Yong Zhou

HW 5

1,

* A), True.

Proof by induction:

* + By definition of a tree which is a connected acyclic undirected graph. We proof by induction on a graph that drop every leave that has one connection with other nodes, the tree will be empty.
    - 1). If the tree has 1 node, it has degree 0, remote the node with degree less than 2. We get G’.
    - 2). If the tree has 2 nodes in G, both 2 nodes have degree of 1, remote the nodes with degree less than 2. We get G’.
    - 3), assume the tree G’ is true with n = k + 1 node.
    - 4), inductive step: For G’ with n = k + 1 node, we add one node to the tree G’ as a leave. By definition of the tree, this new leave has degree of one. When we remote the node, the tree G’ will be empty tree for n = k.
* B). Based on part A), we run DFS function and use color to back track and mark vertex. Keep all the cycles, that is G’

Algorithm:

* insert the edges into an adjacency list.
* Call the DFS function which uses the coloring method to mark the vertex.
* Whenever there is a partially visited vertex, backtrack till the current vertex is reached and mark all of them with cycle numbers. Once all the vertexes are marked, keep it in G’.
* Once DFS is completed, iterate for the edges and push the same marked number edges to another adjacency list.
* Iterate in another adjacency list and print the vertex cycle-number wise.

Proof:

* Based on the part A), this algorithm would delete all

Complexity: O (N + M) as we use adjacency list, where N is number of the vertex and M is the number of edges. Auxiliary Space: O (N + M)

2),

b),

|

========|=

=|=|= ========

===|==

==========

* Observe above graph that if do the more overlap first (red line), need three times to visit the projects. However, the blue line, need visit two times

c), Right

\* Based on she descripts, the set k mutually disjoint, we could not visit two projects at a time. Thus, visit at least k is sufficient. Justify it by contradiction, assume we visit (k-1) time on k projects, there must at least 2 project processing overlap in time. It against the question given.

d), Algorithm: First finish group first for overlaps.

Initial: (, ) 🡨 job i start and finish interval at (, )

Sort all the jobs by earliest finish time first.

For (, ) in ():

If not visited,

visit all unvisited job overlap with i by time

Else:

Continue;

e), Sort jobs take O(n log(n)) and for loop take n. Overall take O(n log(n)).

f), Assume,