CS29002 Algorithms Laboratory

Assignment No: 8

Last date of submission: 28–September–2016

You are at the zeroth (ground) floor of an n-story building, carrying an empty bag. You are told beforehand n positive weights $w_1, w_2, \ldots, w_n$. At every floor (including the ground floor), there is a counter. You will be given gold of weights w_j from the counters. It is your choice in which order you will ask for the weights from the counters. Your goal is to reach the roof with your bag containing gold of weight $w_1 + w_2 + \cdots + w_n$. You then escape with the bag by a helicopter. If you carry a weight of G_i from floor i to floor $i+1$ for $i = 0, 1, 2, \ldots, n-1$ (the roof is assumed to be at the n-th floor), your total effort is $E = G_0 + G_1 + \cdots + G_{n-1}$. You plan to determine the sequence of requesting for the weights so that E is as small as possible. Greedy algorithms work for this problem (Part 1) and a variation of this (Part 2).

Part 1: Assume that at each floor you can ask for exactly one of the weights $w_1, w_2, \ldots, w_n$, that you have not asked for so far. In your bag, you mix the gold that you get, and move up. Write an $O(n \log n)$-time function *mineffort1*() that asks for the weights in the increasing (non-decreasing) order. Print the sequence of your requests, the floor-by-floor efforts $G_0, G_1, \ldots, G_{n-1}$, and your total effort E.

Convince yourself that this greedy approach produces an optimal solution.

Part 2: In this part, you have an option at every floor i: you may first deposit at the counter all the gold (of weight G_{i-1}) that you are carrying from floor $i-1$, and then ask for gold of weight $W + W'$ from the counter. Here, each of W and W' must be either an individual weight w_j or a weight G_k (for $k < i$) that you deposited at a lower floor k. You must not ask for the same w_j or G_k multiple times. You do not have to exercise this new option at every floor, that is, you may opt for not depositing the gold you are carrying. In that case, you ask for a single weight from the counter, and that has to be some w_j or G_k not requested earlier (if it is G_k, this amount must have been deposited at a lower floor k).

For example, take $n = 5$. At the ground floor, you must choose some w_j because you have not deposited any gold earlier. Let it be w_2, so your zeroth-to-first floor effort is $G_0 = w_2$. At the first floor, you ask for w_4, and so $G_1 = w_2 + w_4$. At the second floor, you deposit G_1, and ask for w_1, w_5, so you have $G_2 = w_1 + w_5$. At the third floor, you deposit G_2, and ask for G_1, w_3, so $G_3 = G_1 + w_3 = w_2 + w_3 + w_4$. At the fourth floor, you ask for G_2, and have $G_4 = G_3 + G_2 = w_1 + w_2 + w_3 + w_4 + w_5$. Thus, your total effort is $E = G_0 + G_1 + G_2 + G_3 + G_4 = (w_2) + (w_2 + w_4) + (w_1 + w_5) + (w_2 + w_3 + w_4) + (w_1 + w_2 + w_3 + w_4 + w_5) = 2w_1 + 4w_2 + 2w_3 + 3w_4 + 2w_5$.

Write an $O(n \log n)$-time function *mineffort2*() to solve your problem. Your function should print your entire activity (asking for weights, depositing, floor-by-floor effort), and the total effort E.

As a comment following your function, write a short proof that your algorithm produces an optimal solution.

The *main*() function:

- The reader supplies n and a sequence $w_1, w_2, \ldots, w_n$ of individual weights (positive integers). The input sequence is not assumed to be sorted.
- Call *mineffort1*() to print your activity and total effort, on the given weights $w_1, w_2, \ldots, w_n$.
- Call *mineffort2*() to print your activity and total effort, on the same input weights $w_1, w_2, \ldots, w_n$.

Sample output

```
n = 6
99 13 57 90 69 25

+++ Part 1
    Floor( 0): Adding w[ 2] =  13 to bag, G[ 0] =   13
    Floor( 1): Adding w[ 6] =  25 to bag, G[ 1] =   38
    Floor( 2): Adding w[ 3] =  57 to bag, G[ 2] =   95
    Floor( 3): Adding w[ 5] =  69 to bag, G[ 3] =  164
    Floor( 4): Adding w[ 4] =  90 to bag, G[ 4] =  254
    Floor( 5): Adding w[ 1] =  99 to bag, G[ 5] =  353
--- Total effort = 917

+++ Part 2
    Floor( 0): Adding w[ 2] =  13              to bag, G[ 0] =   13
    Floor( 1): Adding w[ 6] =  25              to bag, G[ 1] =   38
    Floor( 2): Adding w[ 3] =  57              to bag, G[ 2] =   95
    Floor( 3): Depositing G[ 2] =   95
             : Adding w[ 5] =  69, w[ 4] =  90 to bag, G[ 3] =  159
    Floor( 4): Depositing G[ 3] =  159
             : Adding G[ 2] =  95, w[ 1] =  99 to bag, G[ 4] =  194
    Floor( 5): Adding G[ 3] = 159              to bag, G[ 5] =  353
--- Total effort = 852
```

Submit a single C/C++ source file. Do not use global/static variables.