University of Toronto
Scarborough Campus
October 16, 2021

**CSC C73 Midterm Test**
**Instructor:** Vassos Hadzilacos

**Aids allowed:** One $8.5 \times 11$ 'cheat sheet' (may be written on both sides)

**Duration:** Two hours

## READ THE INSTRUCTIONS CAREFULLY

- There should be 6 pages in this exam booklet, including this cover page.
- Answer all questions.
- If possible, print the exam booklet, and put all answers in this booklet, in the spaces provided. If you do not have access to a printer, ***write the answer to each question on a different sheet of paper***.
- At the end of the exam period scan or take a photograph of your answers, and upload them on Crowdmark in PDF, JPG, or PNG form. Make sure that you upload the correct file for each answer, that the images you upload are oriented correctly, and that the photographs you take (if applicable) do not have shadows.
- In your answers you may use any result discussed in this course or its prerequisites merely by stating it.
- Good luck!

| Problem | Marks Rec'ved | Marks Worth |
|---|---|---|
| 1. | | 30 |
| 2. | | 15 |
| 3. | | 15 |
| 4. | | 30 |
| 5. | | 30 |
| TOTAL | | 120 |

**QUESTION 1.** (30 marks)

For each of the statements below, indicate whether it is true or false by circling the appropriate response. Do ***not*** justify your answers. No penalty for wrong answers, but don't rush to guess.

**a.** In the greedy algorithm for interval scheduling, if we initially sort the intervals in increasing start time and all intervals have the same length, the algorithm returns an optimal set of intervals. ..... **True / False**

**b.** Huffman's algorithm runs in $O(n)$ time, where $n$ is the size of the alphabet, if the symbols and their frequencies are given sorted in non-decreasing frequency order. ..... **True / False**

**c.** Let $T$ be a full binary tree that represents a prefix code for alphabet $\Gamma$, $f(a)$ be the frequency of symbol $a \in \Gamma$, and $\mathbf{depth}(a)$ be the depth in $T$ of the leaf that represents $a$. If $\mathbf{depth}(a) \leq \mathbf{depth}(b)$ for every $a, b \in \Gamma$ such that $f(a) \geq f(b)$, then $T$ represents an optimal code for $\Gamma$ and $f$. ..... **True / False**

**d.** Using heaps, we can implement Dijkstra's algorithm so that its running time on a connected undirected graph with $n$ nodes and $m$ edges is $O(m \log n)$. ..... **True / False**

**e.** The fractional knapsack problem for $n$ items can be solved in $O(n \log n)$ time. ..... **True / False**

**f.** If $T(n) = 4T(n/4) + \sqrt{n}$ then $T(n) = \Theta(n)$. ..... **True / False**

**g.** If $T(n) = 8T(n/4) + n^3$ and $S(n) = 16S(n/4) + n^2$ then $T(n) = O\big(S(n)\big)$. ..... **True / False**

**h.** If $p$ and $q$ are the closest pair of points in a set of points $P$ on the plane, then there are at most six points in $P$ whose $y$-coordinates are between the $y$-coordinates of $p$ and $q$. ..... **True / False**

**i.** Given $n^2$ points on the plane we can find two of these points that are closest to each other in $O(n^2 \log n)$ time. ..... **True / False**

**j.** In the deterministic order statistics algorithm, if we partition the given set of numbers in groups of seven, instead of groups of five, the resulting algorithm runs in worst-case linear time. ..... **True / False**

**QUESTION 2.** (15 marks)

Shown below is Dijkstra's shortest-paths algorithm, which computes the weight $d(u)$ of a shortest path from the source node $s$ to every node $u$ of graph $G$, where **wt** is the edge weight function. We assume that $\mathbf{wt}(u, v) > 0$, for every edge $(u, v)$ of $G$. (Note: all edges have ***strictly positive***, not merely non-negative, weight.)

```
DIJKSTRA(G, s, wt)
1  R := ∅
2  d(s) := 0
3  for each v ∈ V − {s} do d(v) := ∞
4  while R ≠ V do
5      let u be a node not in R with minimum d-value (i.e., u ∈ V − R and ∀ u′ ∈ V − R, d(u) ≤ d(u′))
6      R := R ∪ {u}
7      for each v ∈ V such that (u, v) ∈ E do
8          if d(u) + wt(u, v) < d(v) then d(v) := d(u) + wt(u, v)
```

Show how to modify the algorithm, without changing its running time by more than a constant factor, so that it computes the ***number*** $N(u)$ ***of distinct shortest paths*** from $s$ to $u$ (in addition to $d(u)$).

Describe any additional information your modification uses by stating a relevant invariant that it satisfies, and show the modified pseudocode. You can do so by adding statements to the above pseudocode as long as the result is clear; otherwise, write the pseudocode from scratch below. Do ***not*** explain why your algorithm is correct, and do ***not*** analyze its running time.

**ANSWER:**

**QUESTION 3.** (15 marks)

Professor Ignoramus Brainiac claims that the following divide-and-conquer algorithm finds the median of $n$ distinct numbers, when $n$ is a power of 3: Arbitrarily divide the $n$ numbers into $n/3$ groups of 3 numbers each; find the median of each group of 3 (this can be done with at most three comparisons for each group); recurse on the list of $n/3$ group medians until we are left with a group of 3 numbers, in which case we return their median.

**a.** (10 marks) Is the professor right? Justify your answer.

**ANSWER:**

**b.** (5 marks) Use the Master Theorem to analyze the running time of Professor Brainiac's algorithm.

**ANSWER:**

**QUESTION 4.** (30 marks)

A long straight corridor (to be thought of as a line that starts at zero and extends to infinity) contains $n$ offices (to be thought of as points on that line). The location of each office is identified by its distance from the start of the corridor. We wish to place WiFi routers along the corridor to serve the offices. Each router can be placed ***anywhere***, not necessarily in an office, and serves all offices that are at most $d$ meters away from its location (in either direction), for some $d > 0$. We wish to find locations where to place the WiFi routers so that ***all offices are served, using the smallest number of routers***.

More precisely, we want an algorithm that takes as input an array $L[1..n]$, where $L[i]$ is the location of office $i$, and the range $d$ of routers; and produces as output a set $R$ of locations where to install routers so that (a) every office is within distance $d$ of some location in $R$; and (b) the cardinality of $R$ is as small as possible.

**a.** (10 marks) Consider the following greedy algorithm for this problem: Start with the empty set of router locations $R$, and the set $S$ of all $n$ office locations. Find a router location $r$ on the line that serves as many offices in $S$ as possible, add it to $R$, and remove from $S$ all offices served by router location $r$. Repeat until $S$ is empty, and return the set $R$.

Give a counterexample proving that this algorithm is not correct. Explain what the algorithm does in your counterexample, and why that is not optimal.

**ANSWER:**

**b.** (20 marks) Describe an efficient greedy algorithm that solves the problem, and analyze its running time. Do ***not*** justify its correctness. Do not assume anything about the input other than what is stated in the question.

**ANSWER:**

**QUESTION 5.** (30 marks)

For each $k \in \mathbb{N}$, the ***Hadamard matrix*** $H_k$ is a $2^k \times 2^k$ matrix defined recursively as follows:

- $H_0 = [1]$.
- For $k > 0$, $H_k = \left[\begin{array}{c|c} H_{k-1} & H_{k-1} \\ \hline H_{k-1} & -H_{k-1} \end{array}\right]$

**a.** (5 marks) Write the matrix $H_2$.

**ANSWER:**

**b.** (25 marks) Let $\vec{v}$ be a column vector of length $n = 2^k$. Describe a divide-and-conquer algorithm that, given $\vec{v}$, computes the product $H_k\,\vec{v}$ in $O(n \log n)$ time, assuming each arithmetic operation (addition, subtraction, multiplication, or division) takes $O(1)$ time. Justify why your algorithm achieves the required running time. Do ***not*** justify the correctness of your algorithm.

**ANSWER:**

**THE END**