Q4

Zijian Yue

Z5188675

The problem can be solved by solving following subproblem.

Subproblem: for every single day, find the activity that could achieve maximum accumulated enjoyment, the visitor (You in the context) can get.

* For Q4 Assume that all the arrays are started from index 1

* For Q4 Assume the enjoyments of different activities will be stored in a two-dimensional array

    For example A[1][1] ----> the enjoyment of activity 1 on day 1.

Recursion: Opt(d,a) = max{Sum(Opt(d-1,not_a_1)) + A[a][d], Sum(Opt(d-1,not_a_2)) + A[a][d] }

Base case: Opt(0,a) = 0 ,if the day number is reached zero, then just simply return 0 as enjoyment.

1. The variable d is the current day
2. The variable a is the chosen activity
3. The not_a_1, and not_a_2 means the activities that are not chosen. For example Opt(4,1) = max{Opt(4 – 1, 2), Opt(4 – 1, 3)}

We will use another recursion function to trace the sequence.

    From(i) = args max{Sum(Opt(d-1,not_a_1)) + A[a][d], Sum(Opt(d-1,not_a_2)) + A[a][d] }

    From(i) function will be used to trace of sequence of a of function Opt.

Procedure: we start to apply the Subproblem from the last day, which is day N, for day N we call the subproblems three times, and we want to find the maximum enjoyment obtained among these three max{Opt(N, 1), Opt(N,2),Opt(N,3)}. And the returned maximum value will be the maximum enjoyment that a, and From(i) function will return the sequence of activity that would achieve this result.

Time complexity: as we can see from the recursion function that every subproblem is calling two other subproblems, and we are calling three subproblems on day N(starting point), then the complexity is 3 *(n-1)^2 which is O(n^2).