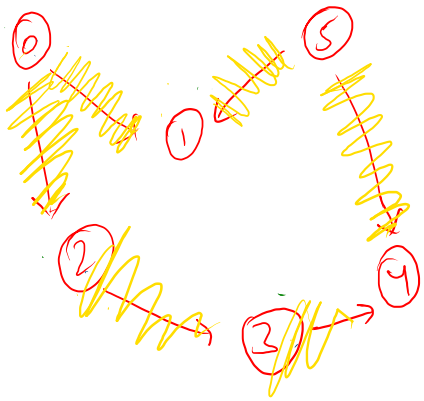
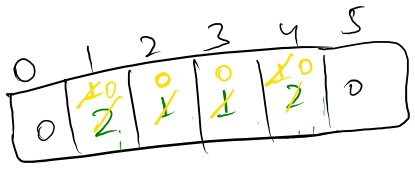
O(V+E)

Toblogical sort
↳ DAG



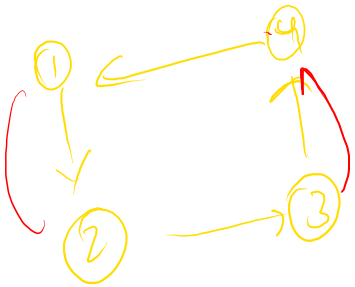
Kahn's algo



0 5 2 1 3 4 ✓

directed
⇓
topological sorting doesn't exist
⇓
cycle detect

DAG
↓
directed
↘ Acyclic

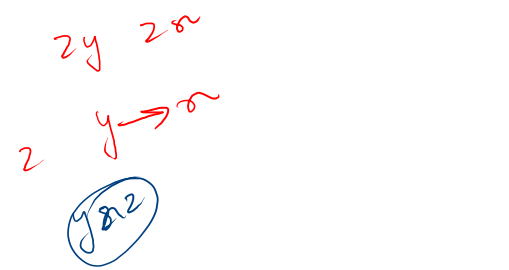


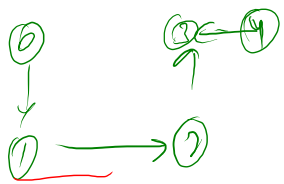
1 2 3 4

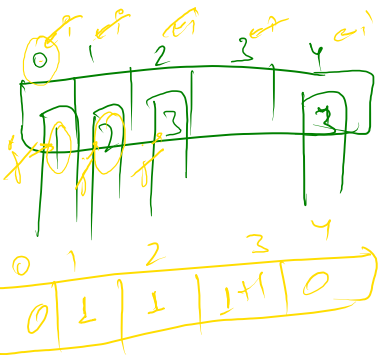
→ 4 1 2 3 → 2 3 4 1

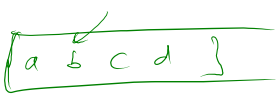
(2

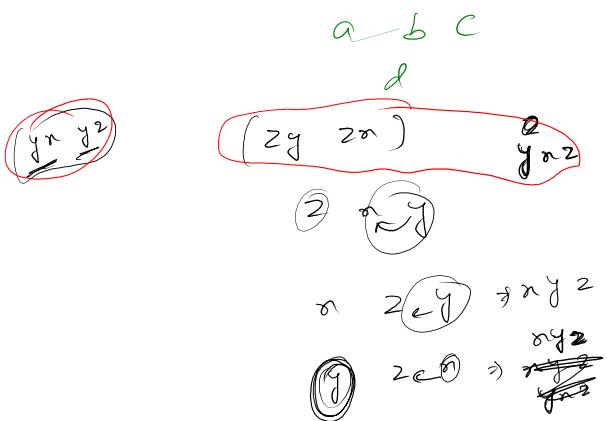
```
public String alienOrder(String[] words) {
    HashMap<Character, HashSet<Character>> graph = new HashMap<>();
    HashMap<Character, Integer> indegree = new HashMap<>();

    for (int i = 0; i < words.length; i++) {
        String st = words[i];
        for (int j = 0; j < st.length(); j++) {
            indegree.put(st.charAt(j), 0);
        }
    }

                }
                flag = true;
                break;
            }
            j++;
        }

        if (flag == false && curr.length() > next.length()) {
            return "";
        }
    }

    LinkedList<Character> queue = new LinkedList<>();
    StringBuilder ans = new StringBuilder();

    for (char ch : indegree.keySet()) {
        if (indegree.get(ch) == 0) {
            queue.addLast(ch);
        }
    }

    int count = 0;
    while (queue.size() > 0) {
        char rem = queue.removeFirst();
        ans.append(rem);
        count++;

        if (graph.containsKey(rem) == true) {
            HashSet<Character> nbrs = graph.get(rem);

            for (char nbr : nbrs) {
                indegree.put(nbr, indegree.get(nbr) - 1);
                if (indegree.get(nbr) == 0) {
                    queue.addLast(nbr);
                }
```











DSU

↳ Disjoint Set union

optimize

(1) Path compression

(2) union by rank

par: 0 1 2 3 4 5 6 7

rank: 1 2 1 1 3 1 1 2

(0,1) (4,5)

(1,2) (0,3)

(3,4)

(6,7)

```
P ↳ int find (int n){
    if (par[n] == n){
        return n;
    }
int temp = find(par[n]);
    par[n] = temp;
    return temp;
}
```

```
P ↳ void union (int n, int y)
ln = find(n);
ly = find(y)
if (ln != ly){
  if (rank[ln] > rank[ly]){
    par[ly] = ln;
  }else if (rank[ln] < rank[ly]){
    par[ln] = ly;
  }else {
    par[ln] = ly;
    rank[ly]++;
  }
}
```

complexity

↳ without opt.

find → O(n)

with Path compression → O(log n)

with union by rank → O(log n)

with both opt. → O(1) → O(α(v))

↳ O(α(v)) → {α(v) < 4}

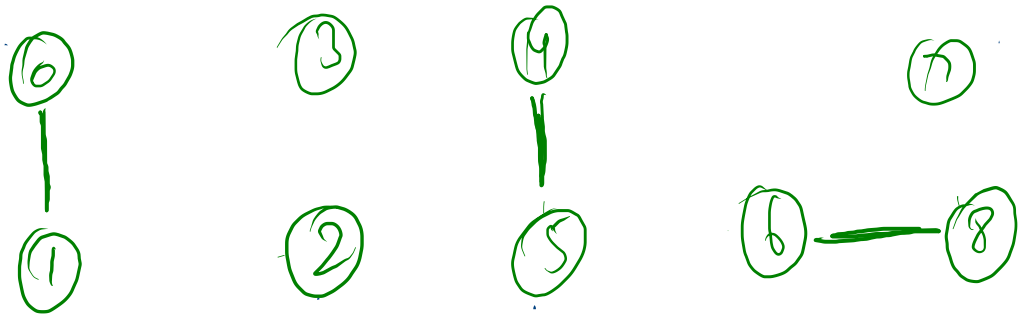
↳ inverse ackermann function

find

↳ amortized analysis

practical

dfs vs dsu



dfs

dsu

(I) $O(V+E_1)$

(II) $O(V+E_2)$

(III) $O(V+E_3)$

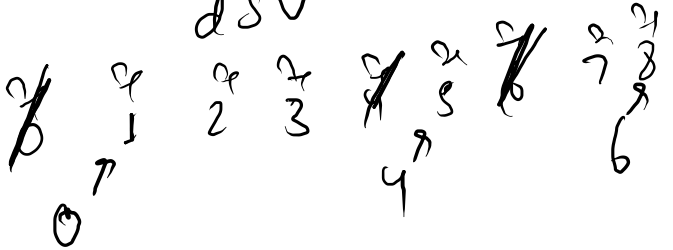
(I) $O(V+1)$

(II) $O(1)$

(III) $O(1)$

$O(V) + O(1)$

dsu



Sets = ~~8~~ ~~8~~ ~~6~~

graph

- Static
  - dfs ≃ dsu
- dynamic
  - dfs
  - dsu ✓