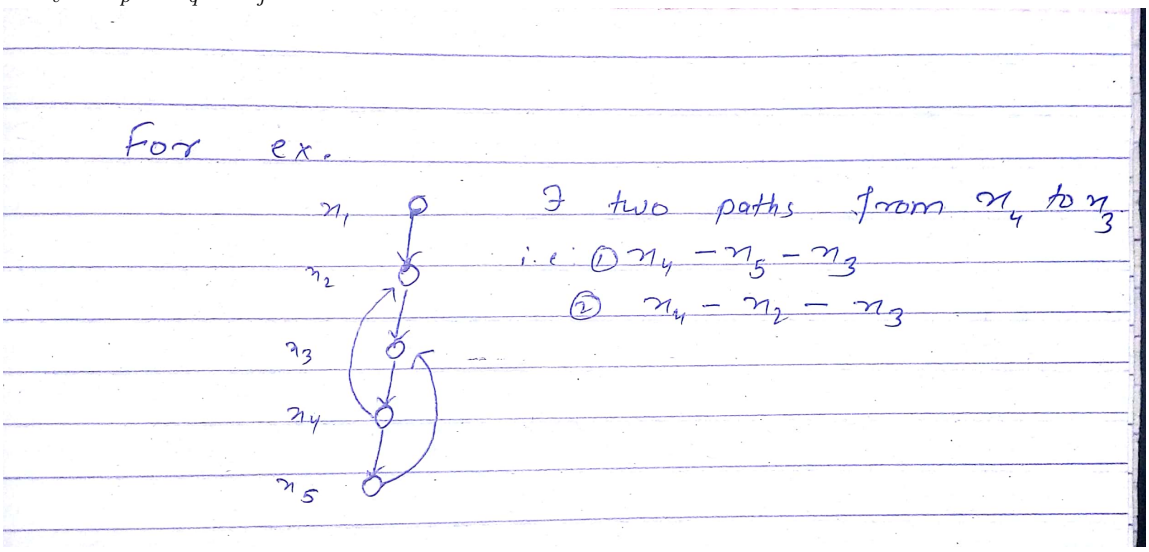
Theoretical Assignment 4

Amit Yadav

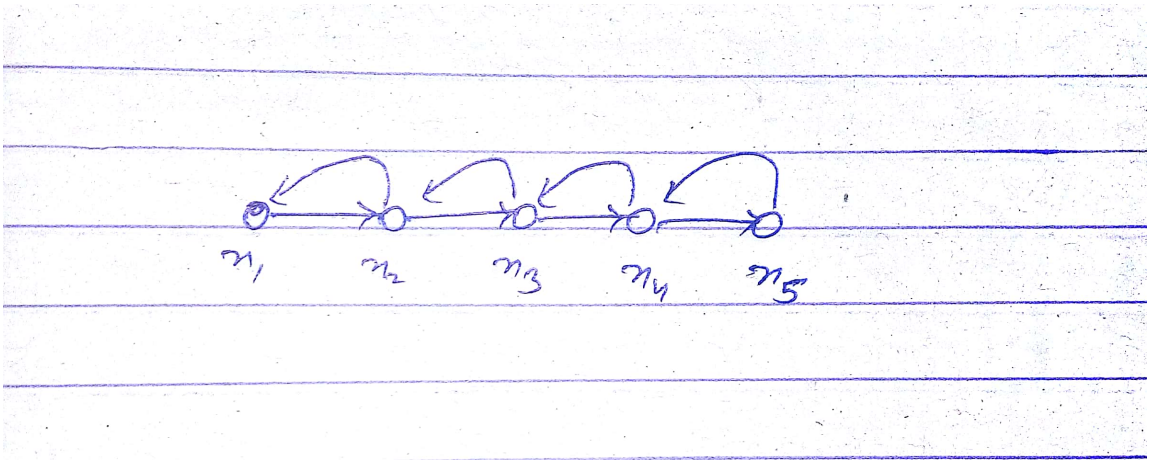
Question 1

The graph can**not** have forward edges or cross edges in a DFS tree of any vertex. But there can be back edges.

- If T_x has forward edge $(u - v)$, it means we can reach from u to v in two different paths, i.e from the path which DFS of x used to visit v, and another using the forward edge directly. So G will no more be unique path graph.
- If T_x has a cross edge $(u - v$, means v had already been visited before u in DFS-visit of x. So we two paths from x to v i.e one that was used by DFS to reach v, while another is $x - u - v$. So G can't be a unique path graph.
- T_x can have back edges, but with some restriction. **Back-edges must not overlap.** Consider there is a path 'p'=$\{x_1 \to x_2 \to \cdots \to x_n\}$ in DFS-traversal of a vertex, let there are two overlapping backedges, $\{x_p \to x_i\}$ and $\{x_q \to x_j\}$, $p, q > i, j$, $p < q$, $i < j$ such that paths x_i *to* x_p and x_j *to* x_q in DFS-tree have some edge $\{x_u, x_v\}$ in common. Then there exists two paths from x_v to x_u in original graph:
  1. x_v to x_p to x_i to x_j
  2. x_v to x_p to x_q to x_j



- Due to this, there can be maximum $\frac{m}{2}$ number of backedges in T_x.*e.g* :



Question 2

Part 1: $a \vee b$ is equivalent to $(\neg a \implies b) \wedge (\neg b \implies a)$
So we $(a \vee b)$ means edge $\{\neg a \; to \; b\}$ and $\{\neg b \; to \; a\}$.
Vertex set=$a, b, \neg a, \neg b$

Part 2: After making the graph, we check if there exist something like $(a \implies \neg a) \wedge (\neg a \implies a)$. It it exists, it means graph is unsatisfiable. Because, 'a' can be either false or true, and in both cases, the expression $(a \implies \neg a) \wedge (\neg a \implies a)$ is always false.
So basically we try to find a cycle including a and $\neg a$ as vertices. Now if we have a such a cycle, means we have $a \implies b \implies \cdots \implies \neg a \cdots \implies a$. Then a and $\neg a$ should have same value, which is not possible. So our expression will be non-satisfiable.

Algorithm:
We find all the SCCs and see if anyone one of them have a variable and its negation both.

1. Run DFS and keep pushing vertices in a stack S as soon as they finish.
2. Take transpose of the graph, i.e invert the direction of all the edges.
3. Run DFS using on vertices by popping them out of the stack (so that we get our desired order i.e decreasing order of their finishing time in previous DFS).
4. We get a DFS forest in which vertices of a tree make a SCC in original graph. Now check if any SCC have a variable and it's own negation. If it is so, the graph is unsatisfiable, else satisfiable.

Part3 :

1. Assign either True or false to any variable, say x_1
2. Assign same value to all variables in the SCC containing that element. So if x_1 is assigned True, then all positive variables in that SCC are assigned true and all negative variables are assigned value false, i.e if $\neg x_2 =$ True, then $x_2 =$ False.
3. Go to another SCC and use previous assiged values to variables, if any assigned variable is present in that SCC. Else, assign True or False to any variable in that SCC and repeat step 2.
4. Do it for all SCCs, and finally we have a valuation.

This assignment satisfies the expression because they are no clashes such that a variable and it's negation are assigned the same value. This comes from the fact that they don't belong to the same SCC. a