Name : Sradha Kedia
Date of Examination : 16 December, 2021
Time of Examination : 9:00 am to 1:00 pm
Examination Roll no. : 20234757053
Semester : III
Unique Paper Code : 223401301
Title of Paper : Design and Analysis of Algorithm
Email-Id : 200083@cs.du.ac.in
Mobile no. of student : ~~200083@cs.du.ac.in~~ 9899519848
Question no. : 1
No. of pages : 7
Name of the program : MCA
Name of the Department : DUCS

4(a) $n=4$, $w_1=1$, $w_2=2$, $w_3=3$, $w_4=5$ and $W=9$

4	0	1	2	3	4	5	6	6	8	9
3	0	1	2	3	4	5	6	6	6	6
2	0	1	2	3	3	3	3	3	3	3
1	0	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7	8	9

```
M[0][W] = 0;
for i = 1 to 4
  {
    for w = 0 to 9
      { if (w < wi)
            opt(i, w) = opt(i-1, w)
        else
            opt(i, w) = max(opt(i-1, W), wi + opt(i-1, W-wi))
      }
  }
```

I- $i=1$, $w_1=1$

$w=0$, if $0<1$ true, $opt(1,0) = opt(0,0)$

$w=1$, if $1<1$ false, $opt(1,1) =$

$max(opt(0,1),\ 1 + opt(0,0))$

$= max(0,\ 1+0) = 1$

$w=2$, if $2<1$ false, $opt(1,2) =$

$max(opt(0,2),\ 1 + opt(0,1))$

$= max(0,\ 1+0)$

$= 1$

$w=3$, if $3<1$ false,
$opt(1,3) = \max(opt(0,3),\ 1+opt(0,2))$
$= \max(0,\ 1+0) = 1$

$w=4$, if $4<1$ false,
$opt(1,4) = \max(opt(0,4),\ 1+opt(0,3))$
$= \max(0,\ 1+0)$
$= 1$

$w=5$, false, $opt(1,5) = \max(opt(0,5),\ 1+opt(0,4))$
$= \max(0,1) = 1$

$w=6$, $w=7$, $w=8$, $w=9$ all false
$\therefore$ ||ly $opt(1,w) = 1$

II $i=2$, $w_2 = 2$
$w=0$, if $(0<2)$ true, $opt(2,0) = opt(1,0) = 0$
$w=1$, if $(1<2)$ true, $opt(2,1) = opt(1,1) = 1$
$w=2$, if $(2<2)$ false, $opt(2,2) = \max(opt(1,2),\ 2+opt(1,0))$
$= \max(1,\ 2+0)$
$= 2$

$w=3$, if $(3<2)$ false, $opt(2,3) = \max(opt(1,3),\ 2+opt(1,1))$
$= \max(1,\ 2+1) = 3$

$w=4$, to $w=9$ all false $\therefore$ in the case
$\therefore$ $opt(2,4) = \max(opt(1,4),\ 2+opt(1,2))$
$= \max(1,\ 2+1) = 3$

opt (2,5) = max(opt(1,5), 2 + opt(1,3))
= max(1, 2+1) = 3

opt (2,6) = max(opt(1,6), 2 + opt(1,4))
= max(1, 2+1) = 3

opt (2,7) = max(opt(1,7), 2 + opt(1,5))
= max(1, 2+1) = 3

opt (2,8) = max(opt(1,8), 2 + opt(1,6))
= max(1, 2+1) = 3

opt (2,9) = max(opt(1,9), 2 + opt(1,7))
= max(1, 2+1) = 3

III $i=3$, $w_3 = 3$
w=0, if (0<3) true, opt(3,0) = opt(2,0) = 0
w=1, if (1<3) true, opt(3,1) = opt(2,1) = 1
w=2, if (2<3) true, opt(3,2) = opt(2,2) = 2
w=3 to w=9, false;

opt (3,3) = max(opt(2,3), 3 + opt(2,0))
= max(3, 3+0) = 3

opt (3,4) = max(opt(2,4), 3 + opt(2,1))
= max(3, 3+1) = max(3,4) = 4

opt (3,5) = max(opt(2,5), 3 + opt(2,2))
= max(3, 3+2) = 5

opt (3,6) = max(opt(2,6), 3 + opt(2,3))
= max(3, 3+3) = 6

$opt(3,7) = max(opt(2,7), 3 + opt(2,4))$
$= max(3, 3+3) = 6$

$opt(3,8) = max(opt(2,8), 3 + opt(2,5))$
$= max(3, 3+3) = 6$

$opt(3,9) = max(opt(2,9), 3 + opt(2,6))$
$= max(3, 3+3) = 6$

$i = 4, \quad w_4 = 5$
$opt(4,0) \Rightarrow if\ (0 < 5)$ true
$\therefore opt(4,0) = opt(3,0) = 0$
$w = 1 < 5$, true, $opt(4,1) = opt(3,1) = 1$
$w = 2 < 5$, ", $opt(4,2) = opt(3,2) = 2$
$w = 3 < 5$, ", $opt(4,3) = opt(3,3) = 3$
$w = 4 < 5$, true, $opt(4,4) = opt(3,4) = 4$

$w = 5 < 5$, false
$w = 6$ to 9, false
$\therefore opt(4,5) = max(opt(3,5), 5 + opt(3,0))$
$= max(5, 5+0) = 5$
$opt(4,6) = max(opt(3,6), 5 + opt(3,1))$
$= max(6, 5+1)$
$= 6$
$opt(4,7) = max(opt(3,7), 5 + opt(3,2))$
$= max(6, 5+1)$
$= 6$
$opt(4,8) = max(opt(3,8), 5 + opt(3,3))$
$= max(6, 5+3) = 8$

$$opt(4,9) = \max(opt(3,9),\ 5 + opt(3,4))$$
$$= \max(6,\ 5+4) = 9$$

The recursive algorithm begins with an empty table opt[][], Starting at value (N, W), which is the last row and column in the table, the recursive algo for computing opt (N, W) considers two alternatives in the 2nd last row in the table mainly (N-1), as given by the recurrence. It chooses the cell that has the larger of these two values, if tie, then choose arbitrarly.
To evaluate the two cells in row N-1, it need to apply compute them recursively. In the worst case, it needs to evaluate 4 cells in row N-2, and 2^j cells in row N-j. and beyond down to row 0. Thus, the recursive method is still O(N.W).

(b) total item $n = 4$
total max. wt. $W = 10$ kg

wt	5	4	6	3
val	10	40	30	60

dp table

			0	1	2	3	4	5	6	7	8	9	10
10	5	0	0	0	0	0	0	0	0	0	0	0	0
40	4	1	0	0	0	0	0	10	10	10	10	10	10
30	6	2	0	0	0	0	40	40	40	40	40	50	50
		3	0	0	0	0	40	40	40	100	40	50	70
60	3	4	0	0	0	60	60	60	60	100	100	100	100

hence with the DP table, we can see that if thief takes item 2 and 4 with wt. 4 & 3 total wt. 7 having maximum worth of 100 $.

here, we have taken a DP table of size (n+1, W+1) intialized with -1.

Algo:

```
for(int i=0; i<n+1; i++) {
    for( int j=0 ; j<w+1; j++) {
        if (i==0 || j==0){
            dp[i][j] = 0;
        }
        if (wt[i-1] <= j) {
            dp[i][j] = max( val[i-1] +
    dp [i-1] [j-wt[i-1]], dp[i-1][j])
        }
        else {
            dp[i][j] = dp[i-1] [j];
        }
    }
```

Run time complexity of the algorithm: $O(n(W))$

where n = no. of elts.
W = max value of knapsack.
as it is an iterative approach and previous computed values is used to calculate current opt.

value and the loop takes place n times with each loop having W iterations.

Since, it is a bounded knapsack problem with bounded no. of items. hence is polynomial time bound problem.