Question 1 Battle Games:

Description

Imagine a group of n players that are located in different floors. The i-th player (for each i such that

1 ≤ i ≤ n, where i is an integer) is characterized by the following:

• pi

, a unique starting floor level

• hi

, a starting HP (Health Points) value

• di

, a direction of movement, either U indicating that the i-th player is going upward or D indicating

that the i-th player is going downward

All players move simultaneously at the same speed in their respective directions. When two players

meet, they engage in a battle. The player with the lower health is eliminated instantaneously, while

the other continue to move in the same direction at the same speed but with a reduced HP by 1. In a

case where two players have identical HP, they are both eliminated simultaneously after the battle.

Your task is to determine the final HP of the surviving players.

Solution:

- `pi`: the starting floor level of the i-th player.

- `hi`: the starting HP value of the i-th player.

- `di`: the direction of movement of the i-th player, either U indicating that the i-th player is going upward or D indicating that the i-th player is going downward.

All players move simultaneously at the same speed in their respective directions. When two players meet, they engage in a battle. The player with the lower health is eliminated instantaneously, while the other continues to move in the same direction at the same speed but with a reduced HP by 1. In a case where two players have identical HP, they are both eliminated simultaneously after the battle.

To determine the final HP of the surviving players, we can use the following algorithm:

1. Sort the players based on their starting floor level `pi`.

2. For each player, calculate the floor level `f` at which they will meet another player. If there is no other player in their direction of movement, set `f` to infinity.

3. Sort the players based on the floor level `f` at which they will meet another player.

4. For each pair of players that will meet, calculate the winner of the battle and update their HP accordingly.

5. Remove all players with HP equal to 0.

6. Repeat steps 2-5 until there are no more battles to be fought.

The final HP of the surviving players is the output of the algorithm.

Question 2 Buried Treasure

Jack Sparrow, the legendary pirate of the Seven Seas, sets sail to an inhabited island in search of a

buried treasure. Guided by his map, he believes the treasure lies within a weird-looking trench. The

trench stretches across a width of n. For any given point from i − 1 to i on the trench’s x-coordinate,

the depth is represented by di (for each i such that 1 ≤ i ≤ n, where i is an integer).

Jack is wondering about the size of the largest treasure that could possibly fit inside the trench.

Wanting to maximize his haul, he turns to you, a trusted member of his crew, to make the calculations.

By largest, Jack means the maximum area – the product of width and height – of the rectangular

treasure chest that can be buried within the trench’s confines. For example, the following figure shows

the largest possible treasure that can fit in a trench with n = 8 and d = [6, 2, 5, 4, 5, 1, 4, 4].

Could you give these calculations a look for our legendary pirate?

Solution:

n: the width of the trench.

di: the depth at the i-th point on the trench’s x-coordinate.

To find the largest treasure that can fit inside the trench, we need to find the maximum area of a rectangle that can be formed using the trench’s confines. The width of the rectangle is equal to the width of the trench, which is n. The height of the rectangle is equal to the maximum depth of the trench.

To find the maximum depth of the trench, we can use the following algorithm:

Initialize a stack S.

For each point i on the trench’s x-coordinate: a. While S is not empty and di is less than or equal to the depth of the point at the top of the stack, pop the top element from S. b. If S is empty, the maximum depth of the trench at point i is di \* i. c. Otherwise, the maximum depth of the trench at point i is (di \* (i - S.top() - 1)). d. Push i onto S.

While S is not empty, pop the top element from S and calculate the maximum depth of the trench at that point using the same formula as in step 2c.

The maximum depth of the trench is the maximum value obtained from the above algorithm. We can then calculate the maximum area of the treasure chest by multiplying the maximum depth of the trench by the width of the trench.