Homework 5

David Bozarth
CES 512
Spring 2005

(1) (a) Two players alternately take the first or last member of a list of numbers. The player whose smallest number is larger is the winner. Design a dynamic programming algorithm to determine for a given list if the first player can win against any player.

Assume all elements of a given list are distinct.

Let $F(i,j)$ be the max score attainable by Player 1, given list $[x_i, \ldots x_j]$.

The score of a player is the value of the player's smallest choice.

$$F(i,j) = \max\left\{\begin{array}{l}\left(\begin{array}{l}\text{max score attainable by Player 1 from}\\ \text{giving Player 2 the list } [x_{i+1}, \ldots, x_j]\end{array}\right),\\ \left(\begin{array}{l}\text{max score attainable by Player 1 from}\\ \text{giving Player 2 the list } [x_i, \ldots x_{j-1}]\end{array}\right)\end{array}\right\}$$

$$F(i,j) = \max\left\{\begin{array}{l}\min\{F(i+2,j),\ F(i+1,j-1),\ x_i\},\\ \min\{F(i+1,j-1),\ F(i,j-2),\ x_j\}\end{array}\right\}$$

Algorithm: Given list of n elements,

If $F(1,n) > \min_x\{[x_1, x_2, \ldots x_n]\}$ then

return true

Else return false.

5.1.a cont'd

Algorithm: (assume positive integers)

Let X be a sequence of n numbers

let $M[i,j]$ be the max score attainable by a player given a sublist of X starting with x_i and ending with x_j.

The value $M[i,j]$ is calculated by $F(X, M, i, j)$.

```
FOR i ← 1 to n
    FOR j ← 1 to n
        M[i,j] ← 0

FOR i ← 1 to n
    FOR j ← 1 to n
        M[i,j] = F(X, M, i, j)

IF M[n,n] ≠ min_i {x_i} THEN RETURN TRUE

RETURN FALSE
```

~~F(X, M, i, j)~~

: IF

D. Bozarth

5.1 a cont'd

F(x, M, i, j) returns integer

:

n ← size(x)

IF n = 0 OR i > j THEN RETURN 0

IF n = 1 OR i = j THEN
    M[i, j] = x_i
    RETURN x_i

IF n = 2 OR i + 1 = j THEN
    M[i, j] = max $\{x_i, x_j\}$
    RETURN M[i, j]

IF M[i+2, j] = 0 THEN
    M[i+2, j] = F(i+2, j)

IF M[i+1, j-1] = 0 THEN
    M[i+1, j-1] = F(i+1, j+1)

IF M[i, j-2] = 0 THEN
    M[i, j-2] = F(i, j-2)

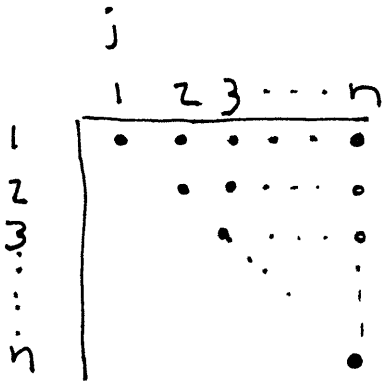
RETURN max { min { M[i+2, j], M[i+1, j-1], x_i },
             min { M[i+1, j-1], M[i, j-2], x_j }
           }

:

(1.a) cont'd

Time complexity:

F(i,j) must be calculated once for each position on or above the diagonal.



So time complexity is $\Theta\left(\frac{1}{2}n^2\right) = \boxed{\Theta(n^2)}$

(1 b) Determine the probability that the first player can win against any strategy when the input list is a random permutation of $\langle 1, 2, \ldots 100 \rangle$. Use algorithm from (a).

(Please see accompanying code listing.)

Results:

#iterations	Pr{ first player wins }
10	1.0
100	0.9
1000	0.993
10000	0.9894
100000	0.9905
1000000	0.989929

This agrees with the fact that the first player wins if their smallest number is any number other than 1. The probability of winning is thus $\frac{99}{100}$.

D. Bozarth

HW 5
Problem 1.c.

Opponent always chooses the larger of {begin, end}. Maximize your score for a given list.

$$F(i,j) = \max\{ \text{max score from choosing } x_i, \text{ max score from choosing } x_j \}$$

$$F(i,j) = \max\left\{ \begin{cases} F(i+2,j) & \text{if } x_j < x_{i+1} \\ F(i+1,j-1) & \text{else} \end{cases} + x_i, \right.$$

$$\left. \begin{cases} F(i+1,j-1) & \text{if } x_{j-1} < x_i \\ F(i,j-2) & \text{else} \end{cases} + x_j \right\}$$

The detailed algorithm would be very similar to that given in part (a). The table is initialized to the lowest ~~possi~~ value ($-\infty$ or 0). The table values are filled in by reference to previous table values wherever possible, and by function calls wherever necessary.

The value $M[i,j]$ contains the max score attainable by either player when starting with the list $\langle x_i \dots x_j \rangle$.

D. Bozarth

HW 5, Problem 2 (Ex. 15.4-5)

Give an $\Theta(n^2)$ algorithm to find the longest nondecreasing subsequence of a sequence of numbers.

Given sequence X.

Let Y = sort(X). $\Theta(n \lg n)$

LCS_LENGTH (X, Y) p.353 our text

$\Theta(n^2)$

PRINT-LCS p.355 our text

$\Theta(m+n) = \Theta(2n)$

Running time = $\Theta(n^2 + n\lg n + m + n)$

$= \Theta(n^2)$

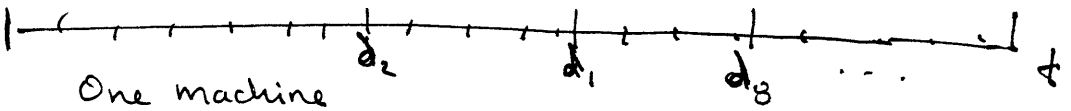
This works because Y is a nondecreasing sequence containing the same elements as X.

Therefore any nondecreasing subsequence of X must also be a nondecreasing subsequence of Y.

So their longest common subsequence must be the longest subsequence of X.

D. Bozarth

HW 5.3



One machine

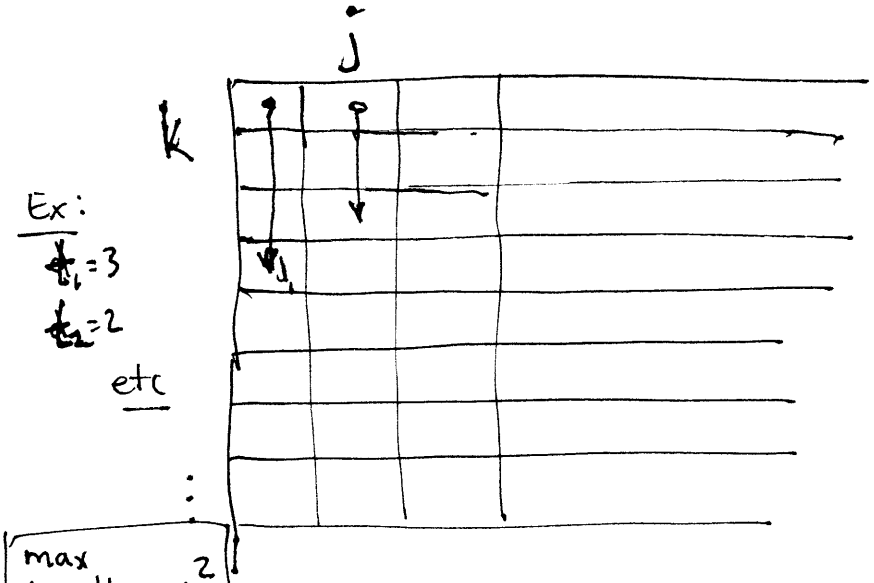
$a_1\ a_2\ a_3 \cdots a_n$ Jobs

Problem 15-7

$t_1\ t_2\ t_3 \cdots t_n$ Running Times (integers between 1 and n)

$p_1\ p_2\ p_3 \cdots p_n$ Profits ← Maximize $\sum p$

$d_1\ d_2\ d_3 \cdots d_n$ Deadlines



- Let P_{jk} be the max profit attainable from starting job a_j at time k.

recurrence

$$P_{jk} = \begin{cases} \max_j \{ P_{j(k-1)} \} + p_j & \text{if } k + t_j - 1 \le d_j \\ \max_j \{ P_{j(k-1)} \} & \text{else} \end{cases}$$

cont'd

5.3 cont'd Algorithm for finding max profit and allocation for max profit.

Given vectors a, t, p, d.
Initialize vectors $P_1, P_2, \cdots P_n$ as NIL.
Initialize vector Z to NIL. # for backtracking

FOR $j \leftarrow 1$ to n DO $P_{j0} \leftarrow 0$; $P_{j1} \leftarrow p_1$

$k \leftarrow 1$
WHILE for some $1 \le j \le n$, $P_{j(k-1)} < P_{jk}$ DO
    $k \leftarrow k+1$
    max $\leftarrow P_{1(k-1)}$; key $\leftarrow 1$

    FOR $j \leftarrow 2$ to n

        IF max $< P_{j(k-1)}$ THEN

            max $\leftarrow P_{j(k-1)}$; key $\leftarrow j$

    # now have $\max_j \{P_{j(k-1)}\}$

    Add key to Z as new list element

    FOR $j \leftarrow 1$ to n

        $P_{jk} \leftarrow$ max

        IF $k + t_j < d_j$ THEN

            $P_{jk} \leftarrow P_{jk} + p_j$

Now the table values can no longer increase

HW 5

D. Bozarth

5.3 cont'd

Get the final values of P^{max}, Z

max ← P_{1k} ; key ← 1

FOR j = 2 to n

IF max < P_{jk} THEN

max ← P_{jk} ; key ← j

Add key to Z as new list element.

Now
- max = maximum profit attainable
- Z is the list of indices $i_1, i_2, \cdots i_n$ for which the sequence of jobs $a_{i_1}, a_{i_2}, \cdots a_{i_n}$ yields max profit.

Running time

For each row in the table there are a constant number of loops, each of size n.

Since each job has time length $t_j \leq n$, and there are n jobs, there are at most n^2 rows in the table.

So the running time is $\Theta(n^3)$.