# 2462. Total Cost to Hire K Workers

You are given a 0-indexed integer array costs where costs[i] is the cost of hiring the ith worker.

You are also given two integers k and candidates. We want to hire exactly k workers according to the following rules:

You will run k sessions and hire exactly one worker in each session.

In each hiring session, choose the worker with the lowest cost from either the first candidates workers or the last candidates workers. Break the tie by the smallest index.

For example, if costs = [3,2,7,7,1,2] and candidates = 2, then in the first hiring session, we will choose the 4th worker because they have the lowest cost [3,2,7,7,1,2].

In the second hiring session, we will choose 1st worker because they have the same lowest cost as 4th worker but they have the smallest index [3,2,7,7,2]. Please note that the indexing may be changed in the process.

If there are fewer than candidates workers remaining, choose the worker with the lowest cost among them. Break the tie by the smallest index.

A worker can only be chosen once.

Return the total cost to hire exactly k workers.

## SOLUTION IN C++

class Solution {

public:

long long totalCost(vector<int>& costs, int k, int candidates) {

long ans = 0;

int i = 0;

int j = costs.size() - 1;

priority\_queue<int, vector<int>, greater<>> minHeapL;

priority\_queue<int, vector<int>, greater<>> minHeapR;

for (int hired = 0; hired < k; ++hired) {

while (minHeapL.size() < candidates && i <= j)

minHeapL.push(costs[i++]);

while (minHeapR.size() < candidates && i <= j)

minHeapR.push(costs[j--]);

if (minHeapL.empty())

ans += minHeapR.top(), minHeapR.pop();

else if (minHeapR.empty())

ans += minHeapL.top(), minHeapL.pop();

// Both `minHeapL` and `minHeapR` are not empty.

else if (minHeapL.top() <= minHeapR.top())

ans += minHeapL.top(), minHeapL.pop();

else

ans += minHeapR.top(), minHeapR.pop();

}

return ans;

}

};