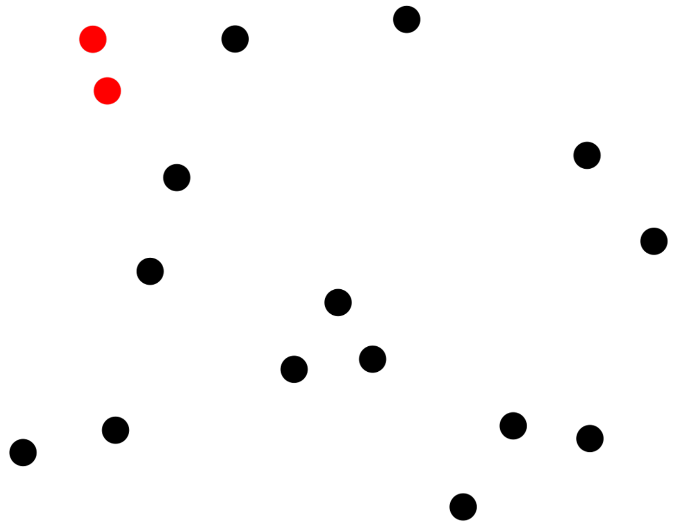
COMP 3711

Design and Analysis of Algorithms

Divide and Conquer

Problem 1: Closest pair of points!

Problem 1: We are given an array of n points in the plane, and the problem is to find out the closest pair of points in the array. Recall the following formula for distance between two points p and q:



$$\|pq\| = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Naive Approach: Check the distance of every pair! Find the minimum.

Complexity? $\binom{n}{2}$ pairs, So $O(n^2)$

Use Divide and Conquer to enhance it!?

Problem 1: Closest pair of points!

Initial Idea:

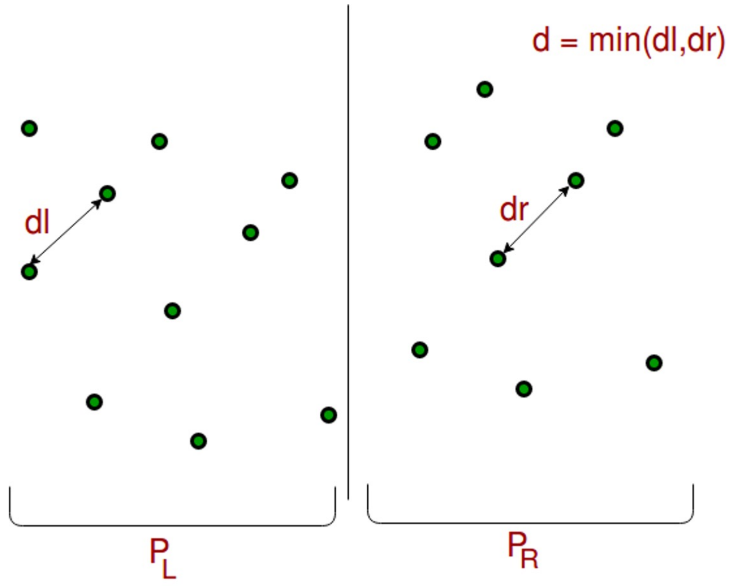
1- Try to "split" the points into two sections. "Left" and "Right".
2- Find the closest pair in each section.
3- Somehow "combine" the solutions to get the final result.

Question : Consider the following example:
Assume d is minimum of dl and dr.

Can we say the closest pair of points
Belongs to the end points of d?

No! (Why?)

Some further checks needs to be done!

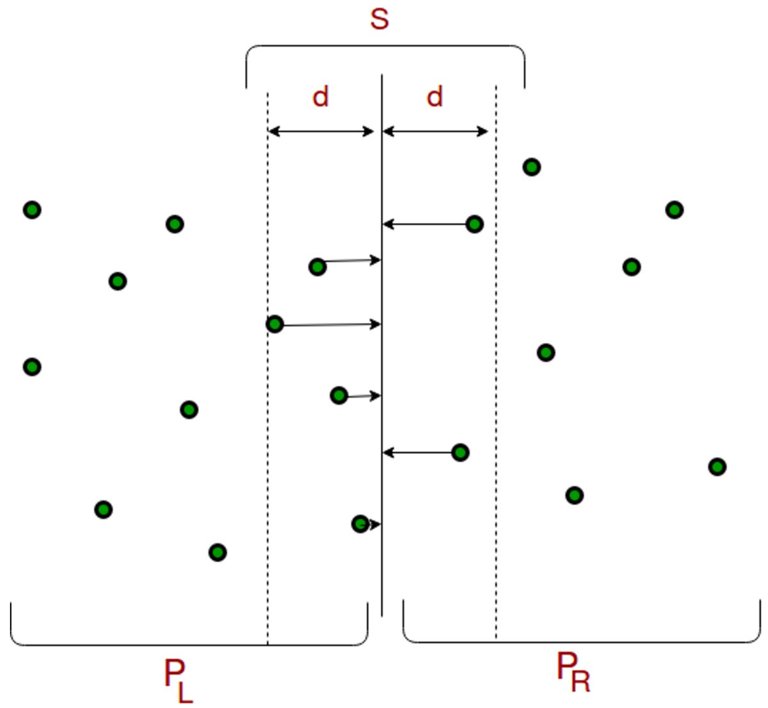


Problem 1: Closest pair of points!

Idea: We need to check the distance points from "left" section to points in "right" section.

Question : Do we need to check every pair?

Idea: Get all the points in this boundary.
Sort these point by their y value.
Now we try to find the closest pair here.



Question : Isn't it problematic? It looks like $O(n^2)$ again?

No! It can be done in O(n)!!! (How? Not obvious!!)

Problem 1: Closest pair of points!

1. Sort inputs by their “x” value.
2. Split them into two groups.
3. Find the closest pair for each half.
4. Find the minimum among them.
5. Sort the point in the boundary by their “y” value.
6. Check if there exists a closer pair that has one point in the left group and one point in the right half.

$$T(n) = 2T(n/2) + O(n \log n)$$

Complexity? $O(n \log^2 n)$

Can be further improved to O(nlog n)

Reference: https://www.cs.cmu.edu/~15451-s20/lectures/lec23-closest-pair.pdf

Problem 2: Majority Element!

Problem 2: Find the majority element in the array. A **majority element** in an array A[] of size n is an element that appears more than n/2 times (and hence there is at most one such element).

Input : A[]={3, 3, 4, 2, 4, 4, 2, 4, 4}
Output : 4

Naive Approach: Go through the array, for each new element count it through the whole array.

Complexity? $O(n^2)$

Problem 2: Majority Element!

Approach 2: First let's sort the array.

Idea: If there exists a majority element, it must be present in the middle index!!! So check the middle index.

Question : Are we done?

No! There might be no majority element?!

Count the frequency of that element in the middle index.

Complexity? $O(n \log n)$

Problem 2: Majority Element!

Approach 3: Use Divide and Conquer

Idea: Split the array into two halves. Find the majority in each. If A has a majority element, It has to be either the majority of the left half or right half (or maybe in both) **(why?)**

Question : So if the majority of left half is “x” and the majority of right half is “y” what should we do?

Just count both, see which one is majority

$$T(n) = 2T(n/2) + O(n)$$

Complexity? $O(n \log n)$

Problem 2: Majority Element!

Approach 4: Use bit manipulation

Idea: Consider each number as a 32-bit integer. Write it down in binary form. Can we exploit this binary form?

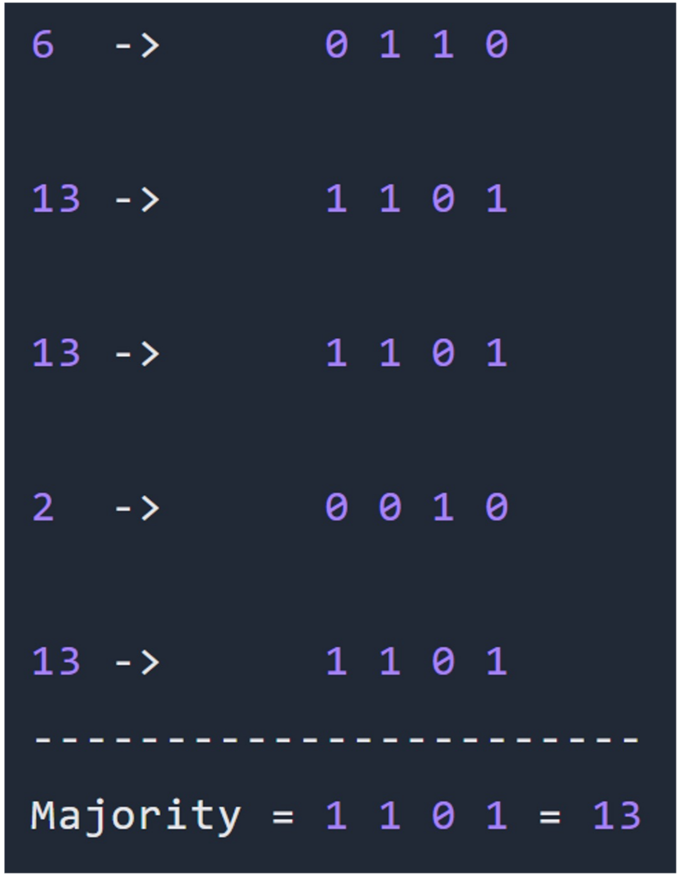
Assume the array is {6, 13, 13, 2, 13}

Question : How to find the majority bit?

Simply add them! If it's bigger than n/2, then "1" is majority. Otherwise its not.

Complexity? $O(n)$

Question : What if there is no majority?



```
6   ->       0 1 1 0

13  ->       1 1 0 1

13  ->       1 1 0 1

2   ->       0 0 1 0

13  ->       1 1 0 1
-----------------------
Majority = 1 1 0 1 = 13
```

Problem 2: Majority Element!

There are other interesting and efficient approaches too.

For example check Boyer-Moore voting algorithm $O(n)$. We will see something similar in next questions!

Problem 3: Maximum subarray product!

Problem 3: You are given a one dimensional array that may contain both positive and negative integers, find the product of contiguous subarray of numbers which has the largest product.

Input: arr[] = {6, -3, -10, 0, 2}

Output: 180

Explanation: The subarray is {6, -3, -10}

Input: arr[] = {-1, -3, -10, 0, 60}

Output: 60

Explanation: The subarray is {60}

Problem 3: Maximum contiguous subarray product!

Naive Approach: Go through all contiguous subarrays. Find the maximum.

Complexity? $O(n^2)$

Idea: Looks very similar to maximum subarray sum!

1. Split the array into two halves.
2. Find the maximum contiguous subarray of the left and right halves. (L,R)
3. Check if there exists a contiguous subarray that starts from the left half and ends in the right half with higher product! Find maximum.

But How?

Problem 3: Maximum contiguous subarray product!

{ -5, -4 , -3, 2, -2, 5, 3, -10}

{ -5, -4 , -3, 2} **{ -2, 5, 3, -10}**

Question : Is this idea correct?

Look at the “suffix” subarray of left half with maximum product.
Look at the “prefix” subarray of right half with maximum produc
Check if their product is bigger than the L and R.

No!!!!!!

We need to also find the suffix subarray with minimum product of left half and prefix subarray with minimum product of right half as well!! (Why?)

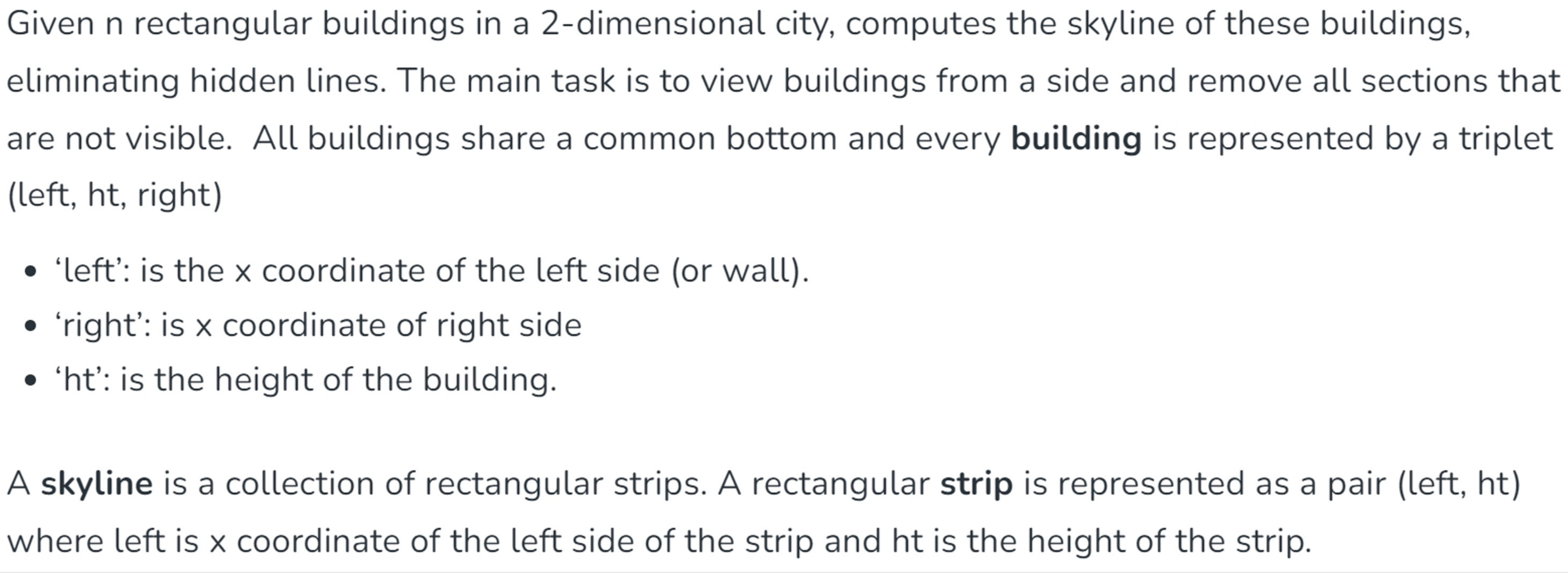
Problem 3: Maximum contiguous subarray product!

Question : How much time does it take to find the largest and smallest product prefix/suffix subarray of right and left half? $O(n)$ (How?)

$$T(n) = 2T(n/2) + O(n)$$

Complexity? $O(n \log n)$

Problem 4: Skyline problem



Given n rectangular buildings in a 2-dimensional city, computes the skyline of these buildings, eliminating hidden lines. The main task is to view buildings from a side and remove all sections that are not visible. All buildings share a common bottom and every **building** is represented by a triplet (left, ht, right)

- 'left': is the x coordinate of the left side (or wall).
- 'right': is x coordinate of right side
- 'ht': is the height of the building.

A **skyline** is a collection of rectangular strips. A rectangular **strip** is represented as a pair (left, ht) where left is x coordinate of the left side of the strip and ht is the height of the strip.

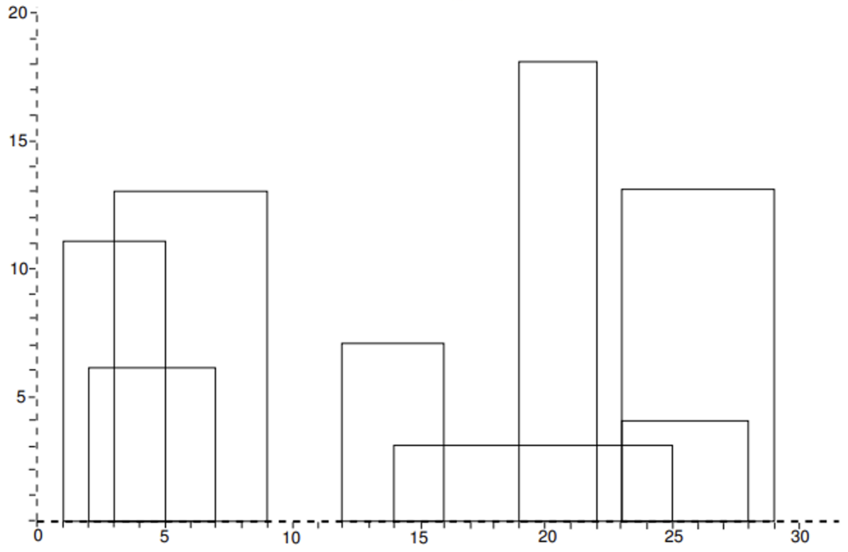
Problem 4: Skyline problem

Example: The skyline of the buildings

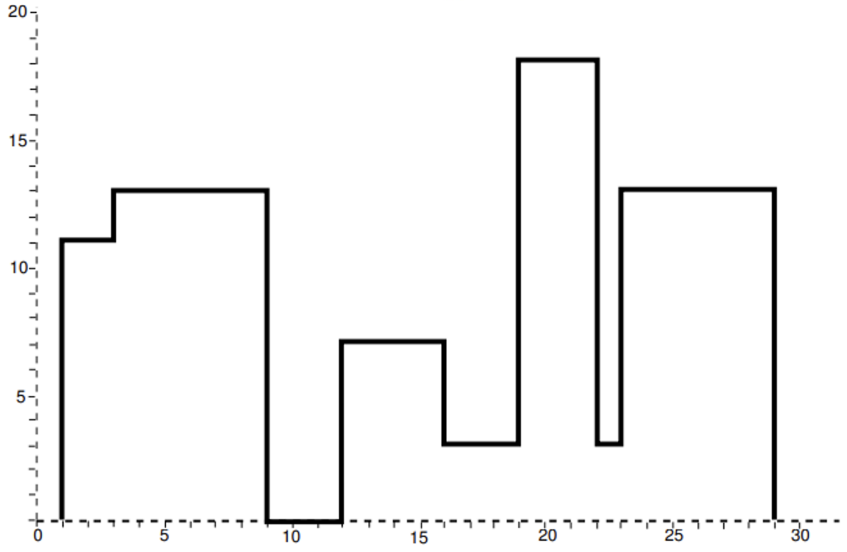
$$\{(\mathbf{3},13,\mathbf{9}),(\mathbf{1},11,\mathbf{5}),(\mathbf{12},7,\mathbf{16}),(\mathbf{14},3,\mathbf{25}),(\mathbf{19},18,\mathbf{22}),(\mathbf{2},6,\mathbf{7}),(\mathbf{23},13,\mathbf{29}),(\mathbf{23},4,\mathbf{28})\}$$

is

$$\{\mathbf{1},11,\mathbf{3},13,\mathbf{9},0,\mathbf{12},7,\mathbf{16},3,\mathbf{19},18,\mathbf{22},3,\mathbf{23},13,\mathbf{29},0\}$$



buildings



skyline

Problem 4: Skyline problem

Idea: Looks very similar to Merge sort algorithm!

1. Sort the buildings based on their starting point.
2. Split these buildings into two halves. (L and R)
3. Calculate the skyline of each half.
4. Combine these two skylines together.

But How? (slightly modify the approach used in merge sort)

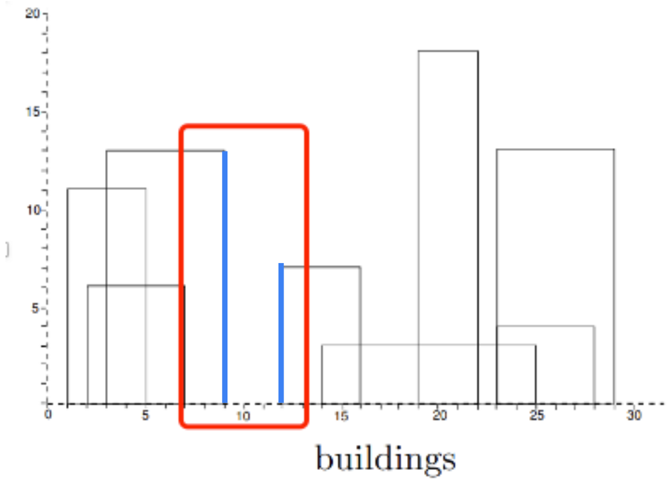
$$T(n) = 2T(n/2) + O(n)$$

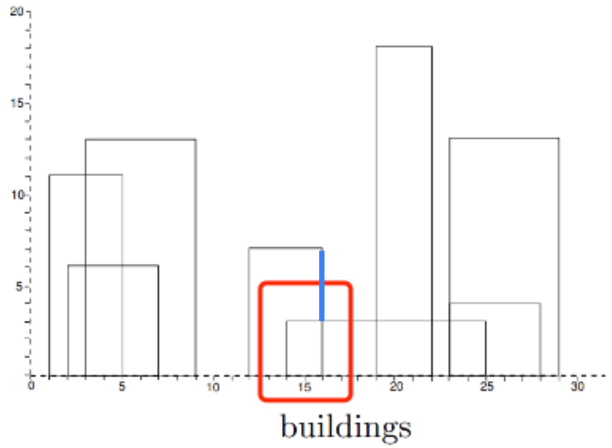
Complexity? $O(n \log n)$

Problem 4: Skyline problem

Idea: Looks very similar to Merge sort algorithm!

How to Combine these two skylines together?





Problem 5: Knights, Rogues and the Princess!

Problem 5: There are n people. More than half of them are knights. Knights **always** tells the true, Rogues can be **both truthful** or **dishonest**. All these people know the identity of everyone. But you don't.

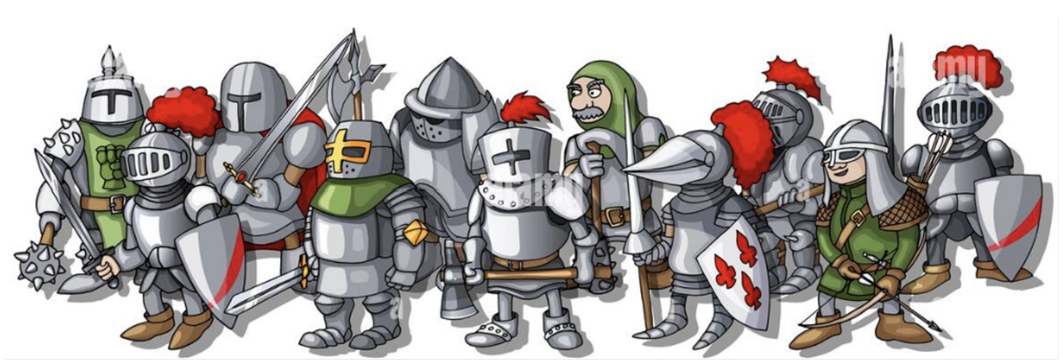
Allowed question: Pick two people, ask the identity of the other person.

Goal: Find a knight to marry!

Source: https://www.alamy.com/kingdom-war-formation-cartoon-illustration-of-different-knights-sketch-image260176794.html

Problem 5: Knights, Rogues and the Princess!

```
1. Initialize an empty stack.
2. For each person $p$ in the given people:
    a) If the stack is empty, push $p$ to the stack.
    b) Otherwise, pit $p$ against the person at the top of the stack.
        I. If both say the other one is a knight, either both are knights or both are rogues.
           Push $p$ to the stack.
        II. Otherwise, at least one of them is a rogue. Pop the stack once.
3. Return the person at the top of the stack.
```



Source: **https://www.alamy.com/kingdom-war-formation-cartoon-illustration-of-different-knights-sketch-image260176794.html**

Problem 5: Knights, Rogues and the Princess!

1. **Check for Empty List**:
    a) If there are no candidates, return None.
2. **Handle Odd Number of Candidates**:
    a) If the number of candidates is odd, set one candidate aside (unpaired).
3. **Pair Candidates and Evaluate**:
    a) Divide the remaining candidates into pairs, e.g., (i, 2*i)
    b) For each pair, ask each candidate about the other.
    c) If both candidates in a pair identify each other as knights, keep one of them. (Why?)
4. **Recursive Call**:
    a) Recursively apply the same process to the remaining candidates.
    b) If the recursive call returns None, use the previously set aside candidate (unpaired) as the solution.
5. **Return the Solution**:
    a) Return the identified knight.

Source: https://www.alamy.com/kingdom-war-formation-cartoon-illustration-of-different-knights-sketch-image260176794.html