TIN093 Algorithms - Assignment 4

Mirco Ghadri

September 1, 2024

Problem 7

Dynamic programming can be used for this problem because it can be expressed in terms of a polynomial number of subproblems. We define $OPT(n)$ as the optimal total payout we can receive if we select jobs from n consecutive days. We denote the optimal solution as S^* and note instantly that it either includes the job on the n'th day or it does not not. If it is included, we know that we can not select the job on the day before it since they are neighbors. Hence the job on day $n-1$ can not be in the optimal solution if the job of day n is. Hence we can write:

$$OPT(n) = \max\{p_n + OPT(n-2), OPT(n-1)\} \tag{1}$$

Problem 8

The OPT function can be defined for n houses using a simple observation. We either end our journey in house 1 or we end our journey in house n. Let $OPT_r(1,n)$ be a time variable which indicates the time it takes to visit all houses from(and including) 1 to n in such a way that you end at house n and satisfy all deadlines. $OPT_r(1,n) = \infty$ if this is not possible, i.e you can not visit all houses in such a way that you satisfy the deadlines. Let $OPT_l(1,n)$ be a time variable that means the same thing with the only difference that you end your journey in the left house, i.e house 1 in this case.

$$OPT(1,n) = \min\{OPT_r(1,n), OPT_l(1,n)\} \tag{2}$$

In case both $OPT_r(1,n)$ and $OPT_l(1,n)$ are inf we have no solution. Otherwise select the smallest value. Obviously if neither is infinite, then both of them will in that case visit all n houses and satisfy deadline but selecting the smallest OPT value would mean that we do so in the fastest total time. So the question becomes, how do we define the recurrence? For

$OPT_l(1,n)$ it becomes:

$$OPT_l(1,n) = \min \left\{ \begin{array}{l} \begin{cases} OPT_r(2,n) + (s_n - s_1), & \text{if } OPT_r(2,n) + (s_n - s_1) \leq d_1 \\ \infty, & \text{if } OPT_r(2,n) + (s_n - s_1) > d_1 \end{cases} \\ , \begin{cases} OPT_l(2,n) + (s_2 - s_1), & \text{if } OPT_l(2,n) + (s_2 - s_1) \leq d_1 \\ \infty, & \text{if } OPT_l(2,n) + (s_2 - s_1) > d_1 \end{cases} \end{array} \right\} \tag{3}$$

For $OPT_r(1,n)$ it becomes:

$$OPT_r(1,n) = \min \left\{ \begin{array}{l} \begin{cases} OPT_r(1,n-1) + (s_n - s_{n-1}), & \text{if } OPT_r(1,n-1) + (s_n - s_{n-1}) \leq d_n \\ \infty, & \text{if } OPT_r(1,n-1) + (s_n - s_{n-1}) > d_n \end{cases} \\ , \begin{cases} OPT_l(1,n-1) + (s_n - s_1), & \text{if } OPT_l(1,n-1) + (s_n - s_1) \leq d_n \\ \infty, & \text{if } OPT_l(1,n-1) + (s_n - s_1) > d_n \end{cases} \end{array} \right\} \tag{4}$$

So what is the time complexity of this dynamic programming algorithm? Calculating 1 OPT value, i.e. $OPT_l(i,j)$ or $OPT_r(i,j)$ takes constant time(if we use dynamic programming!), and we calculate a total of n OPT values. Hence the total running time is $OPT(n)$.