Department of Mathematical Sciences
CARNEGIE MELLON UNIVERSITY

OPERATIONS RESEARCH II 21-393

Homework 1: Due Monday September 15.

Q1 Solve the following knapsack problem:

$$\begin{array}{lccccccc} \text{maximise} & 4x_1 & + & 8x_2 & + & 13x_3 & & \\ \text{subject to} & & & & & & & \\ & 3x_1 & + & 4x_2 & + & 5x_3 & \leq & 16 \end{array}$$

$$x_1, x_2, x_3 \geq 0 \text{ and integer.}$$

Solution

w	f_1	δ_1	f_2	δ_2	f_3	δ_3
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	4	1	4	0	0	0
4	4	1	8	1	8	0
5	4	1	8	1	13	1
6	8	1	8	1	13	1
7	8	1	12	1	13	1
8	8	1	16	1	17	1
9	12	1	16	1	21	1
10	12	1	16	1	26	1
11	12	1	20	1	26	1
12	16	1	24	1	26	1
13	16	1	24	1	30	1
14	20	1	24	1	34	1
15	20	1	28	1	39	1
16	20	1	32	1	39	1

Solution: $x_1 = 0, x_2 = 0, x_3 = 3$. Maximum $= 39$.
Start with $x_1 = x_2 = x_3 = 0$. $\delta_3(16) = 1$ and so we add one to x_3. We have used up 5 units of the knapsack. There are 11 units left. $\delta_3(11) = 1$ and so we add one to x_3. We use up another 5 units and so we are left with 5.

$\delta_3(6) = 1$. We add one more to x_3. There are now 1 units in the knapsack. $\delta_3(1) = 0$ and so we move to column 2. $\delta_2(1) = 0$ and so we move to column 1. $\delta(1) = 0$ and we are done.

Q2: An $m \times n$ rectangle of wood is to be cut into smaller rectangles. An $a \times b$ rectangle is worth $m_{a,b}$. The machine that cuts rectangles can only cut full length or full width. I.e. if after cutting there is an $x \times y$ rectangle then the machine can cut it into two rectangles $z \times y$ and $(x - z) \times y$ for some z or into two rectangles $x \times z$ and $x \times y - z$.

Describe a dynamic programming algorithm for finding the way of cutting into pieces that maximises the total value of the rectangles produced.

Solution: Let $f(i, j)$ be the maximum value obtained from a rectangle with corners $(0, 0)$ and (i, j). Then

$$f(i,j) = \min \begin{cases} \min_{x \leq i}(m(i-x,j) + f(x,j) \\ \min_{y \leq j} m(i,j-y) + f(i,y) \end{cases}$$

Q3 Consider a 2-D map with a horizontal river passing through its center. There are n cities on the southern bank with x-coordinates $a(1)...a(n)$ and n cities on the northern bank with x-coordinates $b(1)...b(n)$. You want to connect as many north-south pairs of cities as possible with bridges such that no two bridges cross. When connecting cities, you can only connect city i on the northern bank to city i on the southern bank. Construct a Dynamic Programming solution to this problem. (You can assume that $a(1) < a(2) < \cdots < a(n)$, but you **cannot** assume that $b(1) < b(2) < \cdots < b(n)$. If both sequences are increasing, then the problem is trivial).

Solution: Let $f(j)$ be the maximum number of bridges choosable if we only use $(a(i), b(i), i \geq j)$. Then

$$f(j) = \max \begin{cases} f(j+1) & \text{do not choose } (a(j), b(j)) \\ 1 + f(\min\{k > j :\ b(k) > b(j)\}) & \text{choose } (a(j), b(j)) \end{cases}.$$