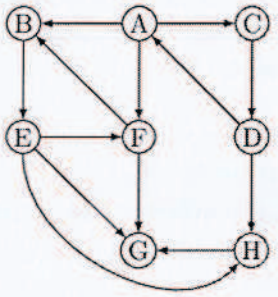
CS204: Algorithms

End Semester, Autumn 2017,

IIT Patna

Duration:3 Hours | Full Marks: 50

1. Comment True/False with proper and **brief justification** (No marks will be awarded without proper justification) (1+1+(2+2)+(1+1)+2=10)
    a. In push-relabel algorithm an edge will disappear if there is a non-saturated push.
    b. Problem X tries to find out if the input undirected graph is having any triangle. Problem X belongs to class NP.
    c. Let G be a weighted connected undirected graph with distinct positive edge weights. If every edge weight is increased by the some constant value c then
        i. Minimum spanning tree of G does not change
        ii. Shortest path between any pair of vertices does not change
    d. Let S be an NP-complete problem and Q is known to be in NP and there is no information about problem R. However, Q is polynomial time reducible to S and S is polynomial-time reducible to R.
        i. Q is in NPC
        ii. R is NP-hard but it may not be in NPC
    e. During a BFS traversal an edge uv is explored from node u. D_u is always less than D_v when D_u and D_v are the distance of node u and v from source s.
2. Prove followings. (4X2.5=10)
    a. A directed acyclic graph must contain at least one source (node which does not have any incoming edge) and at least one target (node does not have any outgoing edge).
    b. There is no path from source to sink in a pre-flow network.
    c. There is no cross edge in an undirected graph.
    d. Vertex cover is a NP complete problem.
3. Answer following questions (4+3+8=15)
    a. Find the below graph (Fig.1). Annotate each edge (using DFS traversal) with edge type (tree edge, back edge, forward edge, cross edge). Whenever there's a choice of vertices, pick the one that is alphabetically first. (show the edge type, no justification is required.)



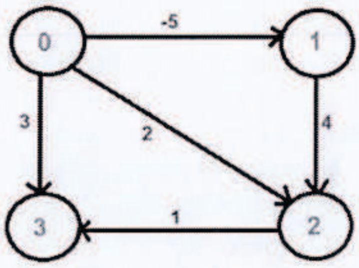


Fig. 1 Fig.2

b. Find the above graph (Fig.2). Apply Johnson algorithm on the graph and show the modified graph with modified edge weight.

c. Assume push-relabel algorithm is being executed in following graph (Fig. 3) after pre-flow initialization. A vertex is enqueued into a queue as soon as it becomes overflowing. When there is a choice, vertex with lower value is enqueued before vertex with higher value. Program will continue while queue is non-empty. During dequeue, a node does either a (push) or (relabel+push) and will be enqueued again if it is still overflowing (if destination node becomes overflowing because of push, it will be enqueued before the source node in case source node remains overflowing after push). Whenever, there is a choice of more than one node to be selected as destination of push, vertex with lower value is always chosen. State of a vertex x is defined as height (x.h) of that vertex x and excess flow (x.e) of that vertex x. Assume time is t_0 after pre-flow initialization. Show states of vertices and queue for all time stamps stating from t_0 until queue becomes empty. Timestamp increases with every dequeue operation. Just show the status, no justification is required.

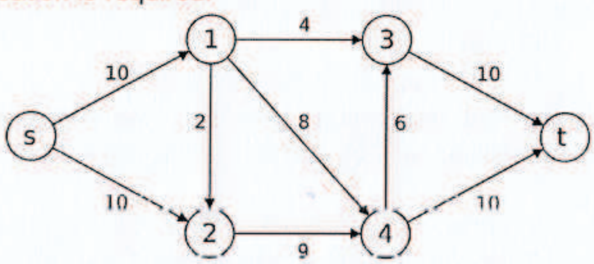


Fig. 3

t_0:

1.e=10	1.h=0
2.e=10	2.h=0
3.e=0	3.h=0
4.e=0	4.h=0

1	2		

4. Given a pattern P[1..m], the prefix function for the pattern P is the function π:{1,2,..m} --> {0,1,..m-1} such that π[q]=max{k: k<q and P_k is suffix of P_q}. Write an O(m) time algorithm for finding π[q] 1<=q<=m of pattern P of size m. Justify that your algorithm is linear. Find prefix of following pattern using your algorithm "ababxabyaba" [Note: P_k is the string consisting of first k characters of P that is P[1..k]] (3+2+2=7)

5. Given a sequence of matrices which are compatible for multiplication. We have to find out the most efficient way to multiply these matrices together. (1+1+1+4+1=8)
   a. Find out number of ways you can multiply a sequence of n matrix
   b. Show that problem satisfy the optimal sub-structure property
   c. Show that sub-problems are overlapping
   d. Propose a dynamic programming algorithm to solve this problem
   e. Discuss about run time complexity of the proposed algorithm