CSCI2100/ESTR2102: Midterm

Name: Student ID:

Problem 1. (10%) Prove $1000 \log_2 n = O(n)$.

Answer. $1000 \log_2 n \le 1000n$ for all $n \ge 1$.

Problem 2. (15%) Consider a function $f(n)$ satisfying $f(1) = 1$ and $f(n) = 4 \cdot f(n/2) + n^2\sqrt{n}$ for $n \ge 2$. Prove: $f(n) = O(n^3)$.

Answer. The recurrence conforms to the template in Master's Theorem with $\alpha = 4, \beta = 2, \gamma = 2.5$. As $\log_\beta \alpha = 2 < \gamma$, the theorem tells us $f(n) = O(n^\gamma) = O(n^{2.5})$. The claim follows from the fact that $n^{2.5} = O(n^3)$.

Problem 3 (10 marks). Suppose that we use binary search to find 90 in the sorted array $A = (5, 12, 35, 43, 55, 78, 82, 90)$. Describe the sequence of integers in A that are compared to 90.

Solution. 43, 78, 82, 90.

Problem 4. (15%) Let S_1 and S_2 be two sets of integers, each with size n. Design an algorithm to report the *distinct* integers in $S_1 \cup S_2$ using $O(n \log n)$ time. For example, if $S_1 = \{1, 5, 6, 9, 10\}$ and $S_2 = \{5, 7, 10, 13, 15\}$, you should output: 1, 5, 6, 7, 9, 10, 13, 15.

Answer. Sort S_1 and S_2 in $O(n \log n)$ time. Then, merge the two sorted sets into one array A with length $2n$, where the integers are arranged in non-descending order. Scan A by the sorted order. For each integer e seen, output e if e is different from the its preceding integer in A.

Problem 5 (15 marks). An integer n is *cubic* if it equals m^3 for some integer m (e.g., 8 and 27 are cubic but 36 is not). You are given a positive integer $n \geq 2$. Design an algorithm to determine whether n is cubic in $O(\log n)$ time.

Answer. We aim to find the largest integer $x \in [1, m]$ such that $x^3 \leq n$. Then, n is cubic if and only if $n = x^3$. We can find x through binary search. First, set $a = 1$ and $b = n$. Iterative the following steps until $a = b$:

- Set $x = (a + b)/2$.
- If $x^3 \leq n$, set $a = x$.
- Otherwise, return $b = x - 1$.

When $a = b$, then $x = a$ is the value we want to find.

Problem 6 (15 marks). Let S_1 be a set of n integers, and S_2 another set of $\log_2 n$ integers (n is a power of 2). Each set is given in an array which is *not* sorted. Report, for every integer $e \in S_1$, its predecessor in S_2. Your algorithm must finish in $O(n \log \log n)$ time.

For example if $S_1 = \{15, 6, 12, 18\}$ and $S_2 = \{16, 7\}$, then you should output: (15, 7) (meaning 7 is the predecessor of 15 in S_2), (6, -) (meaning 6 has no predecessor in S_2), (12, 7), (18, 16).

Answer. Sort S_2. For each element $e \in S_1$, perform binary search on S_2 to find the predecessor of e in S_2.

Problem 7 (20 marks). Let S_1 be a set of n integers, and S_2 another set of $\log_2 n$ integers (n is a power of 2). Each set is given in an array which is *not* sorted. Report, for every integer $e \in S_2$, how many integers in S_1 are greater than or equal to e. Your algorithm must finish in $O(n \log \log n)$ time.

For example if $S_1 = \{15, 6, 12, 18\}$ and $S_2 = \{16, 7\}$, then you should output: (16, 1), (7, 3) because S_1 has only one integer ≥ 16 but has 3 integers ≥ 7.

Answer. For each element $e \in S_2$, obtain a counter c_e which equals how many integers in S_2 have e as the predecessor in S_2. For instance, in our example, the counter of 16 is 1 because only one integer (i.e., 18) in S_2 has 16 as its predecessor in S_2; similarly, the counter of 7 is 2. These counters can be obtained by slightly modifying the algorithm in Problem 6 (every time an element in S_1 finds $e \in S_2$ as the predecessor, increase c_e by 1).

For each element $e \in S_2$, we want to obtain a *suffix counter* s_e, which adds up the counters of all the elements in S_2 greater than or equal to e. The value of s_e is precisely the number of elements in S_1 larger than or equal to e. For instance, in our example, the suffix counter of 7 is 3, which adds up the counters of 7 and 16. If e is the largest element in S_2, $s_e = c_e$. For a general element $e \in S_2$, $s_e = c_e + s_{e'}$ where e' is the element succeeding e in S_2. This gives the following algorithm for obtaining all the suffix counters:

1. $s = 0$
2. **for** $e \in S_2$ in descending order **do**
3. $\quad s \leftarrow s + c_e$
4. $\quad$ output (e, s)