Philippe WS CSC 349

Lab 4

Doc: 4-1, Problem 1

Pseudo Code

//M[i] is the array of i ordered distances of each house from the beginning of the road.

```
greedyPlacement( M[ i ] ){
    int[] towerPlacements;
    towerPlacements.add( M[0] + 4 );

    for (n = 0 while n < i, n++){
        if(towerPlacements.getLast() + 4 < M[ n ]){
            towerPlacments.add( M[ n ] + 4 );
    return towerPlacements;
}
```

Proof By Induction

Proof of correctness: Nearest uncovered house algorithm.

Assumptions: Assume $\Sigma^* = \{ LOC_1^*, \dots , LOC_n^* \}$ is an optimal solution to the most efficient cell tower placement problem with n efficiently placed towers specified by their distance from the start of the road. Let $\Sigma = \{ LOC_1, \dots , LOC_m \}$ be the set of towers determined by the nearest uncovered house algorithm for the same problem. Both sets are sorted by distance.

Proposition: **Let P(k) be the proposition $LOC[k]^* \leq LOC[k]$ and all the houses up to the point are covered by $LOC[k]^*$ and $LOC[k]$**: Want to show this is true for all $1 \leq k \leq n$. This will also be used to prove Σ is optimal, having the same number of towers.

Proof By Induction: Let P(k) be the proposition $LOC[k]^* \leq LOC[k]$ and that all houses up to tower placement k are covered. Want to show this is true $1 \leq k \leq m$.

Base Case: Show P(1) is true, i.e. $LOC[1]^* \leq LOC[1]$ and that the first house is covered.

Proof of Base Case: $LOC[1]^* \leq LOC[1]$ is true, because the first house must be covered and since the algorithm will choose the first house plus four miles, the optimal solution cannot go beyond the algorithm's placement, lest it not cover the first house and not be an optimal solution.

Inductive Case: Show P(k) is true $\rightarrow$ P(k+1) is true for any $1 \leq k \leq m$. Thus, we need to show that $LOC[k]^* \leq LOC[k]^* \rightarrow LOC[k+1]^* \leq LOC[k+1]$:

Proof of Inductive case: LOC[k+1] is the tower placement four miles away from the nearest house not covered by LOC[k].

Claim: Tower placement LOC[k+1]* must be before or up to LOC[k+1].

a. Let M[x] be the next uncovered house.
b. LOC[k] ≤ M[x], Since LOC[k] does not cover M[x] (By definition of M[x])
c. Thus LOC[k]* does not cover M[x] (By the inductive hypothesis)
d. LOC[k] * ≤ LOC[k] < M[x] – 4, must also be true, lest it contradicts the definition of M[x].

a. The greedy algorithms next placement must be LOC[k+1] = M[x] + 4 (By definition of the algorithm)
b. The optimal solutions next placement must be contained by M[x] – 4 ≤ LOC[k+1]* ≤ M[x] + 4. Otherwise, the tower placement will not cover house M[x]. This contradicts that Sigma* is the optimal solution.
c. Thus, since LOC[k+1] = M[x] + 4, LOC[k+1]* ≤ LOC[k+1]

Thus by Principle of Mathematical Induction: P(k) is true for k = 1 … m.

Conclusion: Show m = n. Since the proposition above was proven true, the greedy algorithm is shown to place a tower ahead of or equal to the placement of every tower in a supposed optimal solution. Suppose this was not true and the optimal solution was able to place one less tower than the greedy solution. This contradicts the fact the this is the optimal solution since the greedy ensures coverage of every house, and thus the optimal solution being one tower behind the greedy would then not cover every house. Thus m = n must be true.

Doc: 4-1, Problem 2

Pseudo Code

//m[i] is the array of i ordered distances of each monitoring device from the beginning of the road.

```
greedyPlacement( m[ i ] ){
    int[] roadPlacements;
    roadPlacements.add( m[0] + 2 );

    for (n = 0 while n < i, n++){
        if(roadPlacements.getLast() + 2 < m[ n ]){
            roadPlacments.add( m[ n ] + 2 );
    return roadPlacements;
```

Proof By Induction

<u>Proof of correctness</u>: Nearest uncovered monitoring device algorithm.

<u>Assumptions</u>: Assume Σ* = { R_1^*, … , R_n^* } is an optimal solution to the most efficient access road placement problem with n efficiently placed roads specified by their distance from the start of the pipe. Let Σ = { R_1, … , R_m } be the set of roads determined by the nearest uncovered device algorithm for the same problem. Both sets are sorted by placement ascending.

<u>Proposition</u>: **Let P(k) be the proposition $R[k]^* \leq R[k]$ and all the monitoring devices up to the point are covered by $R[k]^*$ and R[k]**: Want to show this is true for all $1 \leq k \leq n$. This will also be used to prove Σ is optimal, having the same number of access roads.

<u>Proof By Induction</u>: Let P(k) be the proposition $R[k]^* \leq R[k]$ and that all monitoring devices up to access road k are covered. Want to show this is true $1 \leq k \leq m$.

<u>Base Case</u>: Show P(1) is true, i.e. $R[1]^* \leq R[1]$ and that the first monitoring is covered.

<u>Proof of Base Case</u>: $R[1]^* \leq R[1]$ is true, because the first monitoring device must be covered and since the algorithm will choose the first monitoring device plus two miles, the optimal solution cannot go beyond the algorithm's placement, lest it not cover the first monitoring device and not be an optimal solution.

<u>Inductive Case</u>: Show P(k) is true $\rightarrow$ P(k+1) is true for any $1 \leq k \leq m$. Thus, we need to show that $R[k]^* \leq R[k] \rightarrow R[k+1]^* \leq R[k+1]$:

<u>Proof of Inductive case</u>: R[k+1] is the access road placement two miles away from the nearest monitoring device not covered by R[k].

Claim: Access road placement $R[k+1]^*$ must be before or up to R[k+1].

e. Let m[x] be the next uncovered monitoring device.
f. $R[k] \leq m[x]$, Since R[k] does not cover m[x] (By definition of m[x])
g. Thus $R[k]^*$ does not cover m[x] (By the inductive hypothesis)
h. $R[k]^* \leq R[k] < m[x] - 2$, must also be true, lest it contradicts the definition of m[x].
d. The greedy algorithms next placement must be R[k+1] = m[x] + 2 (By definition of the algorithm)
e. The optimal solutions next placement must be contained by $m[x] - 2 \leq R[k+1]^* \leq R[x] + 2$. Otherwise, the tower placement will not cover monitoring device m[x]. This contradicts that Σ* is the optimal solution.
f. Thus, since R[k+1] = m[x] + 2, $R[k+1]^* \leq R[k+1]$

<u>Thus by Principle of Mathematical Induction</u>: P(k) is true for k = 1 … m.

<u>Conclusion</u>: Show m = n. Since the proposition above was proven true, the greedy algorithm is shown to place an access road ahead of or equal to the placement of every access road in a

supposed optimal solution. Suppose this was not true and the optimal solution was able to place one less access road than the greedy solution. This contradicts the fact the this is the optimal solution since the greedy ensures coverage of every monitoring device, and thus the optimal solution being one access road behind the greedy would then not cover every monitoring device. Thus m = n must be true.

<u>Doc: 4-2, Problem 6</u>

a. Assumptions
   - Times all begin at 0.
   - Assume that the jobs by the algorithm have been sorted in ascending order of deadline in the set Σ, in accordance to the greedy algorithm.
   - Assume that there is an ordering Σ* that is the optimal solution but is different from the ordering Σ, the ordering given by the algorithm.
b. Notation
   - Σ: The ordering given by the greedy algorithm. Ordering by ascending deadline.
   - Σ*: The ordering given by a supposed optimal solution. Different ordering than that of Σ.
   - Jobs i and j, the two jobs to be considered
   - C the completion time of all previous jobs before the two considered jobs.
c. If Σ is not an optimal solution, then Σ* must contain at least two jobs (i & j), which are not ordered by ascending deadlines, and are out of order compared to Σ. Let these be the two jobs we consider.
d. Swap

   Since job i's deadline is greater or equal to that of job j's, $d_i – d_j \geq 0$ must be true.

   If we compare the lateness between jobs i and j, before and after the swap.

Lateness (i & j)	Job i	Job j
Before	$(C + L_i) - d_i$	$(C + L_i + L_j) - d_j$
After	$(C + L_j + L_i) - d_i$	$(C + L_j) - d_j$

   Compare Before and After: We should only compare the largest time to completion value in both cases, because any of the lesser duration values will not be the maximum lateness value.

   The jobs with the maximum lateness before and after the swap have duration $(C + L_i + L_j)$. If we compare local maximum lateness before and after the swap by subtracting them: $((C + L_i + L_j) - d_j) – ((C + L_j + L_i) - d_i)$, we end up with

   $d_i – d_j$, which we know previously to be greater than or equal to 0. This means that for this swap, we have either reduced the local maximum lateness or not changed it.

e. Thus, by continuously swapping every pair that is relatively mis ordered in Σ*, we will continue to reduce or not change the local maximum between those two swapped jobs.

   After enough swaps, we will have reduced or not changed the local maxima between every mis ordered pair, and therefore either reduced the overall maximum lateness of the set, or not changed it all, since every swap cannot make the maximum worse.

   The result is that Σ* will become Σ. This means that the greedy solutions maximum lateness is either the same or better than the supposed optimal solution.

f. Hence, the optimal solution is the greedy solution.

Doc: 4-2, Problem 7

Algorithm Description

The greedy algorithm will sort the jobs by ascending duration.

Proof of Correctness: Exchange Argument

Assumptions:

- Assume that the duration of each job is distinct.
- Assume that the jobs by the algorithm have been sorted in ascending order of duration in the set Σ.
- Assume that there is an ordering Σ* that is the optimal solution but is different from the ordering Σ, the ordering given by the algorithm.

Proof:

Since the jobs are ordered by ascending duration, the optimal solution Σ* must contain at least two consecutive jobs, (i & j), where i's duration is less than j's duration, but in Σ*, i comes after j. This must be true, since Σ is sorted by ascending duration, and Σ* deviates from that ordering in some way by the assumed definition of Σ*.

These are the two jobs we will inspect.

Since the duration of $d_i < d_j$, $d_j - d_i > 0$ must be true.

Now consider if the two jobs (i & j) were to be switched, what would happen the Total Time to Completion of Σ*.

Assume that the total time to completion up to job i is some constant C.

Thus the total time to completion before and after the swap are as follows,

	Job j	Job i
Before	C + d_j	C + d_j + d_i
After	C + d_i + d_j	C + d_i

To show their differences, subtract the two sequences, Before – After.

Before – After: $((C + d_j) + (C + d_j + d_i)) - ((C + d_i + d_j) + (C + d_i)) = (d_j - d_i)$

From the previous inequality, we know $d_j - d_i > 0$, and this Before > After.

Thus, After has a smaller time to completion, meaning After is a more optimal ordering.

This contradicts that Σ* is the optimal solution, and thus Σ must be the optimal solution.

Time Complexity

The time complexity is $\theta(n\log(n))$ since that is the time complexity of efficient sorting algorithms.