3121-21T2-HW1-SOLUTION

1. Question 1

You are given an array A of n distinct positive integers.

(1) Design an algorithm which decides in time $O(n^2 \log n)$ (in the worst case) if there exist four **distinct** pairs of integers $\{m, s\}$ and $\{k, p\}$ in A such that $m^2 + s = k + p^2$ (10 points)
(2) Solve the same problem but with an algorithm which runs in the **expected time** of $O(n^2)$. (10 points)

Solution for Question 1(a):

(1) Square all numbers in A, storing in an array S.
(2) Find all combinations of sum $a_i + s_j$, $i, j \in [0, n-1], i \neq j$ storing in an array C with i, j.
(3) Sort C based on sum using merge sort, guarantee worst case $O(nlogn)$.
(4) Loop through C and find two adjacent elements with the same sum and distinct is, js values, and these 4 integers are the indexes of answers.
(5) Time complexity $= O(n) + O(n^2) + O(n^2logn) = O(n^2logn)$.

Solution for Question 1(b):

(1) Square all numbers in A, storing in an array S.
(2) Find all combinations of sum $a_i + s_j$, $i, j \in [0, n-1], i \neq j$ storing in a hash table M, $M[sum] = [(A[i], A[j]), ...]$.
(3) Loop through M and in each slot find if there is more than one pair of $(A[i], A[j])$ that produces the same value of $(A[i])^2 + A[j]$.
(4) Time complexity $= O(n) + O(n^2) + O(n^2) = O(n^2)$.

2. Question 2

You are given a set of n fractions of the form x_i/y_i $(1 \leq i \leq n)$, where x_i and y_i are positive integers. Unfortunately, all values y_i are incorrect; they are all of the form $y_i = c_i + E$ where numbers $c_i \geq 1$ are the correct values and E is a positive integer (equal for all y_i). Fortunately, you are also given a number S which is equal to the correct sum $S = \sum_{i=1}^{n} x_i/c_i$. Design an algorithm which finds all the correct values of fractions x_i/c_i and which runs in time $O(n \log \min\{y_i : 1 \leq i \leq n\})$. (20 points)

Solution for Question 2: In order to find all the correct values of the fractions, we need to find the value of E that all denominators were increased by. Since for all i, $y_i > c_i > 0$ and $E = y_i - c_i$ is positive, we know that

$$0 < E < \min\{y_i : 1 \leq i \leq n\}$$

Then, define

$$P(k) = \sum_{i=1}^{n} \frac{x_i}{y_i - k} = \sum_{i=1}^{n} \frac{x_i}{c_i + E - k}$$

which is strictly increasing for $0 < k < \min\{y_i : 1 \leq i \leq n\}$, with $P(k) = S$ precisely when $k = E$. This means that we can binary search over this range using $P(k)$ to find E in $O(\log \min\{y_i : 1 \leq$

$i \leq n\}$) steps. Each step involves evaluating P once, which takes $O(n)$ time. As such, the overall runtime of our algorithm is $O(n \log \min\{y_i : 1 \leq i \leq n\})$

3. Question 3

You are given an array A consisting of n positive integers, **not** necessarily all distinct. You are also given n pairs of integers (L_i, U_i) and have to determine for all $1 \leq i \leq n$ the number of elements of A which satisfy $L_i \leq A[m] \leq U_i$ by an algorithm which runs in time $O(n \log n)$. (20 points)

Solution for Question 3:
We first sort the array A in $O(n \log n)$ time using merge sort. For the ith query (L_i, U_i), we do binary search twice to find the index of:

(1) the first element with value less than L_i; and
(2) the last element with value greater or equal to U_i.

The difference between these indices is the answer to the ith query.
We can do a binary search in $O(\log n)$ time. There are n queries and we need to do $2 \times n$ binary searches. Therefore, the total time complexity is $O(n \log n)$.

4. Question 4

You are given an array containing a sequence of $2^n - 1$ consecutive positive integers starting with 1 except that one number was skipped; thus the sequence is of the form $1, 2, 3, \ldots, k-1, k+1, \ldots, 2^n$. You have to determine the missing term accessing at most $O(n)$ many elements of A. (20 points)

Solution for Question 4:
Let's denote the array as A. It can be inferred that:

(1) for all $i = 1, 2, ..., k-1$, we have $A[i] = i$.
(2) for all $i = k, k+1, ..., 2^n - 1$, we have $A[i] = i + 1$.

Clearly, if $A[j] = j$ then also for all $i < j$ we have $A[i] = i$ and if $A[j] > j$ then for all $i > j$ we have $A[i] > i$. Thus the task is to find the smallest index i such that $A[i] > i$ and we can do this with a binary search in $O(\log\ (2^n - 1)) = O(n)$ time.

5. Question 5

Read about the asymptotic notation in the review material and determine if $f(n) = O(g(n))$ or $g(n) = O(f(n)$ or both (i.e., $f(n) = \Theta(g(n))$) or neither of the two, for the following pairs of functions

(1) $f(n) = \log_2(n)$; $g(n) = \sqrt[10]{n}$; (6 points)
(2) $f(n) = n^n$; $g(n) = 2^{n \log_2(n^2)}$; (6 points)
(3) $f(n) = n^{1+\cos(\pi n)}$; $g(n) = n$. (8 points)

You might find useful L'Hôpital's rule: if $f(x), g(x) \to \infty$ and they are differentiable, then $\lim_{x\to\infty} f(x)/g(x)$ $\lim_{x\to\infty} f'(x)/g'(x)$

Solution for Question 5:

(a) To show that $f(n) = O(g(n))$ it is enough to show that $\lim_{n\to\infty} f(n)/g(n) = 0$ because this clearly implies that $f(n) < g(n)$ whenever n is large enough. Note that $f(x), g(x) \to \infty$ when $x \to \infty$ and they are both differentiable, so we can apply L'Hopital's rule to this limit formula. In the equations below $\ln n$ denotes the log with the natural basis e; we also use the formula for change of basis: $\log_2 n = \ln n \ln_2 e$

$$\begin{aligned}
&\lim_{n\to\infty} f(n)/g(n)\\
=&\lim_{n\to\infty} f'(n)/g'(n)\\
=&\lim_{n\to\infty} \frac{(\log_2 n)'}{(n^{\frac{1}{10}})'}\\
=&\lim_{n\to\infty} \frac{(\ln n \log_2 e)'}{(n^{\frac{1}{10}})'}\\
=&\lim_{n\to\infty} \frac{\frac{1}{n}\log_2 e}{\frac{1}{10}n^{-\frac{9}{10}}}\\
=&\lim_{n\to\infty} \frac{10\log_2 e}{n^{1/10}}\\
=&0
\end{aligned}$$

Hence, since we have established that $g(n)$ grows much faster than $f(n)$, we have $f(n) = O(g(n))$ but $g(n)$ is not $O(f(n))$.

(b) First, we rewrite both $f(n)$ and $g(n)$ into an exponent of 2:

$$f(n) = 2^{n\log_2 n}$$
$$g(n) = 2^{2n\log_2 n}$$

Hence we have

$$\begin{aligned}
&\lim_{n\to\infty} g(n)/f(n)\\
=&\lim_{n\to\infty} 2^{2n\log_2 n - n\log_2 n}\\
=&\lim_{n\to\infty} 2^{n\log_2 n}\\
=&\infty
\end{aligned}$$

Therefore, $f(n) = O(g(n))$ but $g(n)$ is not $O(f(n))$

(c) Aleks has messed up this one; he meant to write $n^{1+\cos(\pi n)}$ in which case the following reasoning applies:

For n odd, we have $f(n) = n^{1-1} = 1$ and hence $f(n) = O(g(n))$ but $g(n) \neq O(f(n))$.

For n even, we have $f(n) = n^{1+1} = n^2$ and hence $g(n) = O(f(n))$ but $f(n) \neq O(g(n))$.

Thus, for arbitrary n, we have neither $f(n) = O(g(n))$ nor $g(n) = O(f(n))$ (2 points).

However, for $n^{1+\sin(\pi n)}$ we have $\sin(\pi n) = 0$ for all integers n. Thus, $n^{1+\sin(\pi n)} = n$ and consequently $f(n) = \Theta(g(n))$.

Marking note for Question 5: for (c) part full credit is given for both “solutions” despite the fact that only one is correct as the problem is stated.