CSE 431/531 Homework Assignment 4

Due in class on Thursday, Apr 05.

March 22, 2007

There are totally 7 problems, 10 points each. You should do them all. We will grade only 5 problems chosen at my discretion. If it so happens that you don't do one of the problems we don't grade, then no points will be deducted.

Note: this homework is on dynamic programming. To present a DP algorithm, please conform to the following format: (a) description of the idea and argue the correctness, (b) high-level pseudo code, (c) analysis of time **and** space complexity.

More note: also remember to describe how to construct the optimal solution if asked! A recurrence and computation of the optimal objective value is not sufficient in such case.

Example 1 (Alice in Disneyland). Alice, a lazy CSE 431/531 student, plans to go to Disneyland for the Spring break (hmm, what a plan!). She'd like to go to many events to make it worth the trip. However, being lazy she does not want to walk too far. At the start of each hour, there are many events going on at different locations. Events last for 30 minutes each. The other 30 minutes of each hour is sufficient for her to walk from any place to any place in Disneyland (OK, I admit, this is the only thing I made up). She does not mind sitting at one place for a few hours if it could save her some walking time.

Given a set of n events, their starting hours t_i, the distance $d(i,j)$ between event i and event j (note: $d(i,j) = 0$ if $i = j$), and the number k ($\leq n$) of events which Alice would like to participate in. She wants to schedule visits to events at distinct times, so as to minimize the total distance she must walk between events. Give a dynamic programming algorithm to solve lazy Alice's problem.

Sample Solution. Following the standard format:

(a) **Idea.** Re-order the events so that $t_1 \leq t_2 \leq \cdots \leq t_n$. Also, set $d(i,j) = \infty$ if $t_i = t_j$, namely the cost to attend two events at the same time is prohibitedly high. Let l_j be the distance between the entrance and the location of event j. (You can assume that $l_j = 0$.)

For $1 \leq i \leq k$ and $1 \leq j \leq n$, let l_{ij} be the minimum possible walking distance for Alice to visit exactly i events with j being the last event she visits. We are interested in l_{kn}.

Firstly, note that $l_{1j} = l_j, \forall j$.

Secondly, $l_{ij} = \infty$ when $i > j$ because there is no way to visit more events than the number of possible events up to the jth event.

For $i \geq 2$, consider an optimal walk scheduling for Alice to visit i events with j being the last. Let j' be the next to last event that Alice visits, then clearly $l_{ij} = l_{(i-1)j'} + d(j',j)$. In other words, to visit i events and finish at j, we have to visit $(i-1)$ events, finishing at some $j' < j$, and then go from j' to j. Obviously the walking distance up to j' also have to be optimal, thus when $j \geq i \geq 2$

$$l_{ij} = \min_{1 \leq j' \leq j-1} \left\{ l_{(i-1)j'} + d(j',j) \right\}.$$

In summary,

$$l_{ij} = \begin{cases} \infty & \text{if } j < i \\ l_j & \text{if } j \geq i = 1 \\ \min\limits_{1 \leq j' \leq j-1} \left\{ l_{(i-1)j'} + d(j', j) \right\} & \text{when } j \geq i \geq 2 \end{cases}$$

Since we knew $l_{(i-1)j'} = \infty$ when $j' < (i-1)$, we only need to iterate j' from i to $j-1$. The above definition can be rewritten as

$$l_{ij} = \begin{cases} \infty & \text{if } j < i \\ l_j & \text{if } j \geq i = 1 \\ \min\limits_{i-1 \leq j' \leq j-1} \left(l_{(i-1)j'} + d(j', j) \right) & \text{when } j \geq i \geq 2 \end{cases}$$

When filling out the table $L = (l_{ij})$, we are only interested in the entries at or above the main diagonal ($j \geq i$), since the rest are ∞. We first fill out the first row with l_j's ($i = 1$). Entries from the 2nd row and beyond ($d_{ij}, i \geq 2$) depend on entries on the previous row (row $i-1$) and strictly smaller column number ($j' \leq j-1$), hence the upper half of the table L can be filled out from left to right, top to bottom. Note that the table L is of dimension $k \times n$, hence it is not necessarily a square table.

(b) **Pseudo Code.**

```
LAZY-ALICE($D, n, k$)

 1: // We assume the events are sorted in starting time in advance.
 2: for $i \leftarrow 1$ to $k$ do
 3:   for $j \leftarrow i$ to $n$ do
 4:     if $i = 1$ then
 5:       $l_{ij} \leftarrow l_j$
 6:     else
 7:       $l_{ij} \leftarrow \min_{i-1 \leq j' \leq j-1} \left( l_{(i-1)j'} + d(j', j) \right)$
 8:     end if
 9:   end for
10: end for
11: Return $l_{kn}$
```

Although we did not do it here, you should think about how to construct an optimal sequence of events to visit.

(c) **Analysis.**

As far as time is concerned, it takes $O(n \lg n)$ for the sorting, and then $O(kn^2)$ for the three loops we have (the outer loop is of size k, and two inner loops are at most n each). To be precise, the running time after sorting is

$$\Theta(n) + \sum_{i=2}^{k} \sum_{j=i}^{n} \sum_{j'=i-1}^{j-1} \Theta(1) = \Theta(kn^2).$$

Since we had two outer loops of size k and n each, and the min operation of the inner loop takes $O(n)$, we can conclude that the running time is $O(n^2 k)$, for a total of $O(n^2 k)$. Note here that we assume $k \geq 2$, otherwise the total running time for table filling would just be

$\Theta(n)$. In summary, the running time is $O(n \lg n)$ if $k = 1$, and $O(n^2 k)$ if $k \geq 2$. The space requirement is obviously $\Theta(nk)$.

$\square$

Problem 1. Textbook, Chapter 6, Problem 8.

Problem 2. Textbook, Chapter 6, Problem 11.

Problem 3. Textbook, Chapter 6, Problem 14.

Problem 4. Textbook, Chapter 6, Problem 16.

Problem 5. Textbook, Chapter 6, Problem 19.

Problem 6. Textbook, Chapter 6, Problem 24.

Problem 7. Textbook, Chapter 6, Problem 25.