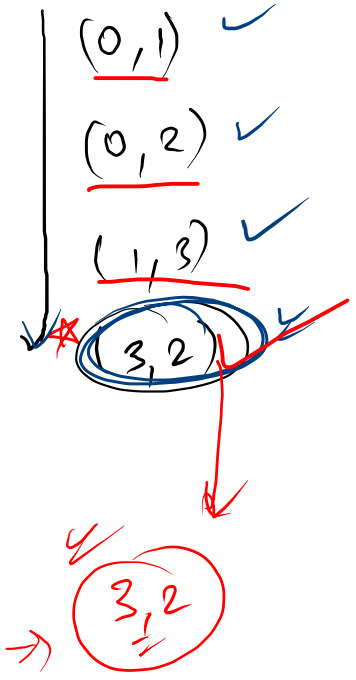
6-7

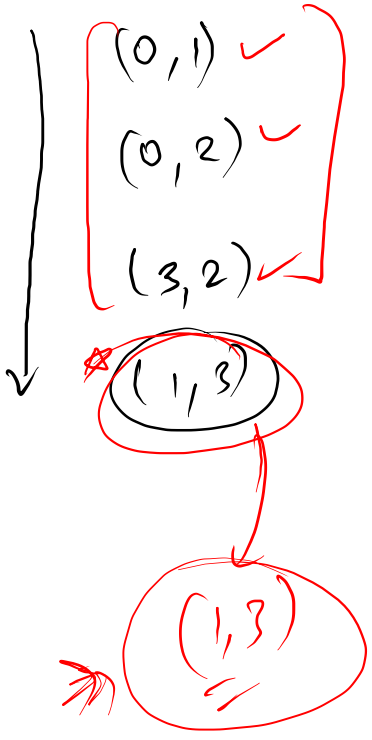
2-6

3-7

0-2

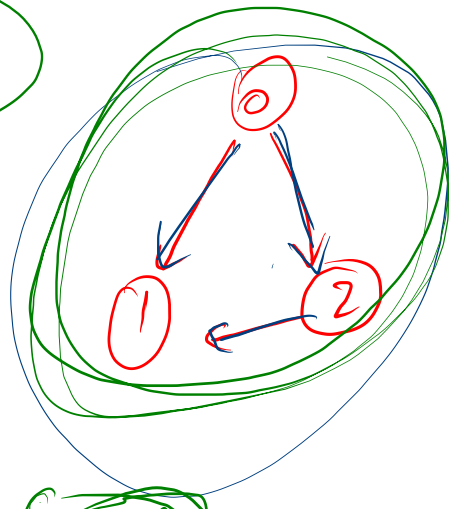


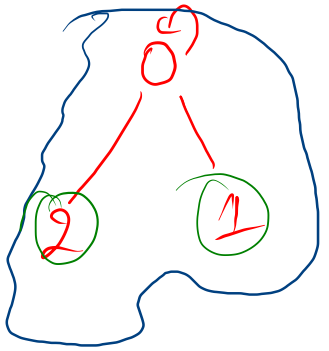




Implement

Directed





Cycle ?

Redundant Connection II

Try

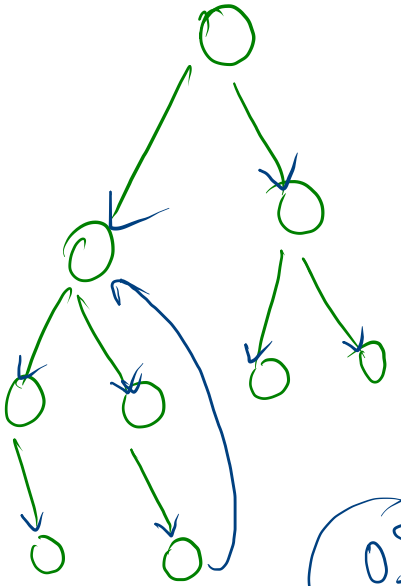
Union Find

directed + Cycle?

(u, v)

u → v

u ⟷ v

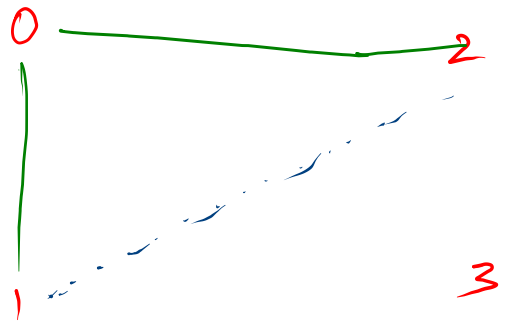


03:25 → 04:10 - 06:00

Doubts

Discussion

Cycle Detection (Undirected Graph) using Union-Find



E(0, 2)

(0, 1)

3