**Optimal Merge Pattern**

**Problem**

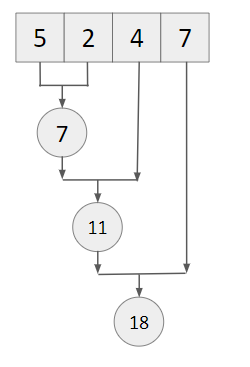
You are given n files, with their computation times in an Array. You can perform the following operation

Operation:

Choose/ take any two files, add their computation times and append it to the list of computation times. {Cost = Sum of computation times}

Do this until we are left with only one file in the array. We have to do this operation so that we can get the minimum cost finally.

**Example**

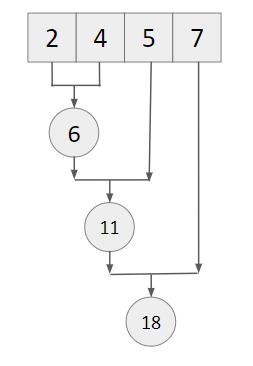


Cost = 7 + 11 + 18 = 36 {may not be minimum}

**Approach**

1. Push all elements to a min heap.
2. Take the top 2 elements one by one, and add the cost to the answer. Push the merged file to the min heap.
3. When a single element remains, output the cost.

**Dry Run**



Cost = 6 + 11 + 18 = 35 {minimum}

**Code**

**#include<bits/stdc++.h>**

**using namespace std;**

**#define int long long**

**signed main() {**

**int n; cin >> n;**

**vector<int> a(n);**

**for(int i=0; i<n; i++) {**

**cin >> a[i];**

**}**

**priority\_queue<int, vector<int>, greater<int>> minheap;**

**for(int i=0; i<n; i++) {**

**minheap.push(a[i]);**

**}**

**int ans = 0;**

**while(minheap.size() > 1) {**

**int e1 = minheap.top();**

**minheap.pop();**

**int e2 = minheap.top();**

**minheap.pop();**

**ans += e1 + e2;**

**minheap.push(e1 + e2);**

**}**

**cout << ans << endl;**

**return 0;**

**}**