Carleton University
Midterm COMP 3804

March 1, 2024

- All questions must be answered on the scantron sheet.
- Write your name and student number on the scantron sheet.
- You do not have to hand in this examination paper.
- Calculators are allowed.

Marking scheme: Each of the 17 questions is worth 1 mark.

Some useful facts:

1. $1 + 2 + 3 + \cdots + n = n(n+1)/2$.

2. for any real number $x > 0$, $x = 2^{\log x}$.

3. For any real number $x \neq 1$ and any integer $k \geq 1$,

$$1 + x + x^2 + \cdots + x^{k-1} = \frac{x^k - 1}{x - 1}.$$

4. For any real number $0 < \alpha < 1$,

$$\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}.$$

Master Theorem:

1. Let $a \geq 1$, $b > 1$, $d \geq 0$, and

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ a \cdot T(n/b) + \Theta(n^d) & \text{if } n \geq 2. \end{cases}$$

2. If $d > \log_b a$, then $T(n) = \Theta(n^d)$.

3. If $d = \log_b a$, then $T(n) = \Theta(n^d \log n)$.

4. If $d < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

1. Recall that $\mathbb{N} = \{1, 2, 3, \ldots\}$ denotes the set of all positive integers. Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ be two functions such that $f(n) = O(g(n))$. Is it true that, for any two such functions f and g,
$$2^{f(n)} = O\left(2^{g(n)}\right)?$$
   (a) This is true.
   (b) This is not true.

2. Consider the recurrence
$$T(n) = \sqrt{n} + T(n/3).$$
   Which of the following is true?
   (a) $T(n) = \Theta(\sqrt{n})$.
   (b) $T(n) = \Theta(\sqrt{n}\log n)$.
   (c) $T(n) = \Theta(n)$.
   (d) $T(n) = \Theta(n \log n)$.

3. Consider the recurrence
$$T(n) = n + T(n/31) + T(29n/31).$$
   Which of the following is true?
   (a) $T(n) = \Theta(n)$.
   (b) $T(n) = \Theta(n \log n)$.
   (c) $T(n) = \Theta(n^2)$.
   (d) None of the above.

4. Consider the following recursive algorithm $\textsc{Power}(a, b)$, which takes as input two integers $a \geq 1$ and $b \geq 1$, and returns a^b:

> **Algorithm** $\textsc{Power}(a, b)$:
> **if** $b = 1$
> **then** return a
> **else** $c = a^2$;
> $\textsc{answer} = \textsc{Power}(c, \lfloor b/2 \rfloor)$;
>   **if** b is even
>   **then** return $\textsc{answer}$
>   **else** return $a \cdot \textsc{answer}$
>   **endif**
> **endif**

Assume that each multiplication, division, and floor-operation in this algorithm takes $O(1)$ time. What is the running time of algorithm $\textsc{Power}(a, b)$?

(a) $T(n) = \Theta(\log(a + b))$.
(b) $T(n) = \Theta(\log(ab))$.
(c) $T(n) = \Theta(\log a)$.
(d) $T(n) = \Theta(\log b)$.

5. You are given m sorted arrays $A_1, A_2, \ldots, A_m$, each of length n. Consider the following algorithm that merges these arrays into one single sorted array of length mn:

- $B = \textsc{Merge}(A_1, A_2)$, where $\textsc{Merge}$ is the algorithm from class that merges the two sorted arrays A_1 and A_2 into one sorted array B.
- For $i = 3, 4, \ldots, m$, $B = \textsc{Merge}(B, A_i)$.

What is the running tims of this algorithm?

(a) $\Theta(mn)$.
(b) $\Theta(mn \log(mn))$.
(c) $\Theta(m^2 n)$.
(d) $\Theta(mn^2)$.

6. You are given m sorted arrays $A_1, A_2, \ldots, A_m$, each of length n. Assume that m is a power of two. Consider the following algorithm MergeManyArrays that merges these arrays into one single sorted array of length mn:

Base case: If $m = 1$, then there is nothing to do.

Non-base case: If $m \geq 2$:

- For each $i = 1, 2, \ldots, m/2$, run the Merge algorithm from class on the two arrays A_{2i-1} and A_{2i}, resulting in a sorted array B_i of length $2n$.

- Recursively run the algorithm MergeManyArrays on the sorted arrays $B_1, B_2, \ldots, B_{m/2}$.

Let $T(m, n)$ denote the running time of this algorithm. Which of the following is correct?

(a) $T(m, n) = \Theta(mn) + T(m/2, n)$.
(b) $T(m, n) = \Theta(mn) + T(m/2, 2n)$.
(c) $T(m, n) = \Theta(m + n) + T(m/2, n)$.
(d) $T(m, n) = \Theta(m + n) + T(m/2, 2n)$.

7. Professor Uriah Heep has designed a new data structure that stores any sequence of numbers, and supports the following two operations:

- Insert(x): Add the number x to the data structure. This operation takes $\Theta(\sqrt{n})$ time, where n is the current number of elements.

- ExtractMin: Delete, and return, the smallest element stored in the data structure. This operation takes $\Theta(\log n)$ time, where n is the current number of elements.

You use Professor Heep's data structure (and nothing else) to design a sorting algorithm. What is the running time of this sorting algorithm on an input of n numbers?

(a) $\Theta(n \log n)$.
(b) $\Theta(n^{3/2})$.
(c) $\Theta(n^2)$.
(d) None of the above.

8. Let S be a set of n distinct numbers. Assume this set S is stored in a min-heap $A[1 \ldots n]$. How much time does it take to use this heap to find the largest number of S?

(a) $\Theta(1)$.
(b) $\Theta(\log n)$.
(c) $\Theta(n)$.
(d) $\Theta(n \log n)$.

9. Let $G = (V, E)$ be a connected undirected graph, and let $n = |V|$. What are the minimum and maximum number of edges that this graph can have?

   (a) 1 and n^2.
   (b) n and $n(n-1)/2$.
   (c) $n-1$ and n^2.
   (d) $n-1$ and $n(n-1)/2$.

10. Let $G = (V, E)$ be a directed graph that is given using adjacency lists: Each vertex u has a list $\textsc{Out}(u)$ storing all edges (u, v) going out of u.
    What is the running time of the fastest algorithm that computes, for each vertex v, a list $\textsc{In}(v)$ of all edges (u, v) going into v?

    (a) $\Theta(|V| + |E|)$.
    (b) $\Theta(|V| \log |V| + |E|)$.
    (c) $\Theta(|V| + |E| \log |E|)$.
    (d) $\Theta((|V| + |E|) \log |V|)$.

11. Let $G = (V, E)$ be an undirected graph with $n = |V|$ vertices, and assume that the vertex set is stored in an array $V[1 \dots n]$. For each i, let $v_i = V[i]$.
    Is it possible to give each edge $\{v_i, v_j\}$ a direction (i.e., replace it by exactly one of (v_i, v_j) and (v_j, v_i)) such that the resulting directed graph is acyclic?

    (a) This is not possible.
    (b) This is possible.

12. Let $G = (V, E)$ be an undirected graph, and assume that this graph is stored using the adjacency matrix. What is the running time of the fastest depth-first search algorithm for this graph?

    (a) $\Theta(|V| + |E|)$.
    (b) $\Theta(|V|^2 + |E|^2)$.
    (c) $\Theta(|V|^2)$.
    (d) $\Theta(|E|^2)$.

13. Let $G = (V, E)$ be a directed acyclic graph, and let s and t be two distinct vertices of V. What is the running time of the fastest algorithm that computes the number of directed paths in G from s to t?

    (a) $\Theta(|V| \cdot |E|)$.
    (b) $\Theta((|V| + |E|) \log |V|)$.
    (c) $\Theta(|V| + |E|)$.
    (d) $\Theta(|E|)$.

14. Let $G = (V, E)$ be a directed graph. We run depth-first search on G, i.e, algorithm DFS(G). Recall that this classifies each edge of E as a tree edge, forward edge, back edge, or cross edge.
    Let (u, v) be an edge of E that is not classified as a tree edge.
    Is the following true or false?
    It is possible to run algorithm DFS(G), where vertices and edges are processed in a different order, such that (u, v) is classified as a tree edge.

    (a) True.
    (b) False.

15. Let $G = (V, E)$ be a directed graph. We run depth-first search on G, i.e, algorithm DFS(G). Is the following true or false?
    If the graph G has a directed cycle that contains a forward edge, then G also contains a directed cycle that does not contain a forward edge.

    (a) True.
    (b) False.

16. Let $G = (V, E)$ be a directed acyclic graph and, for each edge (u, v) in E, let $\textsc{wt}(u, v)$ denote its positive weight. Let s be a source vertex, and for each vertex v, let $\delta_{\max}(s, v)$ be the weight of a *longest* path in G from s to v.
    What is the running time of the fastest algorithm that computes $\delta_{\max}(s, v)$ for all vertices v?

    (a) Since there can be exponentially many paths from s to some vertex v, the running time must be at least exponential.
    (b) $\Theta((|V| + |E|) \log |V|)$.
    (c) $\Theta(|E| + |V| \log |V|)$.
    (d) $\Theta(|V| + |E|)$.

17. After this midterm, you go to a Karaoke Bar and sing the following randomized and recursive song AWESOMEST(n), which takes as input an integer $n \geq 1$:

> **Algorithm** AWESOMEST(n):
> sing the following line n times:
> *COMP 3804 is the awesomest course I have ever taken*;
> **if** $n \geq 2$
> **then** let k be a uniformly random element in $\{1, 2, \ldots, n\}$;
> AWESOMEST(k)
> **endif**

What is the expected number of times you sing *COMP 3804 is the awesomest course I have ever taken*?

(a) $\Theta(n)$.

(b) $\Theta(n \log n)$.

(c) $\Theta(n^2)$.

(d) None of the above.