CS 4820, Spring 2017 Homework 3, Problem 1

Name: Yuxiang Peng

NetID: yp344

Collaborators: jl3455, zl542

1. From the question description we get information as follows:

   Set of locations: X; Time to drive from x to y: D(X, Y); fare to be charged from x to y: F(X, Y);

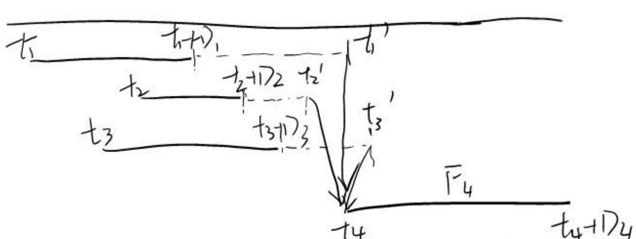
   (Xi, Yi, Ti) refers to from the location Xi to Yi, the time to start is Ti

   This question is quite similar with the weighted interval scheduling problem with minor changes.

   We create an array TD which show all the values of Ti+D(X, Y) and sort it.

   OPT(0) = 0. Let suppose we can start wherever the first OPT has the maximum fare.

   For finding the optimal solution, we assume the first request is T1, for the first request, the OPT(0+T1)= F(X1, Y1). In order to find the second optimal request, we need to traverse all the previous request and find the maximum value of T1+D(Y1, X2) and compare it with T2. It means we need to check whether before the time T2 we can drive from the first place to the second one. If we can make it, then if this is the second optimal request, we do not need to delete it. Otherwise we need to find the optimal solution without adding this possible second request. It is more easily to understand in the following picture.



Suppose we already find the optimal solution before t_4. We get an array OPT. t_1', t_2', t_3' represent if we drive from location 1, 2, 3 to 4, the time we will arrive 4

$\because t_3' > t_4$ $\therefore$ impossible $OPT(t_4+D_4) = OPT(t_3')+F_4$

We find the current maximum index t_i+D_i which satisfy $t_i+D_i+D_{i\ to\ 4} \le t_4$ and t_i+D_i is currently maximum index.

$t_1' \le t_4$, $t_2' \le t_4$, $\because t_2+D_2 > t_1+D_1$

So one possible answer is $OPT(t_2+D_2)+F_4$

At this time, $OPT(t_2+D_2) = OPT(t_2')$

why it is one possible solution?

$\because OPT(t_2+D_2) \ge OPT(t_1+D_1)$

it is the optimal situation of adding 4 into path!

The other optimal solution is without adding path 4 into optimal solution.

It is to find the currently maximum index which is smaller or equal to t_4+D_4

$\therefore$ one possible solution is $OPT(t_3') = OPT(t_3+D_3)$ ($\because t_3' \le t_4+D_4$, t_3+D_3 is the currently maximum index)

$OPT(4) = \max(OPT(t_2+D_2)+F_4,\ OPT(t_3+D_3))$

Algorithm:

OPT[0] = 0

Function ComputeOPT(j)(O = n by using memorization)

If j = 0 return OPT[0]

Else return max(Fj+ComputeOPT(Tj+Dj), ComputeOPT(j-1))

//Here for adding request j into the path, we must satisfy we find the current maximum T(j-1)+D(j-1) (since we have already sort it) which satisfy T(j-1) + D(j-1) + D(from j-1 to j) <= T(j), that is to say we can drive to the place j and before time Tj we find the maximum optimal solution. If we do not add request j into path, then we need to find the optimal solution for request 0 to j-1 which is ComputeOPT(j-1)

So the runtime is O(n) = n

Proof:

Induction: OPT[n] refer to the optimal solution of the fare for all the n requests

Basic status if OPT[0] = 0 is correct

There are only two situations for request n which are adding and not adding it into the solution. If adding to the path, then it must satisfy the condition T(n-1) + D(n-1) + D(n-1, n) <= T(n). It means the car can go to the n place. Also because we sort the Ti + Dj array. The OPT[Tj + Dj] >= OPT[Ti + Di] if Tj + Dj >= Ti + Di. If no adding to the path, then find OPT[Tn + Dn -1]. Compare these two possible solutions and find the maximum value. That is the value of OPT[Tn + Dn]. It is the same with the proof of weighted interval scheduling to construct a binary tree. If OPT[n-1] is correct, then OPT[n] is correct too. Because the base status OPT[0] is correct, the proof is correct too.