CS2040S Tutorial 8

AY 24/25 Sem 2 — github/omgeta

Q1. (a) Since every node is visited at most once, it cannot possibly be visited from a different node from the parent node to form a cycle.

(b) They are identical, with the algorithm on trees being a specific application of on a graph.

Q2. Solution:

1. BFS/DFS on any unvisited node, marking all nodes traversed as visited
2. When there are no more nodes, increment count of CC by 1 and repeat step 1

Q3. Solution:

1. BFS on n nodes, tracking edges encountered
2. Tree iff $n-1$ edges for n nodes and connected

Q4. (a.) 1. Undirected graph with sick and healthy nodes, with edges between contacts. Find any path containing only sick nodes. 2. Find any path containing not too many healthy nodes.

(b.) Bipartite graph with people nodes and (location, time) nodes with edges if a person visited at that location and time. Find any (location, time) node with degree > 1.

(c.) Nodes are jobs and edges indicate overlap. Find jos with no edges between them.

(d.) Undirected graph with nodes as students and edges as similarity. Any non-trivial components are cheaters.

(e.) Bipartite graph with children and present nodes, edges as desire. Find edges not sharing any endpoints.

Q5. (a.) Take each node to represent a possible state where a number if assigned to a letter. For example, start search with three nodes $(A,1),(A,2),(A,3)$ then each node connects to another possible state depending on the remaining letters and number permutations. We are finished when we find the combination which satisfies the equation.

(b.) DFS; all paths are of the same length and in worst-case we must visit all permutations anyway so DFS gives a slight optimization if we hit the correct pemutation early. However, asymptotically same as BFS.

(c.) Search finishes when a combination is found. If we can detect unfeasibility of partial assignment early, then we can skip the remaining character pemutations and move onto the next branch.

Q6. Bipartite graph with nodes as students and edges between nodes if students are on the same card.
Consistency: check bipartitite with BFS/DFS in a way that $|G| > |B|$
Sufficiency: there is exactly one valid assignment. To check for consistency, check bipartite using BFS and count alternating levels as part of the same group; prioritise the larger group size for G. To check for sufficiency, ensure difference in groups between components is 0, except for with 1CC where it is always sufficient.

Q7. 1. Single person: consider $f(x)$ as the number of days required to infect for person x and trivially $f(x) = 0$ for a leaf node. We aim to prioritise infecting nodes with more children. Then $f(x) = max(f(c_1)+1, f(c_2)+2, \cdots)$. Our algorithm is to sort neighbours in descending order of $f(c)$ and find max. Time: $O(n \log n)$ 2. Two persons: we need to determine where to cut path between two infected persons. Use binary search to find where to cut using the single-person algorithm. Time: $O(n \log^2 n)$