MT16121

Ankit Sharma

Problem 1 (part 3):

Strategy used in part 1 and 2:

The solution to the first part of this question has been modelled using object oriented approach wherein a class has been created with value and direction as the attributes. This approach is used as I had to encapsulate 2 attributes for each element. A list of lists has been maintained to serve the purpose of a 2 dimensional matrix. This list of lists contains objects of the class MatrixElement. A method by the name of get_max is written that returns the maximum of two values which are passed as the parameters along with the direction of the move.

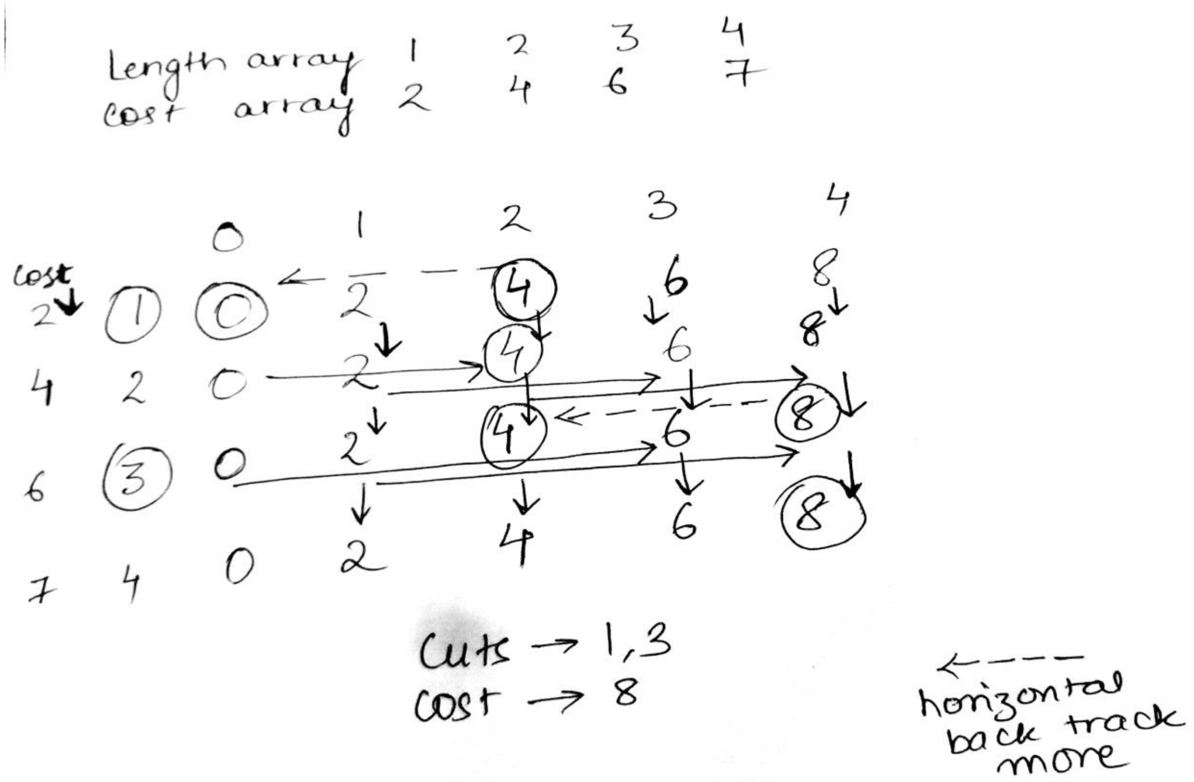
Direction = 0 implies that the move is made vertically

Direction = 1 implies that the move is horizontal

The elements of the matrix are filled with the intention to maximize the cost. The value of the last object is the final cost to be printed.

Directions are used to back track to get the cuts. For every horizontal back track move, the row number is captured as the cut. This is repeated until we back track to the zeroth column.

The entire process is shown using a diagram below:



For the second part, just the cut cost was subtracted while finding the max. Rest of the strategy remains the same.

Time complexity:

The time complexity of the code written is: O(n^2).

Why this strategy is the best?

The dynamic programming strategy used is the best because the asymptotic notation does not take exponential form in the worst case as it does when this is solved using recursion. Recursion takes care of the fact that the problem must be broken down into smaller sub problems but ignores the fact that it has to re-compute solutions to many of the same smaller problems.