CS170 First Midterm 23 Sep 1999

NAME: _____

TA: _____

Key (see below): _____

Be clear and concise. You may use the number of points assigned to each problem as a rough estimate for the number of minutes you want to allocate to the problem. The total number of points is 80.

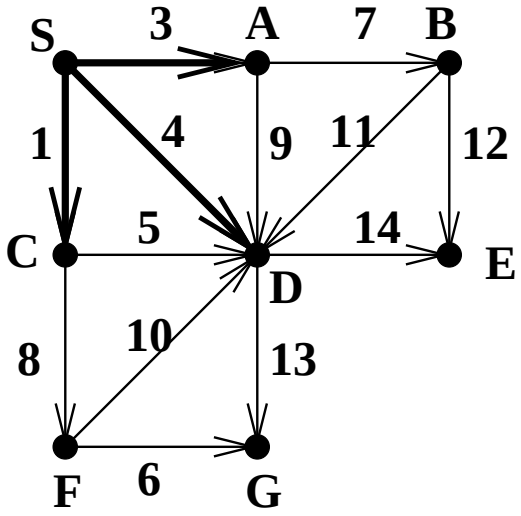
This exam is closed book/closed computer, but you are allowed one sheet of paper (2 sides) for notes to use during the exam. You are allowed to use, without proof, any results proven in class, lecture notes, homework, or the book, but you must state clearly what result you are using.

We are no longer allowed to post grades using SIDs, because of privacy rulings. So we ask you to pick a key that we will use to post your grades on-line. Pick something you can remember, that does not identify you personally, and that is very likely to be unique. Try adding the last 3 digits of your SID to guarantee uniqueness.

1	
2	
3	
4	
Total	

1. **(15 points)** We are running one of these three algorithms on the graph below, where the algorithm has already "processed" the bold-face edges. (Ignore the directions on the edges for Prim's and Kruskal's algorithms.)

   - Prim's for the minimum spanning tree, starting from S.
   - Kruskal's for the minimum spanning tree.
   - Dijkstra's for shortest paths from S.



   Which two edges would be added next in Prim's algorithm? Be sure to indicate the order in which they are added.

   Answer: First (A, B) is added and then (C, F).

   Which two edges would be added next in Kruskal's algorithm? Be sure to indicate the order in which they are added.

   Answer: First (F, G) is added and then (A, B).

   At this point in the running of Dijkstra's algorithm, S has been taken off the top of the heap and marked. Which four vertices would be marked next in Dijkstra's algorithm, i.e. deleted from the top of the heap? Be sure to indicate the order in which they are deleted. Which final edges would Dijkstra's algorithm choose as part of the shortest path to these vertices (i.e. which edge connects to this vertex as part of the shortest path from S)?

   Answer: (1) C is marked, with edge (S, C). (2) A is marked, with edge (S, A). (3) D is marked, with edge (S, D). (4) F is marked, with edge (C, F).

2. **(25 points)** Let $G = (V, E)$ be a directed graph where every edge e has a weight $w(e)$, which may be positive, negative, or zero. The *benefit* of a path consisting of edges $e_1, e_2, ...e_k$ is defined as $\min_{1 \le i \le k} w(e_i)$.

   Give an algorithm which computes the *maximum benefit* path from a given vertex s to another vertex f. For example, if there are two paths from s to f, where Path 1 has edge weights $-1, -2, -3$ and Path 2 has edges weights $-4, 10, 20$, then Path 1 has benefit -3 and Path 2 has benefit -4, so Path 1 has the maximum benefit: $-3 = \max(-3, -4)$. Your algorithm should be as efficient as possible.

   A. Brief description or pseudocode

   Answer: Identical to Dijkstra, except that for each vertex v we maintain the array $W(v)$, which is the min of the weights $w(e)$ on the highest benefit path found so far leading to v. Also we delete the heap entry with the maximum value of $W(v)$. The algorithm is

```
array W, prev; heap H
for each v ∈ V
      W[v] = −∞; prev[v] = nil
W[s] = 0
insert(s, H)
while H not empty
      v = deletemax(H), mark v
      for all edges e = (v, w)
            if w unmarked and W[w] < min(W[v], w(e)) then
                  W[w] = min(W[v], w(e)); prev[w] = v, insert(w, H)
            endif
      end for
endwhile
```

   B. Justification of correctness.

   Answer: Nearly identical to Dijkstra. First, $W[v]$ is always an lower bound on the benefit of the highest benefit path to v, by construction. Making a path longer can only decrease its benefit, so the induction proof that $W[v]$ is the benefit of the best path in the graph examined so far still holds: When v is marked any vertex to which there is a path with a larger benefit has already been explored. Therefore v's maximum benefit path cannot increase after it is marked.

   C. Running time and justification. You may refer to analyses done in class without proof.

   Answer: Identical to Dijkstra: $O(e \log n)$.

3. **(25 points)** A directed graph $G = (V, E)$ is *semiconnected* if for every pair of distinct vertices u and v, there is either a path from u to v, or a path from v to u, or both (u and v lie on a cycle). Show that G is semiconnected if and only if the DAG formed by its strongly connected components has a unique topologically sorted order, i.e. there is a unique way to order the DAG vertices $v_1, v_2, ..., v_n$ such that any edge (v_i, v_j) satisfies $i < j$.

   Answer: Let $\tilde{G}$ be the DAG of SCCs of G. From Homework 3, problem 5 (CLR 23.5-7) G is semiconnected if and only if $\tilde{G}$ has the following property: If v_i, v_{i+1} are consecutive vertices in a topological sort of $\tilde{G}$, then edge (v_i, v_{i+1}) exists in $\tilde{G}$.

   Now assume that G is semiconnected. Then $\tilde{G}$ has the above property. Then there can only be one topologically sorted order, since reordering any vertices creates an edge from v_j to v_k with $j > k$.

   Now assume the topological ordering of $\tilde{G}$ is unique. Then for any pair of vertices v_i and v_{i+1}, we cannot reverse their order. This can only be the case is there is an edge (v_i, v_{i+1}) for all i. Hence G is semiconnected, by the above property.

4. (**15 points**)

True or false? No explanation required, except for partial credit. Each correct answer is worth 1.5 points, but 1.5 points will be *subtracted* for each wrong answer, so answer only if you are reasonably certain.

(a) $n^{2+\sin n} = O(n^2)$ *Answer: False, because* $2 + \sin n$ *is larger than 2.5, say, for infinitely many value of* n.

(b) Dijkstra's algorithm can fail if there are zero edge weights along with positive ones. *Answer: False. The inductive hypothesis can still be proven, since it only requires that making a path longer does* not decrease *its total weight.*

(c) Let G be an undirected, weighted graph. Running an MST algorithm on each biconnected component of G, and then adding the bridge edges to the set of MSTs of each biconnected component, results in a combined MST for the whole graph.
Answer: True, since an MST of G *must connect to the articulation points, and there's no reason an MST algorithm would be affected by not knowing that there was something on the other side of the articulation point, since it must connect to it on both sides anyway to be connected.*

(d) Let $G = (N, E)$ be an undirected graph. Let $N = N_1 \cup N_2$ be a partition of N into nonemtpy disjoint subsets, and G_i the graph consisting of N_i and all edges from E with both endpoints in N_i. Let T_i be an MST of G_i. Then one can construct an MST T of G by connecting T_1 and T_2 by the shortest edge connecting N_1 and N_2.
Answer: False. Let $N_1 = \{b\}$, $N_2 = \{a, c\}$, *and* $E = \{(a, c), (a, b), (b, c)\}$, *with* $w(a, c) = 5$, $w(a, b) = 3$ *and* $w(b, c) = 1$.

(e) $\sum_{i=1}^{n} i^r (\log_2 i)^s = \Theta(n^{r+1}(\log_2 n)^s)$, if $r > 0$ and $s > 0$.
Answer: True, as shown by the two inequalities

$$\sum_{i=1}^{n} i^r (\log_2 i)^s \leq \sum_{i=1}^{n} i^r (\log_2 n)^s = \Theta(n^{r+1}(\log_2 n)^s)$$

$$\sum_{i=1}^{n} i^r (\log_2 i)^s \geq \sum_{i=n/2}^{n} i^r (\log_2 i)^s \geq \sum_{i=n/2}^{n} i^r (\log_2 n/2)^s = \Theta(n^{r+1}(\log_2 n)^s)$$

(f) If we add a directed edge to a directed graph with s strongly connected components, the number of strongly connected components in the new graph can equal any number between 1 and s, but cannot exceed s.
Answer: True: Adding an edge can only create cycles, and so possibly decrease the number of SCCs. Add a single back edge to a chain of length s *to see how to decrease the number of SCCs by any desired number.*

(g) $\sum_{i=1}^{n} (i^2 - i) = n(n+1)(2n+1)/6 - n(n+1)/2$.
Answer: True, by induction.

(h) Run DFS on a directed graph G computing visit times $pre(v)$ and $post(v)$ for each vertex v. An edge (u, v) is a backedge if and only if $pre(v) < pre(u) < post(u) < post(v)$.
Answer: True. For (u, v) *to be a backedge,* u *must be a descendant of* v. u *is a descendant of* v *if and only if DFS first visits* v, *then* u, *then returns from visiting* u *and then returns from visiting* v, *i.e.* $pre(v) < pre(u) < post(u) < post(v)$.

(i) Adding one edge to a DAG must create a cycle.
Answer: False, adding any forward edge will not create a cycle.

(j) You filled in your name, your TA's name, and your key on the first page of this exam.