Qiyao Zhou

Z5379852

Question 4

4.1. Based on the analysis of the question, we can obtain a counterexample to the three approaches A, B, C as followed:

n=2, D=2, P= [[100,150], [110,100]], T= [[10,50], [50,10]].

[A]In day1, he will earn $P [2][1] in country 2 because P [2][1] =110> P [1][1] =100.

In day2, P [1][2] =150> P [2][2] =100, so he will travel to country 1 and his wealth is $(110*50%+150) = $205.

[B] In day1, he will earn $P [2][1] in country 2 because P [2][1] =110> P [1][1] =100.

In day2, 100*T [1][2] % =55> 100*T [2][2] %=11, so he will stay in country 2 and his wealth is $(110*(100-10) %+100) = $199.

[C] In day1, he will earn $P [2][1] in country 2 because P [2][1] =110> P [1][1] =100.

In day2, (110*(100-10) %+100) =199< (110*(100-50) %+150) =205 so he will travel to country 1 and his wealth is $205.

More optimal solution: Day1 and Day2 in country 1 and his wealth is $(100*(100-10) %+150) = $240.240>205>199.

4.2.

From this problem, we can use dynamic programming in the question.

Subproblems: We can solve this problem by considering the subproblem wealth[i][j][k] for 1<=i<=n ,1<=j<=n and 1<=k<=D which means total wealth for day1 in country i and arrival in country j on day k. And we can keep an array itinerary[x][y] to store the best solution in day y for day1 in country x.

Recurrence: itinerary[i][k] = $argmax_j$ (wealth[i][q] [k-1] × (100 − T[j][q]) % + P[j][d]).

And wealth[i][itinerary[i][k]][k] = wealth[i][q][k-1] × (100 − T[itinerary[i][k]][q])% + P[itinerary[i][k]][d].

Base case: wealth[i][i][1] =P[i][1] for 1<=i<=n.

Final answer: The maximum possible wealth Antoni can attain is max(wealth[i][j][D]). And the itinerary he needs to follow can be provided by itinerary[x][y] where x = $argmax_i$(wealth[i][j][D]) and 1<=y<=D.

Time complexity: The time complexity of dynamic programming in this algorithm is O (n^2D) because of wealth[i][j][k] for 1<=i<=n ,1<=j<=n and 1<=k<=D. And we need O (n) to find max(wealth[i][j][D]).So the total time complexity is O (n^2D)+ O (n)= O (n^2D).