TOPOLOGICAL SORT

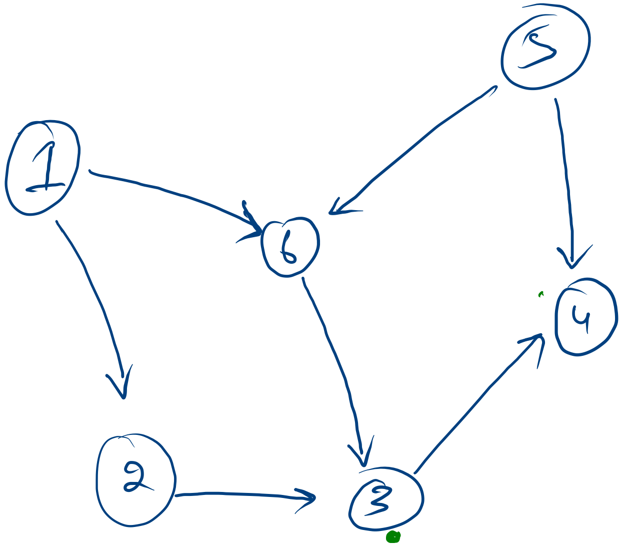
DAG

What

Algo

Application

KHAN'S ALGO



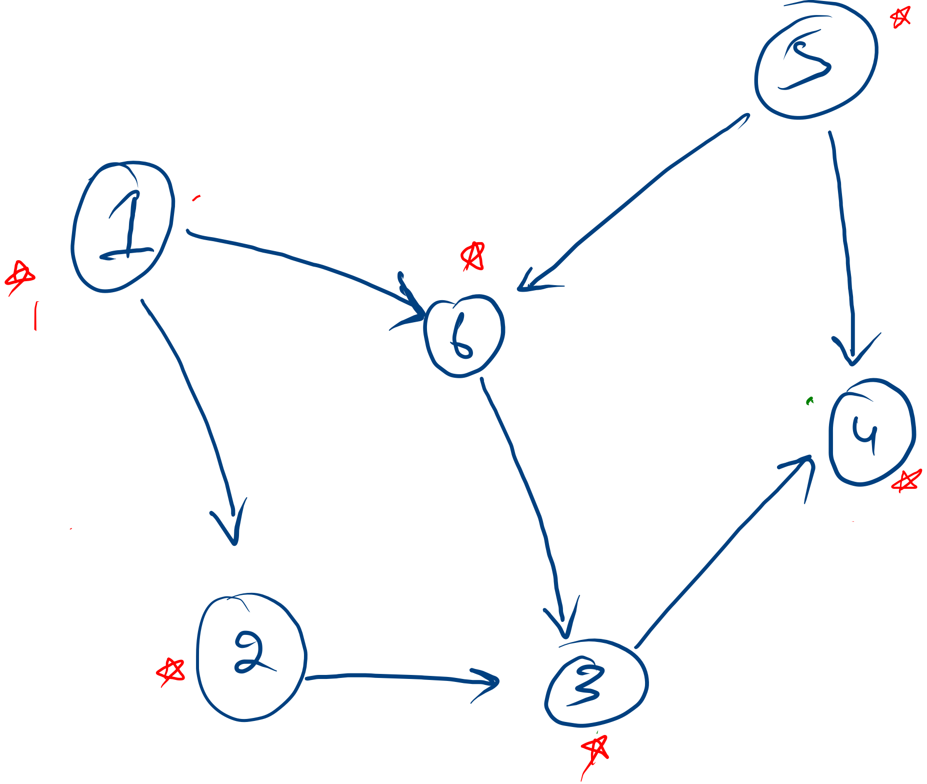
1 5 6 2 3 4 ✓

5 1 6 2 3 4 ✓

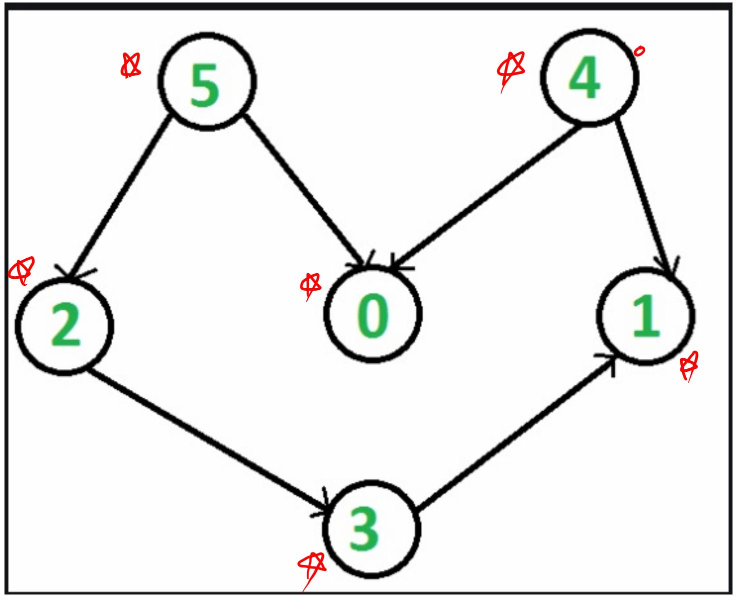
DFS → Recursive

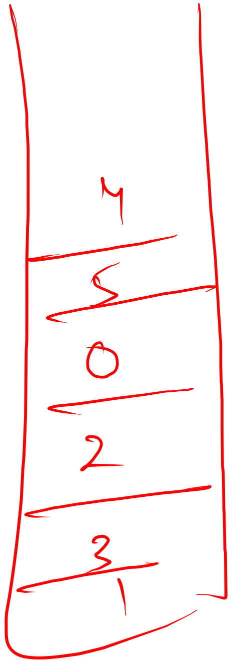
KHAN'S Algo → Iteration

LINEAR ORDERING of vertices with some constraints



5 1 6 2 3 4

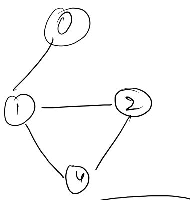




4 5 0 2 3 1

1) dfs

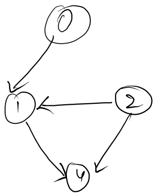
Undirected Edge → degree → no. of edges



TS ⇒ DAG → In / out ← KHAN

directed Edge → In degree

Out degree



vtx	degree
0 →	1
1 →	3
2 →	2
4 →	2

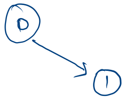
vtx

0 → 1 out degree

1 → 2 in-degree || 1 - out deg.

4 → 2 in-degree || 0 - out deg.

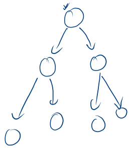
2 → 0 indeg. || 2 - out deg



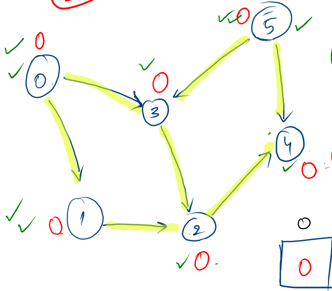
T.S. ⇒ 0 1

vtx	In-degree	Out-degree
0	0	1
1	1	0

DAG



10

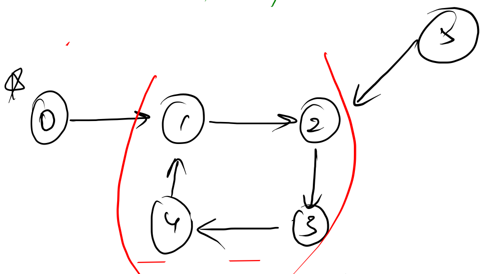


Ans = 0, 5, 1, 3, 2, 4

Count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6 → T.S.

0	1	2	3	4	5
0	1	2	2	2	0
T	T	T	T	T	T

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



~~0~~	~~5~~

Count = ~~0~~ ~~1~~ 2

Cycle

Ans ⇒ 0 5 ✓

T.S. ✗

0	1	2	3	4	5
0	~~2~~ 1	~~2~~ 1	1	1	0
T					T

{ {0, 3} ✓

{1, 2} ✓

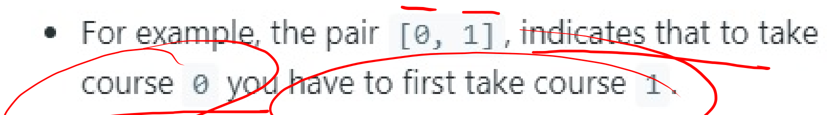
{0, 1} ✓

{3, 2} ✓

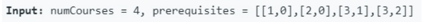
{5, 3} ✓

{5, 4} ✓

{2, 4} } ✓

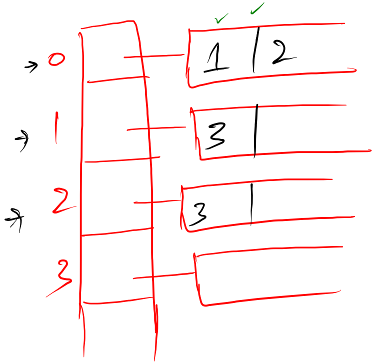


- For example, the pair `[0, 1]`, indicates that to take course `0` you have to first take course `1`.



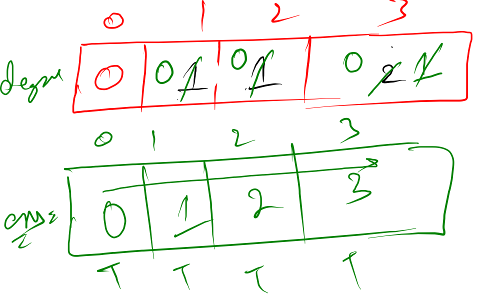
Input: numCourses = 4, prerequisites = [[1,0],[2,0],[3,1],[3,2]]

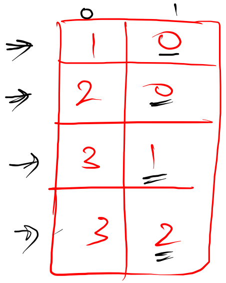
Graph



Count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

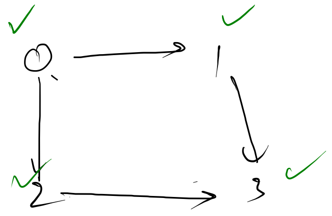
rem = 0

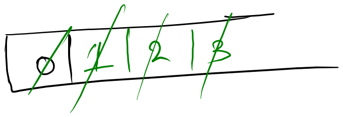


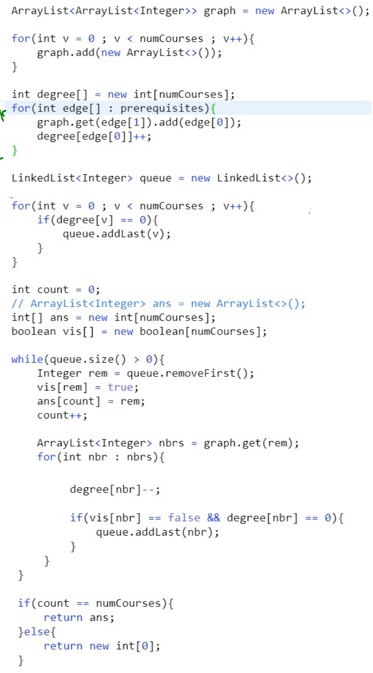


Khans

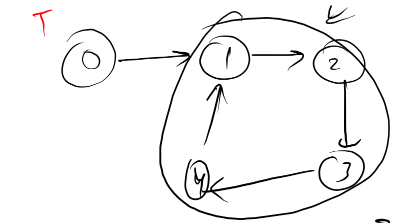
Application

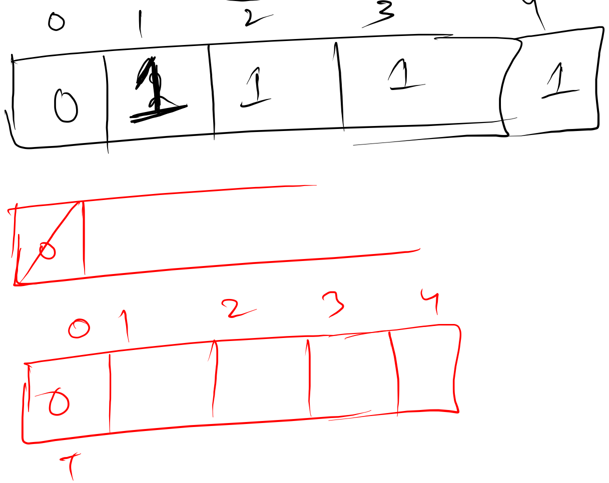




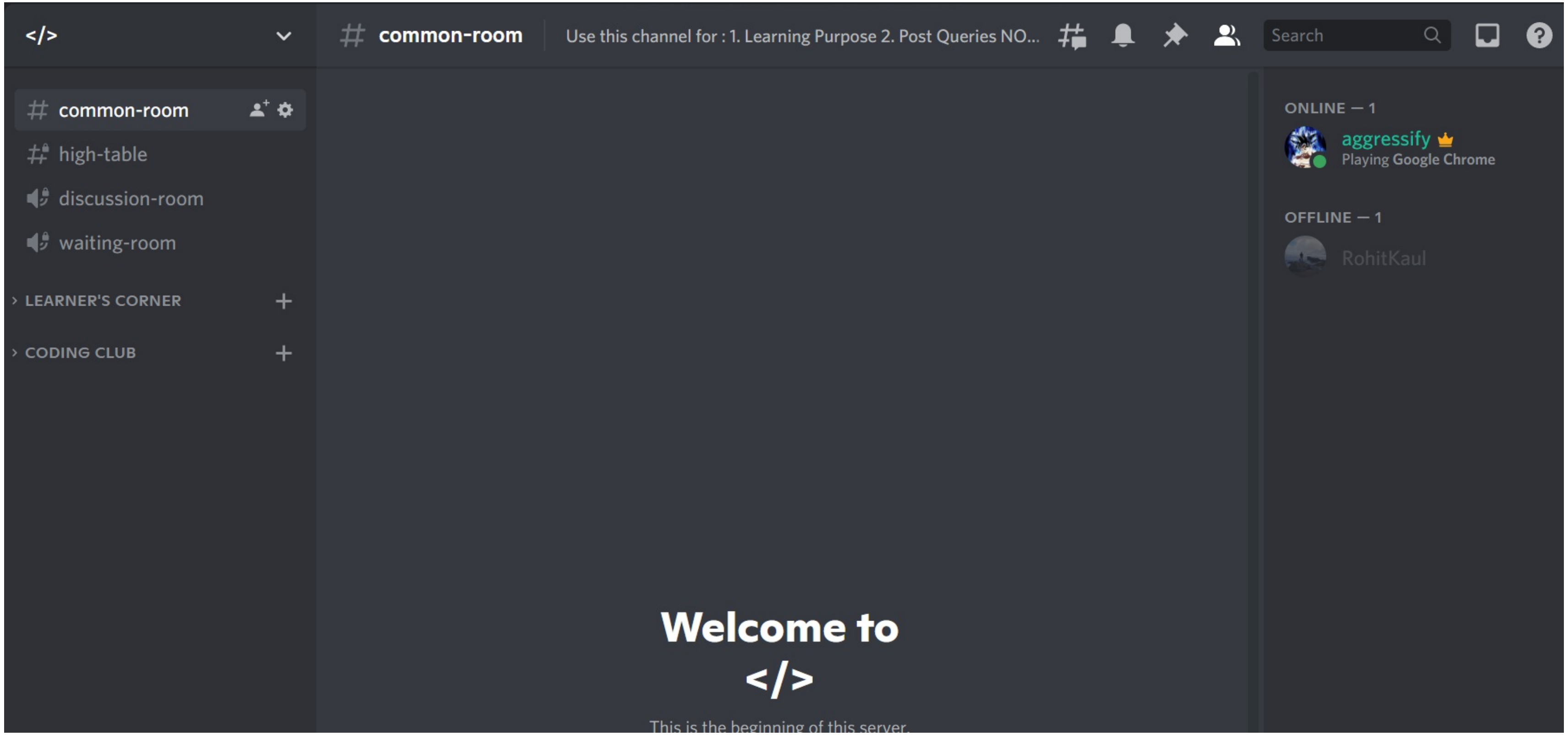


```
ArrayList<ArrayList<Integer>> graph = new ArrayList<>();

for(int v = 0 ; v < numCourses ; v++){
    graph.add(new ArrayList<>());
}

int degree[] = new int[numCourses];
for(int edge[] : prerequisites){
    graph.get(edge[1]).add(edge[0]);
    degree[edge[0]]++;
}

LinkedList<Integer> queue = new LinkedList<>();

for(int v = 0 ; v < numCourses ; v++){
    if(degree[v] == 0){
        queue.addLast(v);
    }
}

int count = 0;
// ArrayList<Integer> ans = new ArrayList<>();
int[] ans = new int[numCourses];
boolean vis[] = new boolean[numCourses];

while(queue.size() > 0){
    Integer rem = queue.removeFirst();
    vis[rem] = true;
    ans[count] = rem;
    count++;

    ArrayList<Integer> nbrs = graph.get(rem);
    for(int nbr : nbrs){

         degree[nbr]--;

         if(vis[nbr] == false && degree[nbr] == 0){
             queue.addLast(nbr);
         }
     }
 }

 if(count == numCourses){
     return ans;
 }else{
     return new int[0];
 }
```





Count = ~~0~~ 1

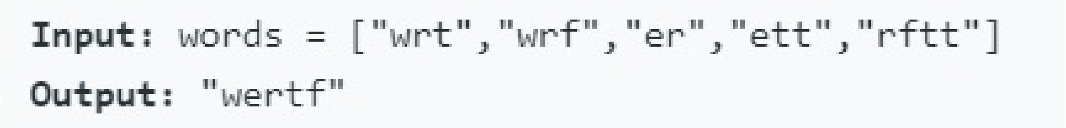


common-room

Use this channel for : 1. Learning Purpose 2. Post Queries NO...

Welcome to
</>

This is the beginning of this server.



```
Input: words = ["wrt","wrf","er","ett","rftt"]
Output: "wertf"
```

rfdc

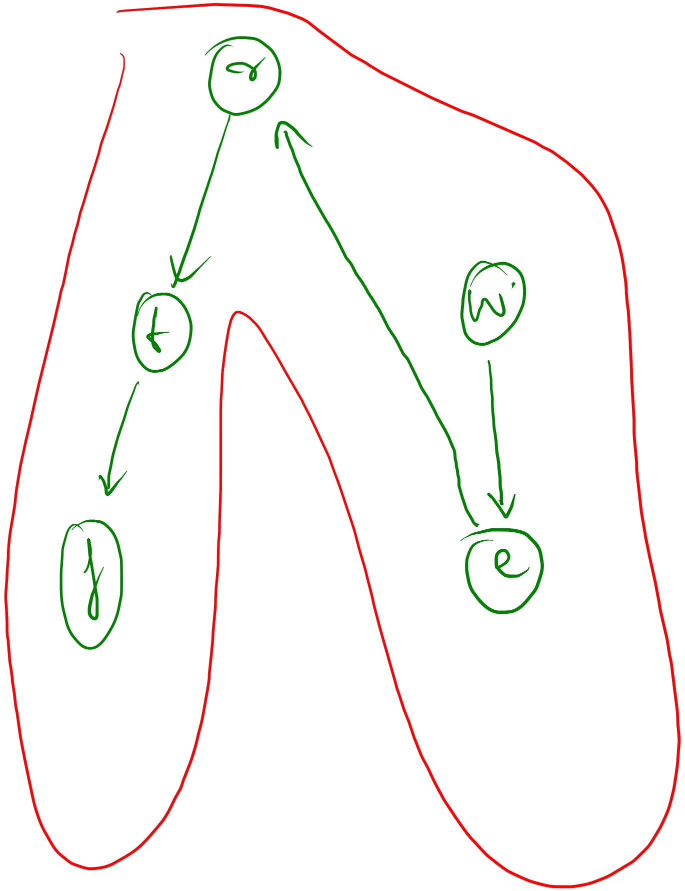
bcmn

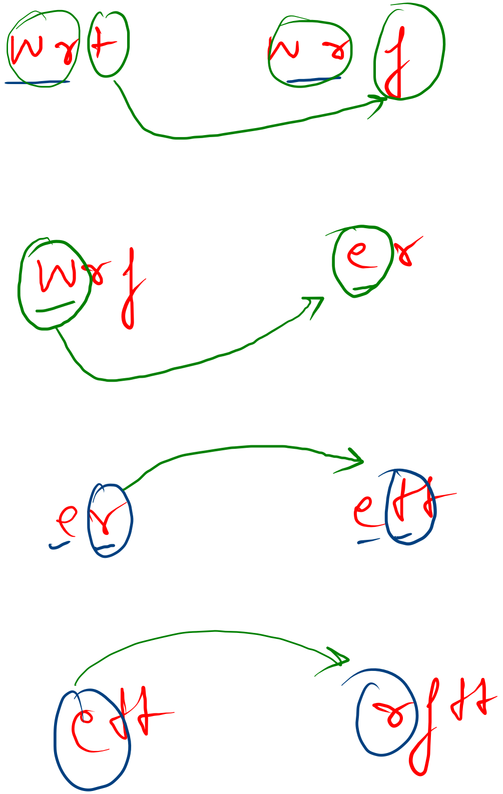
abc

abcd

abcde

wertf

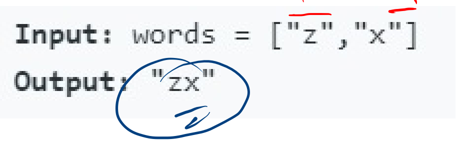




Kahn

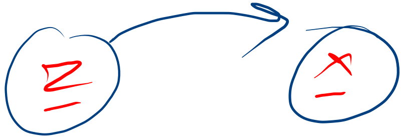
zx , zxy

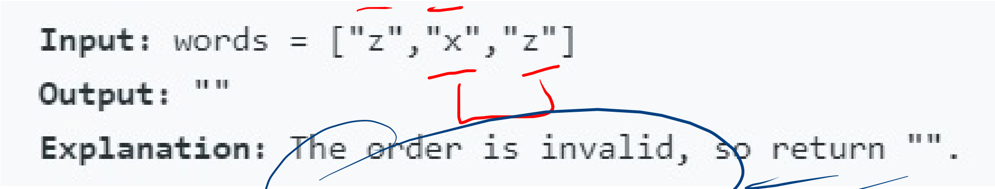
+ Topological



Input: words = ["z","x"]

Output: "zx"

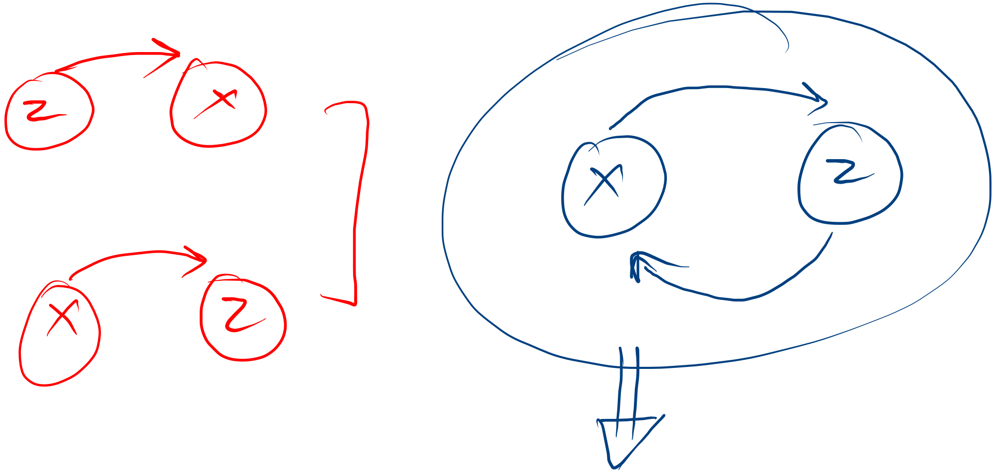




Input: words = ["z","x","z"]

Output: ""

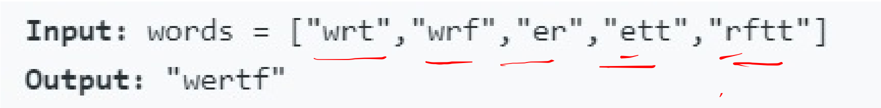
Explanation: The order is invalid, so return "".



Hidden

Input → { z x y , z x o }

Output → ''



```
Input: words = ["wrt","wrf","er","ett","rftt"]
Output: "wertf"
```

st // w ___ ___

HashMap<Character, HashSet<Characters>> graph = new HashMap<>();

t	[f]
w	[e]
r	[t]
e	[r]

ch1 →	ch2
t	f
w	e
r	t
e	r

HashMap<Character, Integer> degree

t	~~0~~ 1
f	1
w	0
e	1
r	~~0~~ 1

In-degree

① graph ✓

② In-degree

degree

③ Visited

④ Count → int

⑤ Queue → FIFO

Khan's

1 ALIEN

2 Kahn → Topological Sort

3 DFS, BFS + DSU