Fatih Çağatay Gülmez

CS372 Assignment 5

1) bestRoute(mountain):

Input: A list of lists describes mountain paths.

Output: The most valuable path from top to down.

Algorithm: First we need an MxM matrix where M is the height of the mountain (len(input)). The matrix consists of a tuple contains the total value of the way until that node, parent x, parent y and initialized as (0, None, None). Each row consists of the level of the mountain, and columns are the nodes at that level. For each level in input, we will iterate the consisting nodes. While doing that if level zero added the matrix entry at 0, 0 will be (VALUE, 0, 0), that is the peak of the mountain. Else if it is not the first level, we will continue with process phase. In this phase, all nodes except first and last one has two probable parent. The algorithm chooses the most valuable parent as parent and saves it in decision matrix, the coordinates of the nodes are same with their own coordinates in input list. That iteration done once per each entry in lists in input. That’s why, the complexity of the algorithm is O(n), which the n is the number of nodes in the input. (Total length of the each list in input.)

2) bestSelection(conferences):

Input: A dictionary (or Map) of Conference Name and a list consists of start time, end time and participants.

Output: Selected conferences and total number of participants at most.

Algorithm: First we create 4 lists as initialization. Which are the lists of conference names, begin times, end times and participant numbers of each conferences named as conference, begin, end and count respectively. The element in same index will be a feature of same conference. Also we need to construct a decision matrix with MxM size where M is the number of conferences in input and initialize it as Zero Matrix (all 0s). The matrix will be consist of the participant numbers added after that conference added to schedule. We will construct an upper zero matrix (upper triangle of the matrix all zeros, example = [200, 0, 0], [300, 300, 0], [500, 500, 500]), the x and y axis is the conferences. Then we take column with the greatest total participant number. The iteration will be continue till the two same conferences paired (that is why the matrix will be upper Zero). The element of the decision matrix at that iteration point is not equal to zero, we will pass that iteration because we calculated that point. If the begin of the row indexed conference (begin[row]) is greater than end of the column indexed conference (end[column]) OR the end of the row indexed conference (end[row]) is smaller than the begin of the column indexed conference (end[column]) than we will fill the row, column index of the matrix with the participant number of the column, if not it will replace the most valuable one if they are fully at the same time. The complexity of the algorithms is O(Nx(N+1)/2) = O(N^2)

3) possibleCombinations(number):

Input: An integer

Output: Possible sequences in a list of tuples.

Algorithm: The result array is initialized as an empty array and a (1,) array. We start with number 2 and iterate all numbers until input. We substract all numbers one by one till the iterating number and combine with the subtracted number indexed array element (sub-problem). Than return the result. Complexity is O(2^n), because for each iteration we do 2 loops for generating problem schema.