COMP3506: Mid-Semester Exam

Note 1: This is the exam paper for **COMP3506**. If you are registered for COMP7505, turn overleaf.
Note 2: Write all your solutions in the **answer book**

Problem 1 (5 marks). Prove: $5n + 3\sqrt{n} = O(n)$.

Solution. $5n + 3\sqrt{n} \le 8n$ for all $n \ge 1$.

Problem 2 (10 marks). Let $f(n)$ be a function of a positive integer n. We know:

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 2 \\ f(n) &= 3 + f(n-2) \end{aligned}$$

Prove $f(n) = O(n)$.

Solution.

$$\begin{aligned} f(n) &= 3 + f(n-2) \\ &= 3 \cdot 2 + f(n-4) \\ &= 3 \cdot 3 + f(n-6) \\ &\dots \\ &= 3 \cdot \lfloor n/2 \rfloor + f(n - 2\lfloor n/2 \rfloor) \\ &\le 3n/2 + f(0) + f(1) = O(n). \end{aligned}$$

Problem 3 (20 marks). Let S_1 and S_2 be two disjoint sets of integers, i.e., $S_1 \cap S_2 = \emptyset$. We know that $|S_1| = |S_2| = n$ (i.e., each set has n integers). Each set is stored in an array of length n, where its integers are sorted in ascending order. Let $k \ge 1$ be an integer. Design an algorithm to find the k smallest integers in $S_1 \cup S_2$ in $O(k)$ time.

Solution. Suppose that S_1 is stored in array A_1, and S_2 in array A_2. Create an array B of size k. At the beginning, B is empty.

Set $i = 1$ and $j = 1$. Repeat the following until there are k integers in B:

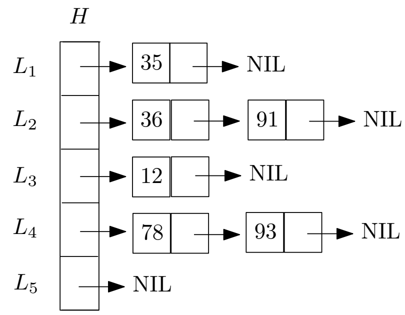
- If $A_1[i] < A_2[j]$, append $A_1[i]$ to B, and increment i by 1.
- Otherwise, append $A_2[j]$ to B, and increment j by 1.

Problem 4 (15 marks). Consider a set of elements $S = \{12, 35, 36, 78, 91, 93\}$. We use a hash function

$$h(k) = 1 + (k \mod 5)$$

to map integers to the domain $\{1, 2, ..., m\}$ where $m = 5$. Draw the resulting hash table on S.

Solution.



Problem 5 (10 marks). Only one of the following statements is true. Which one is it?
A. The quick sort algorithm sorts n integers in $O(n \log n)$ worst case time.
B. The time complexity of counting sort grows slower than that of merge sort.
C. Suppose that a data structure supports an operation in amortized $O(1)$ time, then it supports any sequence of n such operations in $O(n)$ time.
D. Someday Prof. Tao would be able to discover a comparison-based algorithm that sorts n integers in $O(n\sqrt{\log n})$ time.

Solution. C (note: no student answered D).

Problem 6 (20 marks). Let S be a set of n integers in the domain $[1, U]$, where $U = 2^n$. Describe an algorithm that determines if S contains two integers x, y such that $y \le x \le 100 + y$ (i.e., the difference between x and y is at most 100). Your algorithm must finish in $O(n)$ time ($O(n)$ expected time is acceptable).

5 marks given if your algorithm terminates in $O(n \log n)$ time.

Solution. Create a hash table T on S in $O(n)$ time. For every $x \in S$, use T to check whether $x+1, x+2, ..., x+100$ are in T. This requires $100 = O(1)$ queries, which take $O(1)$ expected time in total. The execution time is therefore $O(n)$ expected.

Problem 7 (20 marks). Let A be an array that stores a set S of n integers. We know that there exists some integer $t \in [1, n-1]$ such that

$$A[t+1], A[t+2], ..., A[n], A[1], A[2], ..., A[t]$$

are in ascending order. Given such an array A, the value of n, and an arbitrary integer k, describe an algorithm to determine whether k is in A. Note that the value of t is *not* given. Your algorithm must terminate in $O(\log n)$ time.

For example, suppose that $n = 7$. In $A = (56, 78, 91, 93, 12, 35, 36)$, $t = 4$, whereas in $A = (93, 12, 35, 36, 56, 78, 91)$, $t = 1$. Once again, the actual value of t is unknown.

Solution. The difficult step is to find the value of t in $O(\log n)$ time. After this, the problem can be easily solved by performing binary search in the sequence from $A[1]$ to $A[t]$, and in the sequence from $A[t+1]$ to $A[n]$.

We now explain how to find t, assuming $n \ge 3$ (otherwise, obtain t by simply checking all the integers in A). Let $m = \lfloor n/2 \rfloor$. Proceed as follows:

- If $A[m] > A[m+1]$, return $t = m$.
- If $A[m] > A[1]$, recur on the part of A behind $A[m]$.
- Otherwise, recur on the part of A before $A[m]$.

COMP7505: Mid-Semester Exam

Note 1: This is the exam paper for **COMP7505**. If you are registered for COMP3506, turn overleaf.
Note 2: Write all your solutions in the **answer book**

Problem 1 (5 marks). Prove: $5n + 3\sqrt{n} = O(n)$.

Solution. See the COMP3506 paper.

Problem 2 (10 marks). Let S be a set of n integers stored in an array of length n. You are also given a value of v. Design an algorithm to determine whether S has two integers that add up to v. Your algorithm should terminate in $O(n \log n)$ time.

Solution. Sort S. For each value $x \in S$, binary search for $v - x$. If found, return "yes". If still not found at the end, return "no".

Problem 3 (20 marks). Let S_1 and S_2 be two disjoint sets of integers, i.e., $S_1 \cap S_2 = \emptyset$. We know that $|S_1| = |S_2| = n$ (i.e., each set has n integers). Each set is stored in an array of length n, where its integers are sorted in ascending order. Let $k \geq 1$ be an integer. Design an algorithm to find the k smallest integers in $S_1 \cup S_2$ in $O(k)$ time.

Solution. See the COMP3506 paper.

Problem 4 (15 marks). Consider a set of elements $S = \{12, 35, 36, 78, 91, 93\}$. We use a hash function

$$h(k) = 1 + (k \mod 5)$$

to map integers to the domain $\{1, 2, ..., m\}$ where $m = 5$. Draw the resulting hash table on S.

Solution. See the COMP3506 paper.

Problem 5 (10 marks). Only one of the following statements is true. Which one is it?
A. The quick sort algorithm sorts n integers in $O(n \log n)$ worst case time.
B. The time complexity of counting sort grows slower than that of merge sort.
C. Suppose that a data structure supports an operation in amortized $O(1)$ time, then it supports any sequence of n such operations in $O(n)$ time.
D. Someday Prof. Tao would be able to discover a comparison-based algorithm that sorts n integers in $O(n\sqrt{\log n})$ time.

Solution. See the COMP3506 paper.

Problem 6 (20 marks). Let S be a set of n integers in the domain $[1, U]$, where $U = 2^n$. Describe an algorithm that determines if S contains two integers x, y such that

- $x > y$ and
- $\frac{x}{y}$ is a power of 2 that is between 8 and 256.

Your algorithm must finish in $O(n)$ time ($O(n)$ expected time is acceptable).

Solution. Create a hash table T on S using $O(n)$ time. For each value $y \in S$, probe the hash table to check whether $8y, 16y, 32y, ..., 256y$ are in S. This requires 6 queries which take $O(1)$ expected time. If any of these values is in T, return "yes". If still not found till the end, return "no". The total running time is therefore $O(n)$ expected.

Problem 7 (20 marks). Let A be an array that stores a set S of n integers. We know that there exists some integer $t \in [1, n-1]$ such that

$$A[t+1], A[t+2], ..., A[n], A[1], A[2], ..., A[t]$$

are in ascending order. Given such an array A, the value of n, and an arbitrary integer k, describe an algorithm to determine whether k is in A. Note that the value of t is *not* given. Your algorithm must terminate in $O(\log n)$ time.

For example, suppose that $n = 7$. In $A = (56, 78, 91, 93, 12, 35, 36)$, $t = 4$, whereas in $A = (93, 12, 35, 36, 56, 78, 91)$, $t = 1$. Once again, the actual value of t is unknown.

Solution. See the COMP3506 paper.