CSE 100: Algorithm Design and Analysis
Final Exam

Spring 2021

- This is a take-home exam with no proctoring. You can start at any time once you get access to this exam. You have to submit your solution by due time noted in the assignment page, through CatCourses (Final Exam under Assignments). You can resubmit any number of times until the deadline. If there are any technical issues with uploading, it is extremely important to immediately contact the instructor or the TA.
- This is an open-book exam. The **only** resources you can use during the exam are all **course materials** uploaded to CatCourses plus the **textbook**. In particular, this means that you are NOT allowed to search for solutions on the internet. No calculator is allowed. You have to take the exam by yourself.
- There are 12 problems in total. You can earn some partial points if you show progress even if you can't solve problems completely.
- Please make your answers concise as well as precise. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer.
- If you have questions, you can email the instructor and the TAs. However, we may not answer your questions on time as we can't be available all the time.

You're required to take the following honor pledge prior to taking the exam.

By completing this exam, I acknowledge and confirm that I will not give or receive any unauthorized assistance on this examination. I will conduct myself within the guidelines of the university academic integrity guidelines.

1. (15 points). If the statement is correct, choose 'true,' otherwise 'false.' Each subproblem is worth 1 point.

   (a) $n^2 = \Omega(n \log n)$.
   (b) $2^n = O(n^{100} \log n)$.
   (c) $\sum_{i=1}^{n} i = O(n)$.
   (d) The decision tree of any comparison based sorting algorithm has a height $\Omega(n \log n)$.
   (e) If $f = O(g)$ and $g = O(h)$, then $f = O(h)$.
   (f) If $T(n) = 2T(n/2) + n$, then we have $T(n) = O(n \log n)$.
   (g) If $T(n) = T(\frac{9}{10}n) + n$, then we have $T(n) = O(n)$.
   (h) The adjacency matrix representation of a graph $G = (V, E)$ needs memory $O(|E| + |V|)$.
   (i) If G is undirected, then G and G^T have the same adjacency matrix representation.
   (j) There is a linear time algorithm that finds a median in $O(n)$ time.
   (k) If an undirected graph $G = (V, E)$ has $|V| - 1$ edges, it must be a tree.
   (l) If an undirected graph $G = (V, E)$ is connected and its edges have distinct weights, G has a unique minimum spanning tree.
   (m) One can build a max-heap in $O(n)$ time.
   (n) In the (s-t) max flow problem, the maximum flow value is equal to the minimum cut value.
   (o) If $T(n) = T(n/2) + \Theta(n)$, then we have $T(n) = \Theta(n \log n)$.

2. (6 points). Just give the algorithm name(s):

   (a) (2 points) Name two sorting algorithms whose (worst-case) running time is $O(n \log n)$.
   (b) (2 points) Given a directed graph $G = (V, E)$ where all edges have a unit weight/distance, and two vertices $s, t \in V$, we want to find a shortest path from s to t in $O(E + V)$ time. Which algorithm would you use?
   (c) (2 points) Given a directed graph $G = (V, E)$ where every edge has positive weight/distance, and two vertices $s, t \in V$, we want to find a shortest path from s to t. Which algorithm would you use? You should use the fastest algorithm.

3. (7 points) Assume the following hypothetical divide-and-conquer algorithm for matrix multiplication.

```
MAGIC-MULTIPLY($A, B$)
1. $n = A.rows$
2. let $C$ be a new $n \times n$ matrix
3. if $n == 1$
4.     $c_{11} = a_{11} \cdot b_{11}$
5. else partition each of $A, B$, and $C$ into four $n/2 \times n/2$ sub-matrices indicated by the subscripts (it is not important how this is done but you may assume this takes $\Theta(1)$ time)
6.         $C_1$ = MAGIC-MULTIPLY($A_1, B_2$)
7.         $C_1$ = MAGIC-MULTIPLY($A_2, B_3$)
8.         $C_1$ = MAGIC-MULTIPLY$A_3, B_4$)
9.         $C_1$ = MAGIC-MULTIPLY($A_4, B_1$)
```

10. **return** C

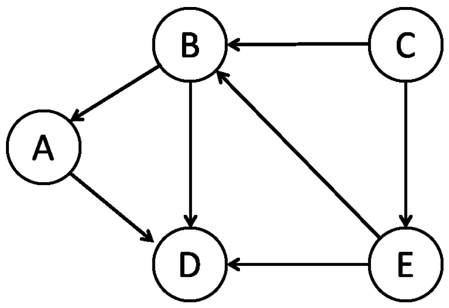
Give the recurrence for the running time of MAGIC-MULTIPLY and use a recursion tree to determine a good asymptotic upper bound on the recurrence. Your solution should be formatted as shown below. We let $T(n)$ denote the running time when input matrices are n by n. You need to provide **all steps** that show you solved it (including, the tree visualization, the tree depth, each subproblem size at depth d, the number of subproblems/nodes at depth d, workload per subproblem/node at depth d, and (total) workload at depth d).

Recurrence:
$T(n) =$

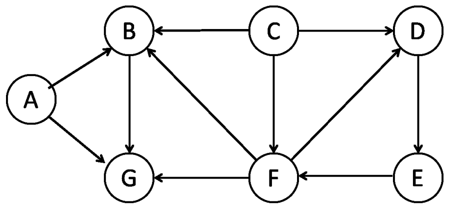
Solving the recurrence using the recursion tree method:

After solving the recurrence,
$T(n) =$

4. (5 points) Find a topological sort of the following graph using DFS. You must *label* each vertex with its DFS finishing time and *list* the vertices according to the resulting topological sort. (Please conside vertices in increasing alphabetical order. Also visit each vertex's neighbors in increasing alphabetical order.)



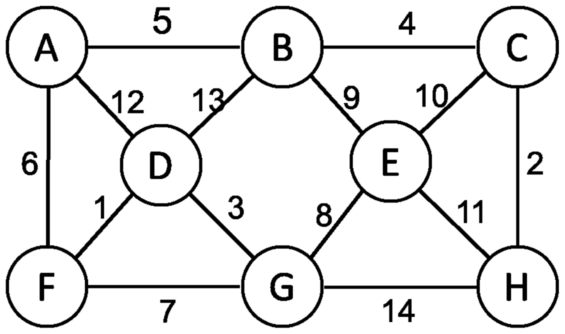
5. (10 points) Consider the following directed graph.



(a) (5 points) Do depth-first search in the graph G (NOT the vertex G), considering vertices in increasing alphabetical order (i.e. start with vertice A). Also visit each vertex's neighbors in increasing alphabetical order. Show the final result, with vertices labeled with their starting (discovery) and finishing times. The DFS forest, or DFF, must be clear.

(b) (5 points) Based on your results, proceed to run the algorithm we learned in class to find the strongly connected components of G (show the result of the DFS with vertices labeled with their starting (discovery) and finishing times. The DFF must be clear.)

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



(a) (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.

(b) (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. (6 points) Suppose we have a disjoint-set data structure that supports the following operations with their respective running time.

- Make-Set(x): $O(1)$ time.
- Union(x, y): $O(\log n)$ time, where n is the number of all elements considered.
- Find-Set(x): $O(1)$ time.

Suppose we implement the following pseudo-code of Kruskal's algorithm using this data structure. We want to analyze the running time. Answer the following three questions. Your answer must be an asymptotic quantity in terms of $|V|$ and/or $|E|$.

```
MST-KRUSKAL($G, w$)
1. $A = \emptyset$.
(E.g., the running time of Line 1 is $O(1)$)
```

```
2. for each vertex $v \in G.V$,
3.     Make-Set($v$)
```

(1 points). What is the running time of Lines 2 and 3?

```
4. Sort the edges of $G.E$ into non-decreasing order by weight $w$ (using the best sorting algorithm)
```

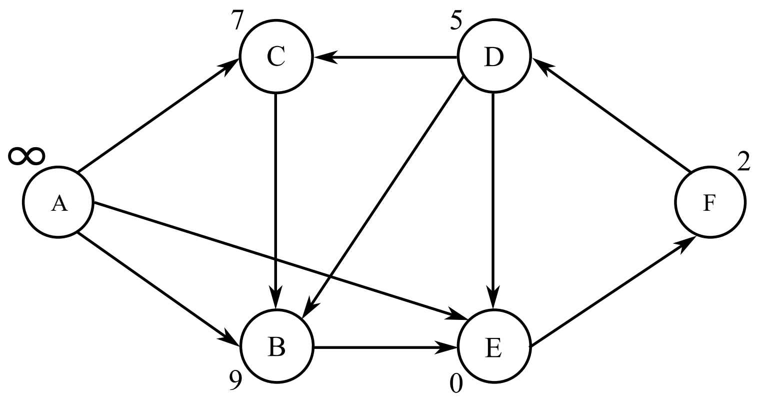
(2 points). What is the running time of Line 4?

```
5. for each edge $(u, v) \in G.E$, taken in non-decreasing order by weight
6.     if Find-Set($u$) $\neq$ Find-Set($v$)
7.         $A = A \cup \{(u, v)\}$
8.         Union($u, v$)
```

(3 points). What is the running time of Lines 5-8?

```
9. Return $A$.
```

8. (15 points) Consider the following weighted graph.



where 1) ach vertex $u \in V = A, B, C, D, E, F$ is labeled with a shortest path distance $\delta(s, u)$ from a certain, but unknown, source vertex $s \in V$; and 2) we do not know what the weight $w(u, v)$ of each edge is, but we are told that all the weights are positive and that $w(C, B) > 3$.

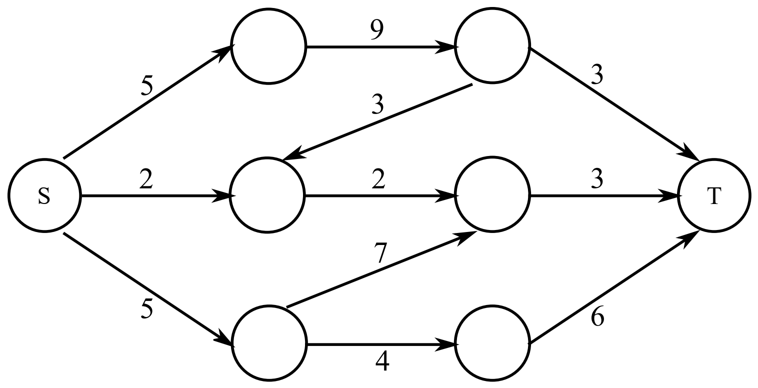
(a) (5 points) Find a source vertex and a value of the weight for each edge that is consistent with the figure. (Usually, we are given the source vertex and the edge weights and we have to find the shortest path distances; here we ask for the inverse problem.) Any solution is fine if there are multiple solutions. Briefly explain how you solved this.

(b) (6 points) Using the source vertex and weight values you found, run Dijkstras algorithm to find the single-source shortest paths starting from the source vertex. Show the intermediate stages of Dijkstras algorithm, i.e., at each stage show the shortest-path estimates at each vertex and the tree edges, and mark which vertices have been already processed. Verify the resulting shortest-path distances equal the ones in the figure above.

(c) (4 points) The Dijkstra's algorithm may fail to find a shortest path if the graph has negative weight edges – in other words, after running the Dijkstra algorithm, we have $v.d \neq \delta(s, v)$ for some $v \in V$. Give such a graph that *has no negative-weight cycle and has no more than four edges.* Note that you must specify the source vertex and the edge weights.

9. (5 points) In the activity selection problem seen in class, we have n activities with starting and finishing times $[si, fi)$, and want to obtain a maximal subset of compatible activities. Consider the following potential algorithm: we greedily select the activity that starts last that is compatible with all previously selected activities, and proceed that way to continue adding activities. Does this satisfy the greedy choice, and thus is the algorithm correct? If yes, provide a proof; if not, find a counterexample.

10. (10 points) For the following graph with the given capacities, show step-by-step how Ford-Fulkerson finds the max flow/min cut between S and T. At each iteration, show the graph of the flow and the residual network.



11. (5 points) Answer the following

    (a) (2 points) As described in class, the longest common subsequence (LCS) algorithm with two strings of lengths m and n uses a matrix of $m \times n$, thus it has a memory cost of $\Theta(mn)$.If the strings are long (e.g. 1000s of symbols), this is a big matrix. Assume that we do not need to find out the actual LCS, but only its length. Describe a simple way to reduce thememory cost to $\Theta(\min(m, n))$.

    (b) (3 points) The Unix **diff** command compares two files line by line and prints, in order, the lines that differ between them. Describe an efficient algorithm that achieves this.

12. (10 points) Suppose an undirected graph G has n vertices. Consider the following algorithm to find the global minimum cut in G: 1. bestCut = None

```
2. for $t = 1, \ldots, \binom{n}{2} ln(\frac{1}{\delta})$:
3.     candidateCut ← Karger($G$)
4.     if candidateCut is smaller than bestCut:
5.         bestCut ← candidateCut
6. return bestCut
```

Here $\delta \in (0, 1)$ is the probaility that Karger fails to return a minimum cut in T trials (for some T). Prove the probability that this algorithm doesn't return a global minimum cut is $\leq \delta$.