1. You are given a set $A$ of $n$ different numbers.

(a) (20 points) Can you find an $o(n^2)$ (little-oh) algorithm to determine if there exist two numbers $x, y$ in $A$ such that $x + y = m$? Please justify why your algorithm is $o(n^2)$.

(b) (25 points) Can you find an $o(n^3)$ (little-oh) algorithm to determine if there are three numbers $x, y, z$ in $A$ such that $x + y + z = m$? Please justify why your algorithm is $o(n^3)$. 
2. Let \( I_1, \ldots, I_n \) be \( n \) intervals, where interval \( I_i \) is defined by set \([a_i, b_i]\), i.e., starting from \( a_i \) and ending at \( b_i \). We assume that the intervals are sorted by the ending time \( b_i \) in non-decreasing order. We also give a value \( v_i \) to interval \( I_i \). We define the following two problems:

**The interval scheduling problem** \( P_1 \): Find a maximum number of disjoint intervals that do not overlap with each other.

Example: Four intervals are given as \( I_1 = [1, 2], I_2 = [2, 3], I_3 = [1, 4], I_4 = [4, 5] \). A solution is \( \{I_1, I_2, I_4\} \).

**The weighted interval scheduling problem** \( P_2 \): Find some subset of disjoint intervals, such that the sum of the values of the disjoint intervals is maximized.

Example: Four intervals and their values are \( I_1 = [1, 2], v_1 = 0.9; I_2 = [2, 3], v_2 = 0.5; I_3 = [1, 4], v_3 = 4; I_4 = [4, 5], v_4 = 2 \). A solution is \( \{I_3, I_4\} \).

(a) (15 points) Please give a linear greedy algorithm to solve the interval scheduling problem \( P_1 \).

(b) (15 points) The following greedy algorithm is proposed to solve the weighted interval scheduling problem \( P_2 \). The intuition is to select intervals with high “density”, i.e., value-to-length ratio.

**GREEDY-WEIGHTED-INTERVAL-SCHEDULER** \((a, b, v)\)

1. Compute the value density \( \rho_i \) of each interval by \( \frac{v_i}{b_i - a_i} \)
2. \( J \leftarrow \text{Sort } I_1, \ldots, I_n \text{ by decreasing value density } \rho_i \)
3. \( DI \leftarrow \{\} \)
4. for \( i \leftarrow 1 \) to \( n \)
5. do  \( \text{Overlap} \leftarrow \text{false} \)
6.   for each interval \( L \) in \( DI \)
7.     do  if \( J_i \) overlaps with \( L \)
8.     then  \( \text{Overlap} \leftarrow \text{true} \)
9.     Break out of the for-loop
10. if  \( \text{Overlap} \) is false
11. then  \( DI \leftarrow DI \cup \{J_i\} \)
12. return \( DI \)

Example: Let four intervals and their values be \( I_1 = [1, 2], v_1 = 0.9; I_2 = [2, 3], v_2 = 0.5; I_3 = [1, 4], v_3 = 4; I_4 = [4, 5], v_4 = 2 \). By following the above algorithm, we can

- compute value densities \( \rho_1 = 0.9, \rho_2 = 0.5, \rho_3 = 4/3, \rho_4 = 2 \)
- sort the intervals by the densities, we get \( I_2, I_3, I_1, I_4 \)
- find disjoint intervals in order from above. Let \( DI \) be the final solution, we get \( DI = \{I_4, I_3\} \). We cannot choose \( I_1 \) or \( I_2 \) any further because they both overlap with \( I_3 \).

It is not hard to verify by enumerating all \( 2^4 - 1 = 15 \) possible selections that \( DI \) gives the maximum total value of 6.

However, this greedy algorithm does not guarantee an optimal solution. Give a counter example to show that it does not work.

(c) (25 points) Can you find an efficient dynamic programming solution to the weighted interval scheduling problem \( P_2 \)?