CSE 101

Midterm Examination

W.A. Burkhard & E. Ettinger

April 24, 2008

One handwritten study sheet is allowed.

name Solution

student identifier ____________

1. ____________

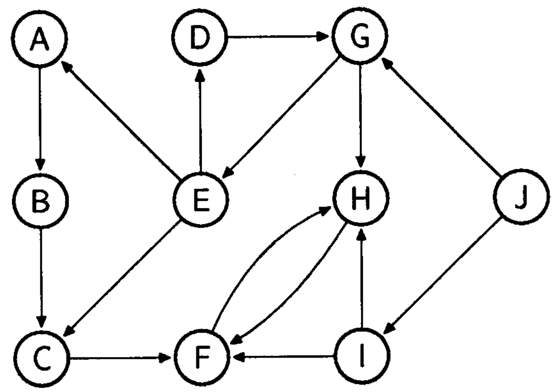
2. ____________

3. ____________

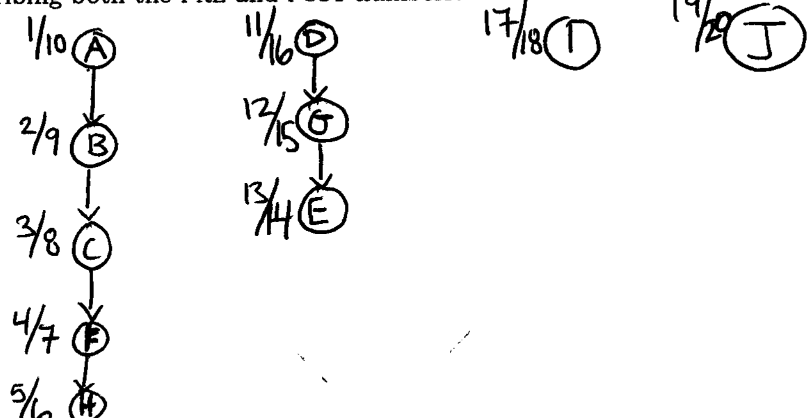
4. ____________

total ____________

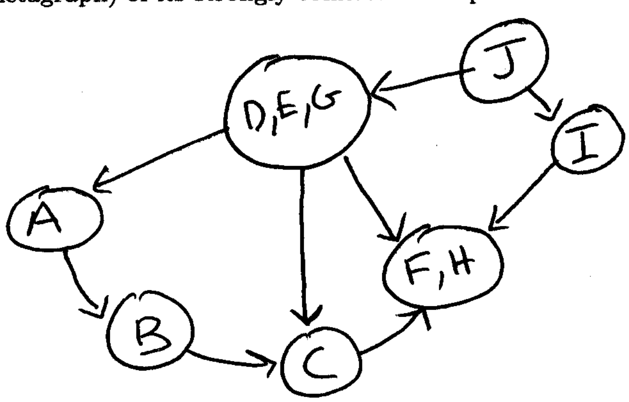
Graph traversal (*10 points*) name Solution



1.a Do a depth first search of the graph, processing nodes in alphabetical order. Show the dfs search forest/tree as well as inscribing both the PRE and POST numbers.



1.b Draw the dag (metagraph) of its strongly connected components.



1.c What is the minimum number of edges you have to include to make the graph strongly connected? What new edges are included? One edge. Connect F,H → J.

Run time calculation (*10 points*) name Solution

Indicate whether $f(n)$ is $O(g(n))$, $\Omega(g(n))$ or $\Theta(g(n))$ and justify your answer.

2.a $f(n) = n^{7.4}$ and $g(n) = 7.4 \log n$.

$f(n)$ is $\Omega(g(n))$ since polynomials dominate logs

2.b $f(n) = n^2$ and $g(n) = 2^{\log_2 \log_2 n}$.

We know $\log \log n < \log n$ and $2^{\log n} = n$

So, n^2 is $\Omega(n)$ meaning $f(n)$ is $\Omega(g(n))$

2.c Determine the run-time of the CountSinkAndSource algorithm. Your answer should given as a function of |V| and |E| only. Briefly explain your answer.

CountSinkAndSource(G = (V,E)):

input: G a directed graph in adjcency list format.

output: sinkCount is set to the number of sink nodes and sourceCount is set to the number of source nodes.

$O(|V|)$ for each v in V: next[v] = 0, previous[v] = 0. sinkCount = 0, sourceCount = 0.

for each v in V:

$O(|E|)$ for each (v, u) in E: next[v]++ , previous[u]++

for each v in V:

$O(|V|)$ if next[v] == 0: sinkCount++

if previous[v] == 0: sourceCount++

+

$O(|V| + |E|)$

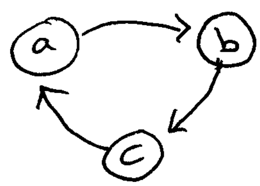
Counterexamples (*10 points*)

name Solution

Each of these statements is *false*; give a counterexample in each case.

3.a Suppose X is a strongly connected component of a directed graph. Then X, with a single node removed, remains a strongly connected component.

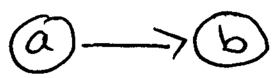
X :



remove c $\Longrightarrow$

no longer an SCC

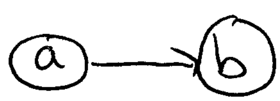
3.b Any directed acyclic graph with n nodes and $\binom{n}{2}$ edges has at least two distinct topological orderings.



$\binom{2}{2} = 1$

a, b is only top. ordering

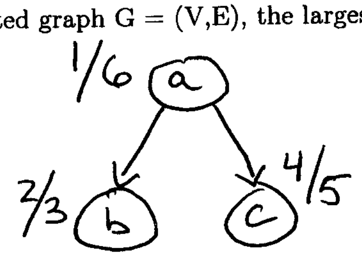
3.c For any directed graph G = (V,E), with every edge having a unit weight/length, there is at least one pair of nodes with shortest path containing exactly |V| edges.



$|V| = 2$

$sp(a,b) = 1 \neq 2$.

3.d For any directed graph G = (V,E), the largest and smallest dfs POST numbers are 2|V| and |V| + 1.



3 is smallest post $\neq |V|+1 = 4$

3.e For any undirected graph, there is a dfs such that at least one pair of nodes u and v satisfies

$$\mathrm{PRE}[v] < \mathrm{PRE}[u] < \mathrm{POST}[v] < \mathrm{POST}[u].$$

This can never happen. Either

$pre[v] < post[v] < pre[u] < post[u]$

or

$pre[v] < pre[u] < post[u] < post[v]$

e.g.

u	v	
a	b	$3 \not< 1$
b	a	$4 \not< 3$
a	b	Symmetric for DFS @ b
b	a	

Algorithm design (*10 points*) name Solution

You are given a strongly connected directed graph G = (V,E) with positive length edges together with a particular node v_0. Specify an efficient algorithm finding shortest paths between *all pairs of nodes* with the restriction that these paths must include node v_0.

Input: Strongly connected directed graph G = (V,E) with positive length edges and node v_0.
Output: A matrix M of lengths of the shortest paths between all pairs of nodes while passing through v_0; i.e. $M_{i,j}$ is the length of the shortest path from node i to node j passing through v_0.

Algorithm

FindAPSPthruV(G, l_e, v_0)

1. Run Dijkstra(G, l_e, v_0) to get $\delta_{v_0}(v)$ = length of SP from v_0 to all $v \in V$.
2. Run Dijkstra(G^R, l_e, v_0) to get $\delta_{v_0}^R(v)$ = length of SP from v to v_0 for all $v \in V$.
3. For all $(u,v) \in V \times V$:
   $M_{u,v} = \delta_{v_0}^R(u) + \delta_{v_0}(v)$
4. Return M

Correctness justification

The SP through v_0 from i to j is of the form $i \to v_0 \to j$ where each portion is a SP in G. δ^R calculates SP from i to v_0 since the reverse graph G^R has such symmetry. δ contains SP lengths from v_0 to all v. Adding them together gives the desired output.

Runtime analysis

1+2) $O((|V|+|E|)\log|V|)$ w/ binary heap PQ

3) $O(|V|^2)$ $\Longrightarrow$ $\boxed{O((|V|+|E|)\log|V| + |V|^2)}$