CS 325: Analysis of Algorithms
Group Assignment 1

Hyun Taek Oh
Busra Dedeoglu
Yu-Hao Shih

January 30, 2024

1 Report A:

There are two arrays A[1,...,m] and B[1,...,n], which include integers between t_{min} and t_{max}. We wrote a code, which finds when a∈A and b∈B and a+b in [t_{min}, t_{max}], it finds valid pairs' count.

Step 1: The code reads inputs, which given in input.txt . We have vecA and vecB, which represent array A and Array B. We used merge_sort function , which sorts an input array in non-decreasing order. We created this function for seeing how works recursion algorithms manually.

Step 2: We used binary search via get_invalid_bmin function on vecB to find the number of elements less than or equal to $t_{max} - a$ for each elements a $\in$ vecA.

Step 3: The get_sum_of_pairs function goes through each number in the first array vecA. For each of these numbers, it modifies the range based on that specific number. Then, it counts how many pairs can be formed by combining this number with elements from the second array vecB within the adjusted range. This counting is done using binary search functions.

Step 4: The print_to_output function saves the final result to output.txt file. It takes the calculated value and writes it into the output.txt file. The code uses specific instructions given when starting the program from the command line. It takes the names of input and output files as inputs. Then, it reads the numbers from the input file, figures out how many pairs meet certain conditions, shows that number on the screen, and saves it in the output file.

2 Calculating Running Time Of A

The code uses a method called merge sort to arrange numbers in a certain order. This method usually takes a reasonable amount of time, specifically O(n · logn), where 'n' is the number of elements. The part where it looks for specific numbers has a method called "binary search," which is also efficient and generally takes O(logn) time. When it goes through the list called vecA and checks each number using the binary search, it takes O(n). In total, the code works efficiently, especially because of the sorting step, and the overall time it takes is O(n · logn), which is considered quite good for these kinds of problems.

3 Proof of Correctness Of A

The correctness of the merge sort algorithm is well-established. We did debug and saw that the code divide a half continuously and checks correctly. The binary search algorithms are correctly implemented, ensuring the correct count of elements satisfying the given conditions. The main part of the program that goes through the list called vecA, changes some values, and counts pairs is accurate as well. It does this correctly by using the accurate binary search method.