ABC123D Cake 123

Given 3 arrays a[x], b[y] and c[z], output the first K sums of a[i] + b[j] + c[k] in descending order.

Solution 1: clever sort

First, make an array d[k \* k] containing all possible a[i] + b[j]. We only need to consider the first k elements of a and the first k elements of b because we only need k elements of d later. Then, sort it and obtain the first K elements. Next, make an array e[k \* k] containing all possible d[i] + c[j]. Similarly, we only need to consider the first k elements of d and the first k elements of e because we only need k elements in the output. Sort e, then output the first k elements.

Time complexity:

P.S. Do you know why it’s not ?

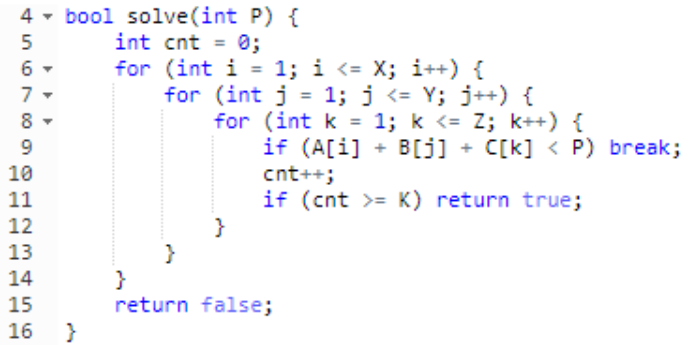
Solution 2: priority queue

First, sort the 3 arrays in descending order. Push {a[0] + b[0] + c[0], 0, 0, 0} into a priority queue. Repeat for K times: Output the top, pop, if top is {a[i] + b[j] + c[k], i, j, k} push {a[i + 1] + b[j] + c[k], i + 1, j, k}, {a[i] + b[j + 1] + c[k], i, j + 1, k}, {a[i] + b[j] + c[k + 1], i, j, k + 1} into the priority queue. Remember to check if a combination of {i, j, k} is already pushed!

Time complexity:

Solution 3: binary search

Binary search for the number such that there are combinations of a[i] + b[j] + c[k] such that their sum is less than N. To calculate how many combinations are behind (during binary search), use the following function:



After binary search, you can use similar method to push all sums into a vector, sort it and output.

Time complexity: or if you optimize

Solution 4: my solution

I created an array d[x \* y] that stores all combinations of a[i] + b[j], and sorted it in descending order. Then, for every element c[i], I pushed {c[i] + d[0], i, 0} into a priority queue. I repeated this for K times: output the top, pop, if the top was {c[i] + d[j], i, j} push {c[i] + d[j + 1], i, j + 1} into the priority queue.

Time complexity:

There was another solution in the editorial but I did’t understand it at all.