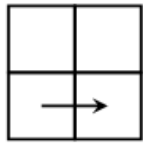


**CSC 323 Algorithm Design and Analysis, Fall 2016**  
**Instructor: Dr. Natarajan Meghanathan**  
**Project 4: Optimum Coin Collection in a Two-Dimensional Grid**

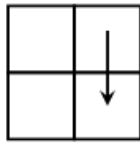
**Due: November 17, 2016: 11.30 AM**

In this project, you will extend the dynamic programming algorithm that we discussed in class for the Coin Collection problem in a two-dimensional grid and implement the same.

The conditions for the robot movement are as follows: at any time, the robot can move one cell down or one cell to the right



One cell to  
the right



One cell down

Each of you are assigned a grid of dimensions  $n$  (rows)  $\times$   $m$  (columns) as specified in the next page. You are required to randomly distribute  $P$  number of coins (where  $P < n*m$ ) across the cells of the grid (at most one coin per cell). The value for a coin assigned to a cell is randomly chosen from the range  $1 \dots V$ . The  $P$  and  $V$  values are also assigned specifically for each student.

**Your tasks are as follows:**

- (1) Implement the dynamic programming algorithm to calculate the optimum (maximum) value of the coins that a robot could collect as it traverses from cell  $(0, 0)$  to any cell in the grid such that at any time, a robot can have one of the two movements mentioned above.
- (2) Extend the dynamic programming algorithm to also keep track of the path traced by the robot to reach any target cell in the grid starting from cell  $(0, 0)$ . Clearly explain the logic of your algorithm to keep track of the path traced.
- (3) As output, your code should print the following:
  - (i) The optimum value of the coins that a robot could collect to reach any target cell in the grid starting from cell  $(0, 0)$ , as shown in the table sample output (see next page).
  - (iii) The sequence of cells that the robot should visit to collect the optimum value of the coins starting from cell  $(0, 0)$  to cell  $(n-1, m-1)$ .
  - (ii) The total number of horizontal movements (one cell to the right) and the individual horizontal movements as well as the total number of vertical movements (one cell down) and the individual vertical movements that the robot needs to make to collect the optimum value of the coins starting from cell  $(0, 0)$  to cell  $(n-1, m-1)$ .

**Submission:**

- (1) **Video:** Record a video explaining the various sections of your project code. Focus more on explaining the following: Your logic to randomly distribute the assigned number of coins to the cells in the grid, the implementation of the dynamic programming algorithm to compute the optimal value for the coins collected, the sequence of cells that the robot should visit from cell  $(0, 0)$  to cell  $(n-1, m-1)$  and the individual horizontal as well as vertical movements traced as part of this path.
- (2) **Softcopy (email to [natarajan.meghanathan@jsums.edu](mailto:natarajan.meghanathan@jsums.edu)):** The report should include your entire code, your explanation for each of the above items (that you are required to cover in the vide) and a screenshot of your output.

## Assignment of Input Values

Student Name	# rows (n)	# columns (m)	# coins (P)	Max value per coin (V)
Kirubel Benalfew	10	12	40	25
Yosef Getachew	10	12	35	35
Quavanti Hart	10	12	30	25
Deonta Kilpatrick	10	12	25	35
Kabinad Melaku	9	10	40	30
Shawndon Portis	9	10	35	20
Nicholas Whitfield	9	10	30	30
Ladarius Felix	9	10	25	20
Alishia Harmon	8	10	30	40
Algesa Haywood	8	10	25	50
Darius Leroy	12	10	40	35
Michael Moore	12	10	35	30
Karmeen Powell-Childress	12	10	30	50
Brian Williams	12	10	25	40

**A sample screenshot** of the execution of the program expected from you is shown in the next page.

```
C:\Big3-Laptop-August2015\3300-laptop\JSU-Teaching\Fall-Semesters\Fall-2016\CSC
323>java coinCollection
Enter the number of rows: 10
Enter the number of columns: 7
Enter the number of coins: 40
Enter the max. value for a coin: 30
```

### Distribution of the Coin Values

```
25    16    4    6    0    18    0
26    0    0    7    0    3    0
0     0    0    26   2    11   21
5     0    0    0    17   0    0
0     4    0    14   0    0    17
0     20   27   0    12   1    26
0     26   29   22   6    0    0
0     13   14   8    0    23   0
0     20   0    0    24   25   12
13    0    12   27   0    28   27
```

### Dynamic Programming Table

```
25    41    45    51    51    69    69
51    51    51    58    58    72    72
51    51    51    84    86    97    118
56    56    56    84    103   103   118
56    60    60    98    103   103   135
56    80    107   107   119   120   161
56    106   136   158   164   164   164
56    119   150   166   166   189   189
56    139   150   166   190   215   227
69    139   162   193   193   243   270
```

```
Path Traversed: [0 0, 1 0, 2 0, 3 0, 3 1, 4 1, 5 1, 5 2, 6 2, 6 3, 7 3, 7 4, 8 4
, 8 5, 9 5, 9 6]
```

### Number of Horizontal Movements: 6

```
[3 0 --> 3 1, 5 1 --> 5 2, 6 2 --> 6 3, 7 3 --> 7 4, 8 4 --> 8 5, 9 5 --> 9 6]
```

### Number of Vertical Movements: 9

```
[0 0 --> 1 0, 1 0 --> 2 0, 2 0 --> 3 0, 3 1 --> 4 1, 4 1 --> 5 1, 5 2 --> 6 2, 6
3 --> 7 3, 7 4 --> 8 4, 8 5 --> 9 5]
```