22C:153 Self-Evaluation Exam

Due: Thursday, 1/23

Notes:

(A) In the following, $\lg n = \log_2 n$, $\log n = \log_{10} n$, and $\ln n = \log_e n$.

(B) In the following, any mention of page numbers, section numbers, and problem numbers refer to your textbook, *Introduction to Algorithms, Second Edition*, by Cormen, Leiserson, Rivest, and Stein.

(C) Write down exactly what you have been asked for. For example, if you are asked to answer a question with a "yes" or a "no," just one word will suffice. It is not necessary to explain your answer.

Determine if the following equations are true or false. In Question 2, F_n stands for the nth *Fibonacci number.*

(1) $\lg(n!) = \Theta(n \log n)$.

(2) $F_n = O((1+\sqrt{5})^n/2^n)$.

(3) $2^{\lg n} = \Omega(n)$.

Solve the recurrence relations in the following three questions (4-6) and express the solution as $T(n) = \Theta(f(n))$ for appropriate $f(n)$. You need to write the final answer only.

(4) $T(n) = T(n-1) + 1/n$.

(5) $T(n) = T(n-1) + \log n$.

(6) $T(n) = 7 \cdot T(n/2) + n^2$.

(7) A function for selection called `SELECT` is described in Section 9.3. `SELECT` takes an array of size n and an integer i, $1 \le i \le n$, and returns the ith smallest element in the array in worst case $O(n)$ time. `SELECT`, as described in the textbook, returns the ith smallest element. For this problem, assume that instead of returning the ith smallest element, `SELECT` returns the index of the ith smallest element. Answer the following question based on this assumption.

On page 146, pseudocode for the function `PARTITION` is given. Into this function insert the following two lines of code *before* Line 1:

```
$index \leftarrow$ SELECT$(A, p, r, (r-p)/2)$;
SWAP$(A, index, r)$;
```

What is the worst case running time of `QUICKSORT` after this change?
Assume that $(r-p)/2$ in the above code equals $\lfloor (r-p)/2 \rfloor$. Also assume that the function call `SWAP`(A, i, j) exchanges the contents of slots i and j in array A.

Given a set of n numbers, we wish to find i largest numbers in the set in sorted order using a comparison based algorithm. In each of the following three questions (9-11), a method to do this is described. Write down the worst case asymptotic running time (as a function of n and i) of the best algorithm you know that implements the given method.

(8) Sort the numbers and list the i largest.

(9) Build a max-priority queue from the numbers and call `EXTRACT-MAX` i times.

(10) Use an algorithm to find the ith largest number, partition around this number, and sort the i largest numbers.

(11) This question relates to the *activity selection problem* in Section 16.1. It is shown in this section that a simple greedy algorithm computes a maximum size set of mutually compatible activities. However, there are other simple greedy algorithms that do not work. Here is an example:

> Order activities in nondecreasing order of duration and process them in this order. At each step, pick the next activity, keep it, if it is compatible with activities already selected; throw it away if it is not compatible with any activity that has been already selected.

Show with an example that this algorithm does not work.

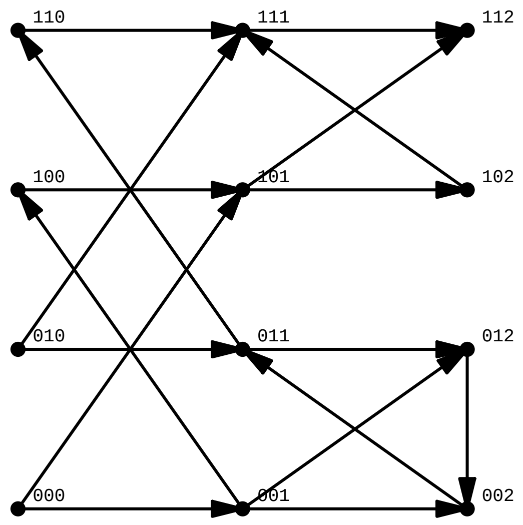
The following nine questions (12-20) pertain to graph algorithms.

(12) The *incidence matrix* of a directed graph $G = (V, E)$ is a $|V| \times |E|$ matrix $B = (b_{ij})$ such that

$$b_{ij} = \begin{cases} -1 & \text{if edge } j \text{ leaves vertex } i \\ 1 & \text{if edge } j \text{ enters vertex } i \\ 0 & \text{otherwise} \end{cases}$$

Describe what the entries of the matrix product $B \cdot B^T$ represent, where B^T is the transpose of B.

(13) Consider the directed graph shown below.

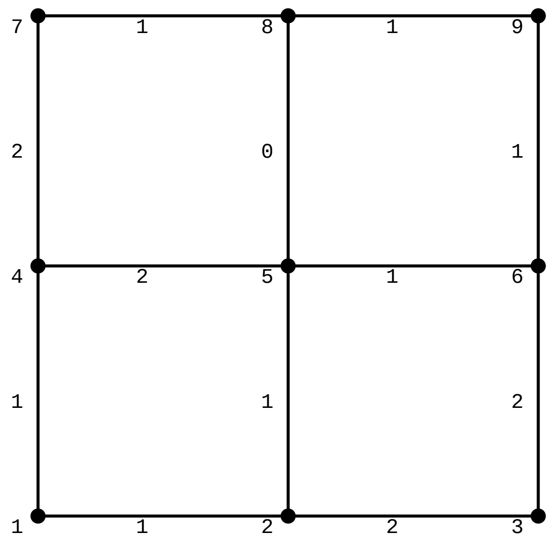


Run `DFS` (described on Page 541) on this graph and write down values of $d[v]$ and $f[v]$ for all vertices v. Assume that in the for-loop in Line 1 of `DFS` vertices of G are processed in lexicographic order of their labels. Also assume that in Line 4 of `DFS-VISIT` vertices adjacent to each vertex u are processed in lexicographic order of their labels.

(14) Identify tree edges, back edges, forward edges, and cross edges for the depth-first search in the previous problem.

(15) For this question consider the undirected version of the graph in Question 13. Draw a breadth-first search tree for this graph for a search that starts at vertex 000. As in the case of depth-first search, assume that whenever vertices are to be processed in some arbitrary order by breadth-first search, they are processed in lexicographic order of their labels.

(16) Here is an edge-weighted graph. Show a *minimum spanning tree* in this graph.



(17) Answer "yes" or "no". Does the following algorithm produce a minimum spanning tree of the given graph?

```
MAYBE-MST-A(G, w)
1.     sort the edges into nonincreasing order of edge weights $w$
2.     $T \leftarrow E$
3.     for each edge $e$, taken in nonincreasing order by edge weight
4.         do if $T - \{e\}$ is a connected graph
5.             then $T \leftarrow T - e$
6.     return T;
```

(18) Here is a *divide-and-conquer* algorithm that supposedly computes a minimum spanning tree of a given graph. Answer "yes" or "no": does this algorithm correctly compute a minimum spanning tree?

> Given a graph $G = (V, E)$ partition V into sets V_1 and V_2 whose sizes differ by at most 1. Let E_1 and E_2 be edges incident only on vertices in V_1 and V_2 respectively. Recursively solve the minimum spanning tree problem for the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select a minimum weight edge in E that crosses the cut (V_1, V_2) and use this edge to unite the spanning trees of G_1 and G_2 into a spanning tree for G.

(19) Show shortest paths from vertex 1 to all other vertices in the edge-weighted graph given in Problem 22.

(20) Show a small example of an edge-weighted graph for which Dijkstra's algorithm produces incorrect answers, but the Bellman-Ford algorithm produces correct answers. Along with the graph, show the source of the search as well.