CSE 100: Algorithm Design and Analysis
Final Exam

Spring 2018

Note: Please write down your name on every page.

Problem	Points earned	Max
1		21
2		7
3		6
4		7
5		6
6		8
7		8
8		8
9		8
10		4
11		6
12		6
13		5
Sum		Max 100

Name _____

1. (21 points). If the statement is correct, choose 'true,' otherwise 'false.' Each subproblem is worth 1.5 points.

   (a) The running time of the Strassen's algorithm for multiplying two n-by-n matrices is $\Theta(n^3)$.
   True False **Sol.** false

   (b) $n \log^5 n = \Omega(n^2)$.
   True False **Sol.** false

   (c) If $f = O(g)$, then $g = O(f)$.
   True False **Sol.** false

   (d) $\log_2 n! = \Omega(n \log n)$.
   True False **Sol.** true

   (e) There is a deterministic $O(n)$ time algorithm for Selection problem; recall that in the Selection problem, we are asked to find the kth smallest element out of n elements.
   True False **Sol.** true

   (f) The average running time of the Randomized Quick-Sort is $O(n \log n)$ if the pivot is chosen uniformly at random.
   True False **Sol.** true

   (g) One can build a max-heap in $O(n)$ time.
   True False **Sol.** true

   (h) If a graph $G = (V, E)$ is connected and has $|V|$ edges, then it has a unique cycle.
   True False **Sol.** true

   (i) Suppose we are given a sequence of n elements of the same key value. If we run a stable sorting algorithm on this input, then the input sequence order remains the same.
   True False **Sol.** true

   (j) The Bellman-Ford algorithm relaxes all edges in each iteration, and performs $|V| - 1$ iterations. Suppose that if $v.d$ is not updated for any vertex in one iteration, then we stop the algorithm. Then, it must be the case that $v.d = \delta(s, v)$ for all $v \in V$; recall $\delta(s, v)$ denotes the shortest path distance from s to v. Assume that there is no negative-weight cycle reachable from s.
   True False **Sol.** true

   (k) In the (s-t) max flow problem, the maximum flow value is equal to the minimum cut value.
   True False **Sol.** true

   (l) Merge-sort is an in-place sorting algorithm.
   True False **Sol.** false

   (m) The running time of Topological sort is $\Theta(E + V \log V)$.
   True False **Sol.** false

   (n) If $T(n) = T(n/2) + \Theta(n)$, then we have $T(n) = \Theta(n \log n)$.
   True False **Sol.** false

2. (7 points) Solve the following recurrence using the *recursion tree method*: $T(n) = 2T(n/2) + n^2$. To get full points, your solution must clearly state the following quantities: the tree depth (be careful with the log base), each subproblem size at depth d, the number of nodes at depth d, workload per node at depth d, (total) workload at depth d. And of course, do not forget to state what is $T(n)$ after all.

   Sol. The tree visualization is omitted. (rough visualization: 1pt)
   For simplicity, let $T(1) = 1$. Tree depth $D = \log_2 n$. 1 pt
   Each subproblem size at depth d: $n/2^d$.(1 pt)
   Number of nodes at depth d: 2^d (1pt)
   WL per node at depth d: $(n/2^d)^2$ (1pt)
   WL at depth d: $n^2/2^d$. (1pt)
   So, $\sum_{d=0}^{D} n^2/2^d = \Theta(n^2)$. (top level dominates) Final answer (1pt)

3. (6 points) You are given n integers, $a_1, a_2, ..., a_n$. Give an algorithm to determine if all the integers are distinct or not. Assume that the simple uniform hashing assumption holds.

   (a) (3 points) Give an $O(n \log n)$ time algorithm without using hashing. **Sol.** Sort n integers in $O(n \log n)$ time. Compare every pair of adjacent integers. If the two are equal for any pair, return false. Otherwise, return true.

   (b) (3 points) Give an $O(n)$ time algorithm via hashing. Briefly explain why your algorithm runs in $O(n)$ time (in expectation). **Sol.** Create a hash table of size $O(n)$ and hash the given integers into the table. If the slot is not empty when trying to insert an element, we test if the slot already contains other elements of the same key value. Since we will perform $O(n)$ insert operations and each insertion takes $O(1)$ time under the uniform hashing assumption, the RT is $O(n)$.

4. (7 points) Interval Selection Problem (ISP). In the ISP, we are given n intervals $(s_1, f_1), (s_2, f_2), ..., (s_n, f_n)$, and are asked to find a largest subset of intervals that are mutually disjoint. For simplicity, let's assume that all s, f values are distinct. We refer to s_i and f_i as interval i's start and finish times, respectively. Also assume that intervals are ordered in increasing order of their finish times. Prove the key lemma that there is an optimal solution that includes the first interval which ends the earliest.

   Sol. See the lecture slides for ch 16

5. (6 points) In the LCS problem, we are given as input two sequences, $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ and would like to find a longest subsequence common to both. Towards this end, we define $c[i, j]$:= length of LCS of X_i and Y_j, where $X_i := \langle x_1, x_2, ...x_i \rangle$ and $Y_j := \langle y_1, y_2, ...y_j \rangle$ and obtain the following recurrence.

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(c[i, j-1], c[i-1, j]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

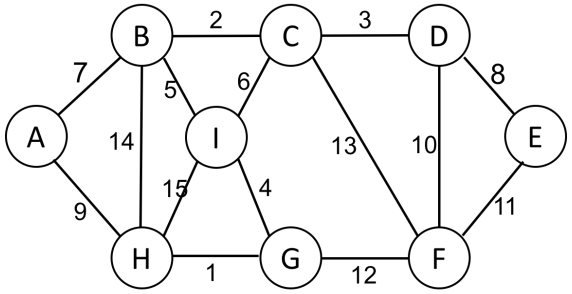
Analyze the (asymptotic) running time:

(a) What is the number of the DP entries/subproblems? **Sol.** $\Theta(mn)$. okay to use $O(.)$.

(b) What is the running time for computing each entry? **Sol.** $\Theta(1)$. okay to use $O(.)$.

(c) What is the running time for computing the optimum? **Sol.** $\Theta(mn)$. okay to use $O(.)$.

6. (8 points) Consider the following weighted undirected graph.

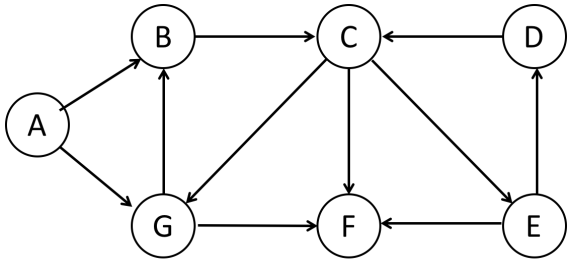


(a) (2 points) Explain why edge (B, I) is safe. In other words, give a cut where the edge is the cheapest edge crossing the cut. **Sol.** $\{A, B, C, D\}$, and the others

(b) (3 points) We would like to run the Kruskal's algorithm on this graph. List the edges appearing in the Minimum Spanning Tree (MST) in the order they are added to the MST.

(c) (3 points) We would like to run the Prim's algorithm on this graph using A as initial vertex. List the edges appearing in the Minimum Spanning Tree (MST) in the order they are added to the MST.

7. (8 points) Consider the following directed graph. We assume that each adjacency list is sorted in *increasing alphabetical* order.



   (a) (4 points) Do depth-first search in G, considering vertices in *increasing alphabetical* order. Show the final result, with vertices labeled with their starting and finishing times, and edges labeled with their type (T/B/F/C). Here T, B, F and C refer to tree, back, forward, and cross edges, respectively.

   (b) (2 points) Based on your results, proceed to run the algorithm you learned in class that finds the strongly connected components of G (show the result of the DFS in G^T, with vertices labeled with their starting and finishing times).

(c) (2 points) Draw the component graph G^{SCC} of G.

8. (8 points) Give a pseudo-code of the Prim's algorithm. Your input is $G = (V, E)$ along with the root vertex $r \in V$ where each edge (u, v) has weight $w(u, v)$. At the end, your algorithm must have computed a MST of G, more precisely $v.\pi$ all $v \in V$. Here, $v.\pi$ denotes v's parent when the MST's root is r. Note that it must be that $r.\pi = NIL$. (You may find the notation $Adj[u]$ useful, which denotes u's neighbors). For your convenience, we give the initialization part of the pseudocode.

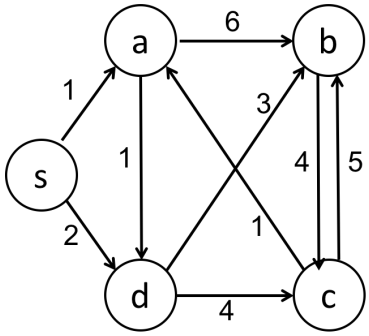
   MST-PRIM(G, w, r)
   1. for each $u \in G.V$
   2. $\quad u.key = \infty$
   3. $\quad u.\pi = NIL$
   5. $r.key = 0$
   6. $Q = G.V$ (here, Q is the minimum priority queue).
   7....

9. (8 points) The following is an incomplete pseudocode of the Bellman-Ford algorithm. The problem is that the algorithm always returns TRUE even when the input graph G has a negative-weight cycle reachable from s. *Correct the pseudocode* **A**.

A. Bellman-Ford(G, w, s)
1. Initialize-Single-Source(G, s)
2. for $i = 1$ to $|G.V| - 1$
3. for each edge $(u, v) \in G.E$
4. Relax(u, v, w)
5. return TRUE

For your notational convenience, we also provide the following pseudocode:
B. Relax(u, v, w)
1. if $v.d > u.d + w(u, v)$
2. $v.d = u.d + w(u, v)$
3. $v.\pi = u$

10. (4 points) Suppose we run the Dijkstra's algorithm on the following graph. Output the vertices in the order they are dequeued from the minimum priority queue maintained by the algorithm.

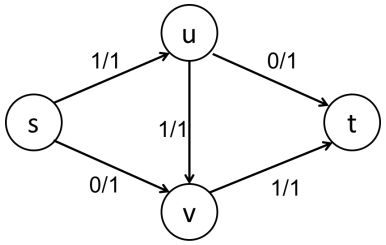


11. (6 points) For a given directed graph $G = (V, E)$, we would like to compute a shortest path from $s \in V$ to $t \in V$ in $O(E + V)$ time. Describe your algorithm *briefly* for each of the following cases, *preferably in words*.

    (a) When all edges have unit weights, i.e., $w(u, v) = 1$ for all $(u, v) \in E$.

    (b) When the graph has no cycle. (Don't say "we run the shortest path algorithm for DAG")

12. (6 points) The following figure shows a flow and flow network where x/y on an edge implies a flow of value x on the edge of capacity y. (i) (2 points) Draw a residual graph G_f for this flow f (do not forget to show each edge's residual capacity). (ii) (2 points) Find an augmenting path. (iii) (2 points) Show the residual graph after adding one unit of flow along the augmenting path.



13. (5 points) In the maximum bipartite matching problem, we are given a bipartite (undirected) graph $G = (V = L \cup R, E)$; note that $L \cap R = \emptyset$ and there are no edges within L or R. To goal is to find a maximum matching. To solve this problem, we modified the input graph into a flow network and found a max flow. How would you modify the following input graph? Do not forget to specify the capacity of each edge.