CMPS 102 — Quarter Spring 2017 – Homework 3

VLADOI MARIAN

May 18, 2017

I have read and agree to the collaboration policy.Vladoi Marian
I want to choose homework heavy option.
Name of students I worked with: Victor Shahbazian and Mitchell Etzel

Solution to Problem 2: Dynamic Programming

a) i. This startegy does not work. If we put a power plant at location i = 1 then we will not consider the power plants at locations 2,3,4,5. The energy of this power plants r_2, r_3, r_4, r_5 might be greater that r_1. An example can be :

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
r_i	10	20	9	7	5	11	3	2	9	1	2	30

In this case the algorithm will choose $r_1 + r_6 + r_{11} = 23W$.
The optimal solution should be $r_2 + r_7 + r_{12} = 53W$.

ii. This strategy also does not work. Let's assume that we choose the most profitable first. Call this position (x_k, r_k). In this case we will not consider all the positions in the interval $[x_{k-4}, x_{k+4}]$. An example can be:

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
r_i	10	1	29	7	5	14	30	29	9	1	2	3

In this case the algorithm will choose $r_7 + r_2 + r_{12} = 34W$.
The optimal solution should be $r_3 + r_8 = 58W$.

b) i. We want to iterate over the indices $(x_i|0 > i > n)$ and not over the values $(r_i|0 > i > n)$. The algorithm will be polinominal in n (n = numbers of pairs). At each step (each location (x_1) we have 2 choices. We ether build a power plant or we do not. If we decide to build a power plant at location x_i then we can not bild another power plant in the interval $[x_{i-4}, x_{i+4}]$. Also , if we decide to build a power plant at position x_i, then we want the maximum energy that we can get up to x_{i-5}. If we decide not to build the power plant at location x_i, then we want the maximum energy that we can get up to x_{i-1}.

ii. We will define a function $\pi(i)$. This function will return $x_k = \pi(i) = x_{i-5}$. It will return the index k , if this index exists or 0 otherwise. In this case , our recursive optimal solution is :
OPT(i) = max $\{OPT(i-1),\ OPT(\pi(i))\ +\ r_i\}$

iii. The pseudocode for an algorithm that calculates the profit of the optimal solution:
1. We know that all our pairs (x_i, r_i) are sorted by x_i in increasing order of i.
2. We create a table were we calculate the $\pi(i)$ function defined in step ii.
3. We create another table OPT than has size n+1. In this table we will store the optimal solution at each step of iteration. OPT [i] has the maximum energy generated for i powerplants.
4. For all pairs in the input set (i = 0 , i $\leq$ n , i++) . We do not use position i_0.
5 . OPT(i) = max $\{OPT(i-1),\ OPT(\pi(i))\ +\ r_i\}$

The value OPT[n] is the maximum energy that we can achieve for the subset of power plants in in the range $[x_1, x_n]$. At each step the algorithm look at the two options described by the recursive solution in part i., and it will choose the maximum. Assume that at step i, the algorithm choose a solution that is not optimal. Then it would have choosen to build or not to build a power plant at x_i. We know that from this 2 choices one is optimal and the other is not. But we checked which choice result in a higher total energy over all previous solutions. Then it is garanteed that the solution choosen by the algorithm is optimal at each step x_i for $0 > i > n$.
The running time of the algorithm is:
1. O(n) for calculating the function $\pi(i)$
2. O(n) for calculating OPT[i] or $0 > i > n$.
So the total running time of this algorithm is O(n).
Space O(n).

iv. The recovery solution can be calculated from the OPT table as following:
1. We start traversing the table from the end to the start.
2. When we find that the values changed, ($OPT_{i-1} \neq OPT_i$)
it means that the power plant was builded at the location i.
3. Print the power plant location .
4. Jump 5 positions. OPT[i] = OPT[i+5] because we know that we can not build two power plants in this region.
6. We repet step 2,3,4, until our jump exceeds the size of the table.

The algorithm is correct because we proved that, when we construct the OPT table, each position i is optimal. Each position has the maximum energy. If we traverse from the end to start , and take in consideration our constraint(every pair of power plants be at least 5 miles apart), it is optimal to make a jump of 5 positions each time the value ($OPT_{i-1} \neq OPT_i$). We know that we build a power plant at that position, and we have to jump 5 positions back.
The running time of this algorithm is O(n) because we scan the Table OPT once.
The algorithm is O(n) space.