CS325: DP project

Prof. Borradaile

Due: in class, November 13, 2014

Your report must be typeset, printed and stapled. *Each team member's name must be listed as well as any resources used to finish the project. If your written report is submitted late, slip it under the door of KEC 3071.*

For this project, you will design a dynamic programming algorithm for the following game that is played on a $n_{row} \times n_{col}$ grid A:

> You start on any square of the grid. Then on each turn, you move either one square to the right or one square down. The game ends when you move off the edge of the board (either the bottom side or the right side). Each square of the grid has a numerical value. Your score is given by the sum of the values of the grid squares that you visit. The object is to maximize your score.

For example, in the grid below, the player can score $8 - 6 + 7 - 3 + 4 = 10$ by following the highlighted route. (This is not the best solution. The best solution has value 15 – can you find it?)

-1	7	-8	10	-5
-4	-9	**8** ⇓	-6	0
5	-2	**-6** ⇓	-6	7
-7	4	**7** ⇒	**-3** ⇓	-3
7	1	-6	**4** ⇓	-9

Your dynamic programming algorithm should be based on the following. Consider the *constrained subproblem* of finding the best solution (taking steps right or down) that ends at entry (i, j) (the i^{th} row and j^{th} column) of A; let $T[i, j]$ denote the value of this solution. Then, if i does not index the first row and j does not index the first column,

$$T[i,j] = \max \left\{ \begin{array}{l} A[i,j] \\ T[i-1,j] + A[i,j] \\ T[i,j-1] + A[i,j] \end{array} \right\} \quad (1)$$

Think about how this formula needs to change if i indexes the first row and/or j indexes the first column and how, given the table of values T, one extracts the value of the best solution. Also think about how one finds the path that corresponds to the best solution. You will need to find the value of the best solution as well as the path corresponding to the best solution.

Specifications

In the directory `http://www.eecs.orst.edu/~glencora/cs325/mw` there are several example input and output files as well as a testing procedure (in python). In the following, all indices start at 0.

For an input instance file:

- The first line specifies the number of rows n_{row} of A.
- The second line specifies the number of columns n_{col} of A.

- Each of the remaining n_{row} lines is a list of n_{col} space-separated values; each represents a row of A.

`example-input-1.txt` is an input file that corresponds to the above example.

Your code should produce an output file according to the following specification:

- The first line gives the score of the solution.
- The second line gives the number of grid squares that your solution visits.
- Each remaining line gives grid squares visited by the solution in order; each line has two numbers separated by a space, the first determines the row, and the second determines the column.

`example-output-1.txt` is an output file that corresponds to the (sub-optimal) solution to the above example and `example-output-1-opt.txt` is an output file that corresponds to the optimal solution to the above example.)

Several more example input and output files are provided to help you in testing your implementation. A testing procedure, `mwv.py` is provided that we will use to verify the validity of your solutions. An example usage is: `python mwv.py example-input-1.txt example-output-1.txt`. Note that there may be more than one solution having the same maximum score. Your implementation may find a different route through A, but it should have the same score as the optimum scores provided in these example cases. You should also create your own instances to test with as well.

Instructions for report

Your report should include the following:

Recursive function Complete Equation (1) to give a complete mathematical description of an optimal solution. That is, Equation (1) does not include base cases and does not specify how to pick out the score of the optimal solution.

Pseudocode Give pseudocode for how to compute the function $T[i, j]$ *efficiently* using *dynamic programming*. Your pseudocode should populate the entries of a dynamic programming *table*. Also show how to determine the grid squares that you visit in obtaining the maximum score.

Running time Analyze the asymptotic time complexity of your algorithm.

Instructions for implementation

One group member should upload to TEACH the following:

Implement Implement your algorithm. Your implementation must be able to read and write the above-described input and output files. Upload your source code to TEACH.

Testing Solve the three instances `test1.txt`, `test2.txt`, and `test3.txt` found in `http://www.eecs.orst.edu/~glencora/cs325/mw`. Upload to TEACH three corresponding output files called test1_grp<i>.txt, test2_grp<i>.txt, and test3_grp<i>.txt where <i> is your group number. Your output files should cause no errors for the testing procedure (so test your files using the provided testing procedure first!).