

CMSC 451: Project, Spring 2004

The project, described in more detail below, is in two parts. The first part is due April 20th, at the beginning of class. The second part is due May 6th, at the beginning of class. These are firm deadlines.

1 The problem

There are n houses on a straight-line street; these houses are numbered $1, 2, \dots, n$ from left to right. The city council is planning to deploy k ($k \leq n$) trash-bins on the street. As a member of the city council, you need to decide the best locations to place these k trash-bins so that the residents will find it convenient to dispose of their trash. The inputs to the problem are:

- (I1) positive integers k and n , with $k \leq n$; and
- (I2) an increasing sequence of non-negative integers h_1, h_2, \dots, h_n , where h_i denotes the distance between the i^{th} house and the leftmost end of the street.

You are only allowed to place the trash-bins in front of houses; residents of each house will go to the closest trash-bin to dispose of their trash. For example, if $k = 2$, $n = 5$, $h_1 = 1$, $h_2 = 3$, $h_3 = 6$, $h_4 = 7$ and $h_5 = 9$, the two trash-bins can be placed in front of houses 2 and 4. In such a case, residents of house 3 will use the latter trash bin, for example. (We cannot place a trash-bin at a distance of 4 from the leftmost end of the street because there is no house there.) Define d_i to be the distance between house i and the closest trash-bin to the house. In the above example, we have $d_1 = 2$, $d_2 = 0$, $d_3 = 1$, $d_4 = 0$ and $d_5 = 2$.

Given the inputs (I1) and (I2), the problem is to choose the locations of the k trash-bins such that the sum $\sum_{i=1}^n d_i$ is minimized.

- (i) Given an instance of the above problem with $k = 1$, show that placing the trash-bin in front of house $\lfloor (n+1)/2 \rfloor$ gives an optimal solution. Recall that given a real number x , $\lfloor x \rfloor$ denotes the greatest integer y such that $y \leq x$. **(3 points)**
- (ii) Develop as efficient an algorithm you can, using dynamic programming, for the given problem. Your algorithm *need not* output where the trash-bins should be placed; it just needs to output the sum of all the d_i in an optimal solution. Analyze the running time of your algorithm. (The problem (i) above may be a useful starting point.) **(7 points)**

The solution to these problems will be given on April 20th.

Important Note 1: For problem (ii) above, you may use the following hint: let $m[i, j, r]$ denote the optimal solution value for serving houses $\{i, i+1, i+2, \dots, j\}$ using r bins. Start by developing a recurrence relation for $m[i, j, r]$. (Note that the final answer we are looking for is $m[1, n, k]$.) To develop the recurrence relation, you can use the following idea. Consider the problem \mathcal{P} of serving houses $\{i, i+1, i+2, \dots, j\}$ using r bins, where $r \geq 2$. Given some solution \mathcal{S} to this problem \mathcal{P} , let $last(\mathcal{S})$ denote the trash-bin that is placed in front of the highest-numbered house. (For instance, if \mathcal{S} places trash-bins in front of houses 2, 6, 9, then $last(\mathcal{S})$ denotes the trash-bin placed in front of house 9.) Suppose someone gave you a value ℓ which lies in the range $\{i, i+1, i+2, \dots, j\}$ and told you the following: “there is an *optimal* solution \mathcal{S} to the problem \mathcal{P} , in which people living in houses $\{\ell, \ell+1, \dots, j\}$ use the trash-bin $last(\mathcal{S})$, while people living in houses $\{i, i+1, i+2, \dots, \ell-1\}$ use trash-bins other than $last(\mathcal{S})$.” How can you use this information to develop the recurrence relation? And, of course, nobody is going to give us this information – we will have to find it out. How can we find it out?

Important Note 2: As you are doing the above, start coding up a simple “exhaustive search” based algorithm for the problem, which basically tries out all possible ways of placing k trash-bins, and chooses the optimal solution from these. Please do not submit this code with Part I; it is due in Part II of the project. But it will help you greatly to start this implementation right away.

2 Part II of the Project

This part is due on May 6th, at the beginning of class. Here, you will code up the exhaustive search algorithm mentioned above, as well as the dynamic programming algorithm (which is the heart of Part I, and whose description will be given on April 20th); you will then compare their running times. The details of how to generate problem instances, report running times, etc., will be announced soon. This part is worth 10 points.