Week 7: Lecture 13 - Lecture 14

#graph

Lecture 13

Problem 1

Connected Components in a Graph | Practice Problems

Solution:

Number of CC depends on number of DFS run in the graph

Problem 2

Monk and the Islands | Practice Problems

Solution:

Just run BFS from the starting point

Problem 3

Contest Page | CodeChef SNSOCIAL

Solution:

Lecture 14

Problem 1

Problem - 1365D - Codeforces

Solution:

Observations

- if a good person neighbors a bad person, the answer is NO
- we should block all empty cells neighboring a bad person
- finally check if all good persons can reach the destination

Bipartite Graph

All edges in a graph (u, v) so that u belongs to part A and v belongs to part B

Verify if a graph is bipartite: use BFS/DFS

- all trees are bipartite graphs

Problem 2

CSES - Building Teams

Verify if an undirected graph contains cycles: checks if neighbor has visited
Verify if a directed graph contains cycles: the previous approach doesn't work. See the following example

```
1 -> 2
|.   |
3 -> 4
```

Directed edges: (1, 2), (1, 3), (2, 4), (3, 4)
Approach: create another array inAction. When starts checking this node I, inAction(i) = 1, when we are done with this node, inActon(i) = 0.
Or, we can count in degree of nodes. Start from a node in graph whose in degree is zero, and remove nodes from graph whose in degree is zero. If finally all nodes are removed in the graph, the directed graph has no cycle.

Topological Sort

- only for directed, acyclic graph

Implementation: code // to do

```
algo: (use DFS)
maintain a stack, for a node u, when all its neighbors v have been visited,
then only push u to the stack.
This makes sure that when popping out the stack u is popped first before the
neighbors.
```

Union Find
Offline RMQ (range minimum query) in $O(\alpha(n))$ on average / Arpa's trick - Competitive Programming Algorithms
Practice problem: Leetcode 952