Problem 1 (50 points). There is a long, lonely highway. There are \( n \) houses along the highway, which we call \( h_1, h_2, ..., h_n \). We can think of the highway as a straight line, and the house \( h_i \) (for \( i = 1, 2, ..., n \)) has coordinate \( x_i \). For convenience, let us assume \( x_1 < x_2 < ... < x_n \). We would like to build grocery stores along the highway, such that every house is within distance \( k \) from at least one grocery store. Design an algorithm to minimize the number of grocery stores you have to build. Prove the correctness of your algorithm, and analyze its time complexity.

Solution. Idea: We scan the houses from \( h_1 \) through to \( h_n \). Find the first house that doesn’t have a grocery store within distance \( k \) from it, say this house \( h_i \), and build a store at the coordinate \( x_i + k \). Repeat this till all the houses are dealt with.

Pseudocode:

Input: \( x_1, x_2, ..., x_n, k \)
Output: A set of stores locations \( R \)

1: let \( R = \emptyset \)
2: let \( r = x_1 + k \)
3: \( R.add(r) \)
4: for \( j = 2 \) to \( n \) do
5: if \( x_j > r + k \) then
6: let \( r = x_j + k \)
7: \( R.add(r) \)
8: end if
9: end for

Correctness: Assume \( R = (r_1, r_2, ..., r_m) \) be the output of the Greedy Algorithm, and one optimal solution is \( O = (o_1, o_2, ..., o_{m'}) \), where \( r_1 < r_2 < ... < r_m, o_1 < ... < o_{m'} \) are the coordinates. If \( m' = m \), we are done. Otherwise \( m > m' \). We claim that \( o_i \leq r_i \) for \( i = 1, 2, ..., m' \). \( o_1 \leq r_1 \) obviously since otherwise there will not be a store within distance \( k \) of the first house. Since \( r_j \) is built within distance \( k \) of the first house \( h_i \) that \( x_i > r_{j-1} + k \), \( r_i \geq o_i \) holds. However, if \( m' < m \), there must be some house that doesn’t have a store within distance of \( k \). Thus \( m' = m \).

Time complexity is \( O(n) \).

Problem 2. Given a sequence of \( n \) real numbers \( A_1, A_2, ..., A_n \), find a subsequence (not necessarily continuous) of maximum length in which the values in the subsequence form a strictly increasing sequence. (For example, if \( n = 6 \) and \( (A_1, ..., A_6) = (1, 4, 2, 6, 8, 7) \), then \( 1, 2, 6, 7 \) is such a longest increasing subsequence.) Note: You should present your algorithm, prove its correctness, and analyze its time complexity.
**Solution.** Let $M[i]$ be the length of the longest increasing sequence that ends with $A_i$. Then

$$M[i] = \max_{j<i, A_j < A_i} \{M[j] + 1\}$$

---

**Input:** $A_1, A_2, \ldots, A_n$

**Output:** A strictly increasing subsequence $R$

1. let $M[i] = 1$ for $i = 1, 2, \ldots, n$
2. for $i = 2$ to $n$ do
3.   for $j = 1$ to $i - 1$ do
4.     if $A_j < A_i$ and $M[j] + 1 > M[i]$ then
5.       let $M[i] = M[j] + 1$
6.     end if
7.   end for
8. end for
9. let $k$ be the index that $M[k]$ is the largest
10. let $R = \{A_k\}$
11. while $M[k] > 1$ do
12.   for $i = 1$ to $k - 1$ do
13.     if $A_i < A_k$ and $M[i] + 1 == M[k]$ then
14.       $R = R \cup \{A_i\}$
15.       let $k = i$
16.       break
17.     end if
18.   end for
19. end while

Time complexity is $O(n^2)$. 

---

2