CSC373— Algorithm Design and Analysis, Fall 2010

Cell Phone Tower Placement Problem

Example for Greedy Algorithm Design and Correctness Proof

Placing Cell Phone Towers. Suppose there is a long straight country road, with n houses sparsely scattered along the road. Positions along the road are specified by distance in kilometers from one end, say in terms of distance $d \in [0, D]$. The position of the k^{th} house along the road is given by $d = h_k$. Here you can assume the h_k's are integers.

Assume a cell tower has a range of R km with $R > 0$ a fixed constant. That is, if the j^{th} tower is placed at kilometer t_j along the road, then any house k with $|h_k - t_j| \leq R$ would have service from that tower.

Tower Placement Problem. Given the house locations, $\{h_k\}_{k=1}^{n}$, and the range $R > 0$, the problem is to place the **minimum** number of cell phone towers along the road, say at distances $\{t_j\}_{j=1}^{J}$, so that each of the n houses can get service from at least one of the towers. That is, the tower placements must satisfy

$$\max_k \left[\min_j |h_k - t_j| \right] \leq R,$$

and we wish to minimize J.

Suppose the indices k are provided in the order the houses first requested service from the cell phone company.

1. **Online Greedy**. Consider the greedy algorithm which treats the customers in the order they arrive (i.e., in increasing k). A new cell phone tower is built at $t_{j+1} = h_k$ if and only h_k is beyond the range of all of the previously built towers $\{t_r\}_{r=1}^{j}$. Prove that this algorithm is not optimal.

2. **Correct Greedy**. Describe a greedy algorithm that generates an optimal solution for the general problem stated above. Note the towers can be built at any distance d along the road, and not just beside houses (as was done in part (a)).

   Illustrate how your algorithm works on a simple example (say, requiring three cell phone towers).

3. **Runtime order.** Give a big-Oh estimate of the runtime of your algorithm. Briefly explain how you arrived at this estimate.

4. **Proof.** Prove your algorithm is correct.

Sample Solution: Placing Cell Phone Towers.

1. **Online Greedy**. Prove that the online algorithm is not optimal. Let R = 2, and the house locations be $(h_1, h_2) = (0, 3)$. Then the online greedy algorithm sets $t_1 = h_1 = 0$. But then $h_2 = 3$ is out of range R, so a second tower gets allocated with $t_2 = h_2$.

   This is not optimal, since a single tower built at any $t \in [1, 2]$ (e.g., $t = 1.5$) would have both houses within the range R. This completes the proof.

2. **Correct Greedy**.

   Given R and $\{h_k\}_{k=1}^{n}$.

Sort the house locations in increasing order, removing any duplicates.
Reset the indices k (and possibly n) to reflect this sorted order.
So $h_1 < h_2 < \ldots < h_n$.

```
if n == 0
  return { }

t_1 = h_1 + R      % First tower is as far down the road as possible.
p = 1              % Index of latest tower.
for k = 2 to n
    if abs(h_k - t_p) > R    % House h_k doesn't have service.
       p = p + 1
       t_p = h_k + R         % Build next tower as far down the road as possible.
    end
end
return {t_1, ..., t_p}
```

Example: $h_k = k$ for $k = 1, \ldots, 7$ and $R = 1$.

Output: $(t_1, t_2, t_3) = (2, 5, 8)$.

Optional: Proof of minimality for this example. This is not explicitly asked for in the assignment handout, but I will include it anyway. The above solution is a minimal solution, since we claim there can be no solution that requires two towers. We prove this claim by contradiction. Notice if there was a solution with two towers, one of these towers would have to cover at least four out of the seven houses. And any four houses are at least a distance 3 apart. But the range of a tower is only 2 (i.e. $\pm R$), so one tower cannot cover 4 houses.

3. **Runtime order.** $\Theta(n \log n)$ for the sorting. The loop itself runs in time $\Theta(n)$. (Optional: If the distances are small integers, and you know the range D, you could use radix sort to do the sort in $\Theta(n)$ time, thereby reducing the overall time to $\Theta(n)$.)

4. **Proof.** First, let us deal with the trivial case $n = 0$, i.e., no houses. In this case both the greedy algorithm and the optimal solution allocate no towers, and so the greedy algorithm is optimal in this case.

   Secondly, if there are duplicate house locations in the input, i.e, $h_j = h_k$ for $j \neq k$, then we can simply remove all but one of every duplicated house location. This is true since whether or not any set of towers is a solution only depends on the set of house locations, not on the number of houses at any location. We omit a formal proof of this.

   For the remainder of the proof we assume $n \geq 1$ and $h_1 < h_2 < \ldots < h_n$, that is, there is at least one house, there are no duplicate house locations, and the locations are sorted.

   Let O be any optimal solution. Suppose O has q towers at distances $d_1 < d_2 < \ldots < d_q$. (Note, two towers cannot be at the same location in an optimal solution, otherwise we could remove one of the duplicate towers and not lose any coverage. This would contradict O being optimal. Hence the implicit assumption above that all tower positions are distinct.)

   Let $\{t_1, t_2, \ldots, t_p\}$ be the output from the algorithm. We will show the algorithm stays ahead (or, at least, stays neck and neck) with the optimal solution O in the following sense.

   Defn. Property $A(s)$: $q \geq s$ and $d_s \leq t_s$.

   That is, property $A(s)$ ensures that the optimal solution has at least s towers and the tower location d_s in

the optimal solution is not further down the road than the corresponding tower, t_s, allocated by the greedy algorithm.

Claim: $A(j)$ is true for all $j \in \{1, 2, \ldots, p\}$.

Before proving the claim, we will show that it implies that the algorithm returns an optimal solution.

From the claim it follows that $A(p)$ is true, so $q \geq p$. But it is easy to verify from the algorithm that it returns a set of towers $\{t_1, \ldots, t_p\}$ such that every house has service. (Every house is given service, if it doesn't already have it and, once given, service is never removed.) Thus the set of towers returned is a solution to the problem of providing service to every house. Therefore any optimal solution must have no more towers than this solution. So $q \leq p$. Thus we have shown $p = q$, that is, the solution the algorithm returns is optimal.

Proof of Claim:

Case A(1): Note that the algorithm allocates the first tower at $t_1 = h_1 + R$. We will prove $A(1)$ is true by contradiction.

Suppose $A(1)$ is false. Then either $q = 0$ or $d_1 > t_1$. Since we are assuming $n \geq 1$ and O is a solution, then house h_1 must get coverage from some tower. Therefore the number of towers in the optimal solution, q, must satisfy $q \geq 1$. As a consequence the second clause of $A(1)$ must fail, that is, $d_1 > t_1$. Given the sorted order of the towers in O it must also be the case that $d_k \geq d_1 > t_1$ for all $k = 1, \ldots, p$. So $\min_k(d_k - h_1) > t_1 - h_1 = R$. That means h_1 does not have any service given the tower placement in O. This contradicts O being a solution, thereby proving $A(1)$ must be true.

Remaining Cases, $1 < s \leq p$: We now use contradiction on the whole claim. Suppose the claim is not true. Then let s be the smallest index for which $A(s)$ is false. From the previous case we know $s > 1$. Therefore it must be the case that $A(s-1)$ is true but $A(s)$ is false.

Since $A(s-1)$ is true we have $q \geq s - 1$ and $d_{s-1} \leq t_{s-1}$. Let $k(s-1)$ denote the loop index k at the point in the algorithm that the tower t_{s-1} was allocated. We know that t_{s-1} was allocated to cover some previously uncovered house at location $h_{k(s-1)}$. Then, from the algorithm, we have $t_{s-1} = h_{k(s-1)} + R$.

Since $s - 1 < p$ the algorithm continues looping through k after allocating t_{s-1} and eventually allocates a new tower at t_s. Let $k(s)$ denote the k at which the tower at t_s is allocated. Then, from the algorithm, we have $t_s = h_{k(s)} + R$ and also $k(s) > k(s-1)$. Moreover, the trigger for allocating the tower on loop $k = k(s)$ must have applied, that is, $|h_{k(s)} - t_{s-1}| > R$.

We next show how to combine the above facts to argue that

$$h_{k(s)} > t_{s-1} + R \tag{1}$$

That is, $h_{k(s)}$ must be **further down** the road than the tower at t_{s-1} can reach. To see this, it is convenient to first summarize the information provided in the previous two paragraphs as

$$\begin{aligned} &|h_{k(s)} - t_{s-1}| > R, \\ &h_{k(s-1)} = t_{s-1} - R, \\ &h_{k(s)} > h_{k(s-1)}, \end{aligned}$$

where the last inequality follows from $k(s) > k(s-1)$ and the sorted order of the house locations. The only way for these three conditions on $h_k(s)$ to be satisified is for (1) to be true.

Moreover, from $A(s-1)$, we also know the greedy solution is ahead at step $s-1$, so $d_{s-1} \leq t_{s-1}$. Therefore, from (1) we have the following inequality for the case $j = s - 1$,

$$h_{k(s)} > d_j + R, \quad \text{for } j = 1, \ldots, s-1. \tag{2}$$

The remaining cases in (2) follow from the sorted order of the d_j's. Notice (2) states that the house at $h_{k(s)}$ does not get service from any of the towers up to the $(s-1)^{st}$.

Since O is a solution, this house at $h_{k(s)}$ must be within range of some tower, so there must be at least one more tower in O. That is, $q > s-1$. Now using the assumption that $A(s)$ is false, along with $q > s-1$, we find that $d_s > t_s$, i.e., the greedy solution has just fallen behind at tower t_s. But, using the placement of the tower t_s, we have $t_s = h_{k(s)} + R$. Finally, using the sorted order for the d_k's, and $d_s > t_s$, we have

$$d_j > t_s = h_{k(s)} + R, \text{ for } j = s, \ldots, q. \tag{3}$$

Therefore we see by (2) and (3) that the house at $h_{k(s)}$ must be outside the service of all the towers in O. This contradicts O being a solution, and thereby proves the claim.