Stephen Woodbury 4/30/17 CMPS 102
I have read and agree to the collaboration policy. Stephen Woodbury.
Collaborators: none

Assignment 2_4 : Greedy Algorithm II:

Assumptions: 1) River runs from left to right.
2) We're given an array of size n+1 where index 0 is the left most town on the river and index n is the right most town on the river. n miles stand between city at index 0 and the city at index n. There is one mile between each index. If there is a town at an index, then the element at that index is a 1. If there is no town at said index, then there is a 0.
3) We can place a power plant in the same vicinity as a town.

Components: T:Array of size n+1 locations of towns, filled with 1's and 0's
P:Array of size n+1, for locations of power plants, a power plant at index 4 would be 4 miles from the left most city along the river. Filled with 1's and 0's. 1 being a location for a power plant and 0 being not a place for a power plant.
covered: boolean, true if all towns have power, false otherwise
LB: Left most bound of where we need power.
RB: Right most bound of where we need power.

Algorithm:

```
1) Initialize P[0->n] = 0, LB=0, RB=n, covered = false. T already init.
2) while(!covered)
3)   P[LB+20] = 1;
4)   LB = (LB+40) + 1;
5)   if(LB-1 >= RB)
6)     covered=true;
7)   else
8)     while(T[LB] == 0 && !covered)
9)       if( (RB-LB) <= 40 )
10)        P[LB+20] = 1;
11)        covered = true;
12)      else
13)        LB++;
14)return P;
```

Proof of Correctness : Termination

Claim: Our while loop terminated after at most n calls.
Pf: Every time the while loop is called, we move at least one square forward on our town array. There are only n jumps one can make between indexes in our town array. Once the end of the town array is reached, all of our towns would have been provided with power and the while loop wouldn't be called again.
Note: We would never make n calls of while as we are constantly searching 40 spaces ahead, never is the LB closer than 40 spaces to RB while still needing to call while again.

Proof of Correctness : Optimality

Claim: Our Greedy algorithm G is the most optimal
Pf (by Contradiction): Assume C is the most optimal algorithm. The solutions given by the Greedy Algorithm are: PG1, PG2,..., PGr,..., Pgk. The solutions given by the Optimal Algorithm are: PC1, PC2,..., PCr,..., PCk'. PG# is Greedy Power plant location choice # while PC# is Optimal Power plant location choice #. The Greedy Algorithm and the Optimal

Algorithm share at most the first r number of solution choices. PCr+1 is the first solution to differ from the Greedy algorithm, which chose PGr+1. We know that PCr+1 must be closer or equal to the distance that PGr+1 is from the left town due to the nature of the Greedy Algorithm. Since we know PGr+1 is a valid choice, we can sub PCr+1 with PGr+1 in the optimal solution with no extra cost incurred. Yet, this is a contradiction as now the optimal solution would share r+1 solution choices with the Greedy Solution, therefore, our Greedy Algorithm is our most optimal algorithm as it stays ahead. QED

RunTime Complexity:

1) O(1)
2) O(n)
3) O(1)
4) O(1)
5) O(1)
6) O(1)
7) no cost
8) O(n)
9) O(1)
10) O(1)
11) O(1)
12) no cost
13) O(1)
14) O(1)

total : O(1) + O(n)(8*O(1) + O(n)) + O(1) =* ***O(n^2).***

Space Complexity:

T : O(n)
P : O(n)
LB, RB, covered : O(1)

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