

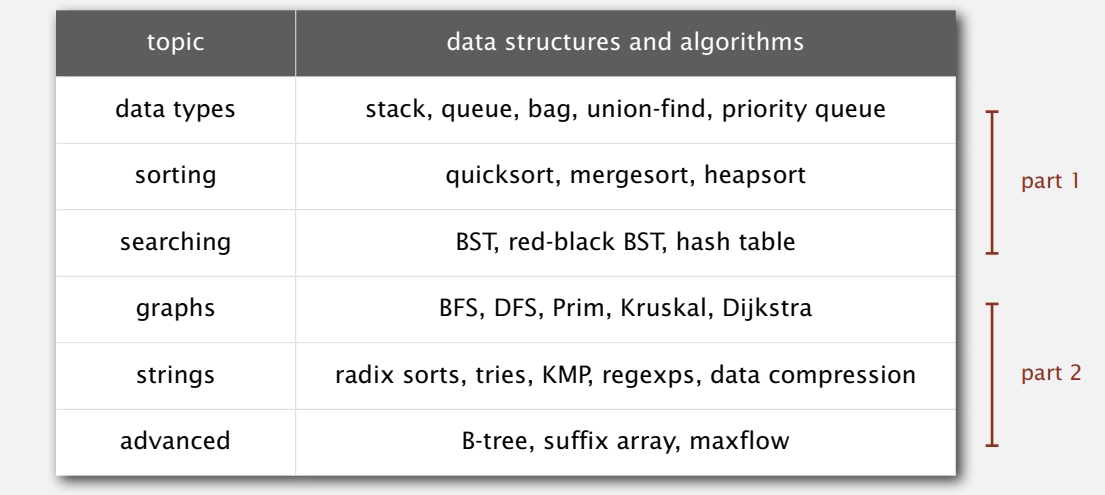
# Algorithms Stanford



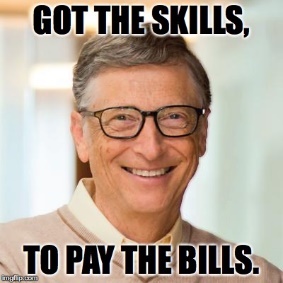
## Week 1

Introduction, Union find and analyse of Algorithms.

### Introduction



#### Reason to study algorithms

1. Fun and profit.
2. ~~Power~~ AI 
3. Better code
4. Brain stimulation ([Maths build brain muscles](https://www.theguardian.com/education/2001/aug/28/schools.highereducation))
5. Become best programmer.

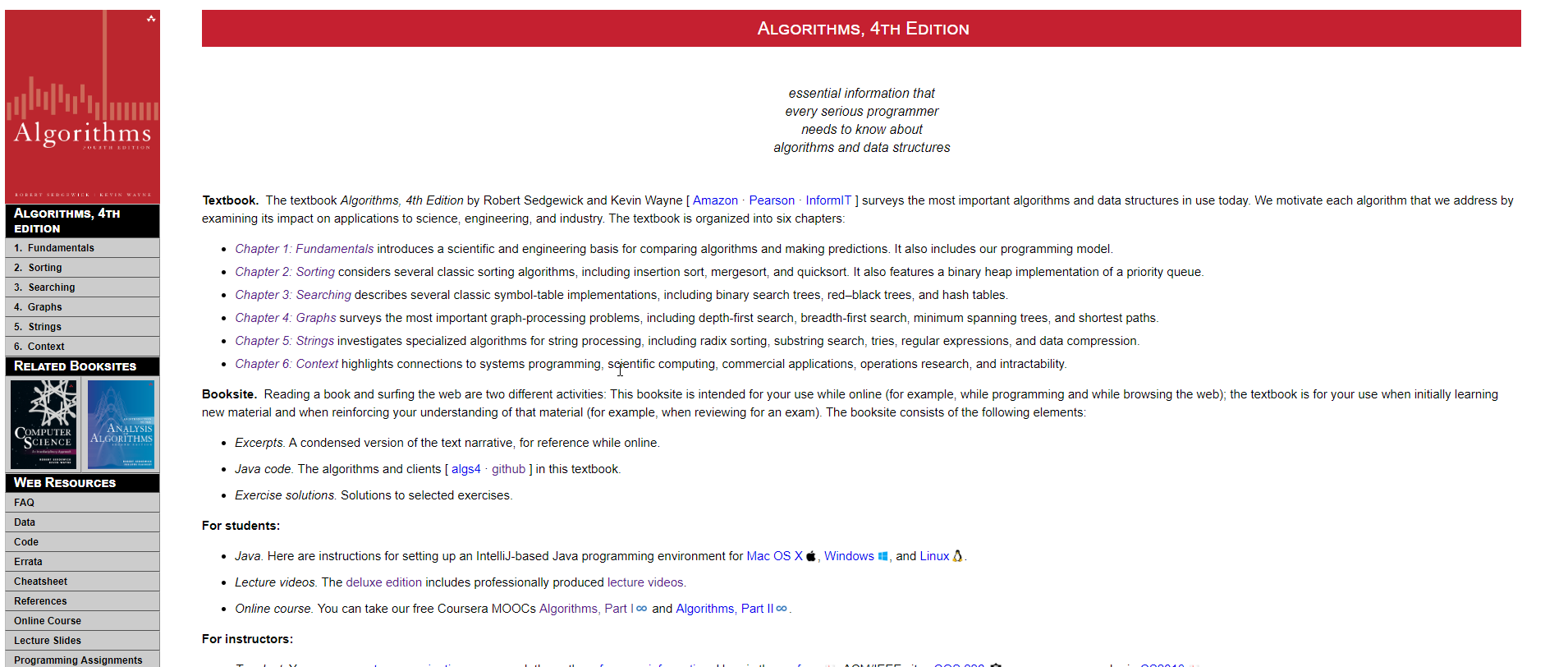


#### Extra Materials

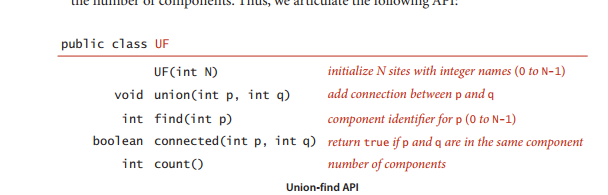
Book



[Book web page](https://algs4.cs.princeton.edu/home/)

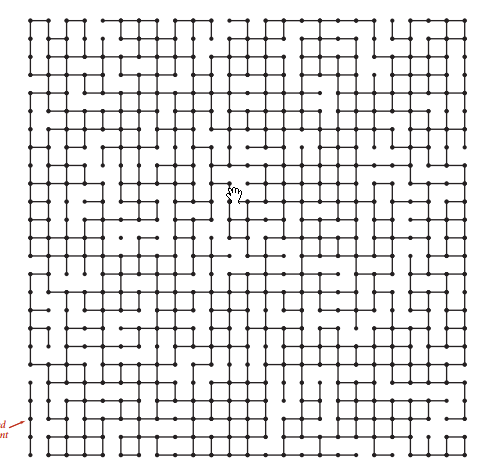


### Union Find



#### Problem

The problem is when the big variety of object

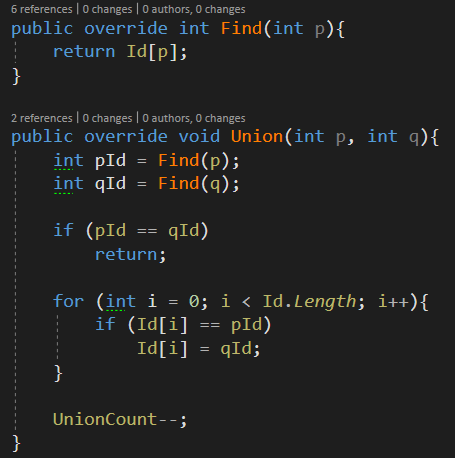


Applications involved in in connectivity problem

1. Pixels in digital photos
2. Computers in a networks
3. Friends in social network
4. Transistors in a computer chip
5. Elements in mathematical set
6. Variables in Fortran program
7. Metallic seites in composite system

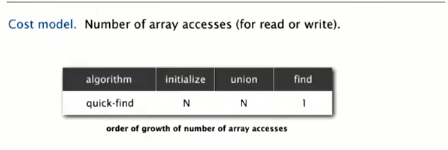
#### Quick Find

It is very simple to implement



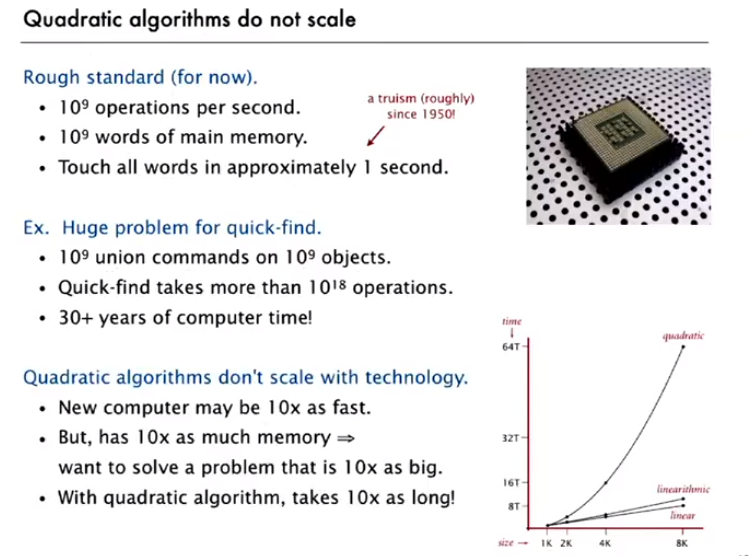
##### Analyze

So the algorithm is very slow

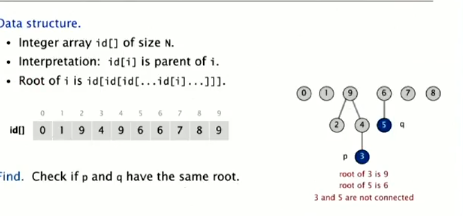


To Union all elements it will take to run Union Operation N times what will be O( what is super slow.

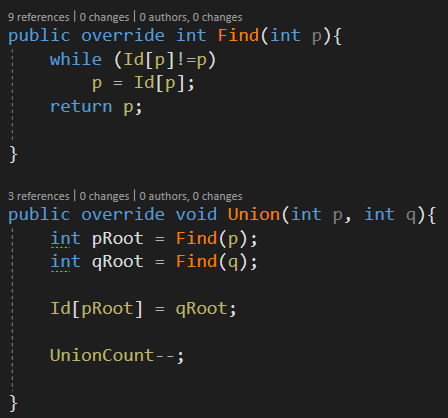
And again quadratic algorithm is to slow to be preform. So we need to find better way.



#### QuickUnion

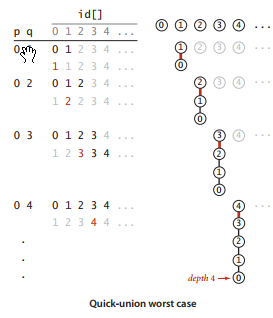
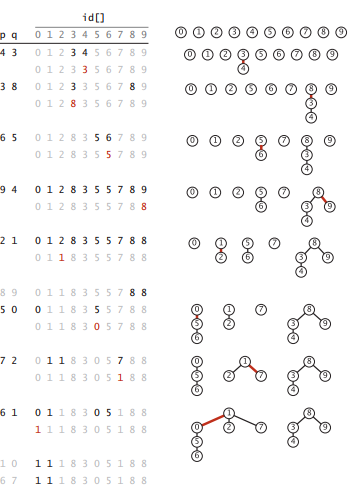


Implementation



##### Analyze

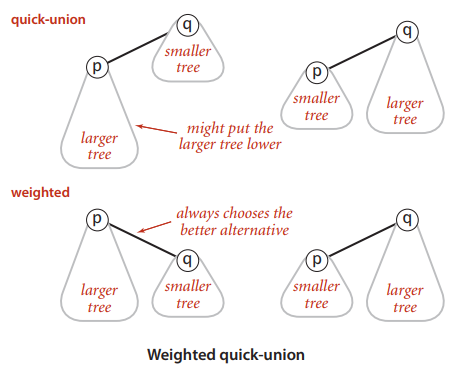
So in perfect world we have those trees that speed up over find operation.

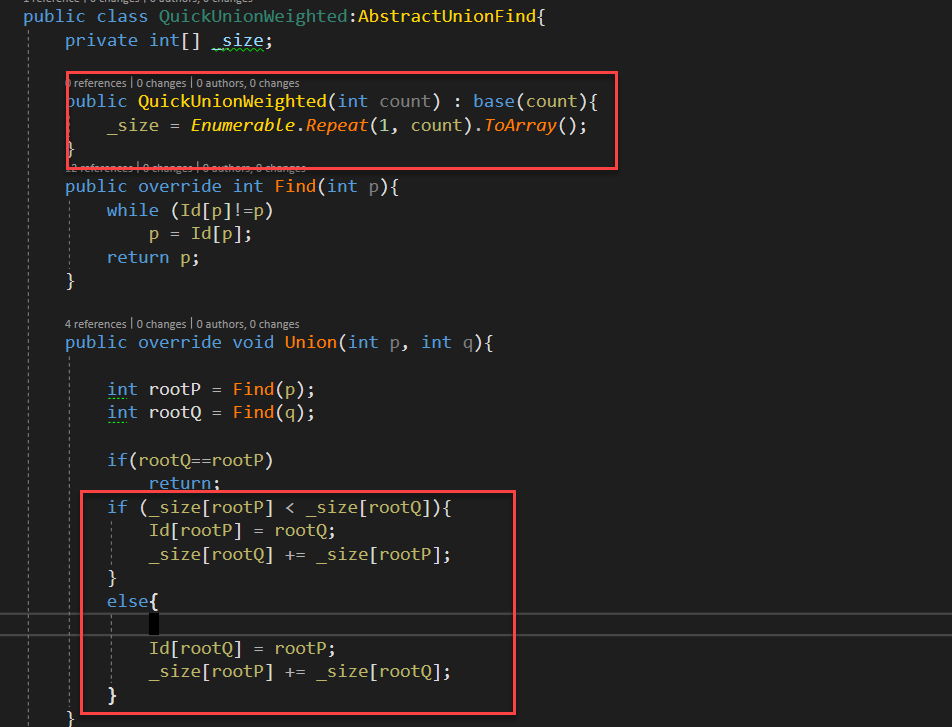


We

But some times we will have it linear tree. So it would be to slow to use Quick Union

#### Weighted Quick Union

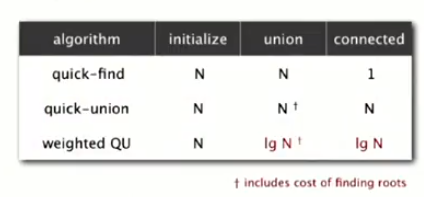




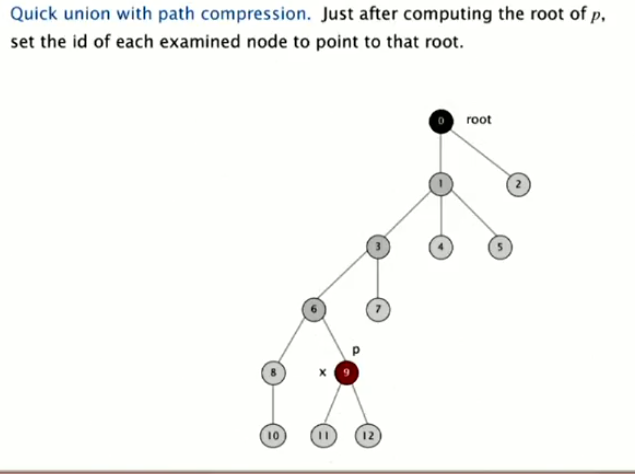
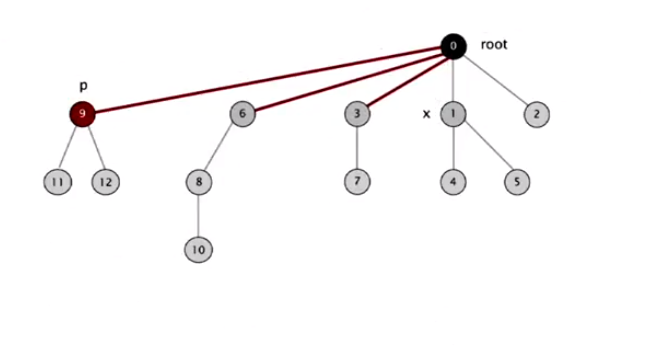
In that way our tree can maximum can be

##### Analyze

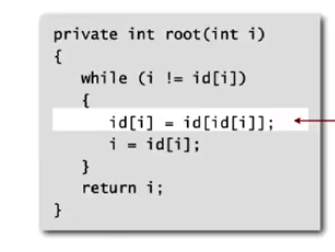
lg N

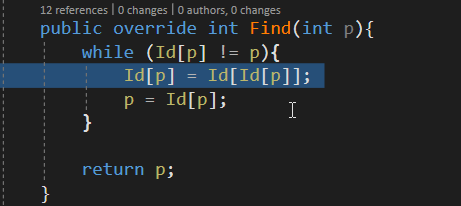


#### Path Compression

1 Line of Code!!!!



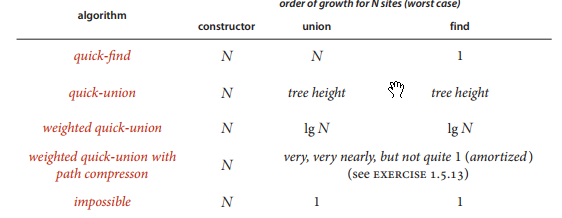


##### Analyze

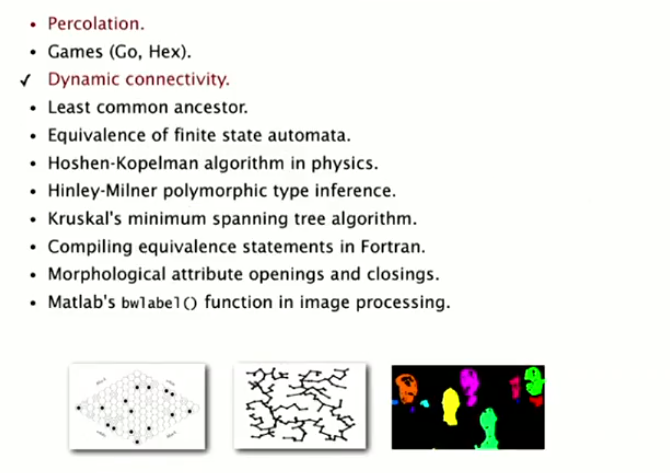


Path Compression is close to be linear.

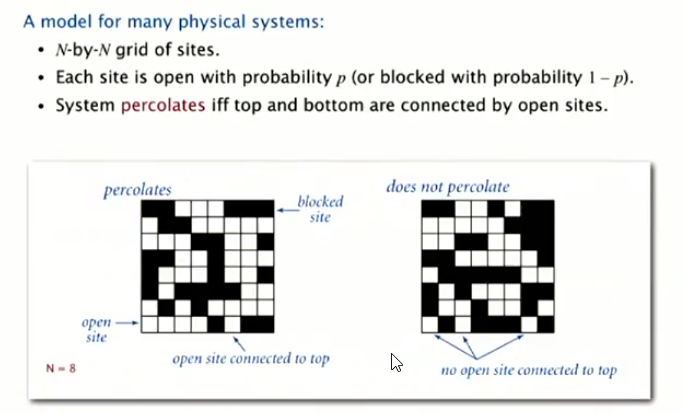
#### Analyze



#### Application



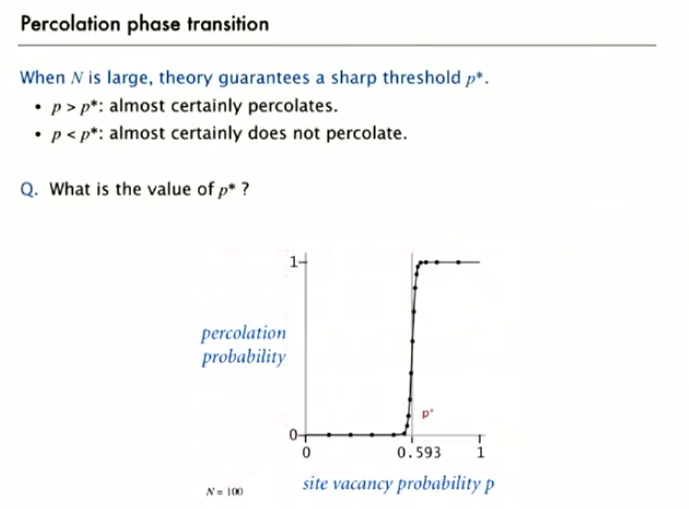
##### Percolation



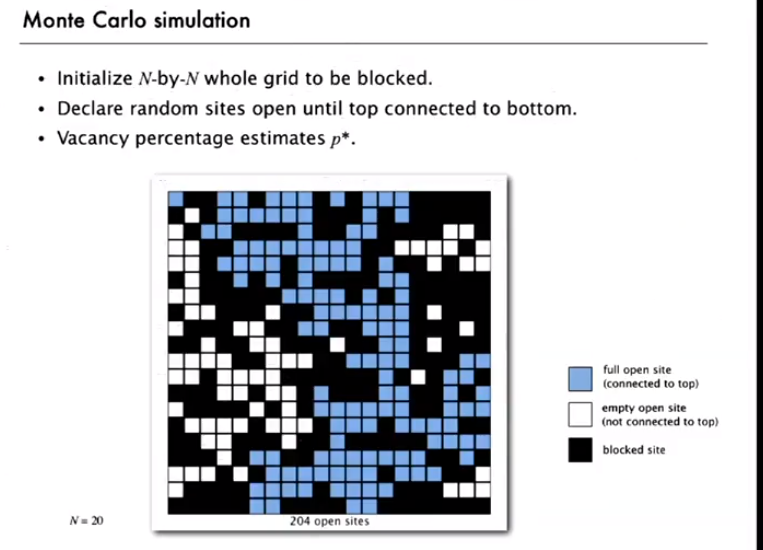
###### Likelihood of percolation



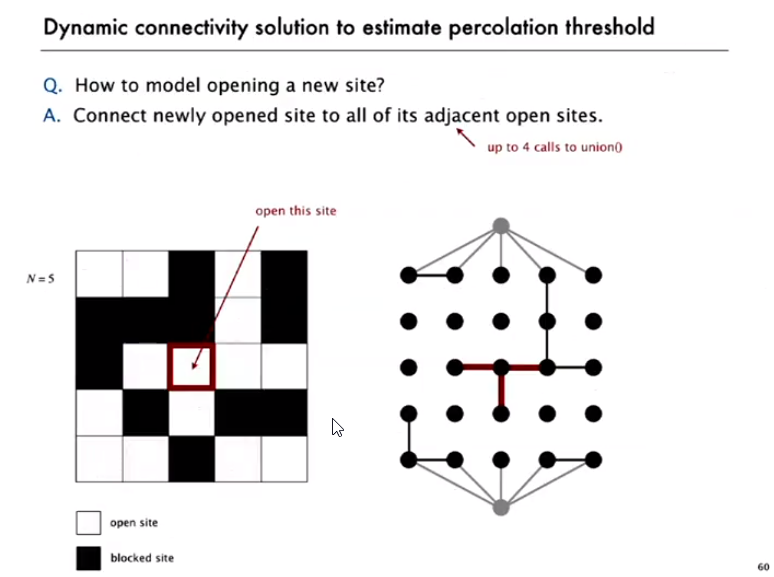
###### Probability of percolation



Monte Carlo simulation



There is Monte Carlo simulation where we open block by block to see is there perculation or not



Here is the way to represent the Percoalation problem using Quick Union way.

I can do it on my computer and run it infinite amount of times and get result for my self.

# Codility



So codility have this lesson sequence that is used for TopTal but I would like to get max score to apply for toptal.

## Lesson 1 : Iteration

For, while, foreach

Basically it talks about iterations.

### Binary Gap

The task is basicly to made while cycle together with while operation.

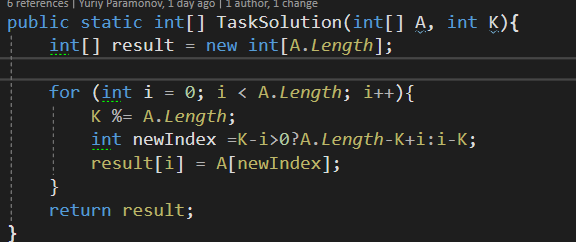
So it is very basic and siple

## Lesson 2: Arrays

Reading material focus on basic arrays operations like element value chaning and array length determination. It focus on reversing the array

### CyclicRotation

The code is very basic just shift the array for count of elements.



#### Hidden trick

So the trick to the task was in one non describe detail

Number to shift K can be longer then array length

In such case for shifting our array we need to remove extra length by preforming

K%=A.Length;

And also you need to done task in O(N) time.

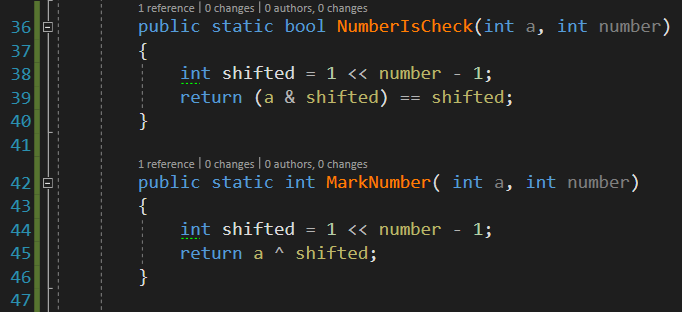
### OddConcurrencyInArray

So all you need is mark number that alredy be counted.

And then check all the checked number and find one that wasn’t dubble checked.

### Hidden trick

The number can be to big so you need to go smart to mark it. You can’t use as mutch space as you would like so you need to find way to count them on minimum space so I used trick with binary operations where each bit represent one number

And then using this technique I achived enough in total to get what I need.

## Lesson 3: Time complexity

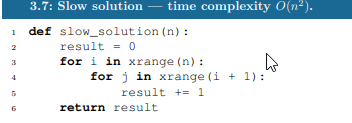
Now it basically talking about the O(n)

About types I already well-known O(n) O() O() ,O() .

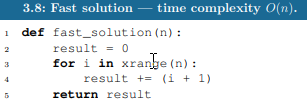
The good example case in the end in task where you need count

Arethmetical progression 1+2+3+4+…+N instead of using

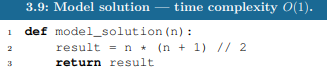
You can do it very bad:



Bad:



Or basicly remember that it was in math and it’s basic arithmetical progression and you can easly apply to get result.



### FrogJmp

So the solution is just count the count of jumps in arithmetical way instead of any iteration.

#### Hidden Trick

So ther is alwas can be situation when you have or not have extra jump

But the calculation is based on divisionso you need manully preoform opperaiton if there is some rest after devision then add one extra jump because frog need to jump one more time toget throw that distance.

### PermMissingElement

The task was simply to find missing element using the lowest possible algorithm complexity.

#### Hidden trick

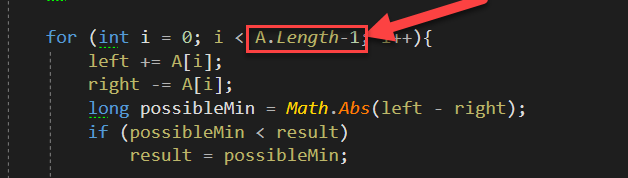
1. Using algebraial progression
2. Using longs to get totalsum of elemetns.

### TypeEquilibrium

So the task was kind of tricky. So the solution was to made sum of all elemnts.

#### Hidden trick

1. Use longs for calculation sum
2. Pay attention to the minun numbers
3. Attention on ittration of two slices it counts to much elemnts but last shouldn’t be counted



So if I have the three number

{100,5, -100}

We should have 2 slices

P[0] = |100 - -95| = 195

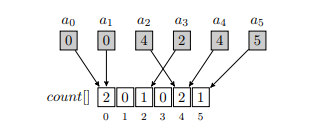
P[1] = |105 - - 100| =205

But if we let got all ittration of array what 3 then we will have one extra elemtn

P[2] = [5 -0] = 5

That would be mistake and it would not be seen in positive number but in negative it would be issaly missanderstand.

## Lesson 4 : Counting Elements



So here we have system where each element is counted and count index stores the count of elements with such number But for such operation we need first of all know the max element.

And there are were interesting example with swaping one element

Where obvious solution is O() but my the smart moves we come to the length O(n+m).

Very smart and interesting example.

### FrogRiverOne

So the trick was in arithmetical progression Using counting technique in mix.

#### Hidden trick

1. A lot of test
2. Using longs for total sum
3. Mixed technique of counting.

### Perm Check

So the task was just detect is there any missing element or not.

It was really basic if you think about it but. There is one major thing I should get noted about.

#### Hidden trick

1. Mix counting technique



1. IN the task description is clue that our integer can be bigger than our array. So we just check is array max bigger then array length.
2. Our our main integer could be smaller then aray length.

### Missing integer

So I need to find minimal positive inateger but ther could minus number just minus number and just queue of numbers.

#### Hidden trick

Unit testing. Unit testing for ever evething.

### Max Counters

It’s easy you just do old trick

#### Hidden trick

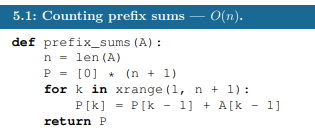
It’s sounds like you should made O(N\*M) by description, but you should actually made O(N+M)

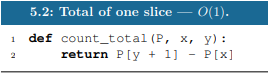
## Lesson 5: Prefix sum

So we have been give the some array and we create new array containing the prefix sums of all possible combinations of first array.



It allow us to count the





### Mushroom picker

So there is this exercise where you need to count how many mushrooms can pic the mushroom picker.

About the guy who pick the mushrooms.



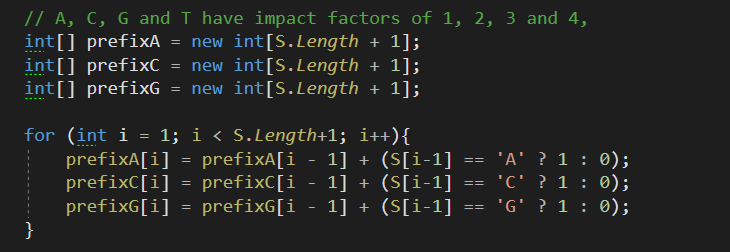
The exercise is kind of complex so I would like to sort of rewrite it.

### Passing Cars

Simple task where we just need to use the prefix sum technique.

### Genomic range query

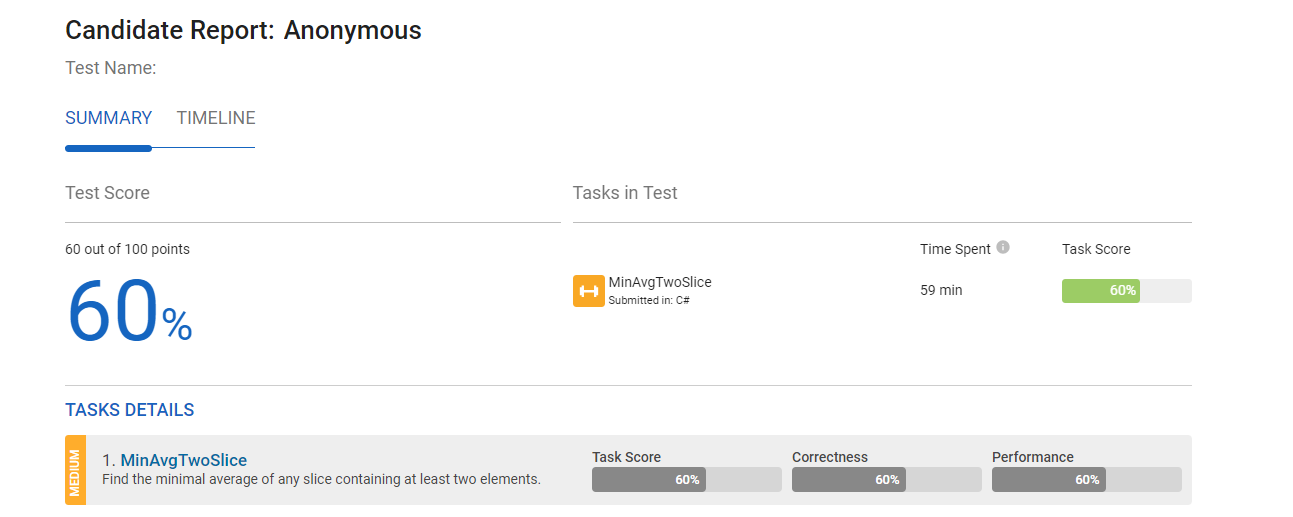
More fun task But the basic is the same all you need to do is made multiple prefix sums.



I would say not that hard.

### Minimal average of two slice

Task is hard or sort of and from the first attempt I get only 60%

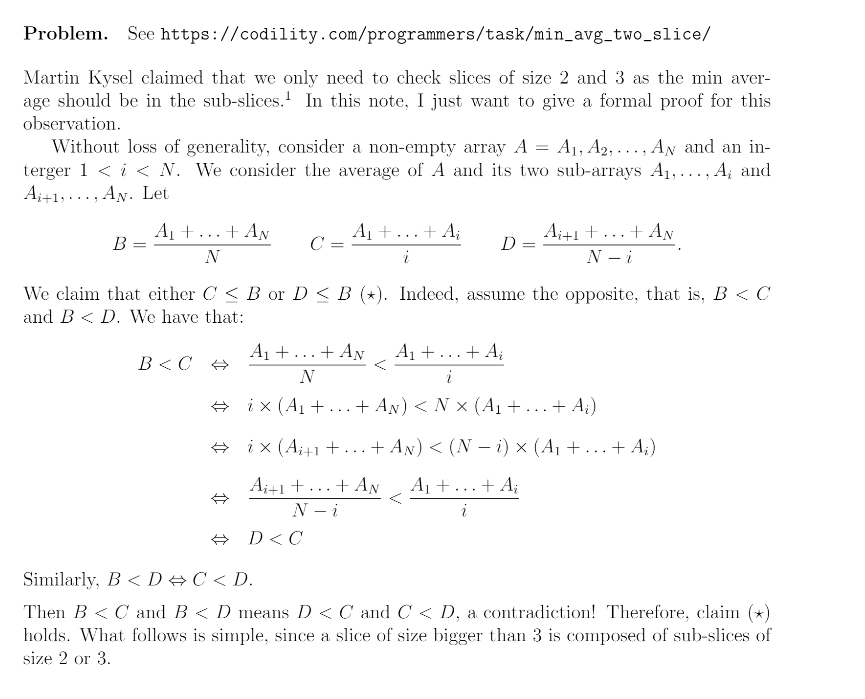


Than I found that guy who has kind of hobby of solving codility and Hacker rank tasks.



Prove by mathematical that any sequence can be made out of sub-sequence of 2 or 3 elements.

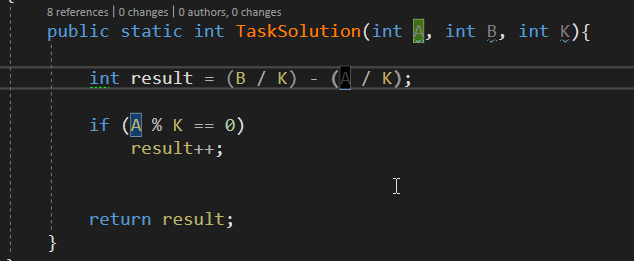
And even mathematical prove that:



What is sort of awesome.

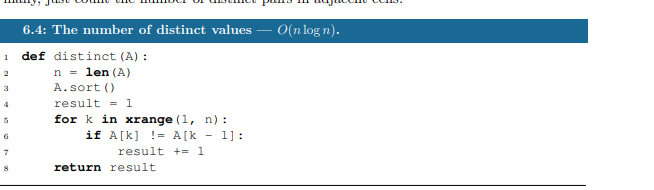
### CountDiv

So the task was to count the div between number find the amount of number devidebla by K between A and B. The solution is so simple that I would even not write the description.



## Lesson 6: Sorting

So whole idea of the lesson goes around our opportunite to sort things and we have this idea of using counter of elemts as sorting method.



In the example task we need to find count of unique values in the array N.

And what w

### Distinct

Basically the same task as it was in the beginning but array can be empty so just notice that detail but otherwise normal sorting

### MaxProductOfThree

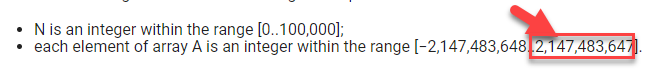
Task was to find max multiplication of 3 elements in the array the hidden trick was that there could be minus number. If the biggest are two lest first numbers

So it can be A[0]\*A[1]\*A[N] or A[N]\*A[N-1] \* A[N-2] if the biggest are positive numbers.

Easy task.

### Triangle

So the thing was to find triangle. The solution was easy just sort it. And look for 3 elements one after other. If there such sequence then we have triangle one trick in task description



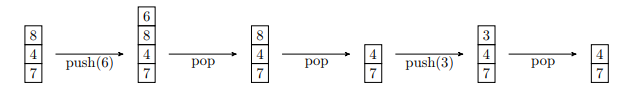
The sum of 2 sides can be sum of 2 int.MaxValue what is more then int so we need to use long.

But meanwhile it is simple task.

# Lesson 7: Stack and Queues

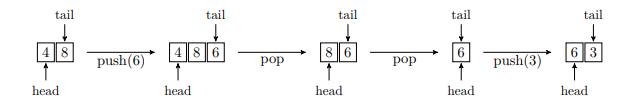
 - in C#

### Stack



FIFO(First In First Out)

### Queue

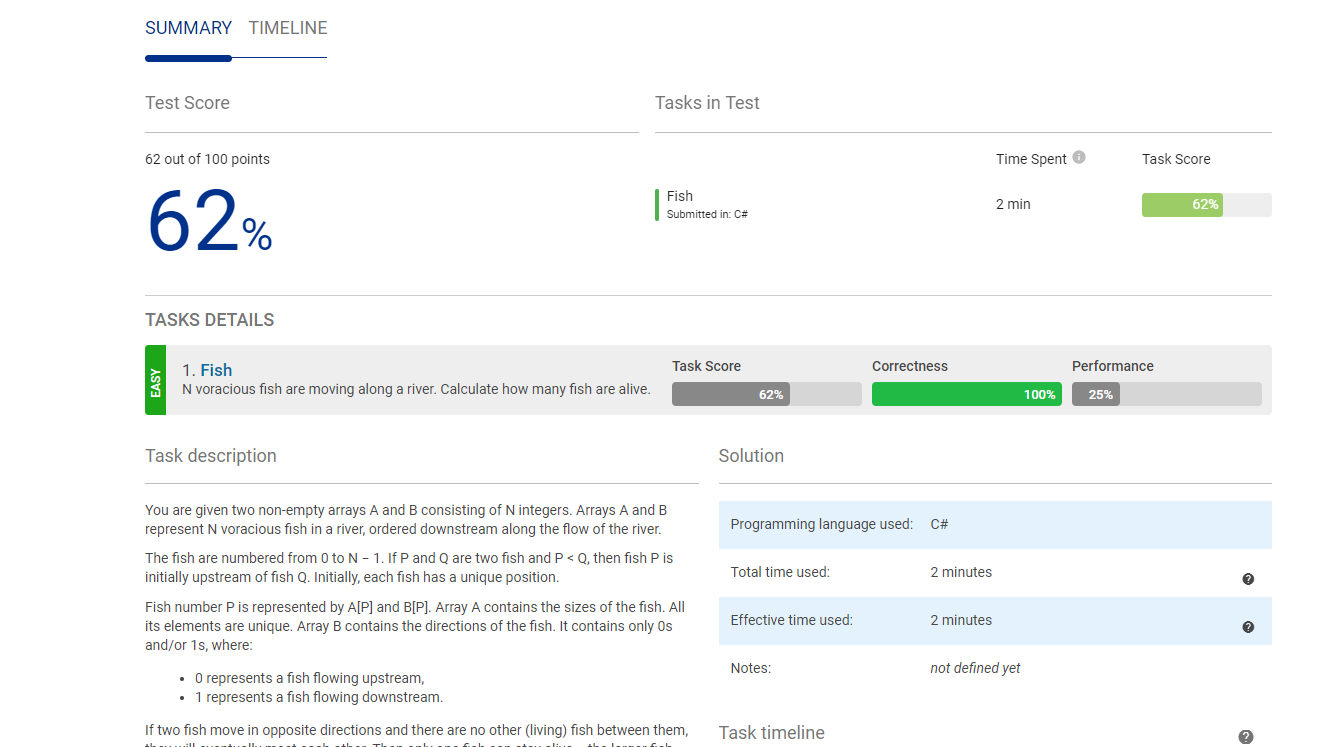


LIFO (Last In First Out

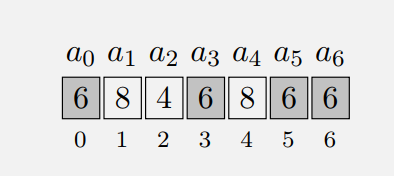
## Fish

I found solution with Stack but currently it is 62%

I should look for other solution. Tomrrow or maybe next time.

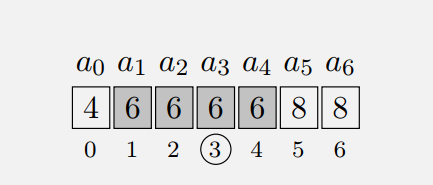


# Lesson 8: Leader

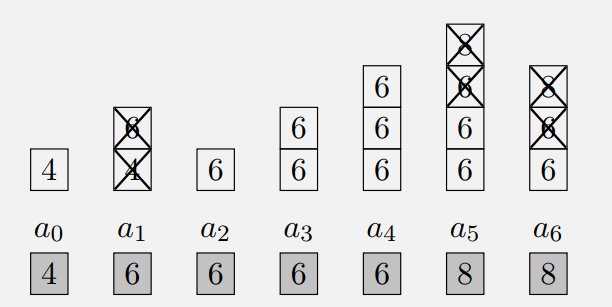


Basically saying we looking for element that occurs more then times. In given sequence A.

And there is basically two solutions:



1. Is simply sorting all elements and then choosing central one Because in order to be leader who should occur more then more then times



1. Smarter one using stack. We just add each element to stack and check if we have same element on the top of our stack the same we plan to currently add to the stack .

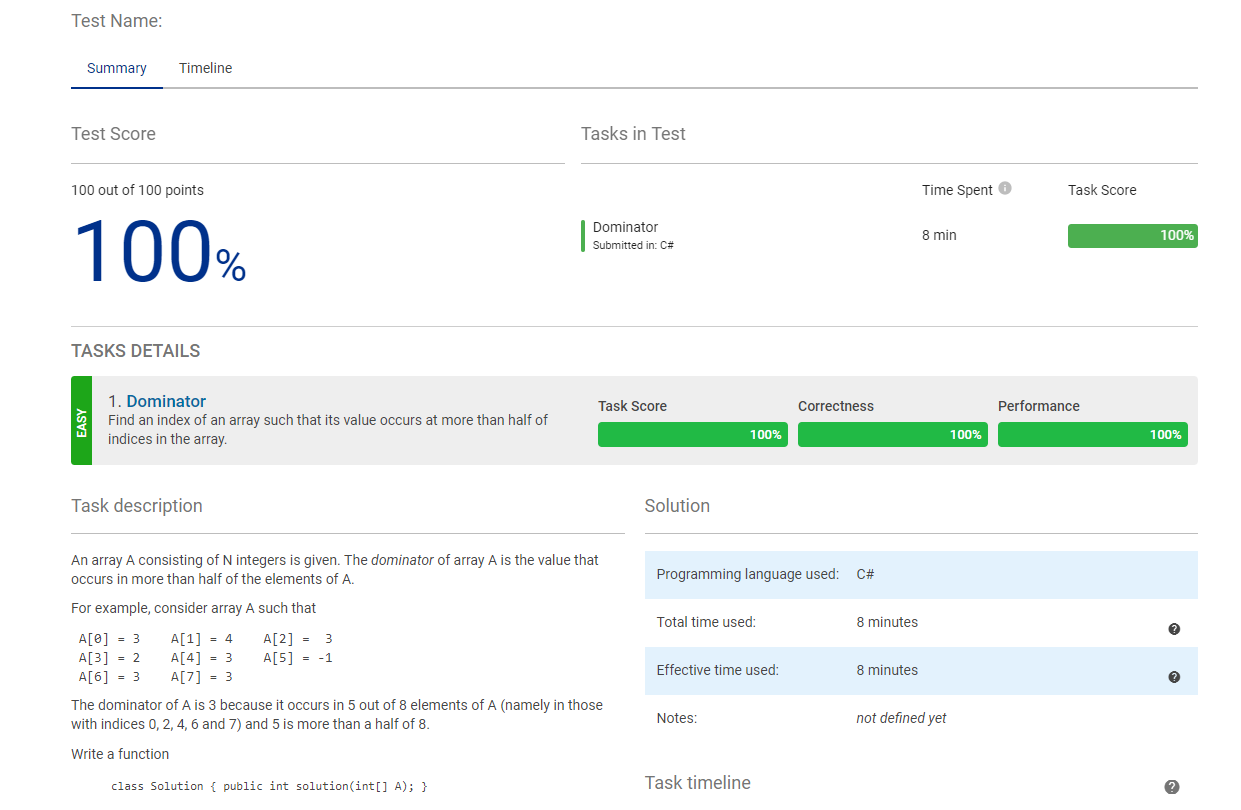
But in such cause it’s not guaranty to be the leade it’s only nown to be candidate for leader role.

So we need to check is value occurs more then time.

If yes then we have our leader else we need to return -1.

And after we recicle throw the array to check is our leader correct.

## Dominator

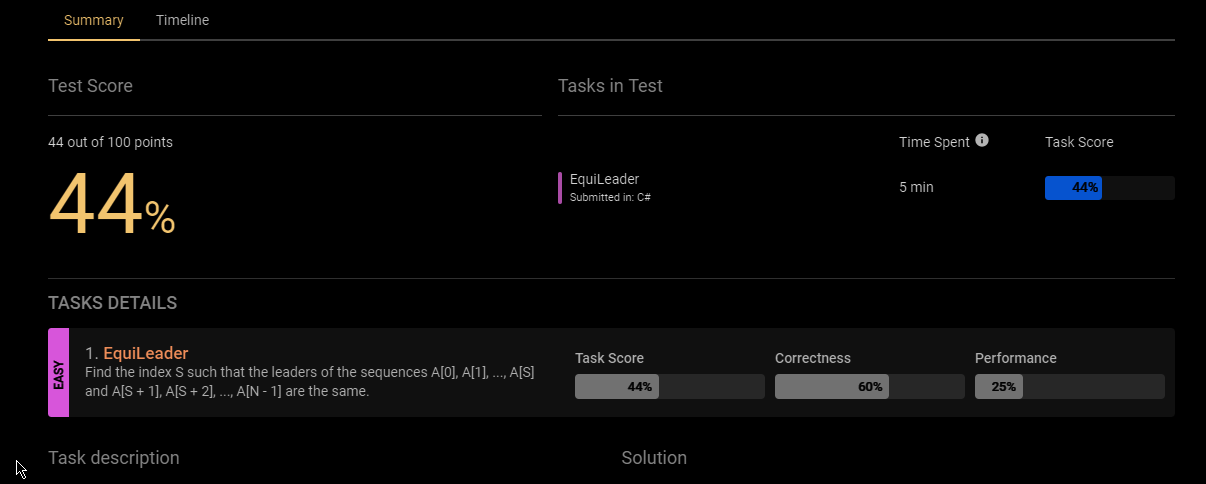


It was simply the copy of the explanation task but with one small difference. It returns us the index instead of number itself .

Very easy task. Just perfect to get back

## EquiLeader

I believe I found solution for the Equla Leader it is sort of obvious we just use 2 stacks instead of one.

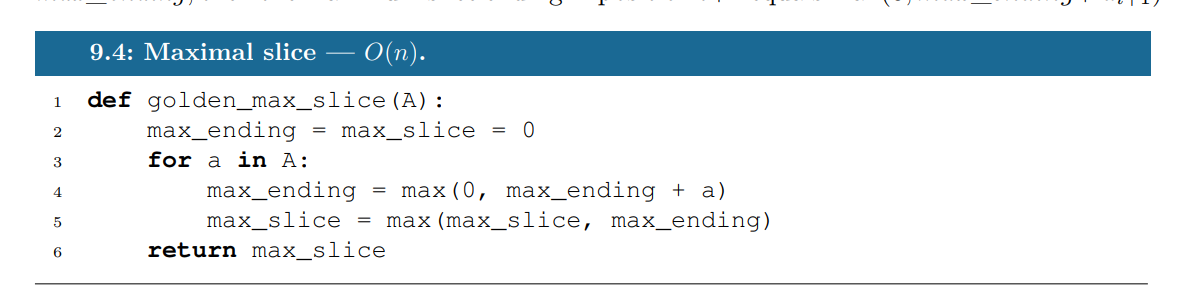


# Lesson 9: Max Slice



So for given sequency we have covered in grey the maximum sum.

So In word there is sort of way of inserting next b



Okay so it’s sort of interesting I get max slic by using this simplet implementation in the world.

But it’s still workable.

Okay so I run out of time but the concept is with me.

# Lesson 10: Prime and composite number

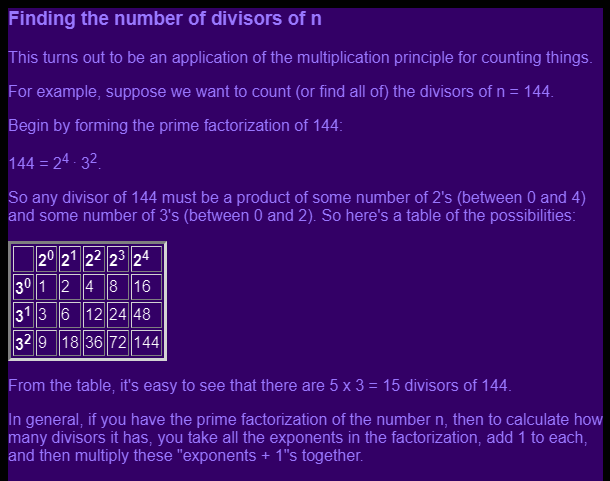
Prime number – the one that can be only divided by itself and by one

Composite number – has more then two devisors.

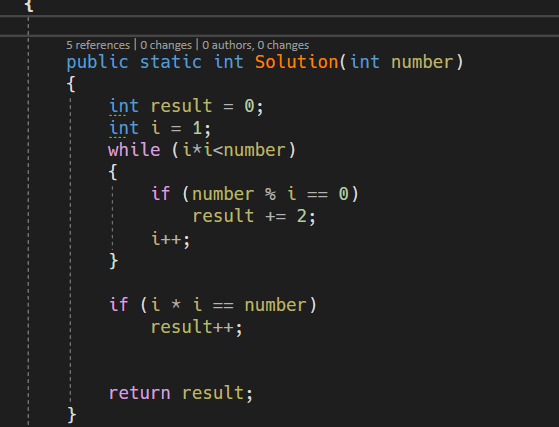


White – prime numbers

Gray – composite numbers.

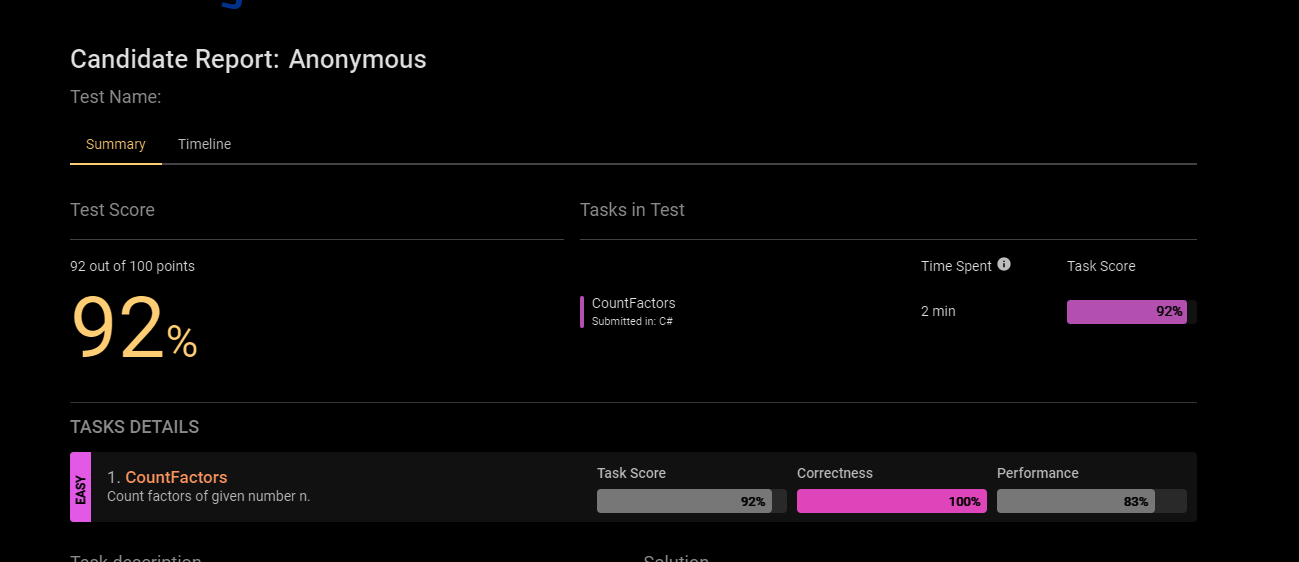


## Algorithm



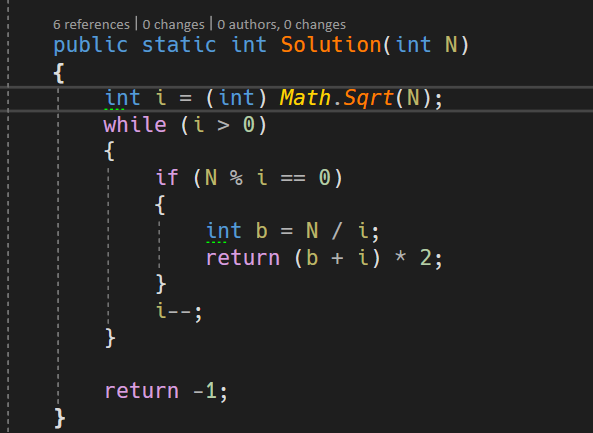
To find all prime numbers till specific number we do this shit. I mean it’s strange but still

## Count factors



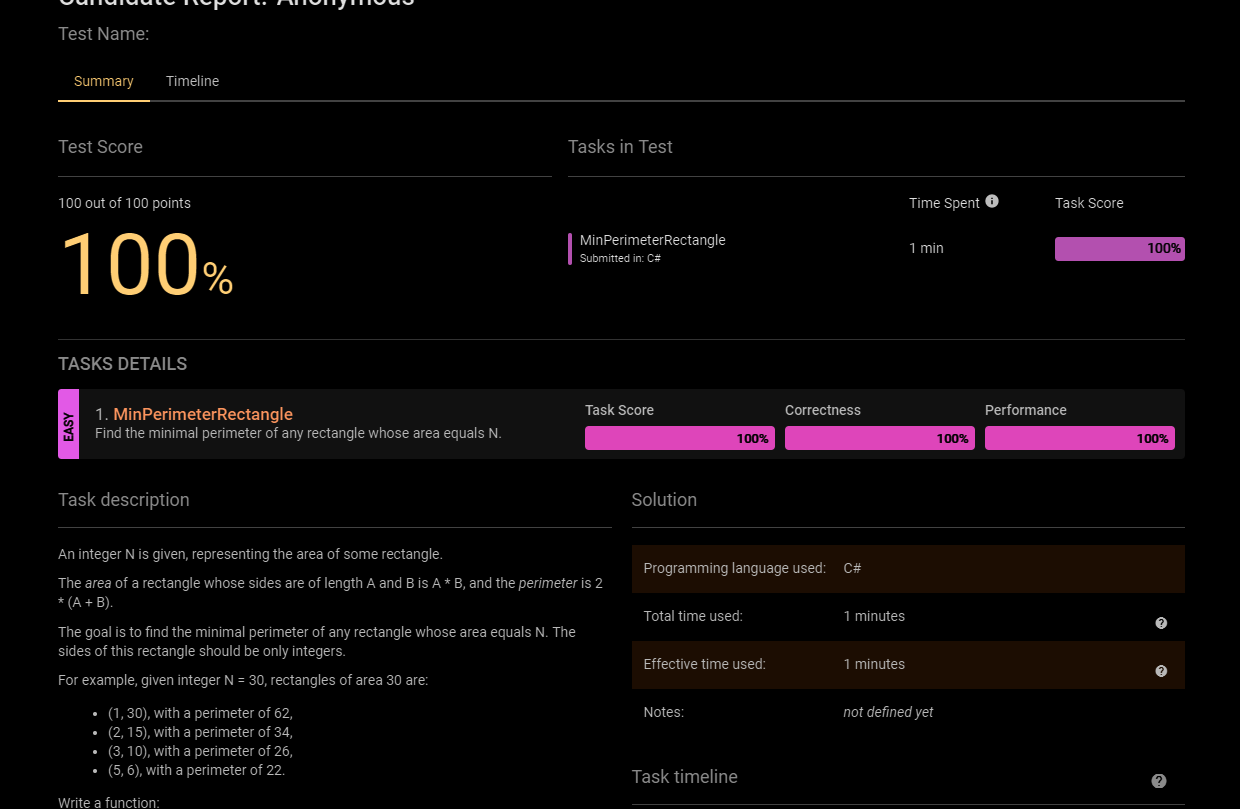
It was just rewriting the algorithm given in the book. And it’s sort of sas that there should be other way of doing that.

## MinPeremeterRectangle



By the given formula I know that at least one of two multipliers should be less then sqrt(N)

And therefore I can start my search there



By that solution I have the best performance garnty in many cases it will be almost linier.

Sieve of Eratosthenes

# Lesson 11: Sieve of Eratosthenes

# Lesson 11.1: Fractolization