Question 9 Solution

COMP3121/9101 21T3 Final Exam

This document gives a model solution to question 9 of the final exam. Note that alternative solutions may exist.

1. You are given an array A of n positive integers, each at most M. For each pair of distinct indices $1 \leq i < j \leq n$, consider the corresponding sum $A[i] + A[j]$.

   Design an algorithm which determines the kth largest of these sums and runs in $O(n \log n \log M)$ time.

   You must provide reasoning to justify the correctness and time complexity of your algorithm.

   The input consists of the positive integers n, M and k where $k \leq \frac{n(n-1)}{2}$, as well as n positive integers $A[1], \ldots, A[n]$ where each $A[i]$ satisfies $1 \leq A[i] \leq M$.

   The output is the ith largest sum $A[i] + A[j]$ where $1 \leq i < j \leq n$.

   For example, suppose $n = 4$, $k = 4$ and the array elements are $2, 5, 3, 4$. Going over pairs of distinct indices, we encounter the corresponding sums $5, 6, 7, 7, 8, 9$, so the correct answer is 7. Note that 7 appears twice in the list: it is both the third largest sum and the fourth largest sum.

   Hint: for a given positive integer S, can you determine the number of pairs of indices with corresponding sum greater than or equal to S in $O(n \log n)$ time?

   Solution for hint: Sort the elements. Then, for each index i, you can find the set of other indices $j > i$ that would form sufficiently large pairs. This set consists of all $j > i$ such that $A[j] \geq S - A[i]$, and since we sorted the array, it must therefore be simply the range $j \geq \max(j^*, i+1)$ where j^* is the smallest suitable index. We can find j^* in $O(\log n)$ using binary search (or $O(1)$ amortised using a two pointers approach). Repeating this for every i and computing the total brings the time complexity to $O(n \log n)$.

   Full solution: We know that each pair must have sum at most $2M$.

   For some $0 < S \leq 2M$, we can count the number of pairs with sum at least S in $O(n \log n)$ as above. Then:

   1. If the number of such pairs is less than k, the answer must be strictly smaller than S.

2. If the number of pairs is greater than or equal to k, the answer must be larger than or equal to S.

We want to find the largest S such that the number of pairs equal or larger than S is not less than k. The above criterion allows us to find this value of S by binary search.

This binary search requires $O(\log M)$ steps, each of which takes $O(n \log n)$, so the overall runtime is $O(n \log n \log M)$.

Note the termination condition we used; other choices might be incorrect. In particular multiple pairs could have the same size, so finding an S with *exactly* $k-1$ larger pairs won't work. Only minor penalties were incurred for such mistakes.