Problem 1. *There are n villages on a straight road at $x-$coordinates $x_1 < x_2 < \ldots < x_n$. You have to place k cellphone towers, each in one of the villages. Any village will connect to its nearest tower and incur a delay equal to the distance between them. The goal is to place the k towers to minimize the maximum delay of any village. Give an efficient algorithm for this.*

Let V denote any vector $x_1, x_2, \ldots, x_n$ of x-coordinates of villages. Then suppose we can efficiently solve the following decision version of our problem

Problem 2. *Suppose we are given village locations at V and k towers. Additionally, we are given a number d. Then, is the optimum value for problem 1 at most d ?*

Then, we can simply binary search on d to solve our original optimization problem. More precisely, given an algorithm $\texttt{DECIDE}(V, k, d)$ for problem 2, the algorithm $\texttt{MINDELAY}(X, k)$ solves problem 1.

Algorithm 1 Optimization via decision version

procedure $\texttt{MINDELAY}(V, k)$ $\quad\triangleright$ V is a vector $x_1, x_2, \ldots, x_n$ of coordinates
    $D \leftarrow x_n - x_1$. $\quad\triangleright$ Maximum possible delay.
    Initialize $a \leftarrow 0, b \leftarrow D$.
    while $b > a$ **do** $\quad\triangleright$ Invariant : the optimum delay lies in the interval $[a, b]$
        $d \leftarrow \lfloor (a+b)/2 \rfloor$
        if $\texttt{DECIDE}(V, k, d)$ **then**
            $b \leftarrow d$
        else
            $d + 1$

            return a $\quad\triangleright$ $a = b$

The correctness of algorithm 1 follows from the invariant that at the start of each iteration of the while loop, the optimum delay lies in the interval $[a, b]$.

Time Complexity of $\texttt{MINDELAY}$ Algorithm 1 makes $O(\log_2(x_n - x_1))$ calls to $\texttt{DECIDE}$ because $b - a$ halves after each call to $\texttt{DECIDE}$ and initially $b - a = D = x_n - x_1$ (and $a = b$ at the end).

Now to solve problem 2, it is enough to solve its following optimization version.

Problem 3. *Given village locations at V and a number d, find the minimum number of towers needed so that maximum delay of any village is at most d.*

Given an algorithm $\texttt{MINTOWER}(V, d)$ to solve problem 3, $\texttt{DECIDE}(V, k, d)$ simply returns whether $\texttt{MINTOWER}(V, d)$ is at most k, see algorithm 2.

Algorithm 2 Decision version

procedure $\texttt{DECIDE}(V, k, d)$ $\quad\triangleright$ V is a vector $x_1, x_2, \ldots, x_n$ of coordinates
    return $(\texttt{MINTOWER}(V, d, 1) \leq k)$
end procedure

Hence, we now focus on solving problem 3.

Minimizing the number of towers needed to achieve a given maximum delay We give a natural greedy algorithm for problem 3 : place a tower at the rightmost village i such that $x_i - x_1 \leq d$. Delete the villages whose delay by this tower is at most d, and recurse on the remaining villages. See algorithm 3 to recursively solve the problem for villages $j, \ldots, n$. Run this algorithm for $j = 1$.

Algorithm 3 Minimizing towers to achieve a given maximum delay for villages $j, j+1, \ldots n$

---

procedure MINTOWER(V, d, j) ▷ V is a vector $x_1, x_2, \ldots, x_n$ of coordinates
    for all $i = j, j+1, \ldots$ **do**
        if $x_i - x_j > d$ **then** ▷ Place tower at $i-1$
            for all $k = i, i+1, \ldots$ **do**
                if $x_k - x_{i-1} > d$ **then**
                    return $1 +$ MINTOWER(V, d, k) ▷ k is the leftmost village not covered by our tower $i-1$
                end if
            end for
        end if
    end for
    return 1 ▷ only one tower suffices
end procedure=0

Proof of correctness:

Proof. Let us fix some feasible subset S of villages, and let G be the subset returned by the greedy algorithm. Clearly G is feasible by construction. We want to show that $|G| \leq |S|$. The following claim is the key to everything

Claim 1. *Suppose the left most tower in G (respectively S) is at village g (resp. s). Then, $s \leq g$ or equivalently, $x_s \leq x_g$.*

Proof. First let us see why $x_s \leq x_g$. Since S is feasible, $x_s - x_1 \leq d$. Since $x_i - x_1 > d$ for all $i > g$ (by the greedy choice of g), it must be that $s \leq g$. Hence, $x_s \leq x_g$. □

Claim 1 shows that after the choice of the left most tower, the greedy solution G is 'ahead' of any other solution S in the following sense :

Claim 2. *$S - s$ is feasible for the subset of villages not covered by g i.e. the villages $k, k+1, \ldots, n$ in the algorithm 3. Or equivalently, any village within distance d of s is also within distance d of g.*

Proof. It is enough to show that any village within distance d of s is within distance d of g. To see this, note first that any village to the left of x_g is within distance d of g (since this property is true for the leftmost village). Also, any village to the right of x_g is only closer to g than s (since $x_g \geq x_s$). This completes the proof. □

This means that one can essentially exchange s for g in S i.e. $S - s \cup g$ is a feasible solution of the same value as S. We can now finish the proof by exchanging off all towers of S with those of G, but I prefer to do a more slick induction. We can inductively assume that greedy is optimal for any smaller subset of villages than V, hence $G - g$ is an optimal solution for the set of villages not covered by g. Since, $S - s$ is feasible for the set of villages not covered by g, we have that $|S - s| \leq |G - g|$ and hence $|S| \leq |G|$.

$\square$

Time complexity of the greedy algorithm: In algorithm 3, we look at any village at most once in the iterative loop. Hence, the time complexity is $O(n)$.

Final time complexity There are $O(\log(x_n - x_1))$ calls to `DECIDE` which makes a single call to `MINTOWER`. `MINTOWER` takes $O(n)$ time, hence the overall running time is $O(n \log O(x_n - x_1))$.