**Task 7**

1. **Detailed assumptions:**

* The hiding spots are located along a straight line.
* The target moves to an adjacent hiding spot between every two consecutive shots.
* The shooter can hit any of the hiding spots.
* The shooter cannot see the target.
* The shooter wants to hit the target with the fewest number of shots possible and make sure it will be shot.

1. **Problem Description:**

* We have a computer game that has a shooter and a moving target.
* The shooter can hit any hiding spots which are located along a straight line in which the target can hide. The number of hiding spots can’t be less than one up to n.
* The shooter can never see the target and all he knows is that the target moves to an adjacent hiding spot between every two consecutive shots.
* The goal is to find a strategy that guarantees hitting the target in the minimum number of shots.

1. **Explaining solution:**

Greedy Algorithm:

• Let's begin by counting the concealing places from 1 to n from left to right.

The gunner should fire the opening shot at spot 2 (or, symmetrically, at spot n 1) as these are the only shots that can ensure either hitting the target.

target or ensuring that the target cannot move after the first shot to a location (1 or, respectively, n). We'll start by thinking about the scenario in which the aim was initially.

in an area with an even number. After the initial shot, the target either moved to a location marked with an odd number greater than or equal to 3 or the shooter hit the target. As a result, if the shooter hits spot 3 with his second shot, either he will reach the target or ensure that it will be at a location marked with an even number larger than or equal to 4. So, if the shooter keeps firing at the same numbered positions (4, 5,..., n 1), they will undoubtedly hit the target.

The target will not be hit by the n 2 shots indicated above if it was previously located at an odd-numbered spot since its location and the shots' locations will always have the opposite parities. However, the target will move to a position with the same parity as the parity of n1 following the shot at spot n 1,so we will consider the following pseudocode:

1. At the beginning, assume that the target is hiding in the middle hiding spot along the straight line.
2. Shoot at the middle hiding spot.
3. If the target is hit, stop. Otherwise, continue.
4. If the target moved to the left of the middle hiding spot, shoot at the middle hiding spot between the left-most hiding spot and the middle hiding spot.
5. If the target moved to the right of the middle hiding spot, shoot at the middle hiding spot between the middle hiding spot and the right-most hiding spot.
6. Repeat steps 3-5 until the target is hit.

**Explanation:**

This algorithm works by using a divide and conquer approach. At each step, we divide the remaining hiding spots in half and determine which half the target is in. We then shoot at the middle hiding spot in that half. This allows us to narrow down the possible hiding spots for the target with each shot.

The algorithm is guaranteed to hit the target because, at each step, we are reducing the number of possible hiding spots by half. Since the target can only move to adjacent hiding spots between shots, it will eventually be forced into a single hiding spot that we will be able to hit with our final shot.

1. **Pseudo code:**

A screenshot of a computer

Description automatically generated

1. **Implementation:**

Graphical user interface, text, application

Description automatically generated

1. **Time and Space Complexity:**

To analyze the time complexity of the algorithm, we can break it down into its individual steps:

1. Reading input (single integer n) - O(1)

2. Checking for special case n = 2 - O(1)

3. Shooting at consecutive spots from 2 to n-1 - O(n)

4. Shooting at spot n-1 - O(1)

5. Shooting at even-numbered spots from n-2 down to 2 - O(n/2)

Therefore, the time complexity of the algorithm can be expressed as:

O(1) + O(1) + O(n) + O(1) + O(n/2) = O(n)

The space complexity of the algorithm is O(1) since it only uses a fixed amount of memory regardless of the size of n.

In summary, the time complexity of the algorithm is linear in the input size.

1. **Another solution:**

An alternative technique to hit the target would be to use a **binary search approach**. However, this approach would not guarantee hitting the target in the minimum number of shots. The algorithm presented above guarantees hitting the target in at most 2(n-2) shots, while the binary search approach would take at least **log2(n)** shots to hit the target.

**Steps**

1. Define a function binarySearchTarget(hidingSpots, left, right) that takes an array of hiding spots, and the indices of the left and right bounds of the search range as parameters.
2. Calculate the middle index of the search range: middle = (left + right) / 2.
3. Shoot at the hiding spot at the middle index.
4. If the target is hit, return true.
5. If the target is not hit, determine whether the target moved to a hiding spot to the left or right of the middle index. To do this, compare the current index of the shooter to the middle index. If the shooter's current index is less than the middle index, the target moved to a hiding spot to the left. Otherwise, the target moved to a hiding spot to the right.
6. If the target moved to a hiding spot to the left, call binarySearchTarget(hidingSpots, left, middle-1).
7. If the target moved to a hiding spot to the right, call binarySearchTarget(hidingSpots, middle+1, right).
8. If the function reaches the end of the search range without hitting the target, return false.
9. In the main program, call binarySearchTarget(hidingSpots, 0, n-1), where hidingSpots is an array of the n hiding spots.

A screenshot of a computer

Description automatically generated

1. **Input and outputs:**

**Input: An integer n > 1 representing the number of hiding spots.**

**Output: A sequence of 2(n-2) shots**

1. **Conclusions:**

**The target will be hit with the fewest number of shots feasible according to the algorithm's guarantee. It functions by firing at specific hiding places that either hit the target or restrict the target's movement to a smaller set of hiding places where it can be hit with later rounds.**

1. **References:**

*"Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein*