Lindsey Bieda and Joe Frambach
Dynamic Programming Problems
10.14.2011

24. Give a polynomial time algorithm for the following problem. The input consists of a sequence $R = R_0, \ldots, R_n$ of non-negative integers, and an integer k. The number R_i represents the number of users requesting some particular piece of information at time i (say from a www server. If the server broadcasts this information at some time t, the requests of all the users who requested the information strictly before time t are satisfied. The server can broadcast this information at most k times. The goal is to pick the k times to broadcast in order to minimize the total time (over all requests) that requests/users have to wait in order to have their requests satisfied.

Tree definition:
Each level represents a time step. At each time step, we may decide to broadcast or not. So, each node has two children. Each node is indexed by its level, ℓ, and the number of broadcasts used so far, b. The nodes each store two values: the total wait time since the last broadcast, σ, and the total wait time, Σ. The tree is rooted at $[0,0] = [0,0]$, the state before any requests have come in.
In the case of a broadcast, σ is reset to the current level's user-count; else σ is increased by the current level's user-count. Σ is then increased by the new value of σ.

Pruning Rules:

1. For all levels if any node has used more than k broadcasts prune that node.
2. If there are more broadcasts remaining than remaining levels, prune that node. Since, it would be impossible to use k broadcasts.

Algorithm:
$A[*,*] = \infty$ *Initialization*
$A[0,0] = 0$

For $\ell = 0$ to $n-1$:
    For $b = \max(0,\, k-1)$ to $\min(k,\, n-\ell)$: *Take care of the pruning rules*
        A$[\ell+1,\, b]$ = A$[\sigma + R_{\ell+1},\, \Sigma + \sigma + R_{\ell+1}]$
        A$[\ell+1,\, b+1]$ = A$[R_{\ell+1},\, \Sigma + R_{\ell+1}]$

25. Assume that you are given a collection $B_1, \ldots, B_n$ of boxes. You are told that the weight in kilograms of each box is an integer between 1 and some constant L, inclusive. However, you do not know the specific weight of any box, and you do not know the specific value of L. You are also given a pan balance. A pan balance functions in the following manner. You can give the pan balance any two disjoint sub-collections, say S_1 and S_2, of the boxes. Let $|S_1|$ and $|S_2|$ be the cumulative weight of the boxes in S_1 and S_2, respectively. The pan balance then determines whether $|S_1| < |S_2|$, $|S_1| = |S_2|$, or $|S_1| > |S_2|$. You have nothing else at your disposal other than these n boxes and the pan balance. The problem is to determine if one can partition the boxes into two disjoint sub-collections of equal weight. Give an algorithm for this problem that makes at most $O(n^2L)$ uses of the pan balance. For partial credit, find an algorithm where the number of uses is polynomial in n and L.

2. Show that if there is an $O(n^k)$, $k \geq 2$, time algorithm for inverting a nonsingular n by n matrix C then there is an $O(n^k)$ time algorithm for multiply two arbitrary n by n matrices A and B.

Matrix Multiplication $\leq$ Matrix Inversion.

Program Matrix Multiplication:

read A, B

$$C = \begin{bmatrix} \mathrm{I} & A & 0 \\ 0 & \mathrm{I} & B \\ 0 & 0 & \mathrm{I} \end{bmatrix}$$

$C^{-1} = Inversion(C)$

We know that:

$$C^{-1} = \begin{bmatrix} \mathrm{I} & -A & AB \\ 0 & \mathrm{I} & -B \\ 0 & 0 & \mathrm{I} \end{bmatrix}$$

AB = top right "ninth" of C^{-1}

Output AB