Lindsey Bieda and Joe Frambach
Dynamic Programming Problems
10.17.2011

27. Give a polynomial time algorithm for the following problem. The input consists of two dimensional array $R$ of non-negative integers, and an integer $k$. The value $R_{t,p}$ gives the number of of users requesting page $p$ at time $t$ ( say from a www server. At each integer time, the server can broadcast either 0 or 1 pages. If the server broadcasts page $p$ at time $t$, the the requests of all the users who requested page $p$ strictly before time $t$ are satisfied. The server can make at most $k$ broadcasts. The goal is to pick the $k$ times to broadcast, and the pages to broadcast at those $k$ times, in order to minimize the total time (over all requests) that requests/users have to wait in order to have their requests satisfied. So problem 24 is a special case of this problem where there is only one page.

**Tree Description**:
All of the nodes in the tree are stored as [time $t$, broadcasts used $b$, page currently broadcasted $p$]. The nodes store the values as [total wait time $W$, wait time since last broadcast $w$]. Each of the children of any node represent the decision to either broadcast no page or broadcast one of the requested pages. The tree is rooted at $[0, 0, 0]$ reprensenting the state in which no pages have been broadcast yet and no requests have come in yet.

**Pruning Rules**:

1. For all levels if any node has used more than $k$ broadcasts prune that node.

**Algorithm**:

```
$A[*, *] = [\infty, \infty]$ *Initialization*
$A[0, 0] = [0, 0]$

$n$ = total number of requests
$n_p$ = total number of pages
For $t = 0$ to $n - 1$:
    For $b = 0$ to $\min(k, n - \ell)$: *Take care of the pruning rule*
        $[W, w] = A[t, b, p]$ *for any locations where W or w are cited the values are from this location in the array*
        A$[\ell + 1, b, 0] = [W + w + \Sigma R_{t+1,*}, w + \Sigma R_{t+1,*}]$ *case: do not broadcast anything*
        For $p = 1$ to $n_p$:
            A$[t + 1, b + 1, p] = [W + w\Sigma R_{t+1,*} - R_{t+1,p}, w + \Sigma R_{t+1,*} - R_{t+1,p}]$
```

3. Show that if there is an $O(n^k)$, $k \geq 1$, time algorithm for squaring a degree $n$ polynomial, then there is an $O(n^k)$ time algorithm for multiplying two degree $n$ polynomials. Assume that the polynomials are given by their coefficients.

Polynomial Multiplication $\leq$ Polynomial Squaring

```
Program Polynomial Multiplication:
    read $I$,$J$
    $P = I + J$
    $Q =$ Polynomial Square ($P$)
    $Q = Q - I^2 - J^2$
    output $Q$
```

Constructing $Q$ takes linear time since if $I = [i_1, i_2, \ldots, i_n]$ $J = [j_1, j_2, \ldots, j_n]$ $Q = [i_1+j_1, i_2+j_2, \ldots, i_n+j_n]$.

4. Consider the following variant of the minimum Steiner tree problem. The input is $n$ points in the plane. Each point is given by its Cartesian coordinates. The problem is build a collection of roads between these points so that you can reach any city from any other city and the total length of the roads is minimized. The collection of roads should be output as an adjacency list structure, that is, for each point $p$ of intersection of roads there are a list of roads that meet that point. Roads must terminate when they meet one of the original input points, or when they meet another road. For example, if the points are (1, 1), (-1, 1), (1,-1), (-1,-1), (2, 2) the output would be: From the point (0, 0) there are roads to the points (1, 1), (-1, 1), (1,-1), and (-1,-1). From the point (1, 1) there are roads to (0, 0) and (2, 2). From the point (-1, 1) there is a road to (0, 0). From the point (1,-1) there is a road to (0, 0). From the point (-1,-1) there is a road to (0, 0). Roads from (1, 1) to (-1,-1), and from (1,-1) to (-1, 1) would not be allowed because they cross. A road from (-1,-1) to (2, 2) would not be allowed since it would cross the point (1, 1). Show by reduction that if you can solve this problem in linear time, then you can sort n numbers in linear time.

Sorting $\leq$ Steiner Tree Problem

Program Sorting:

Read $I$.

Create $I' = (i, i)$ for each element $i$ in list $I$. The mapping on a cartesian plane is linear.

$O =$ Steiner$(I')$.

The output of Steiner is a set of edges joining each point to the next closest one.

In linear time, we can find the minimum element of $I$. Starting here, there is only one road emerging from that city to the next highest input. From there, continue in linear time to the highest element of $I$.