Stephen Woodbury 5/22/17 CMPS 102
I have read and agree to the collaboration policy. Stephen Woodbury.
Collaborators: none

Assignment 3_2 : Dynamic Programming : River Power Stations

a) Disproving Foreman

I) *Next Available Location:* Consider Pairs ((1,1),(2,10),(3,10), (4,10), (5,10),(6,1)). NAL would yield electricity generation of 2 MW, when the optimal would be 10 MW.

II) *Most profitable first*: Consider pairs ((1,1),(2,10),(3,10),(4,10), (5,13),(6,10),(7,10),(8,1),(9,1)). MPF would yield 13 MW while the most optimal would yield 20 MW.

b) Algorithm Creation

Algorithm:

```
1 Compute p(1), p(2),...,p(n)
2 for(j=1 to n)
3   M[j] = empty
4 M[j] = 0
5 List optimalStationLocations = nothing

6 Run M-Compute-Opt(n)
7 Run Find-Solution(n)

8 return optimalStationLocations

9 M-Compute-Opt(j) {
10  if(M[j] is empty)
11    M[j] = max(rj + M-Computer-Opt(p(j)), M-Compute-Opt(j-1))
12  return M[j]
13}

14Find-Solution(j) {
15  if(j = 0)
16    //do nothing
17  else if(rj + M[p(j)] > M[j-1])
18    add xj to optimalStationLocations
19    Find-Solution(p(j))
20  else
21    Find-Solution(j-1)
22}
```

Notes:
p(i) represents the closest power station that is 5 miles or more away from our station at xi (with a position that is less than xi)
M is an array that for every index gives the value of the max electricity able to be generated from index 1->our index with our power station restrictions.

Algorithm Description:
Our Algorithm takes in our n pairs and immediately computes p() for every possible power station location. Once that's calculated, init M[] for every index to be zero, then run M-Computer-Opt. What this does is determine for every index 1->n the max electricity one is able to produce with a solution involving indexes 1->said index. The functions returns the max of all possible combinations. The function itself works by starting at the

furthest point from the spring, splits the problem into two subproblems, one problem to find the max electricity generation of the index's electricity generation capacity + that of p(of said index) and the other problem is to find the max electricity generation of the previous index's power station. They do this by calling the function on itself. Once both problems are solved, the max is returned. Once this is done, Find-Solution is called on the furthest point from the spring. This uses the values calculated for M in M-Compute-Opt and compares the results of our two sub-problems listed above to either 1) add the current index to the optimal stationLocation or 2) to call the function on itself on the index right before.

Proof of Correctness : Optimality

Claim: Our Algorithm yields the most optimal solution

Pf: M-Compute-Opt computes every possible scenario of power station placement and returns the maximum possible electricity generation whereas the Find-Solution function just prints out said solution. So so far as Function logic, our algorithm does in fact yield the most optimal solution. However, why does M-Compute-Opt give the greatest amount of electricity that can be generated using index's 1-> the index specified by the function. When we look at M-Compute-Opt(j), it separates our problem (finding the greatest elect. Production one can get using index's 1->j) into two subproblems. 1)Finding max elect. Production using the station at xj and the next closest station outside 5 miles to the left of the xj station and 2)finding the max elec. Production of the station at x(j-1). This recurses all the way down until we go out of bounds with x(j-1). This only happens when j=1. The solution to the second sub problem is 0 and for the first problem it's rj. Then M[1] is returned. Then the solution builds itself up.

Runtime Analysis:

1-2)O(n) 3-5)O(1)

Recurrences:

M-Compute-Opt: $T(n) = 3 + T(n-5) + T(n-1)$

$T(n) = O(n)$

Find-Solution: $T(n) = 3 + T(n-5) + T(n-1)$

$T(n) = O(n)$

total : $O(n) + O(1) + 2*(O(n)) =$ ***O(n)***

Space Analysis:

O(n) For M array.