**Submission instructions as in previous homeworks.**

## 1 (100 PTS.) Break up!

You are given a vertical pile of $n$ books. The books are numbered $1, 2, \ldots, n$ (in this order from the bottom of the pile). The $i$th book has weight $w_i$. Given a pile of books $[\![i \,..\, j]\!]$, you can pick an index $i \le k < j$, and break the pile into two piles $[\![i \,..\, k]\!]$, and $[\![k+1 \,..\, j]\!]$. Since the books are heavy, performing such an operation requires $\sum_{\ell=i}^{j} w_\ell$ units of energy, where $w_\ell$ is the weight of the $\ell$th book.

The task at hand is to break the given pile of $n$ books into $n$ piles of single books, using the above breakup operation. Note, that given a pile of books, the energy required to break it up into two piles is always the same – it is the total weight of the pile. In particular, moving repeatedly just the top book of the pile is probably not going to be the optimal solution.

For example, if the initial pile is $B_0$ is $[\![1 \,..\, 5]\!]$ a solution might be a sequence of commands:

- Break $[\![1 \,..\, 5]\!]$ into two piles $[\![1 \,..\, 2]\!]$ and $[\![3 \,..\, 5]\!]$.
- Break $[\![1 \,..\, 2]\!]$ into two piles $[\![1]\!]$ and $[\![2]\!]$.
- Break $[\![3 \,..\, 5]\!]$ into $[\![3 \,..\, 4]\!]$ and $[\![5]\!]$
- Break $[\![3 \,..\, 4]\!]$ into $[\![3]\!]$ and $[\![4]\!]$.

It is easy to verify that the total energy required to implement this solution is $2w_1+2w_2+3w_3+3w_4+2w_5$.

**1.A.** (70 PTS.) Let $\mathcal{E}$ be the minimum total energy required by any solution that breaks the given pile into singletons. Describe an algorithm, as fast as possible, that computes $\mathcal{E}$. Bound the running time of your algorithm.

**1.B.** (30 PTS.) Describe how to modify your algorithm in (A) so that it computes the solution itself (i.e., the sequence of breakup operations in the optimal solution). Describe the algorithm/procedure that outputs this solution once it is computed.

## 2 (100 PTS.) Wireless routers.

You are given $n$ locations $\ell_!, \ldots, \ell_n$ of people living on Red street. Here $\ell_i$ is the distance of the $i$th person from the beginning of the street in feet (Red street is straight), where $\ell_1 < \ell_2 < \cdots < \ell_n$.

We would like to install $k$ wireless routers to serve these people need. Specifically, for a set of locations $Y = \{y_1, \ldots, y_k\}$, the cost of this solution to the $i$th customer is the distance of $\ell_i$ to its nearest wireless routers in $Y$ to the power four. Formally, it is $\mathrm{cost}(\ell_i, Y) = \min_{y \in Y} (y - \ell_i)^4 = (\ell_i - \mathsf{nn}(\ell_i, Y))^4$, where $\mathsf{nn}(\ell_i, Y)$ is the location of the nearest point to $\ell_i$ in $Y$. Indeed, the further a person's computer is from a wireless router, the stronger the signal his computer has to use to communicate with the router, and the energy of this signal grows as (say) a fourth power of the distance.

The ***cost*** of the solution $Y$ is $\mathrm{cost}(Y) = \sum_i \mathrm{cost}(\ell_i, Y)$.

Given the $n$ locations $\ell_1, \ldots, \ell_n$ and $k$, provide an algorithm, as fast as possible (in $n$ and $k$), that computes the set $Y \subseteq \{\ell_1, \ldots, \ell_n\}$ of $k$ routers, such that $\mathrm{cost}(Y)$ is minimal. What is the running time of your algorithm? (Note, that the routers can be placed only in the given locations of the houses[1].)

[1] The variant where the routers can be placed anywhere is slightly harder – you can think about this variant for fun. Then you can think about your life, and what you do for fun.

## 3 (100 PTS.) A bridge too far.

You are planning a military campaign, and you had decided to break your army into two corps $P$ and $Q$ (a corps is a military unit bigger than a division). Corps $P$ would occupy cities $p_1, p_2, \ldots, p_n$ (in this order), while the other corps $Q$ would occupy cities $q_1, \ldots, q_m$ (in this order). The problem of course is that if $P$ is in city $p_i$, and it is being attacked, then the second corps $Q$ (which might be at city $q_j$), should be close enough to be able to provide support to $P$[2].

To this end, you are given numbers $nm$ numbers, where $d(i, j)$ is the distance between $p_i$ and $q_j$, for $i = 1, \ldots, n$ and $j = 1, \ldots m$. You are also given a threshold $\ell$. You need now to schedule the movements of the two corps. Specifically, you need to output a schedule $(i_1, j_1), \ldots, (i_t, j_t)$. Here $(i_k, j_k)$ denotes that in the beginning of the $k$th week of the campaign $P$ would be at city $p_{i_k}$, and $Q$ would be at city $q_{j_k}$.

Formally, the requirements on the computed schedule are the following:

(I) $i_1 = 1$ and $j_1 = 1$ (i.e., the campaign starts with $P$ at $p_1$ and $Q$ at $q_1$).

(II) $i_t = n$ and $j_t = m$ (i.e., the campaign ends with $P$ at $p_n$ and $Q$ at $q_m$).

(III) For any $k$, $1 \leq k \leq t$, we have $d(i_k, j_k) \leq \ell$.

(IV) For any $k$, we have that either $i_{k+1} = i_k + 1$ or $j_{k+1} = j_k + 1$ (but not both – only one corps can move at any point in time). [As such, $t = n + m - 1$.]

**3.A.** (50 PTS.) Given the above input, provide an algorithm, as efficient as possible, that computes whether there is a feasible schedule that complies with all the above conditions. What is the running time of your algorithm? Argue shortly why your algorithm is correct (and why the stated running time is correct).

**3.B.** (10 PTS.) Describe how to modify the algorithm in (A) so that it outputs the feasible schedule.

**3.C.** (40 PTS.) Modify your algorithm so that it computes and outputs the smallest value of $\ell$, such that there is still a feasible schedule. What is the running time of your algorithm?

[2] Failing to have such close by support might end up in a military disaster – see the movie "a bridge too far".