

DS-ALGS - Lab 1.1

Centauri Prime - Problem Statement

Alice and Bob are playing the game called Centauri Prime. Centauri Prime represents a planet which is split into several independent kingdoms. Before they start playing the game, they have to choose rulers for each kingdom. Alice and Bob mutually decided to distribute kingdoms based on the letter the kingdom's name ended with. Alice loves vowels and decided to rule kingdoms whose names ended in a vowel. Bob loves consonants and decided to rule kingdoms whose names ended in a consonant. Both of them do not like the letter 'y'(case insensitive) and thus, all kingdoms whose names ended in the letter 'y' are left without a ruler. Can you write a program that will determine the rulers of several kingdoms, given the kingdoms' names?

List of vowels for this problem is ['A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u']

Centauri Prime - Test Sets

Mollaristan

Auritania

Zizily

In Sample Case #1, the name of the kingdom ends with 'n' which is a consonant and thus, it is ruled by Bob.

In Sample Case #2, the name of the kingdom ends with an 'a' which is a vowel and thus, it is ruled by Alice.

In Sample Case #3, the name of the kingdom ends with 'y' which is not liked by both of them and thus, it is ruled by nobody.

Centauri prime - Solution

```
public static String getRuler(String kingdom) {  
    String ruler = "Bob";  
    String[] vowels = {"A", "E", "I", "O", "U", "a", "e", "i", "o", "u"};  
    String[] ys = {"Y", "y"};  
    String lastLetter = kingdom.substring(kingdom.length() - 1);  
  
    for (String v: vowels) {  
        if (v.equals(lastLetter)) {  
            ruler = "Alice";  
            break;  
        }  
    }  
  
    for (String y: ys) {  
        if (y.equals(lastLetter)) {  
            ruler = "nobody";  
            break;  
        }  
    }  
  
    return ruler;  
}
```

1.

By default, ruler is Bob. The reason it is Bob is because the consonants are most numerous.

2. Get the last letter

3. Loop through vowels, if any vowel is last letter, reset the ruler to Alice.

4. Loop through y's. If y is last letter, reset the ruler to nobody.

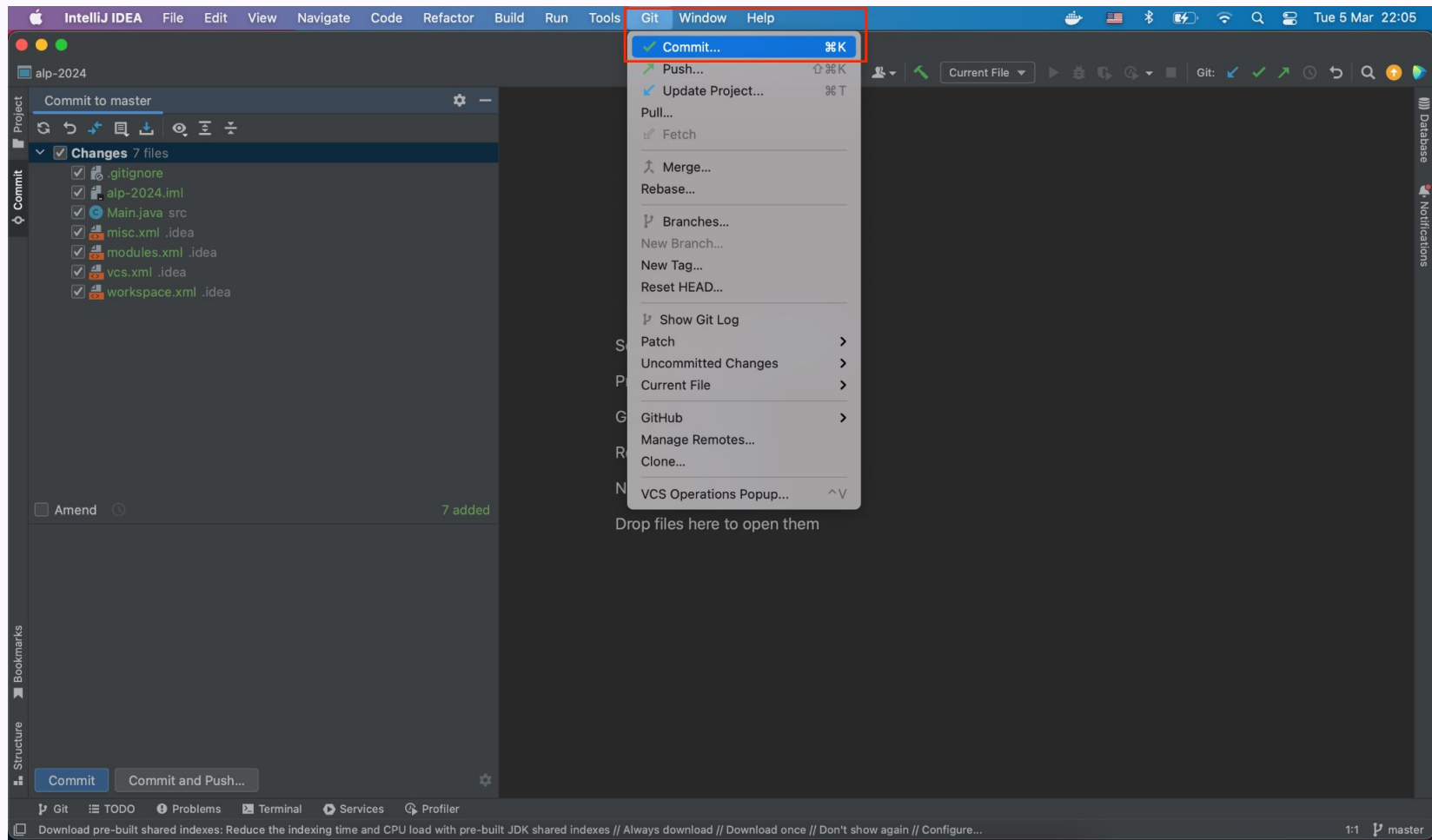
If at 3 and 4 the ruler is not reset, then last letter is a consonant and the ruler remains Bob.

5. Return the ruler

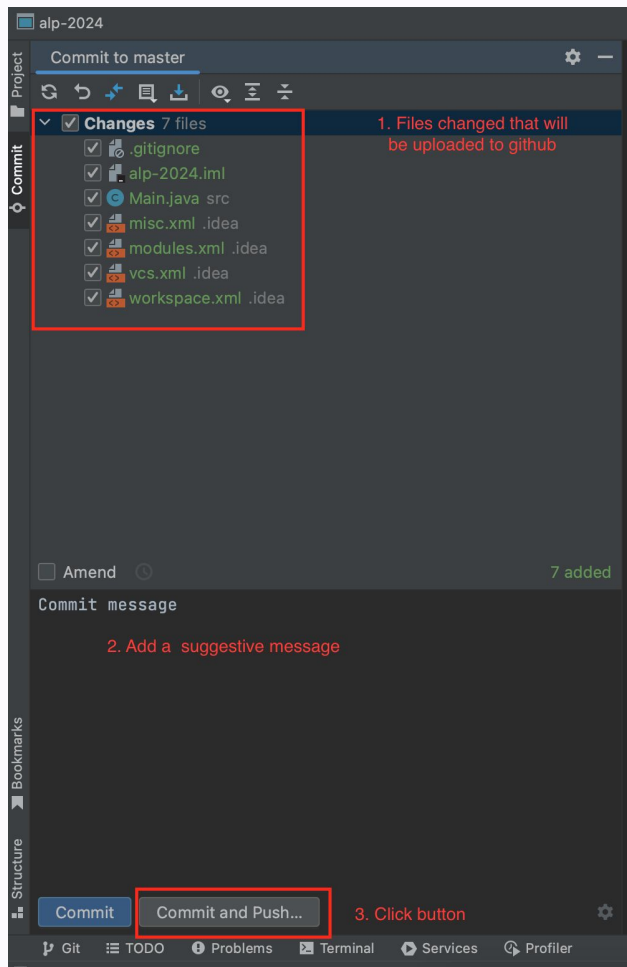
Centauri Prime - Tests

```
public static void main( String[] args) {  
    Mollaristan Auritania Zizily  
  
    String kingdom = "Mollaristan";  
  
    String ruler = getRuler(kingdom);  
  
    System.out.println(kingdom + " is ruled by " + ruler + ".");  
}
```

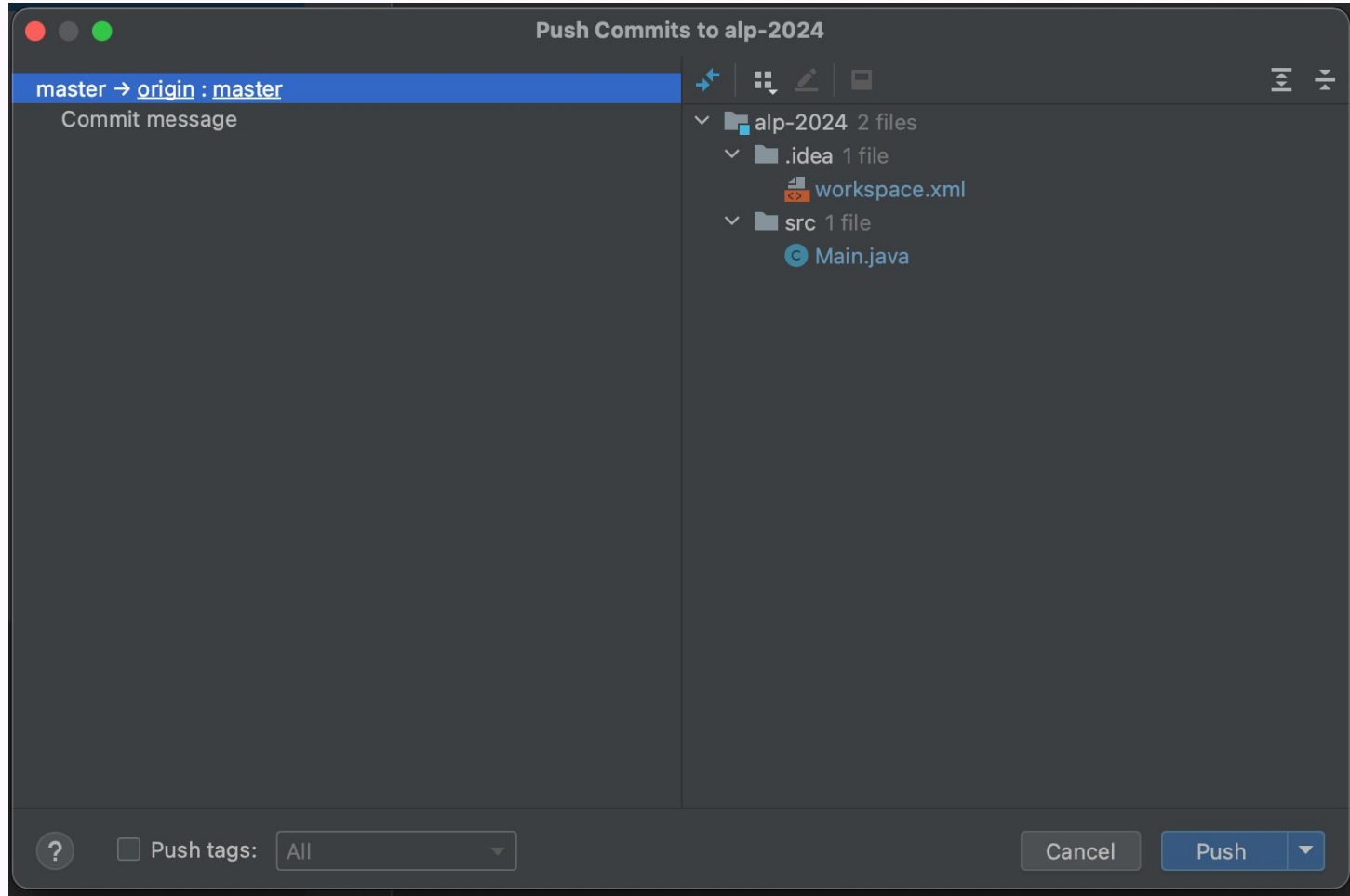
Adding to github




Adding to github



Adding to github




Adding to github


 **alp-2024-copy**


Private

Unwatch 1

 master

Our branch

 1 Branch


 0 Tags

Go to file

t

Add file





<> Code

 **OwlHowl3** Commit message

b50f95e · 6 minutes ago

Our commit that was pushed

🕒 2 Commits

 .idea	Commit message	6 minutes ago
 src	Commit message	6 minutes ago
 .gitignore	Commit message	8 minutes ago
 alp-2024.iml	Commit message	8 minutes ago

Allocation - Problem Statement

There are N houses for sale. The i -th house costs A_i dollars to buy. You have a budget of B dollars to spend.

What is the maximum number of houses you can buy?

Allocation - Test Set

4 100

20 90 40 90

4 50

30 30 10 10

3 300

999 999 999

Allocation - Test Set

In Sample Case #1, you have a budget of 100 dollars. You can buy the 1st and 3rd houses for $20 + 40 = 60$ dollars.

In Sample Case #2, you have a budget of 50 dollars. You can buy the 1st, 3rd and 4th houses for $30 + 10 + 10 = 50$ dollars.

In Sample Case #3, you have a budget of 300 dollars. You cannot buy any houses (so the answer is 0).

Allocation - Visuals



100



20



90

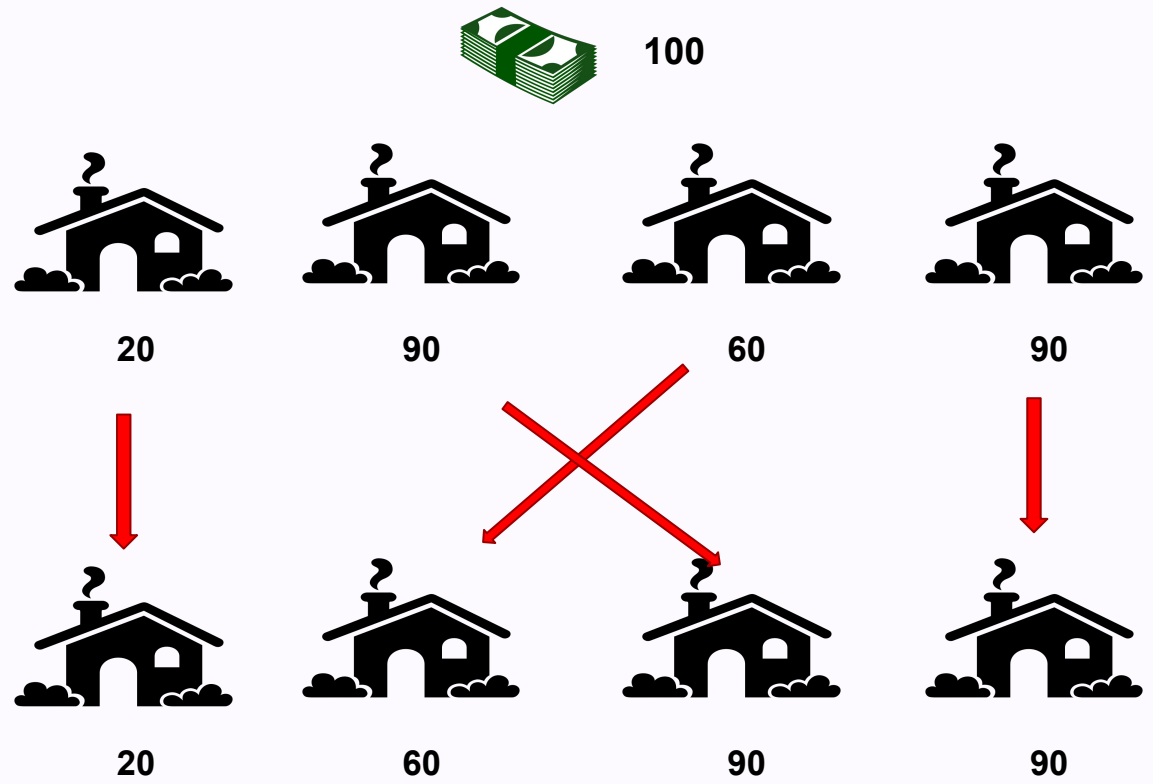


60

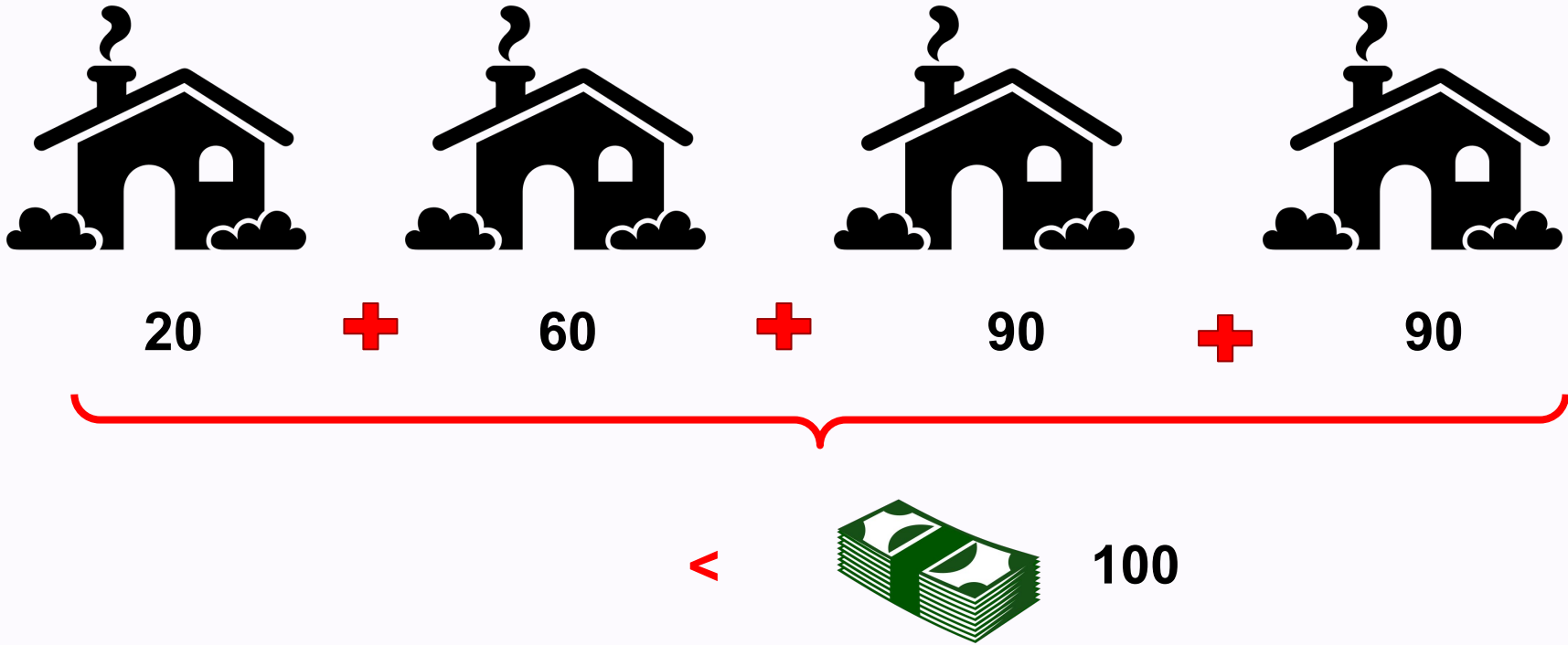


90

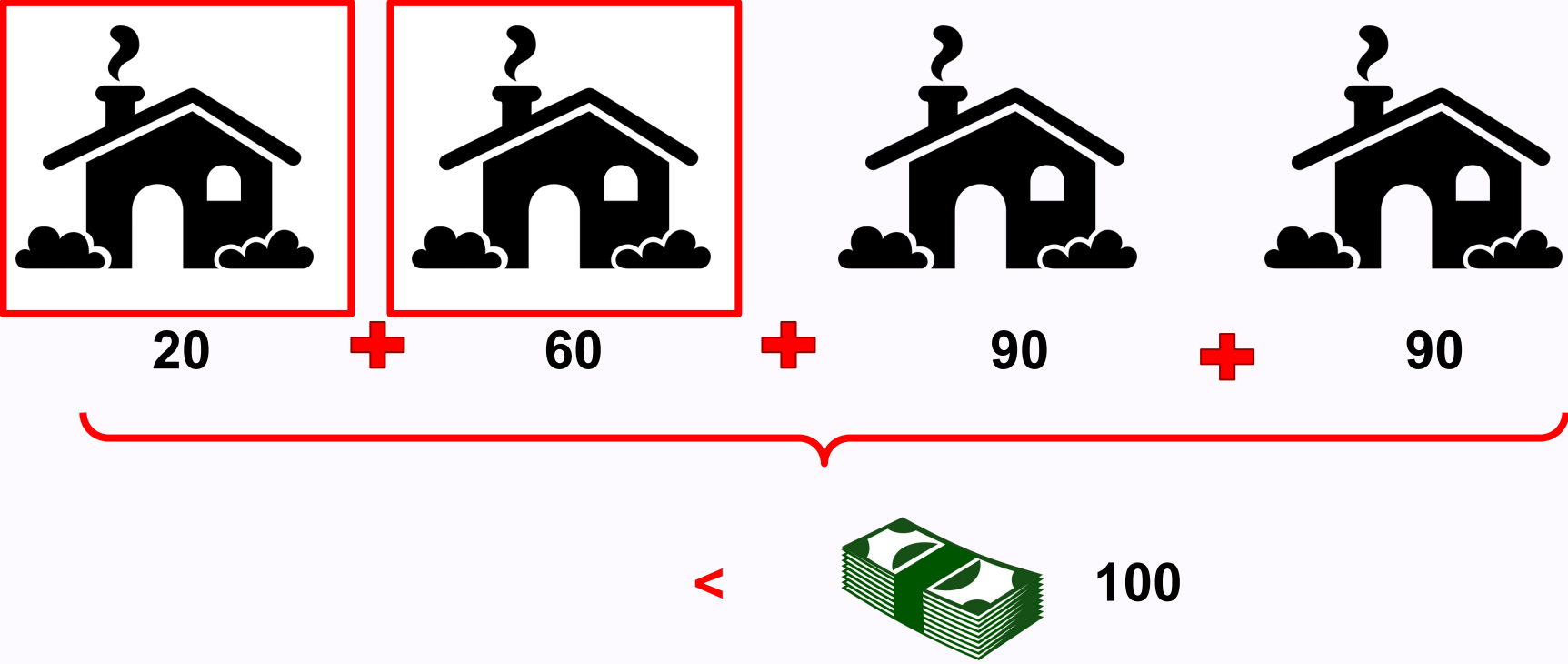
Allocation - Visuals



Allocation - Visuals



Allocation - Visuals



Allocation - Solution

```
public static int getMaxNumberOfHouses(int budget, int[] prices) {  
    Arrays.sort(prices);  
  
    int countHouses = 0;  
    int totalSpent = 0;  
  
    for (int price: prices) {  
        totalSpent += price;  
  
        if (totalSpent <= budget) {  
            countHouses +=1;  
        } else {  
            break;  
        }  
    }  
  
    return countHouses;  
}
```

1. Sort the prices asc

2. Loop through the prices

3. add to the total spent

4. If we are within the budget,
increment the number of houses.
Otherwise - exit the loop

5. return the solution

Allocation - Tests

```
public static void main( String args[] ) {  
    int budget = 100;  
  
    int[] prices = {20, 90, 60, 90};  
  
    int maxNumber = getMaxNumberOfHouses(budget, prices);  
  
    System.out.println("You can buy a maximum of " + maxNumber + " houses.");  
}
```

Doors and Keys - Problem Statement

The knight is standing in front of a long and narrow hallway. A princess is waiting at the end of it.

In a hallway there are three doors: a red door, a green door and a blue door. The doors are placed one after another, however, possibly in a different order. To proceed to the next door, the knight must first open the door before.

Each door can be only opened with a key of the corresponding color. So three keys: a red key, a green key and a blue key — are also placed somewhere in the hallway. To open the door, the knight should first pick up the key of its color.

The knight has a map of the hallway. It can be transcribed as a string, consisting of six characters:

R, G, B — denoting red, green and blue doors, respectively;

r, g, b — denoting red, green and blue keys, respectively.

Each of these six characters appears in the string exactly once.

The knight is standing at the beginning of the hallway — on the left on the map.

Given a map of the hallway, determine if the knight can open all doors and meet the princess at the end of the hallway.

Doors and Keys - Test Set

rgbBRG

RgbrBG

bBrRgG

rgRGBb

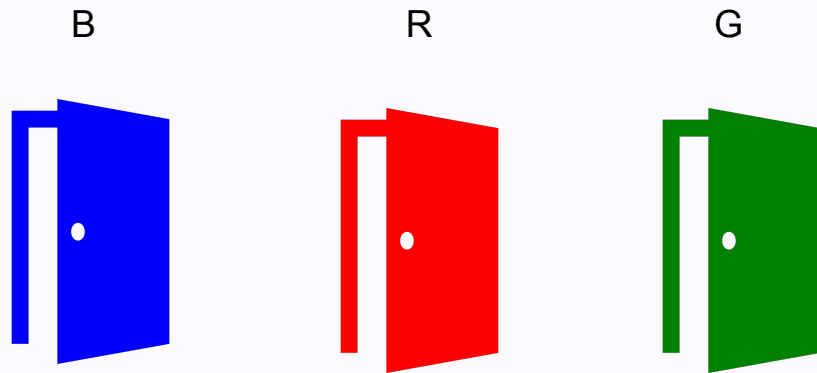
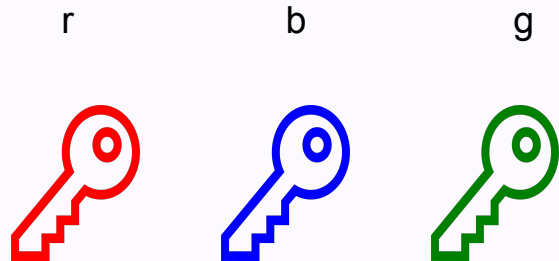
In the first testcase, the knight first collects all keys, then opens all doors with them.

In the second testcase, there is a red door right in front of the knight, but he doesn't have a key for it.

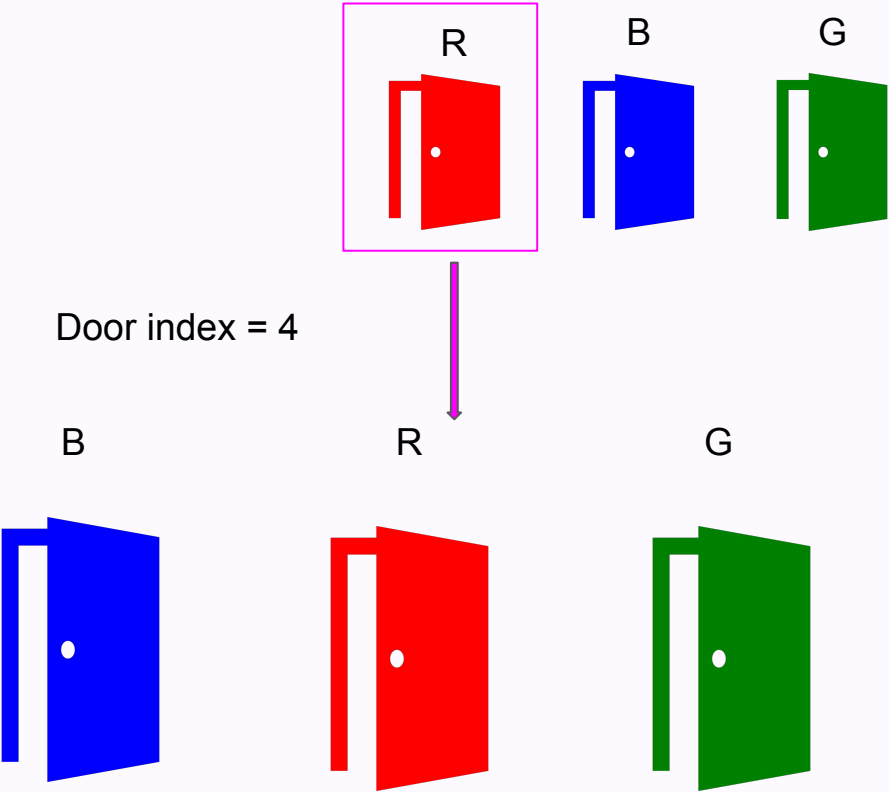
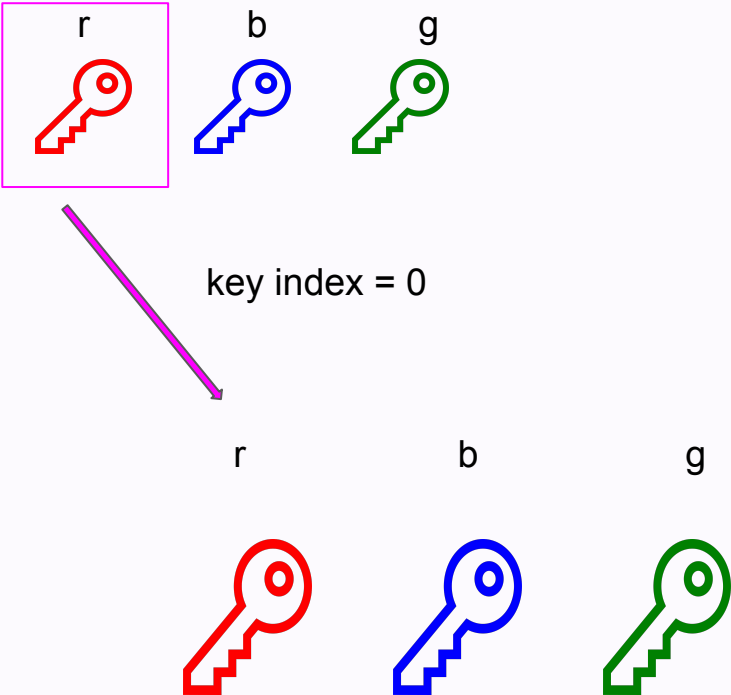
In the third testcase, the key to each door is in front of each respective door, so the knight collects the key and uses it immediately three times.

In the fourth testcase, the knight can't open the blue door.

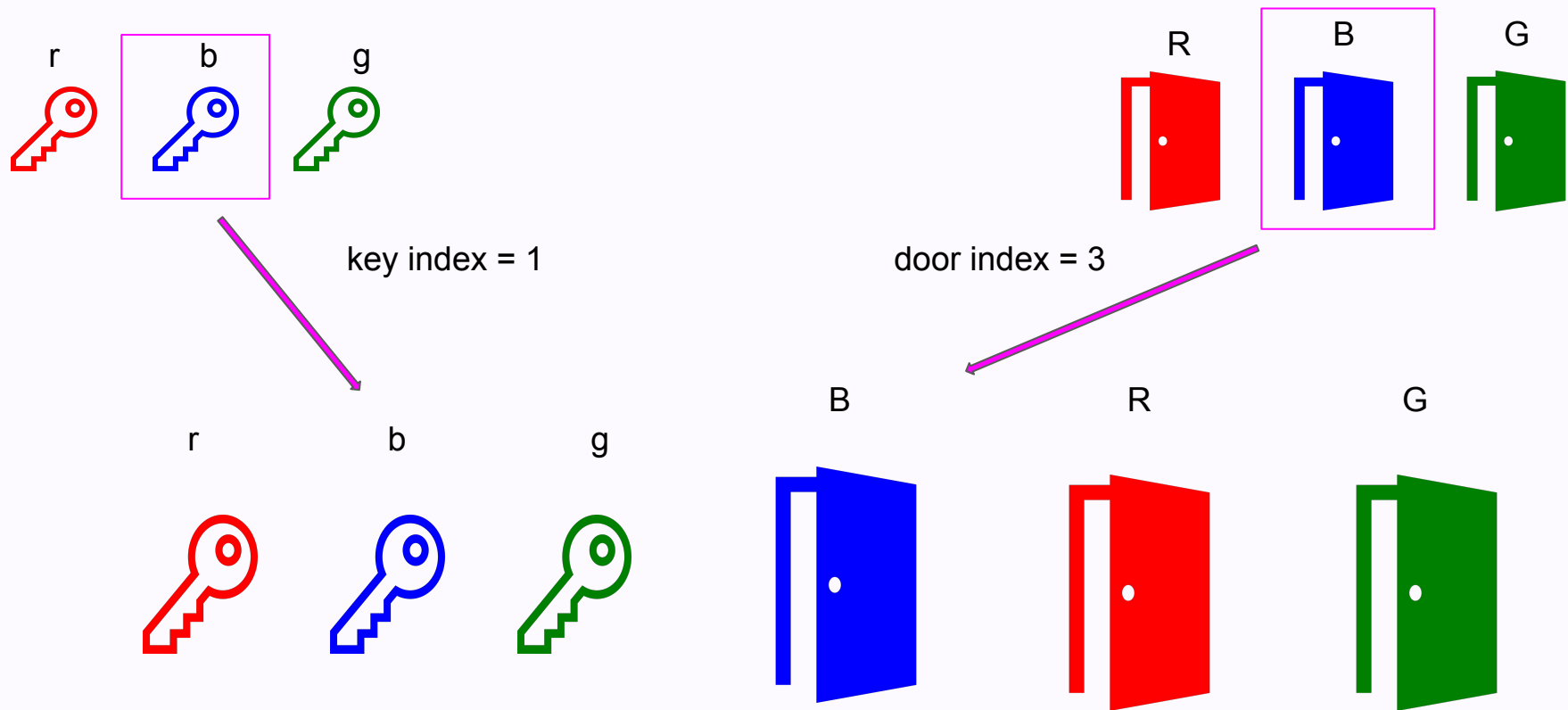
Doors and Keys - Visuals



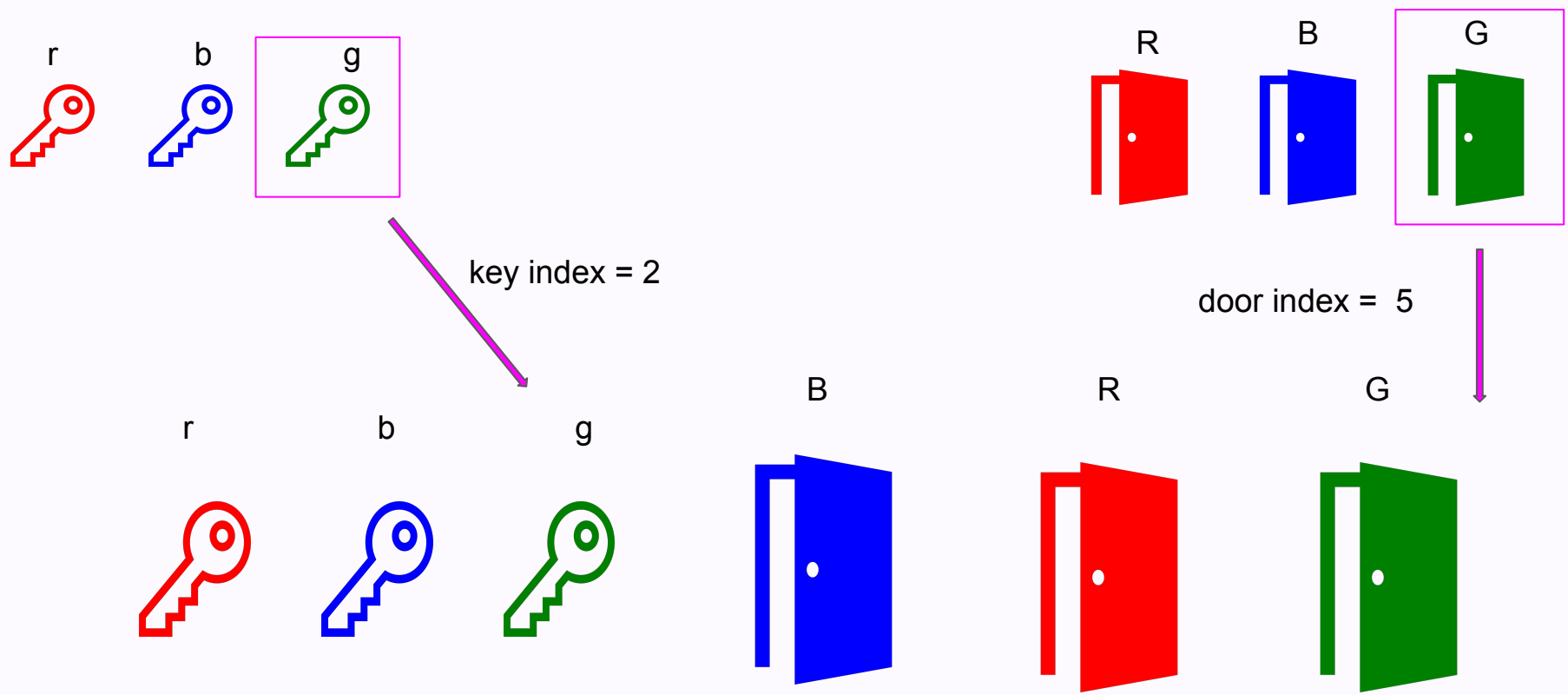
Doors and Keys - Visuals



Doors and Keys - Visuals



Doors and Keys - Visuals



Doors and Keys - Solution

```
public static void main(String args[]) {  
    // rgbBRG  
    // RgbrBG  
    // bBrRgG  
    // rgRGBb  
    String testCase = "rgbBRG";  
    |  
    String[] doors = {"R", "G", "B"};  
    String[] keys = {"r", "g", "b"};  
  
    boolean canGo = true;  
    for (int i = 0; i < doors.length; i++) {  
        int doorPosition = testCase.indexOf(doors[i]);  
        int keyPosition = testCase.indexOf(keys[i]);  
  
        if (doorPosition < keyPosition) {  
            canGo = false;  
            break;  
        }  
    }  
  
    if (canGo) {  
        System.out.println("YES");  
    } else {  
        System.out.println("NO");  
    }  
}
```

Angry Professor - Problem Statement

A professor has a class of students. Frustrated with their lack of discipline, the professor decides to cancel class if fewer than some number of students are present when class starts. Arrival times go from on time ($\text{arrivalTime} \leq 0$) to arrived late ($\text{arrivalTime} > 0$).

Given the arrival time of each student and a threshold number of attendees, determine if the class is cancelled.

Angry Professor - Test Set

4 3

-1 -3 4 2

4 2

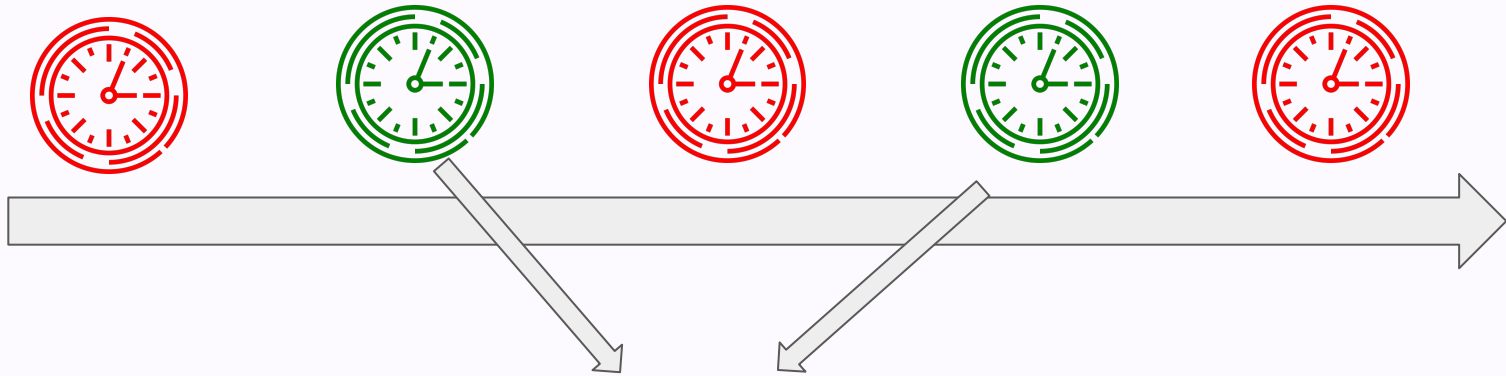
0 -1 2 1

For the first test case, the professor wants at least 3 students in attendance, but only 2 have arrived on time (-3 and -1) so the class is cancelled.

For the second test case, the professor wants at least 2 students in attendance, and there are 2 who arrived on time (0 and -1). The class is not cancelled.

Angry Professor

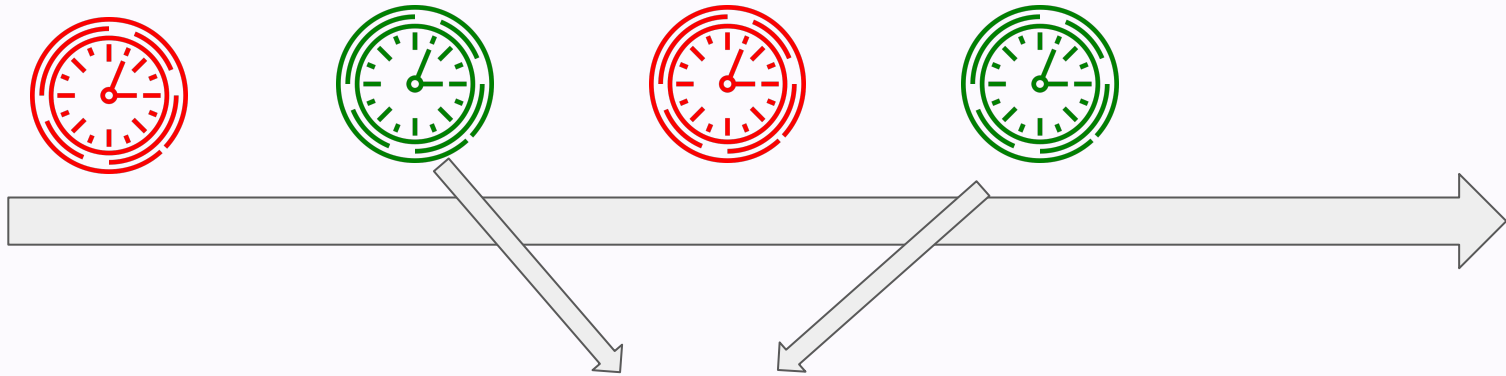
Should be present min = 3 students



$2 < \text{threshold } 3 \Rightarrow \text{class is canceled}$

Angry Professor

Should be present min = 2 students



$2 \geq \text{threshold } 2 \Rightarrow \text{class is NOT canceled}$

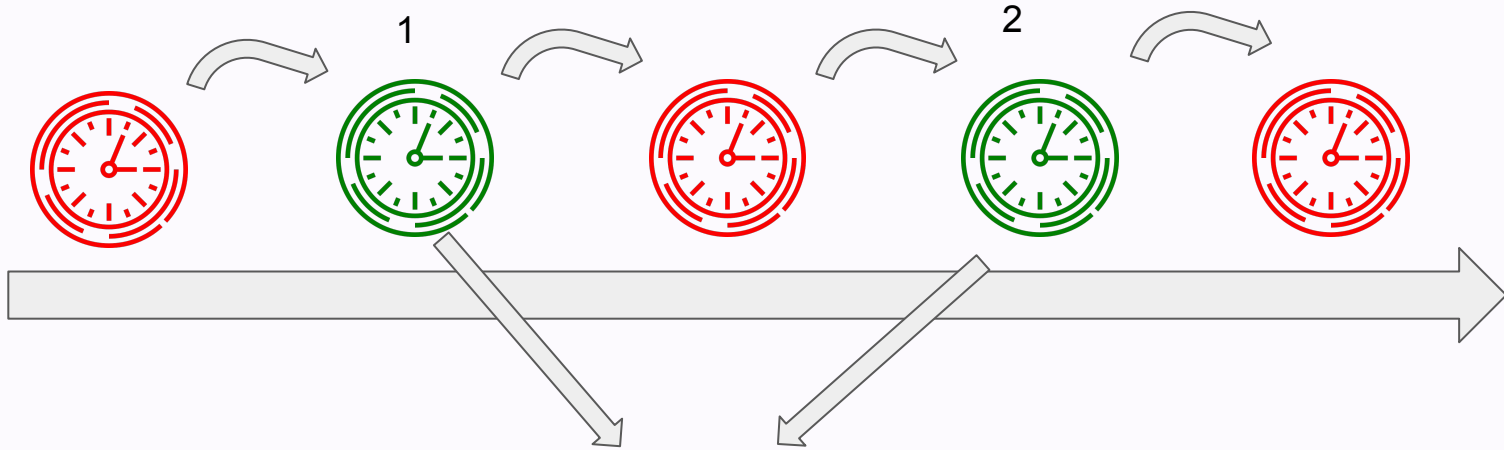
Solutions

1. Iterate over and count, then compare to threshold
2. Filter on time students, then compare the count of those to threshold
3. Sort the times, then check the item at index = threshold - 1, compare it to 0.
If this item is >0 , class is canceled, otherwise not.

Angry Professor

Should be present min = 3 students

1

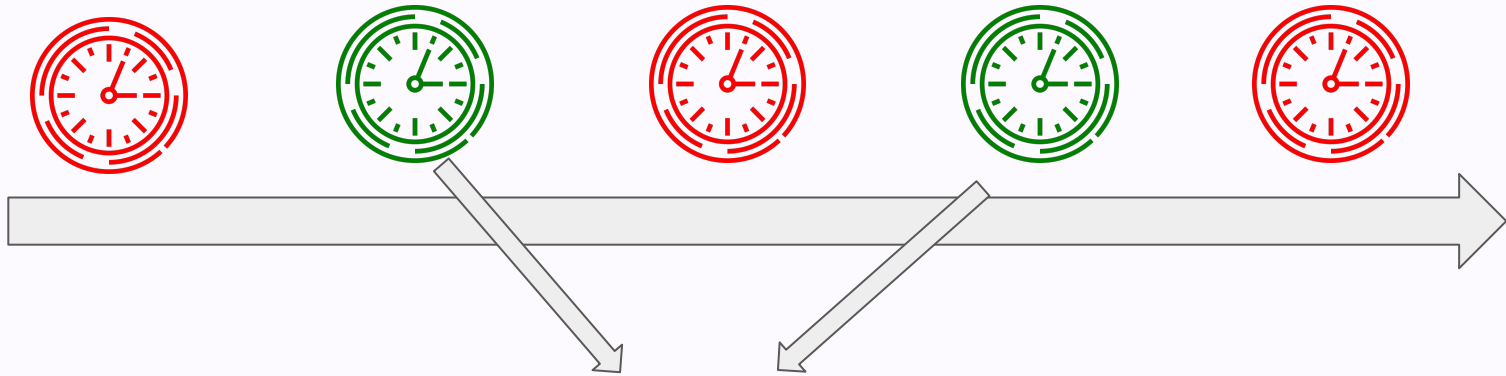


$2 < \text{threshold } 3 \Rightarrow \text{class is canceled}$

Angry Professor

Should be present min = 3 students

2

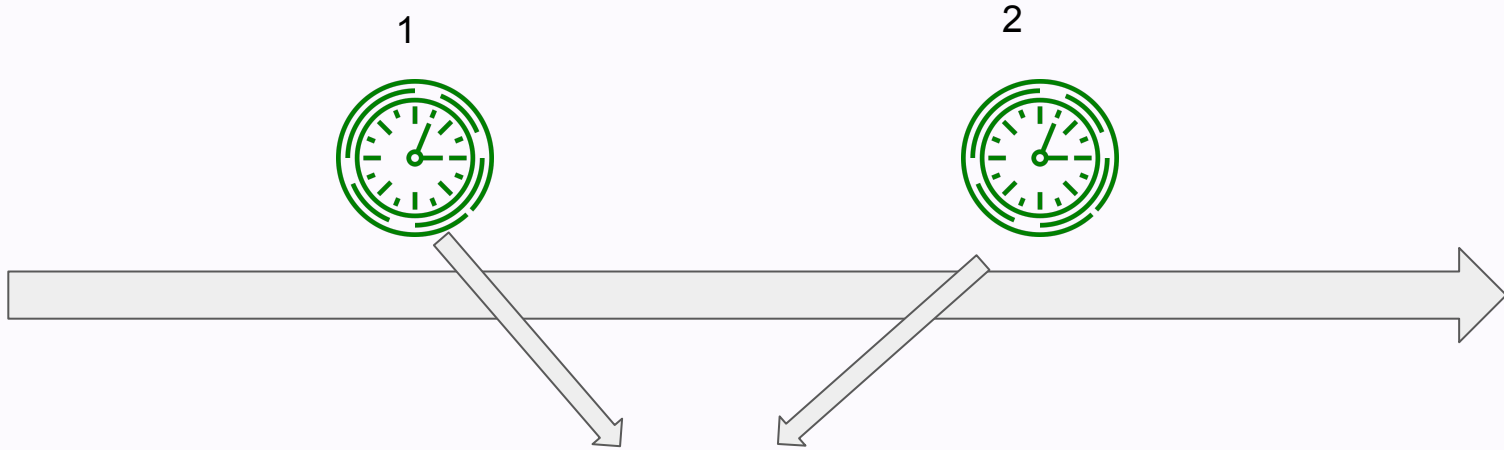


$2 < \text{threshold } 3 \Rightarrow \text{class is canceled}$

Angry Professor

Should be present min = 3 students

2

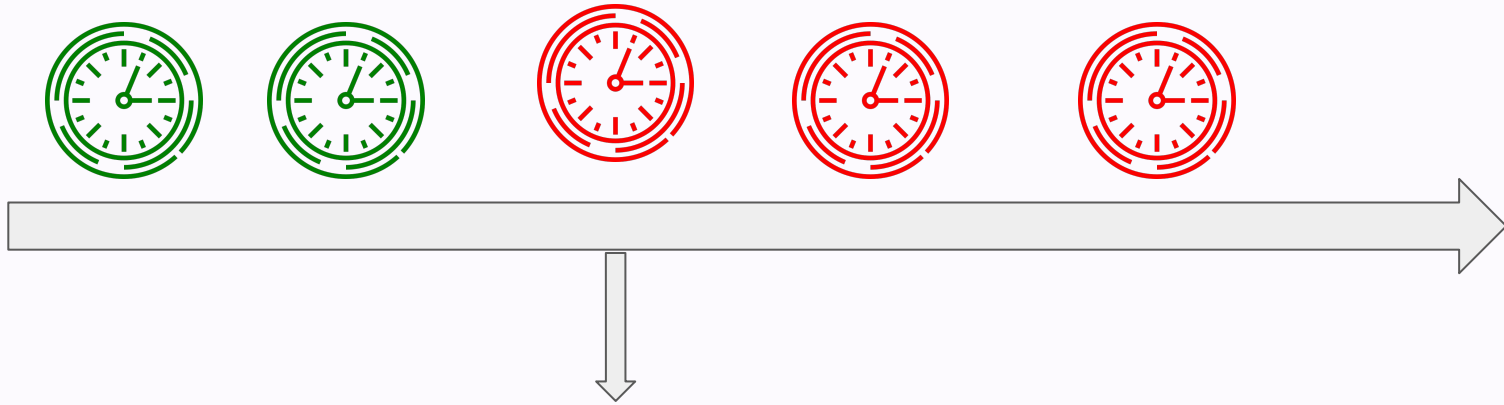


$2 < \text{threshold } 3 \Rightarrow \text{class is canceled}$

Angry Professor

Should be present min = 3 students

3



Item at index threshold - 1 is late, so
the class is canceled

Pair with given sum - Problem Statement & Test Set

Given an unsorted integer array, find a pair with the given sum in it.

nums = [8, 7, 2, 5, 3, 1]

target = 10

Output: Pair found (8, 2) or Pair found (7, 3)

nums = [5, 2, 6, 8, 1, 9]

target = 12

Output: Pair not found

Max product of two integers in an array

Given an integer array, find the maximum product of two integers in it.

For example, consider array $\{-10, -3, 5, 6, -2\}$. The maximum product is the $(-10, -3)$ or $(5, 6)$ pair.

Not Shading - Problem Statement

There is a grid with n rows and m columns. Some cells are colored black, and the rest of the cells are colored white.

In one operation, you can select some black cell and do exactly one of the following:

- color all cells in its row black, or

- color all cells in its column black.

You are given two integers r and c . Find the minimum number of operations required to make the cell in row r and column c black, or determine that it is impossible.

Not Shading - Test Set

The first line of each test case contains four integers n , m , r , and c ($1 \leq n, m \leq 50$; $1 \leq r \leq n$; $1 \leq c \leq m$) — the number of rows and the number of columns in the grid, and the row and column of the cell you need to turn black, respectively.

Then n lines follow, each containing m characters. Each of these characters is either 'B' or 'W' — a black and a white cell, respectively.

Not Shading - Test Set

3 5 1 4

W B W W W

B B B W B

W W B B B

Output => 1

4 3 2 1

B W W

B B W

W B B

W W B

Output => 0

Not Shading - Test Set

2 3 2 2

WWW

WWW

Output => -1

2 2 1 1

WW

WB

Output => 2

Not Shading - Test Set

1 1 1 1

B

Output => 0

1 1 1 1

W

Output => -1

Not Shading - Test Set

1 2 1 1

WB

Output=> 1

2 1 1 1

W

B

Output => 1

Not Shading - Test Set

5 9 5 9

WWWWWWWWWWW

WBWBWBBBW

WBBBWWBWW

WBWBWBBBW

WWWWWWWWWWW

Output => 2

Not Shading

W	W	W	W	W
W	W	W	W	W
W	W	W	W	W

Result = 1

Not Shading

B	W	W
B	B	W
W	B	B
W	W	B

Result = 0

Not Shading

W	W
W	B

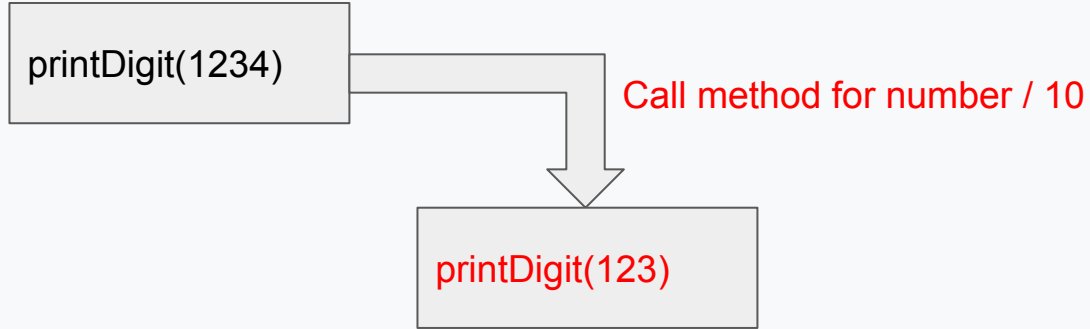
Result = 2

DS-ALGS - Lab 2

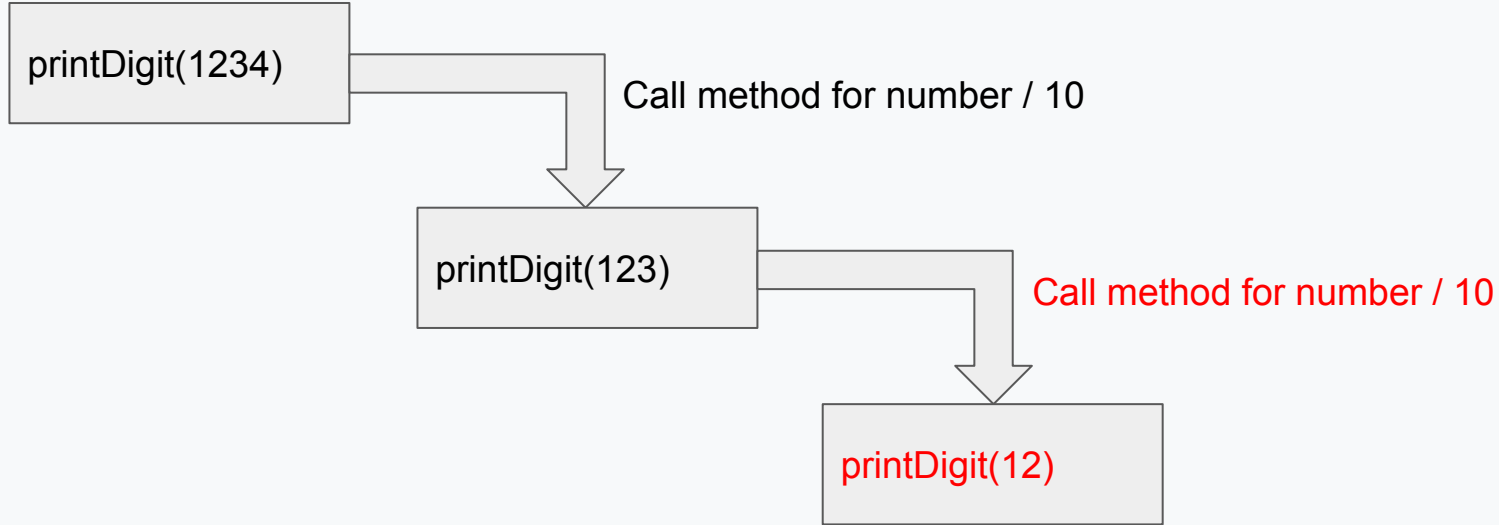
1. Print all digits of a number

```
printDigit(1234)
```

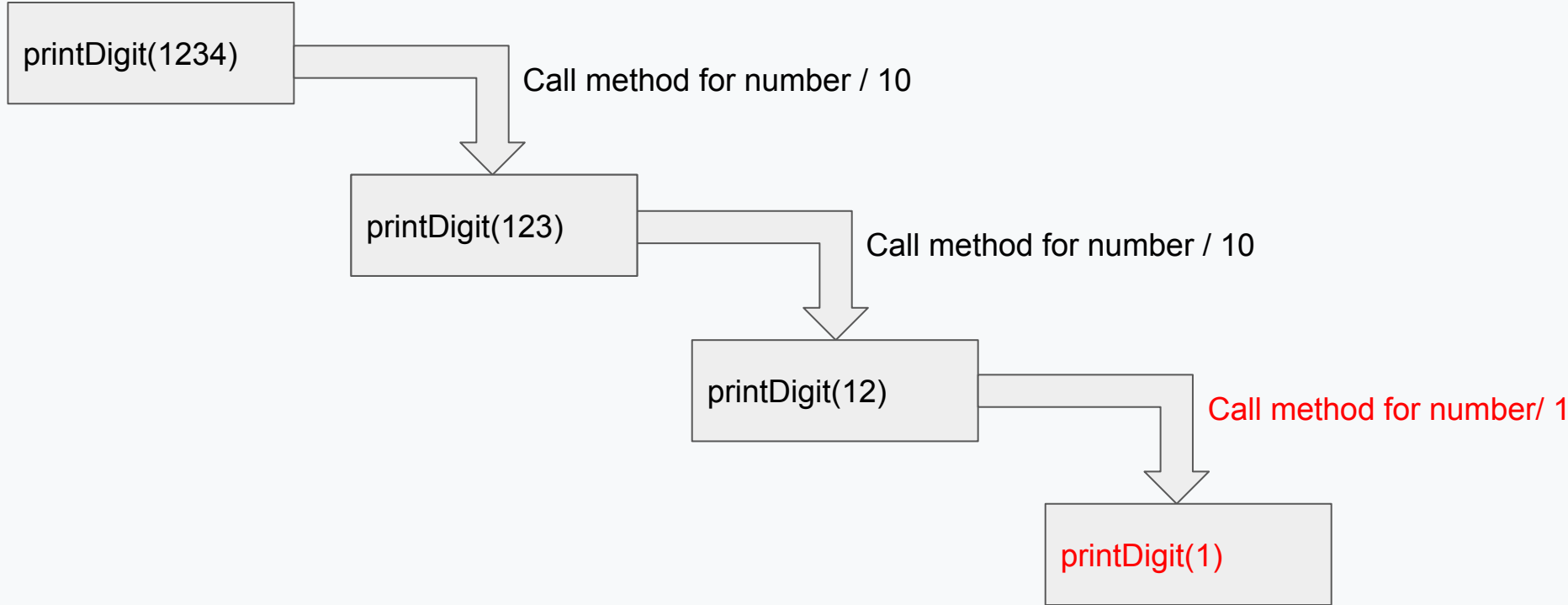

1. Print all digits of a number



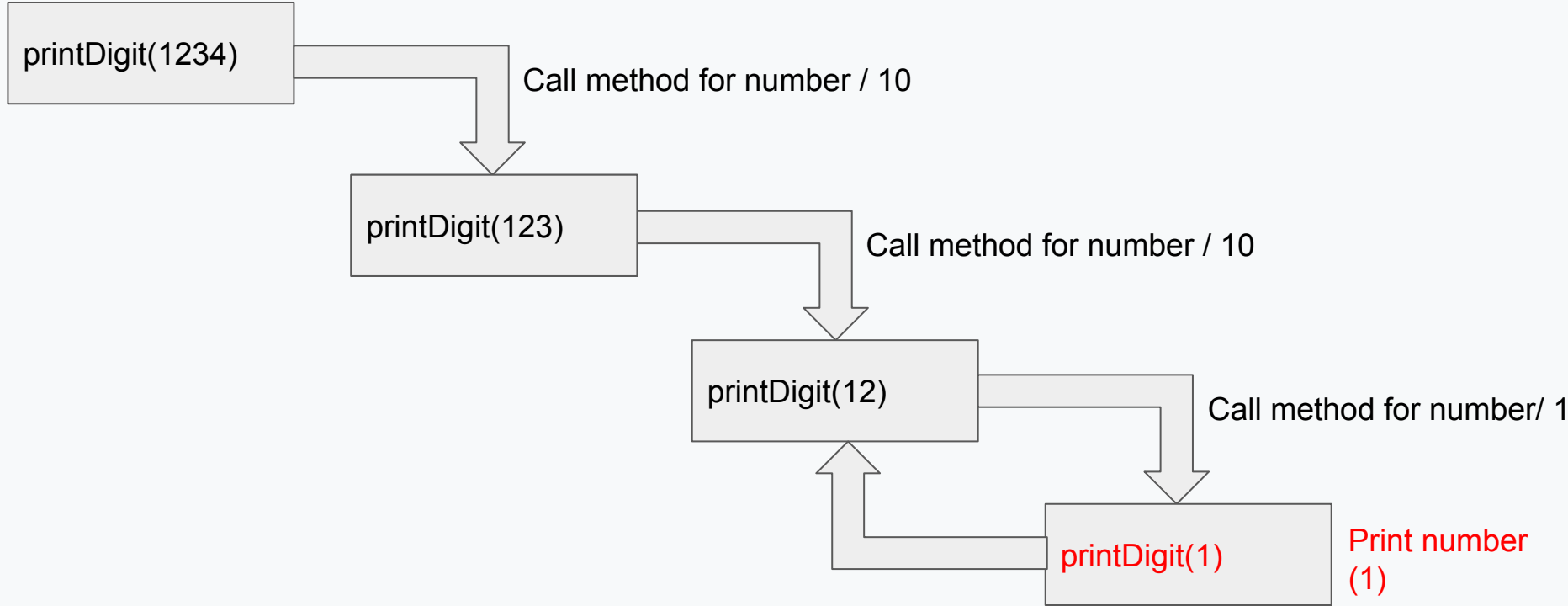
1. Print all digits of a number



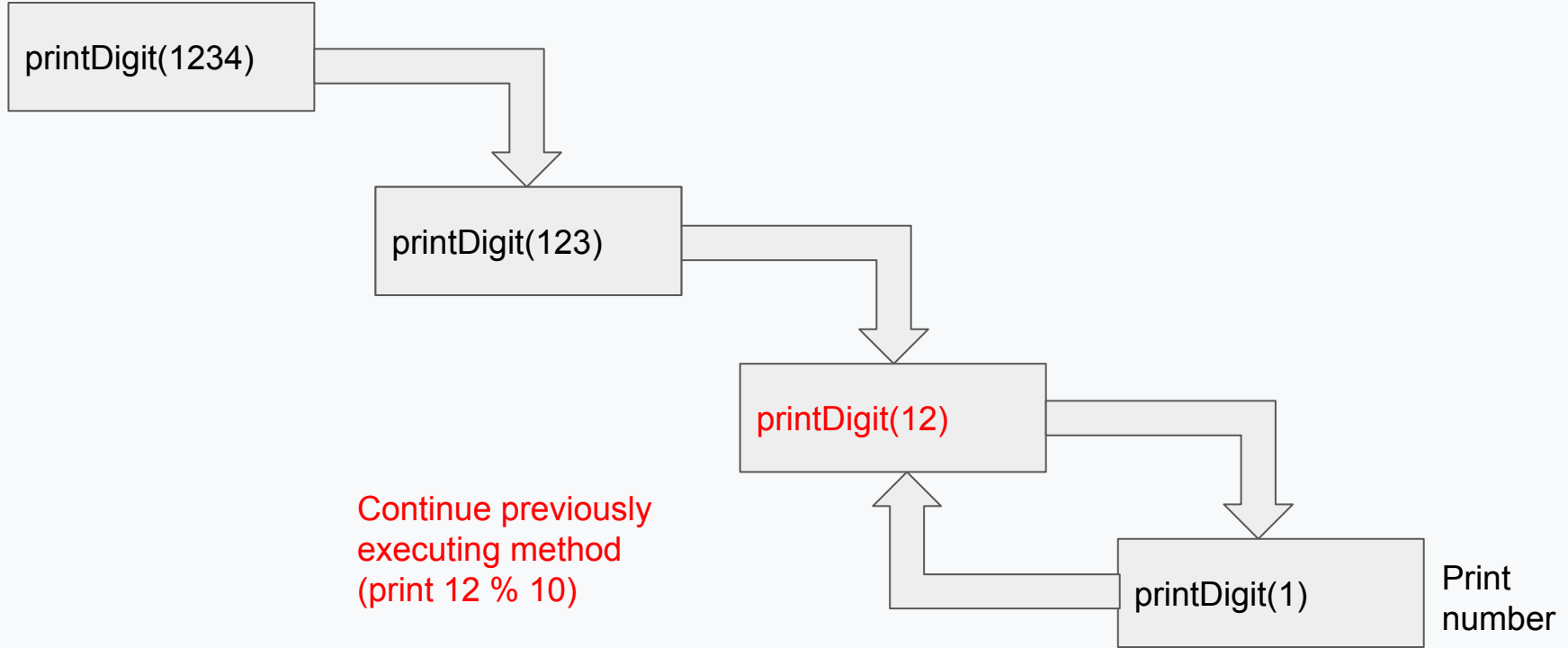
1. Print all digits of a number



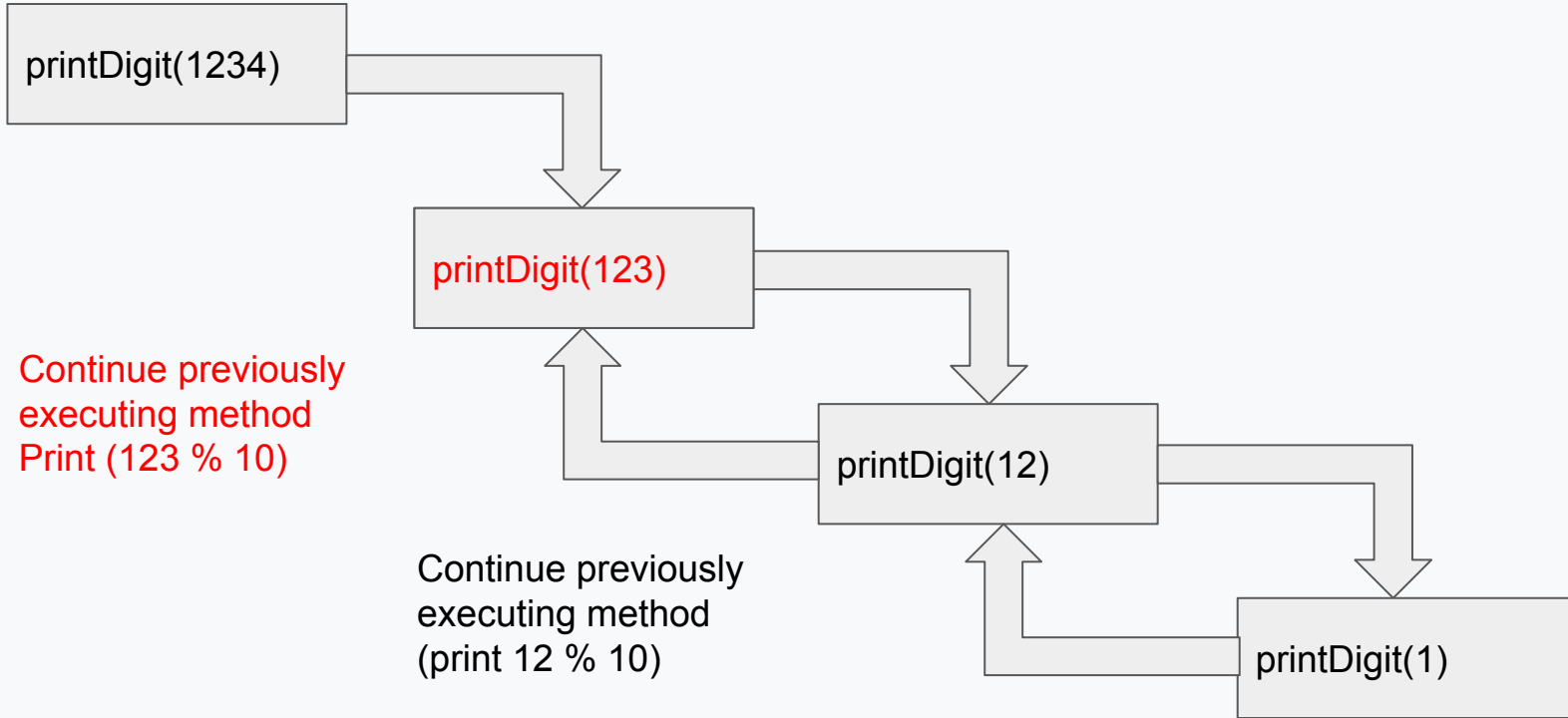
1. Print all digits of a number



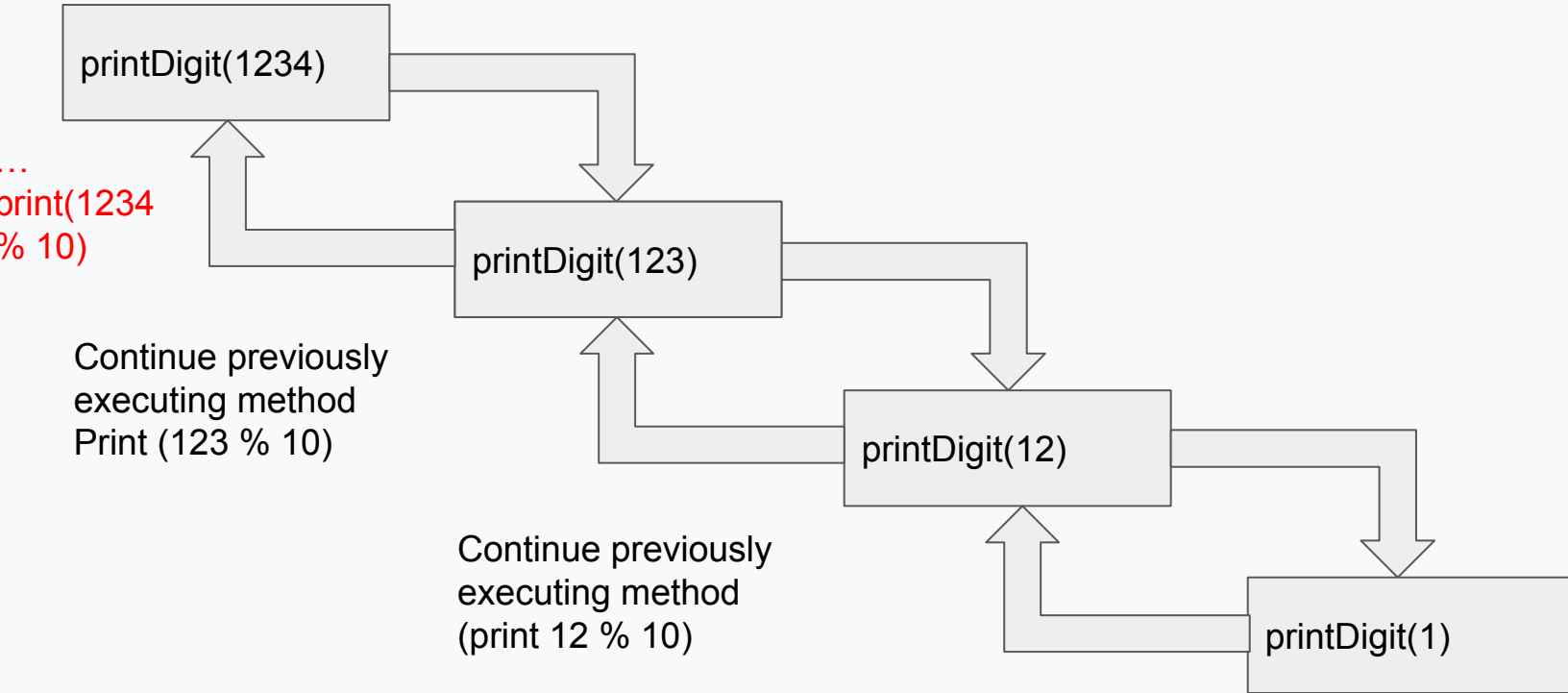
1. Print all digits of a number



1. Print all digits of a number



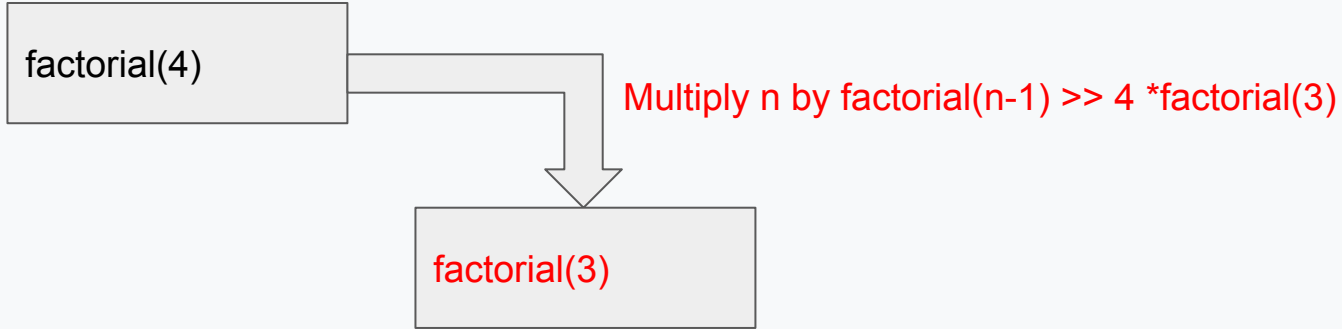
1. Print all digits of a number



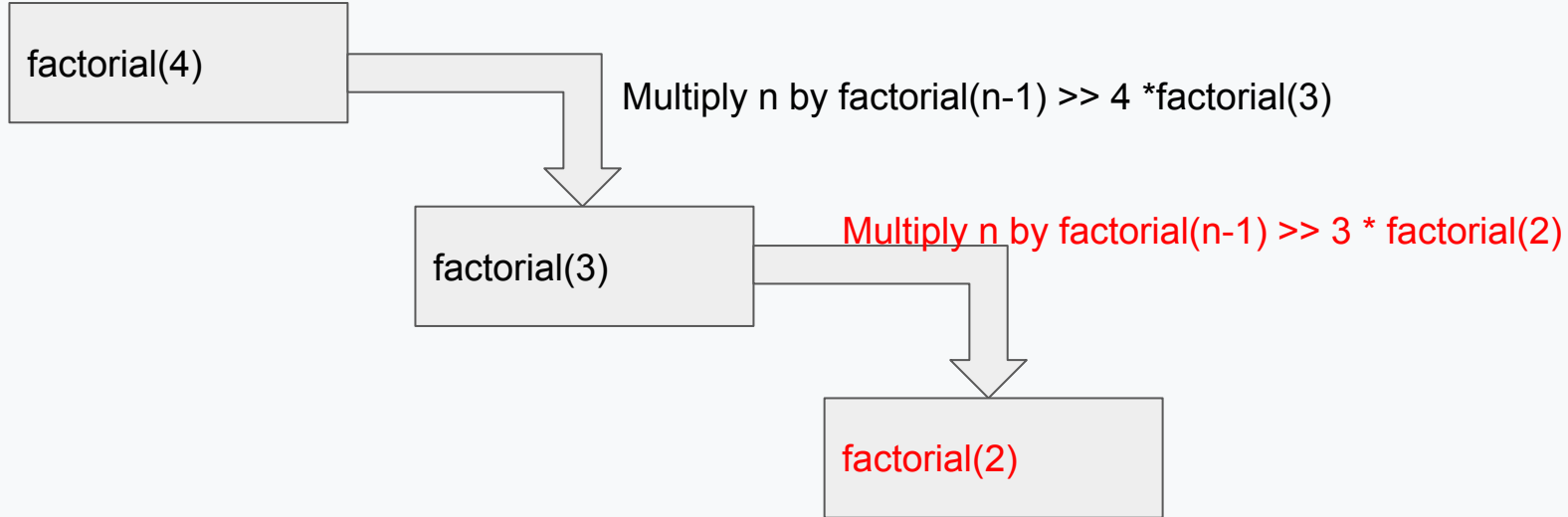
2. Factorial

```
factorial(4)
```

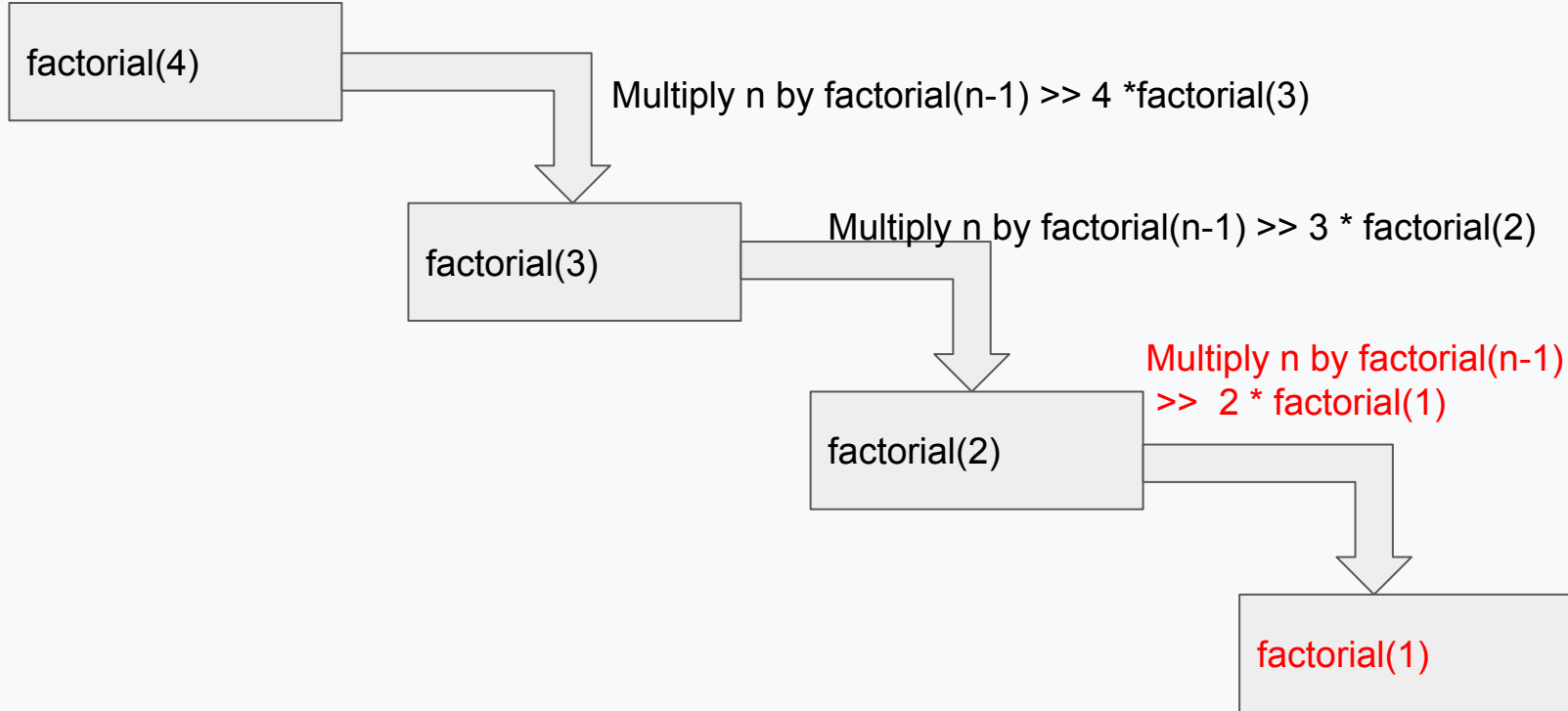

2. Factorial



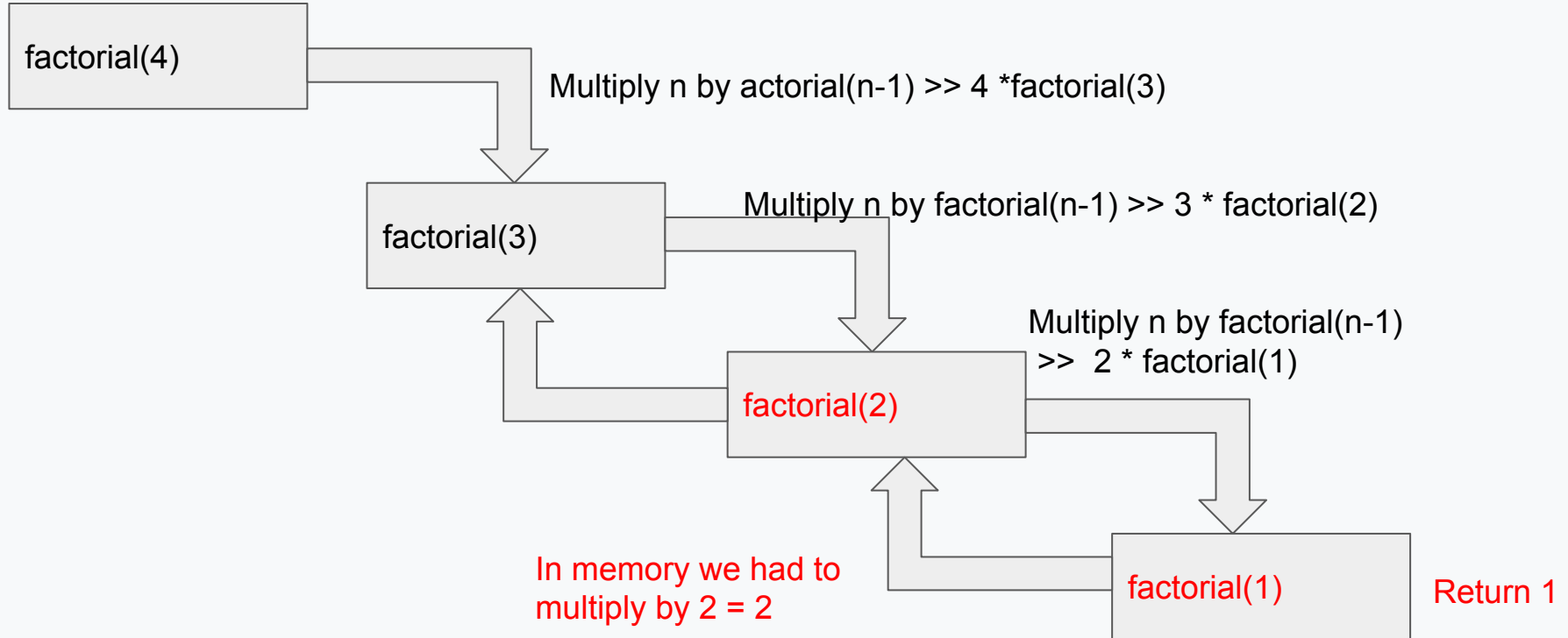
2. Factorial



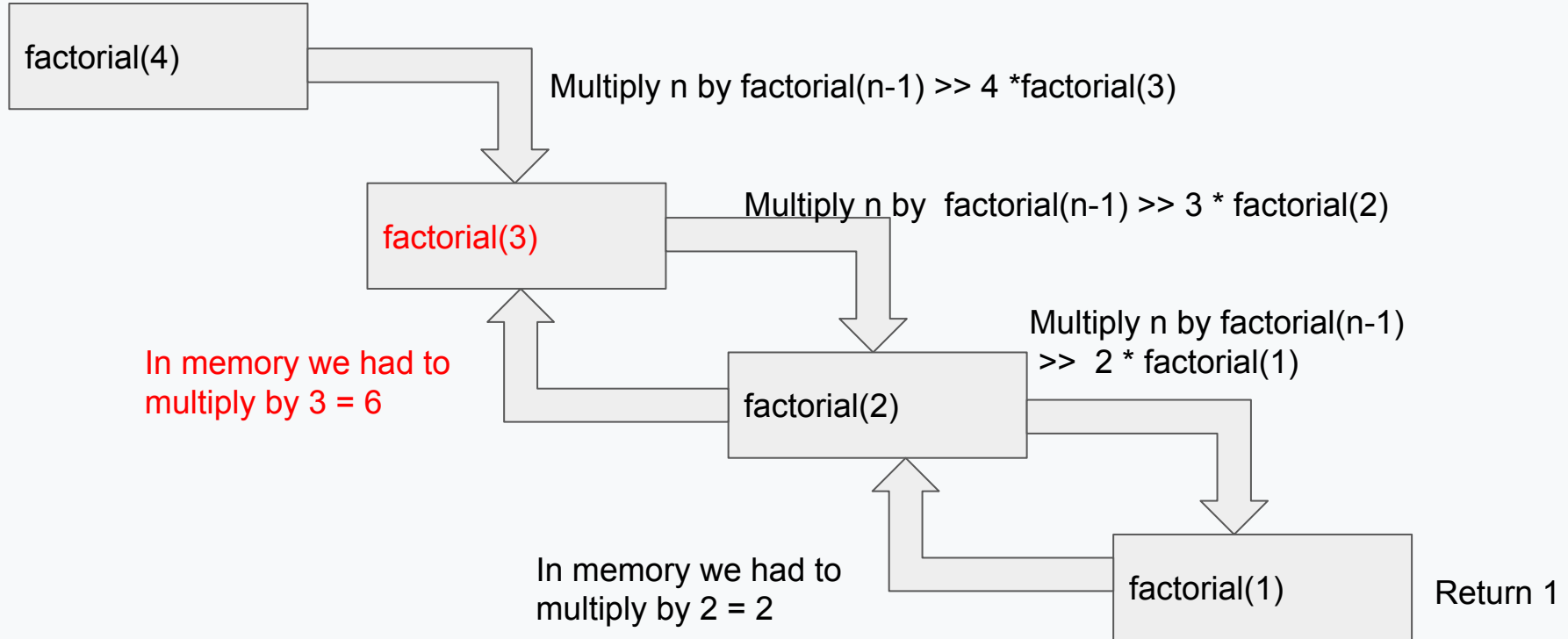
2. Factorial



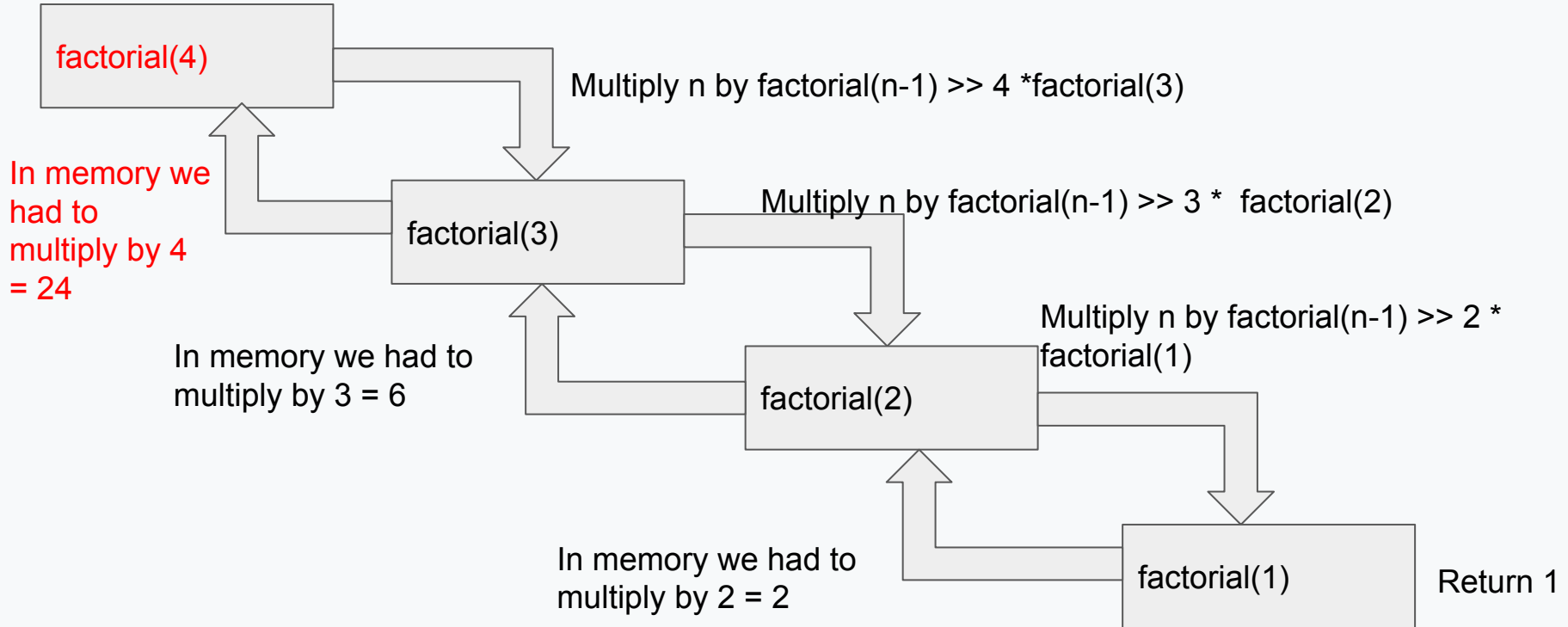
2. Factorial



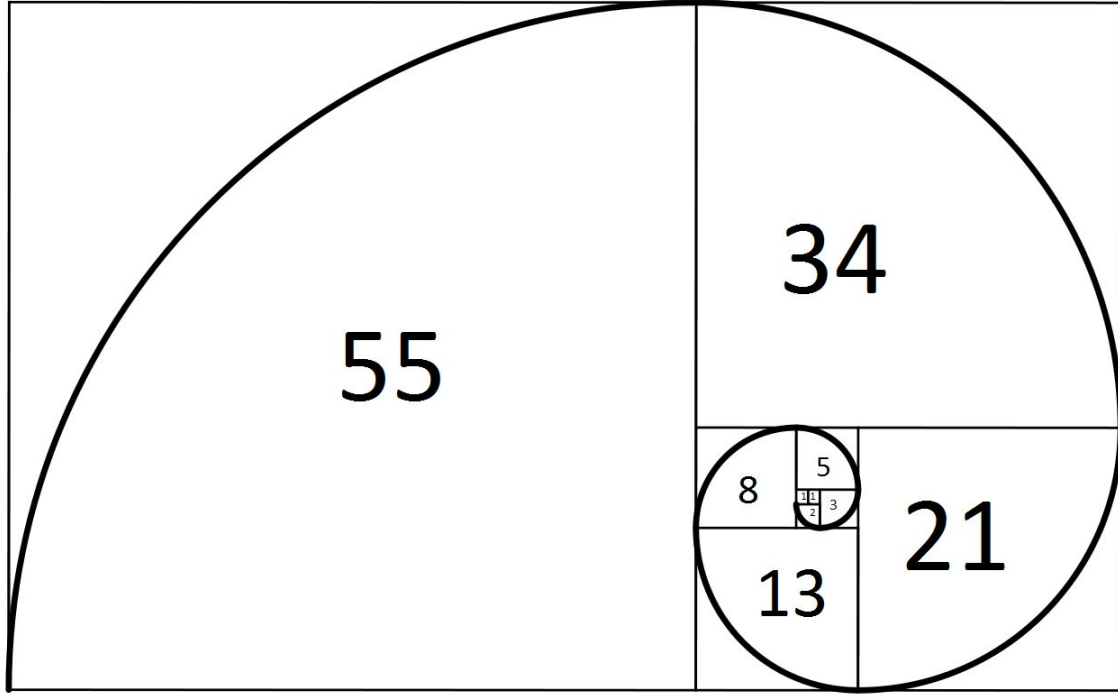
2. Factorial



2. Factorial



3. Fibonacci Sequence

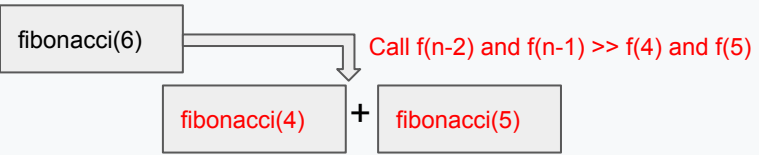


[Fibonacci in nature](#)

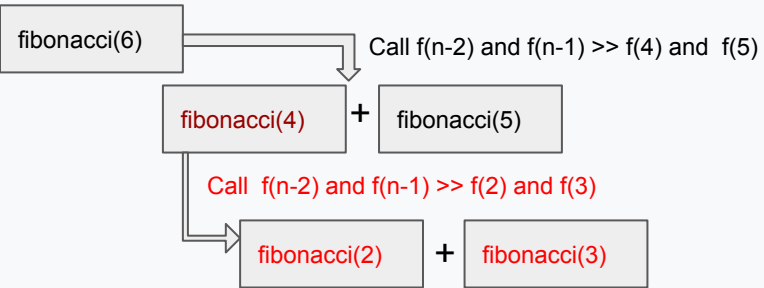
3. Fibonacci Sequence

```
fibonacci(6)
```

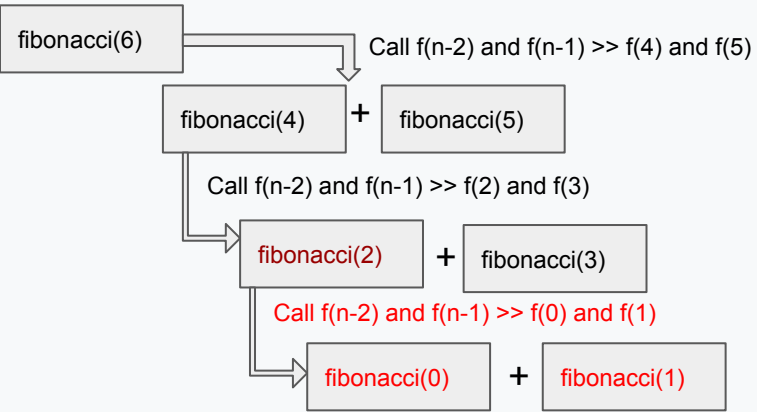

3. Fibonacci Sequence



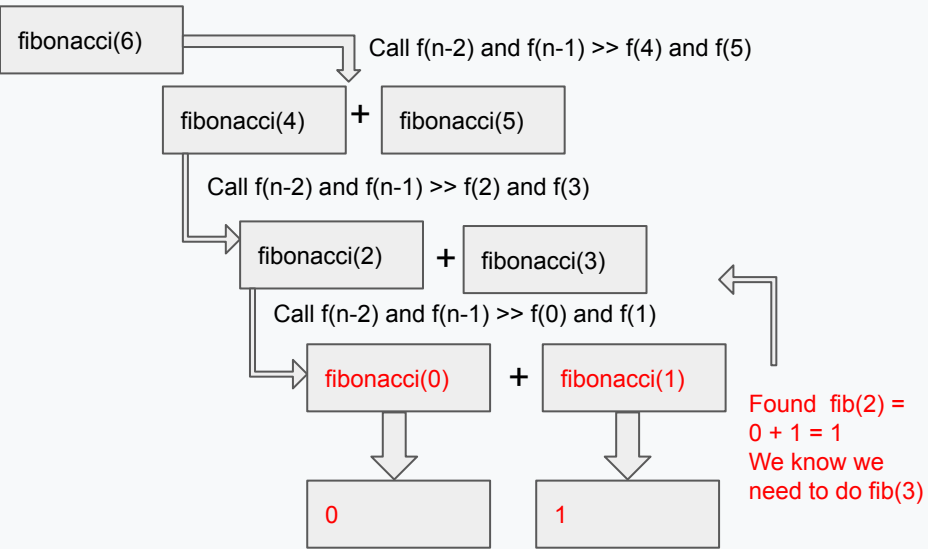
3. Fibonacci Sequence



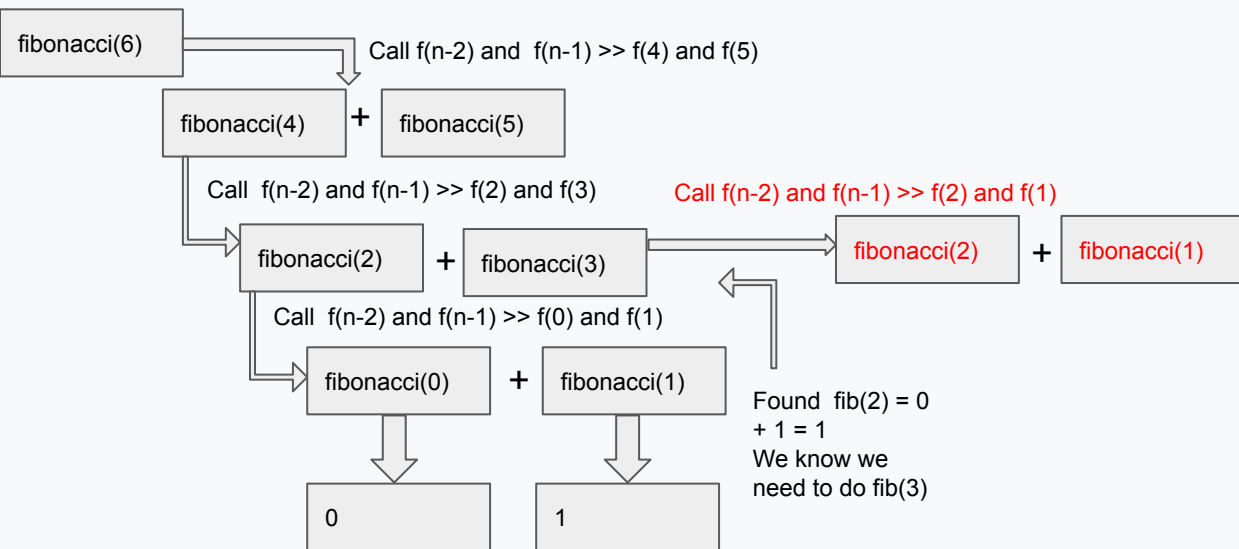
3. Fibonacci Sequence



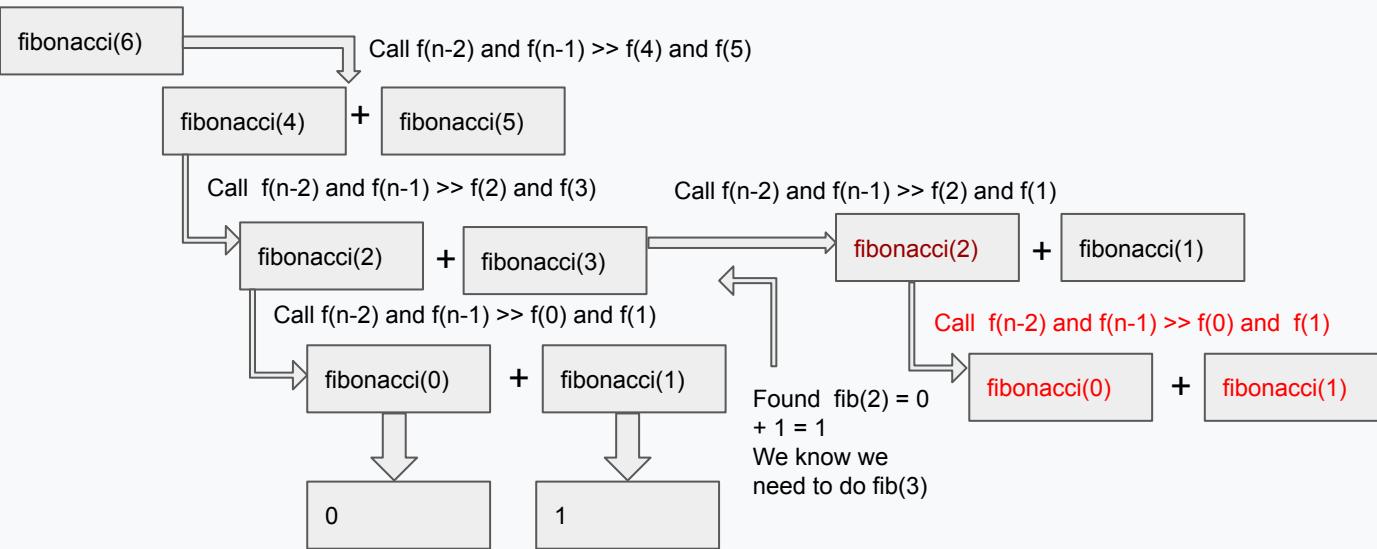
3. Fibonacci Sequence



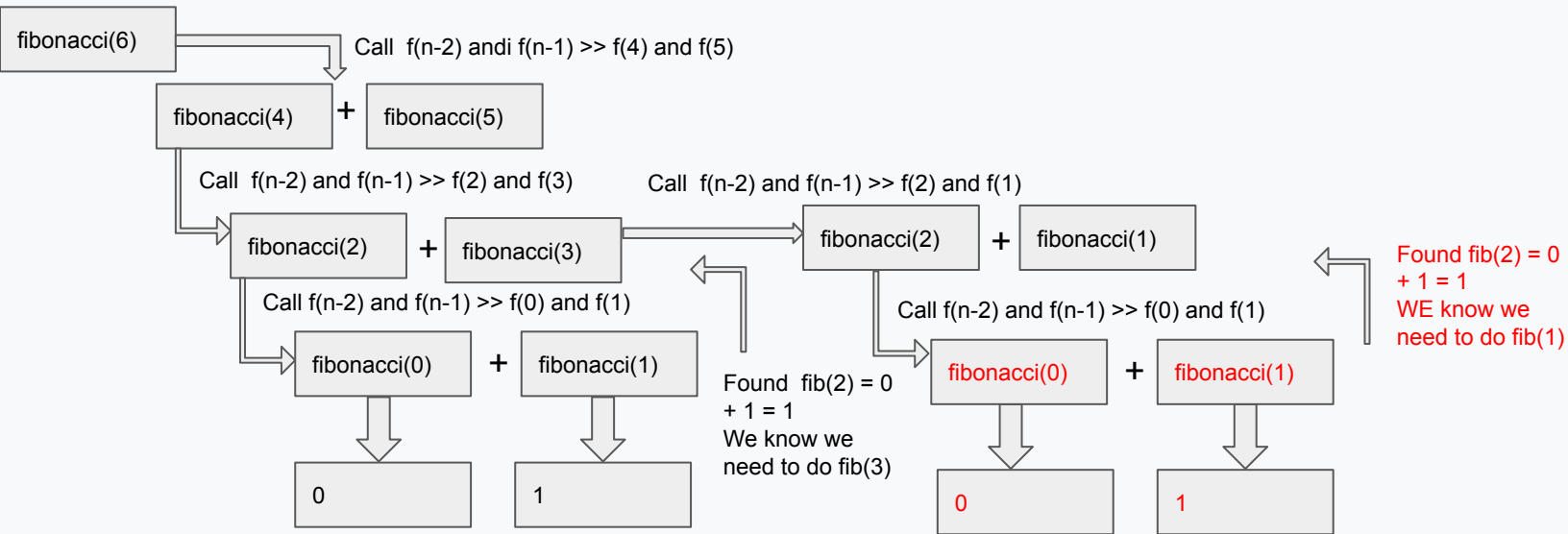
3. Fibonacci Sequence



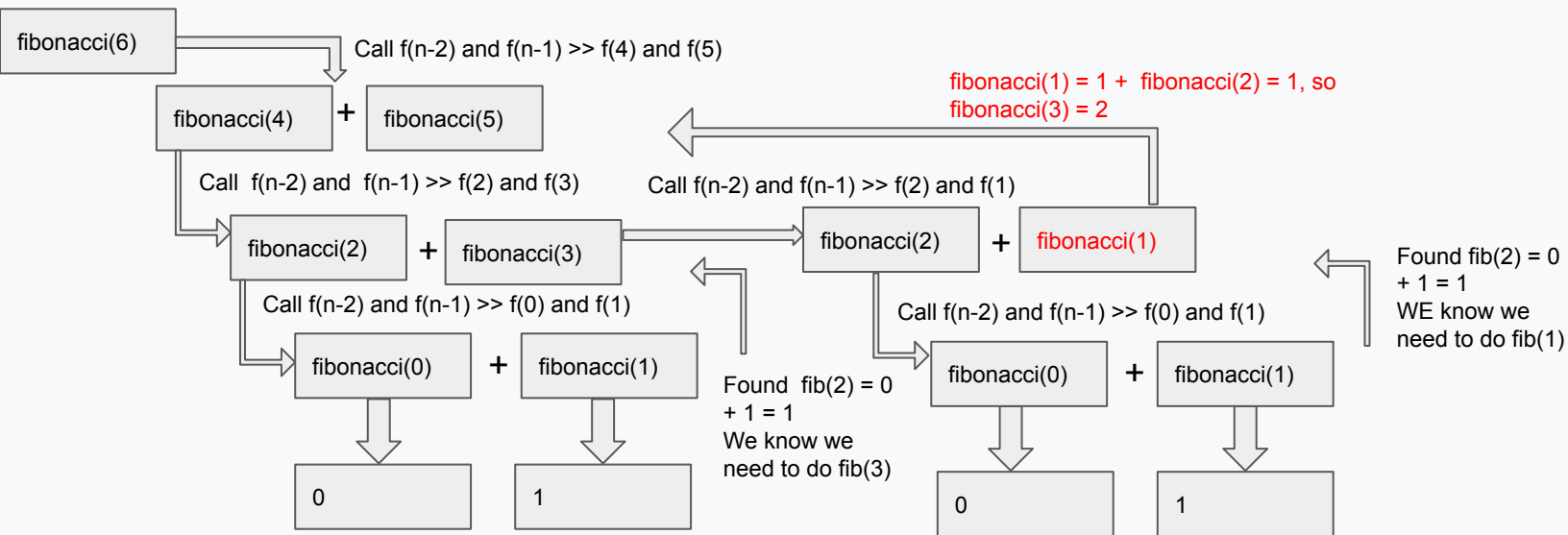
3. Fibonacci Sequence



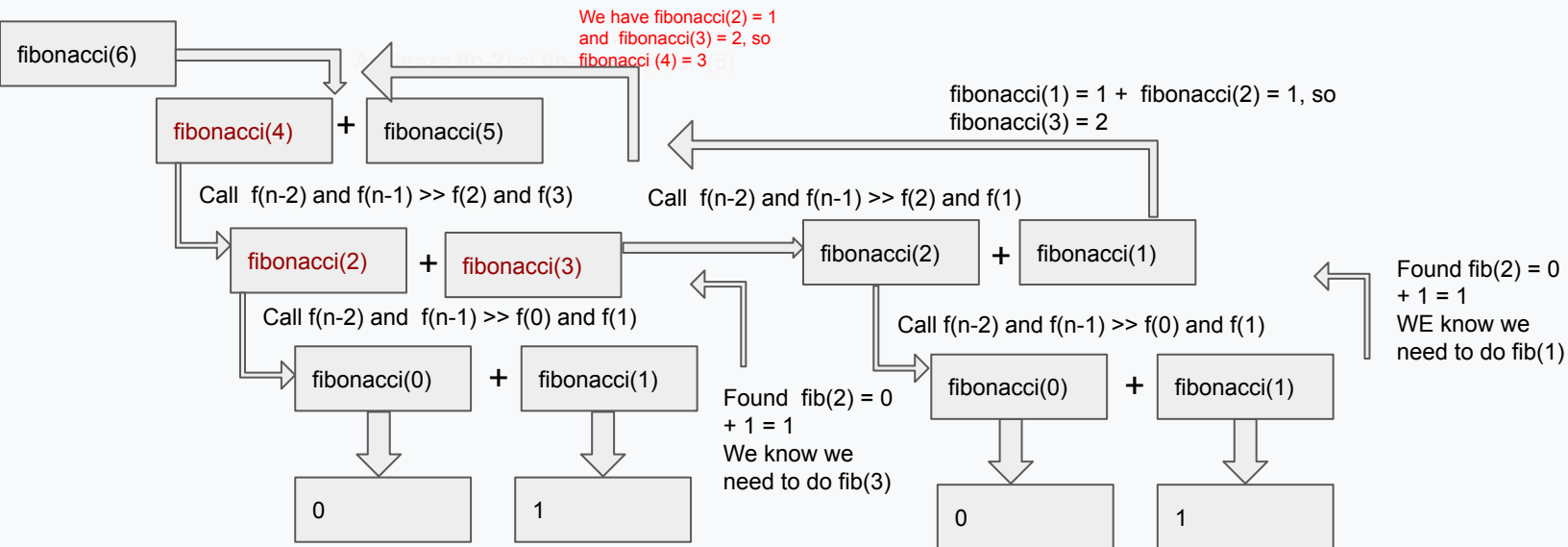
3. Fibonacci Sequence

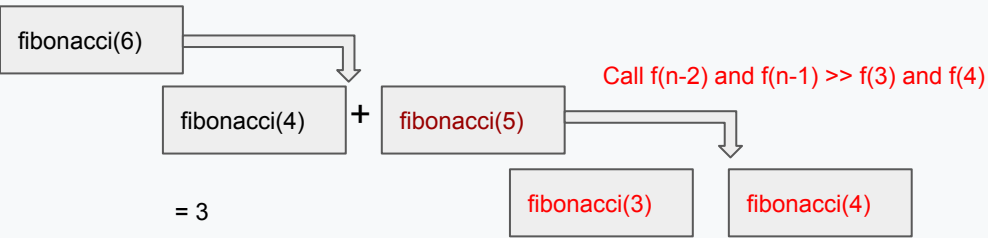


3. Fibonacci Sequence



3. Fibonacci Sequence

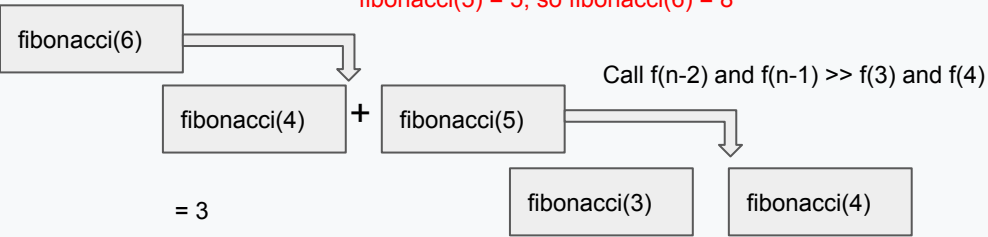




Same process is followed in branch `fibonacci(5)`

We know we'll have `fibonacci(3) = 2` and `fibonacci(4) = 3`. So `fibonacci(5) = 5`

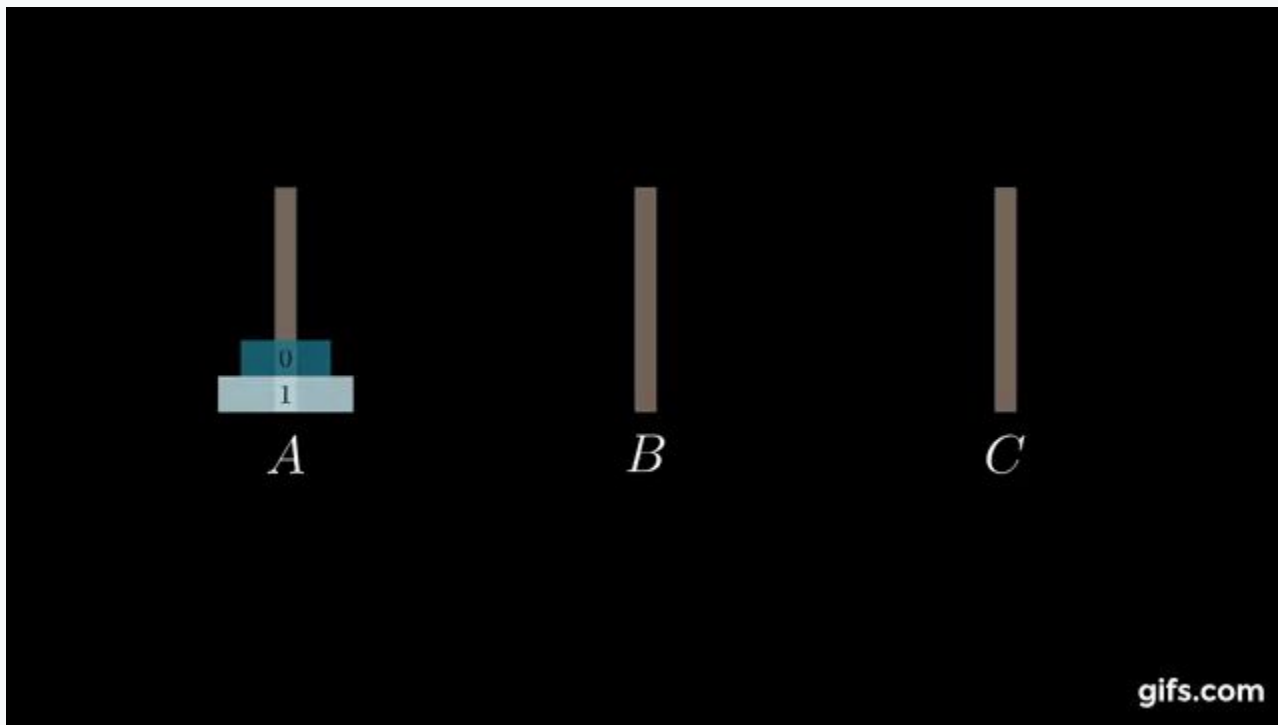
After all iterations we have $\text{fibonacci}(4) = 3$ and $\text{fibonacci}(5) = 5$, so $\text{fibonacci}(6) = 8$



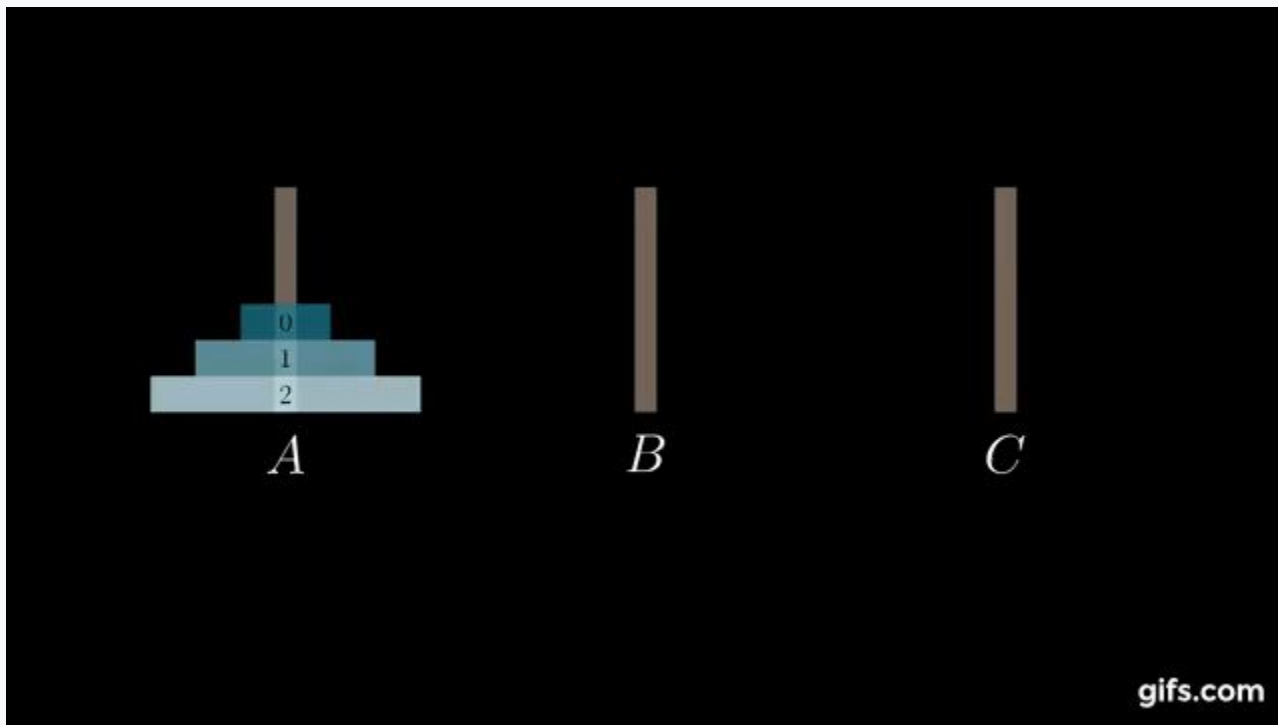
Same process is followed in branch $\text{fibonacci}(5)$

We know we'll have $\text{fibonacci}(3) = 2$ and $\text{fibonacci}(4) = 3$. So $\text{fibonacci}(5) = 5$

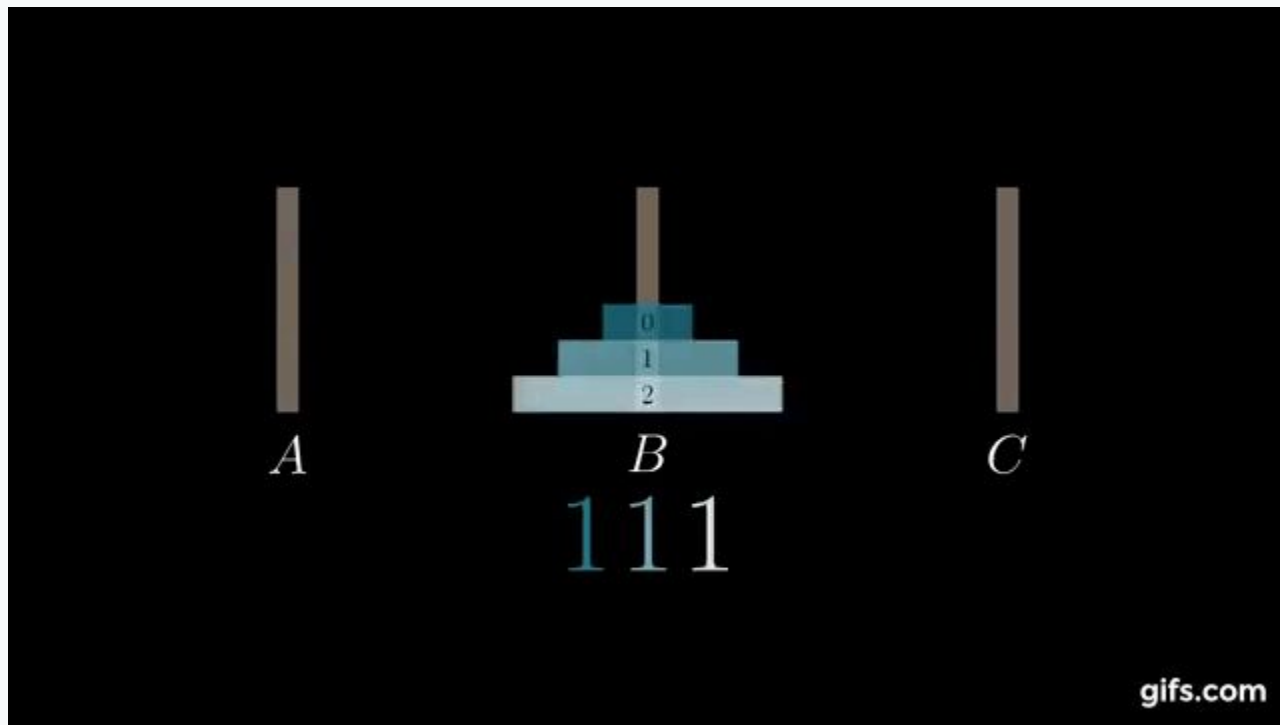
4. Hanoi



4. Hanoi



4. Hanoi

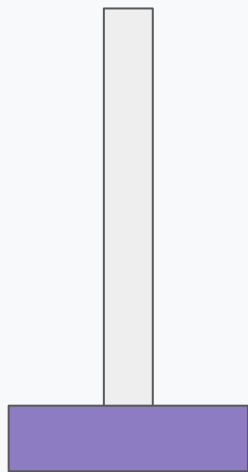


https://www.youtube.com/watch?v=2SUvWfNJSsM&ab_channel=3Blue1Brown

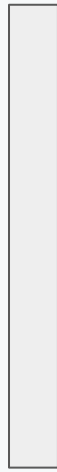
https://www.youtube.com/watch?v=bdMfjfT0IKk&ab_channel=3Blue1Brown

https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw

4. Hanoi (1)



A



B



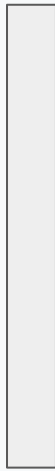
C

4. Hanoi (1)

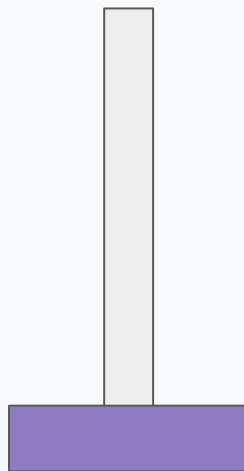
Pas 1:



A

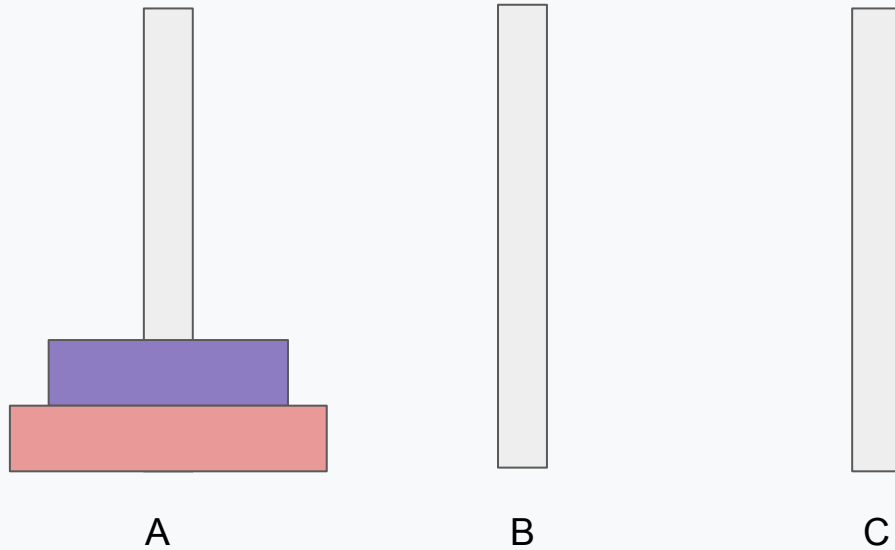


B



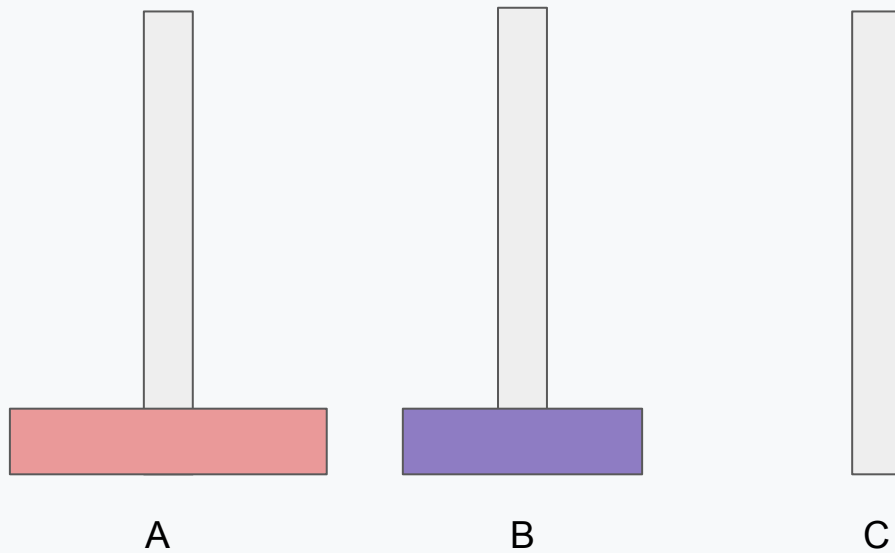
C

4. Hanoi (2)



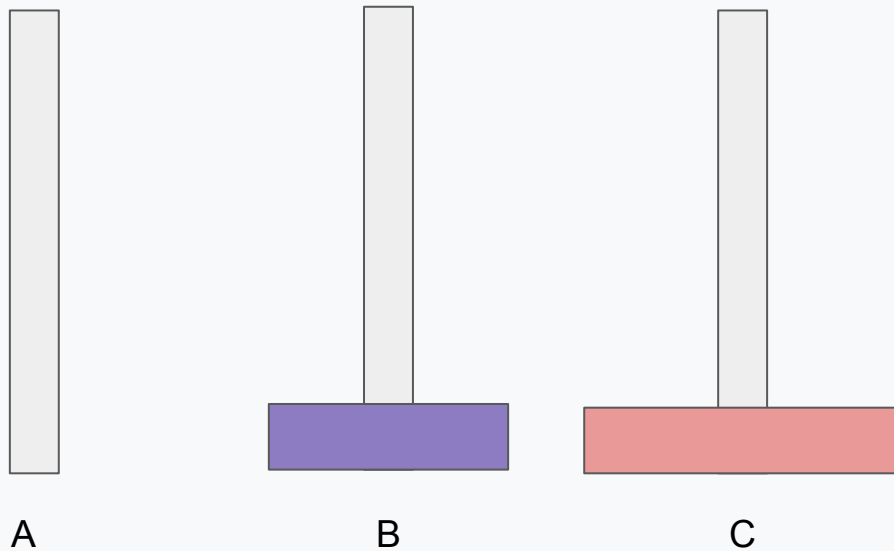
4. Hanoi (2)

Pas 1:



4. Hanoi (2)

Pas 2:



4. Hanoi (2)

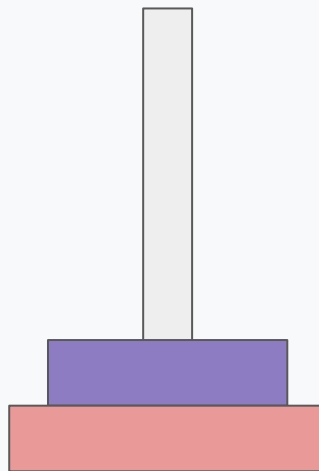
Pas 3:



A

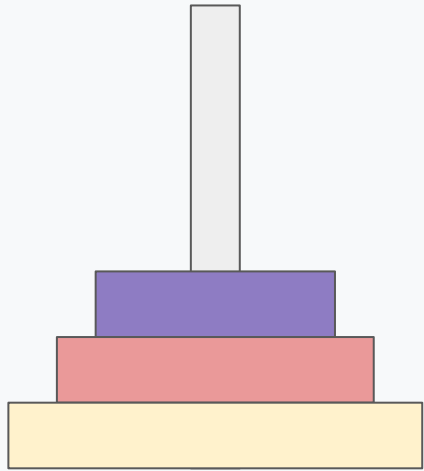


B



C

4. Hanoi (3)



A



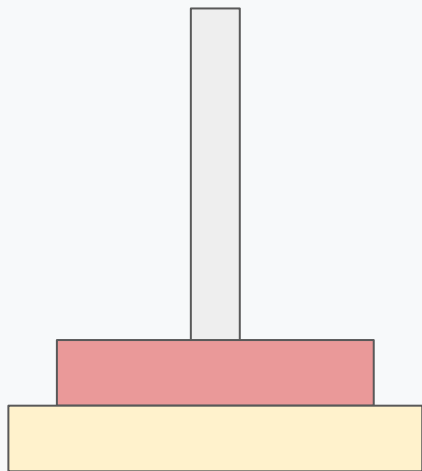
B



C

4. Hanoi (3)

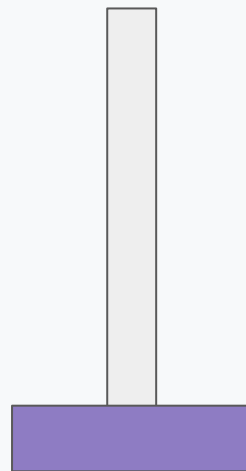
Pas 1:



A



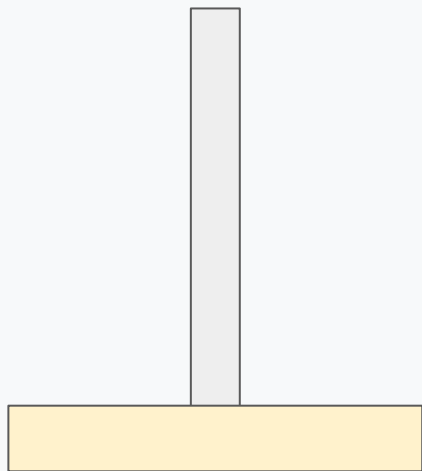
B



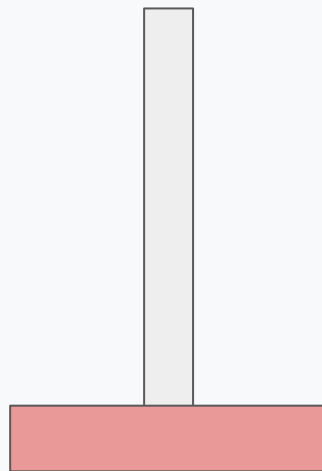
C

4. Hanoi (3)

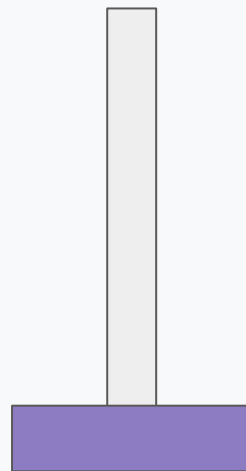
Pas 2:



A



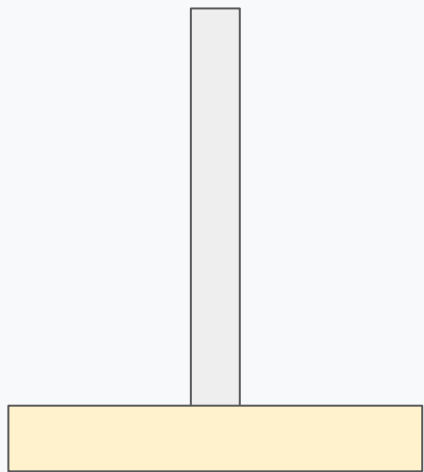
B



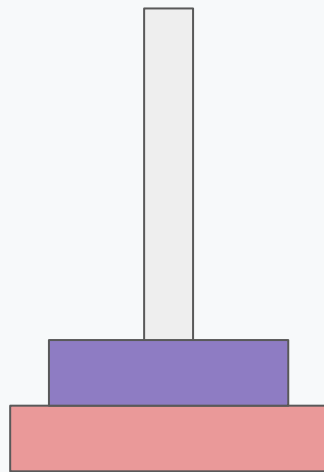
C

4. Hanoi (3)

Pas 3:



A



B



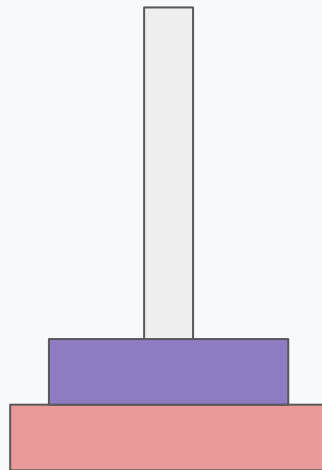
C

4. Hanoi (3)

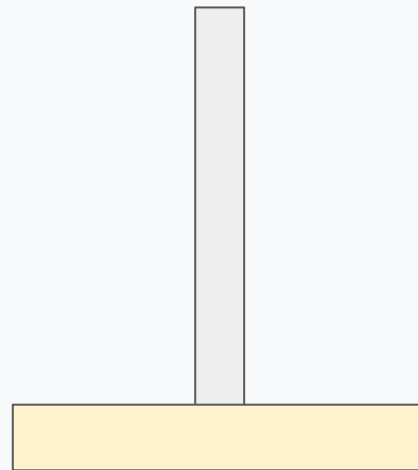
Pas 4:



A



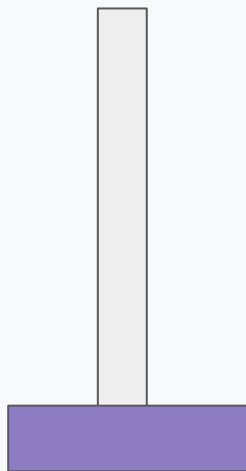
B



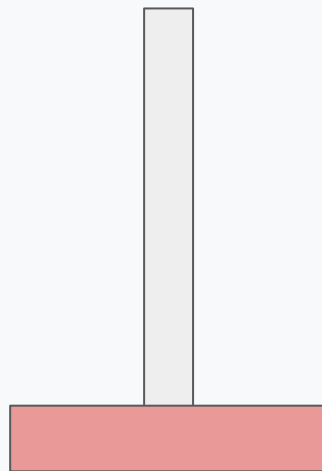
C

4. Hanoi (3)

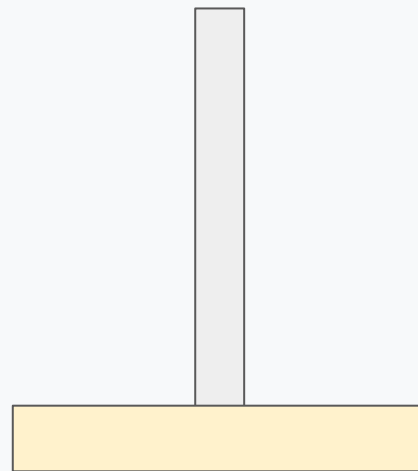
Pas 5:



A



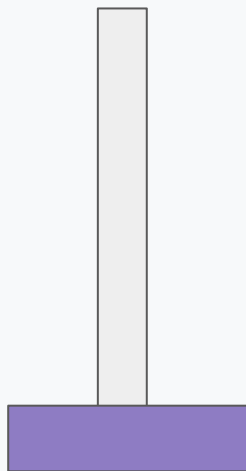
B



C

4. Hanoi (3)

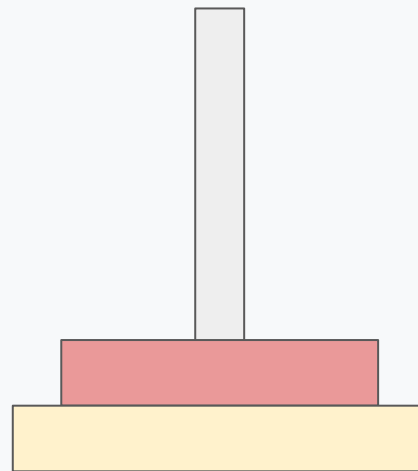
Pas 6:



A



B



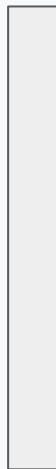
C

4. Hanoi (3)

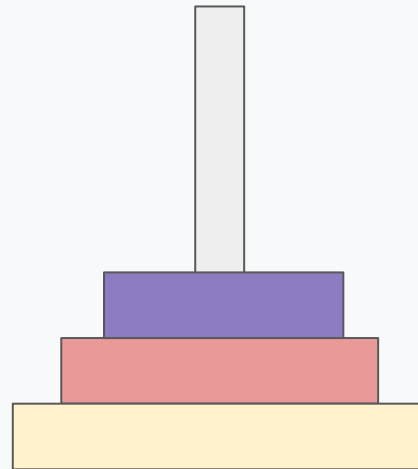
Pas 7:



A

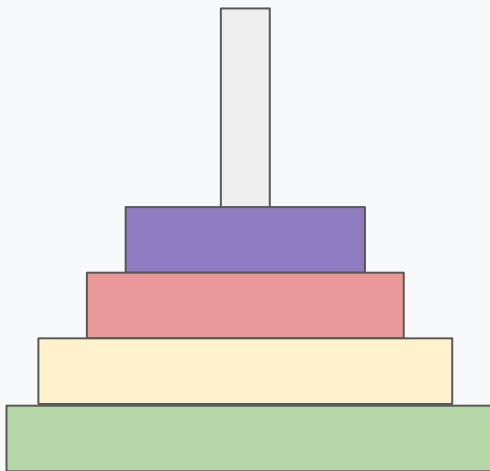


B



C

4. Hanoi (4)



A



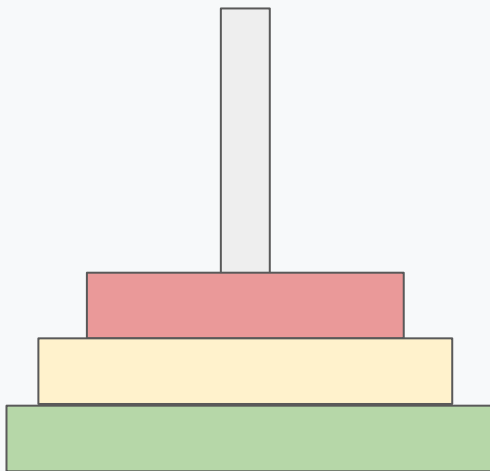
B



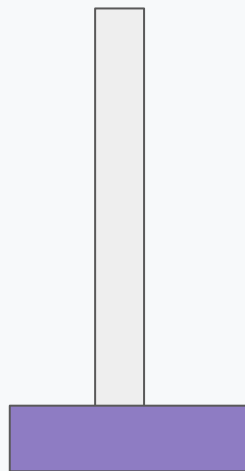
C

4. Hanoi (4)

Pas 1



A



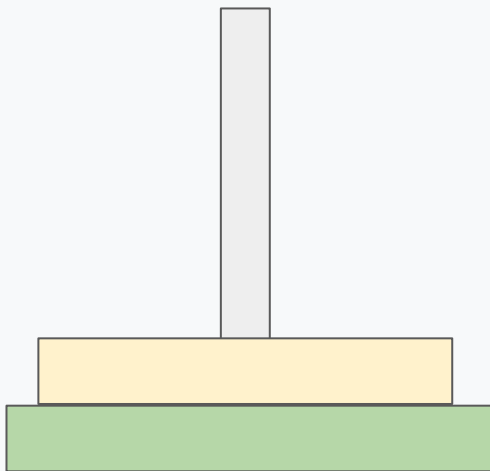
B



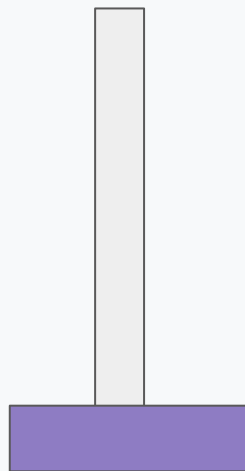
C

4. Hanoi (4)

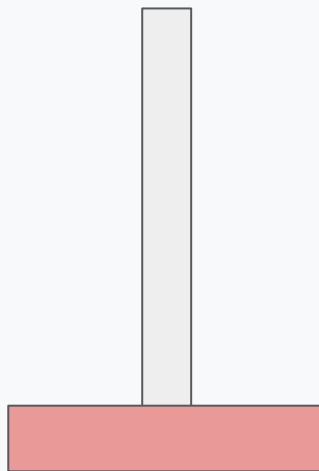
Pas 2



A



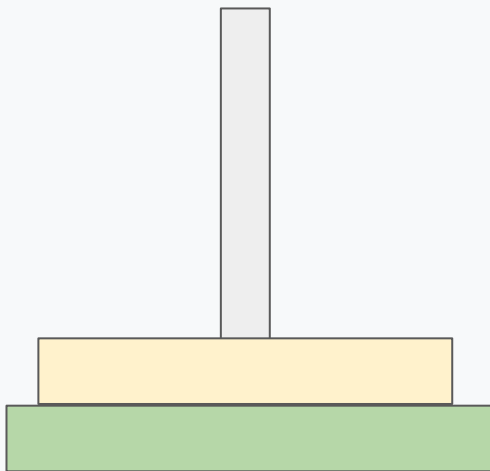
B



C

4. Hanoi (4)

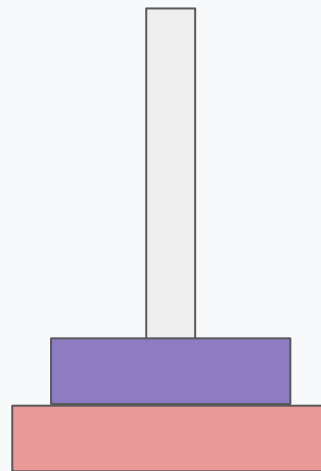
Pas 3



A



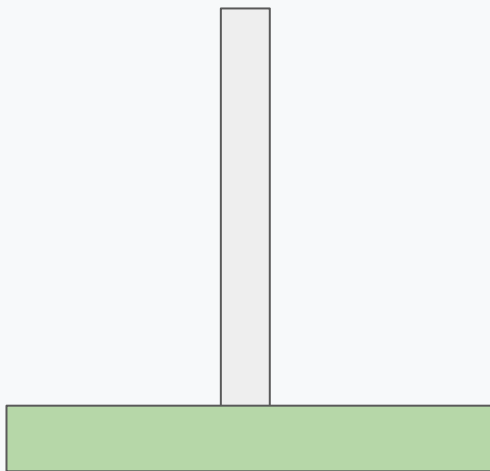
B



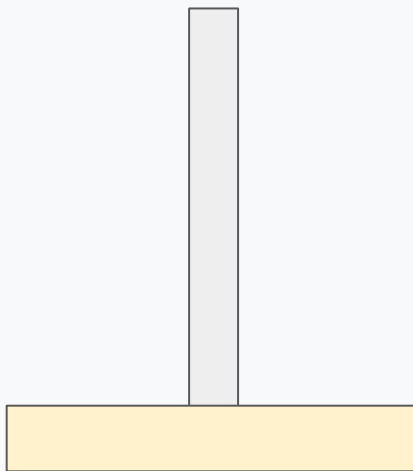
C

4. Hanoi (4)

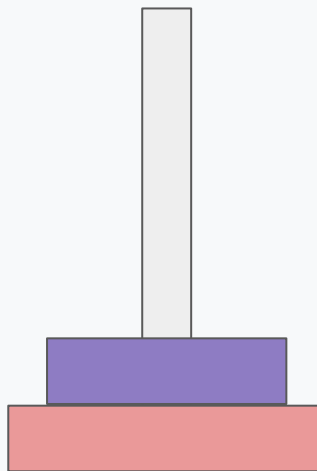
Pas 4



A



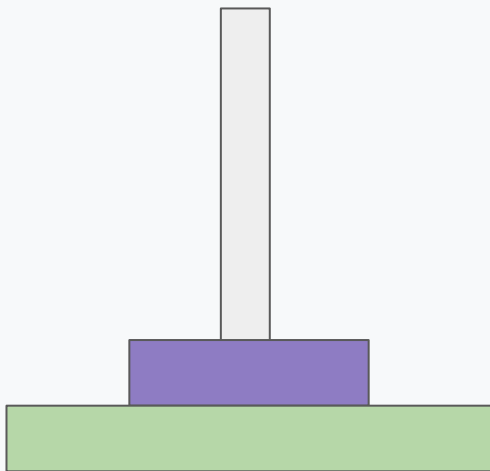
B



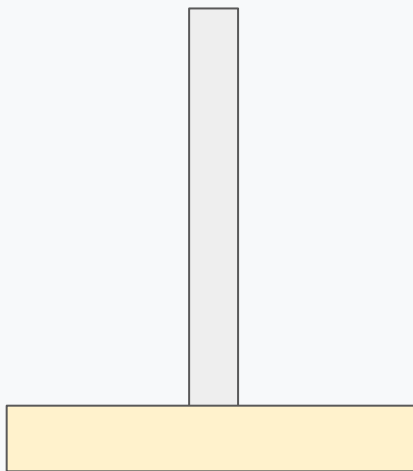
C

4. Hanoi (4)

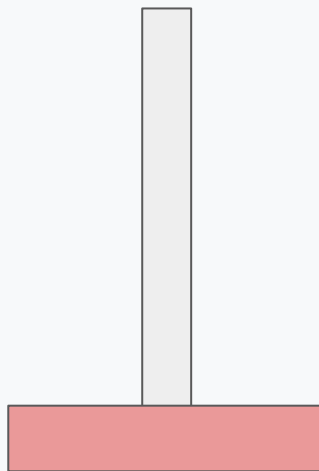
Pas 5



A



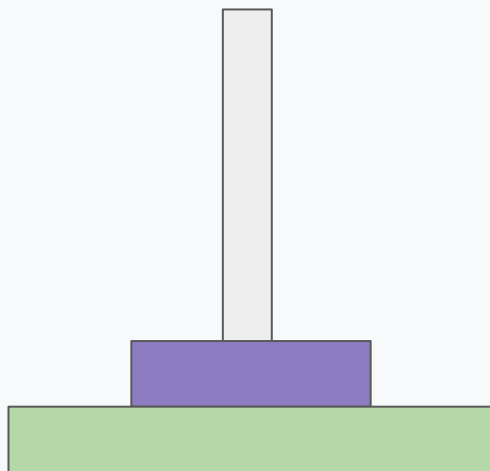
B



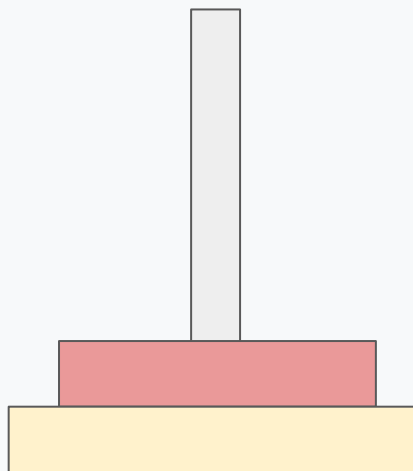
C

4. Hanoi (4)

Pas 6



A



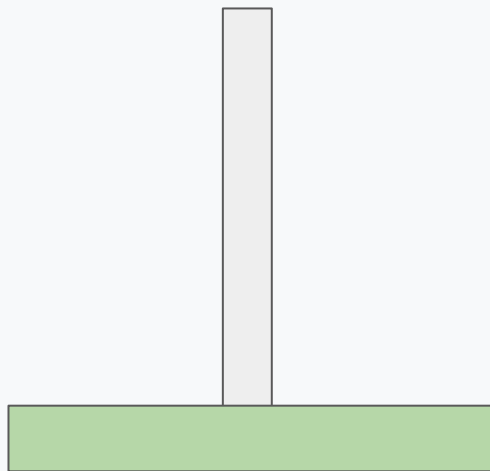
B



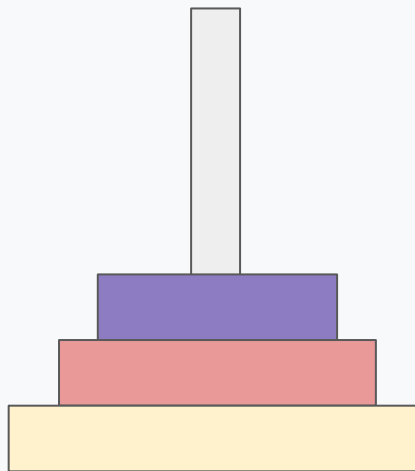
C

4. Hanoi (4)

Pas 7



A



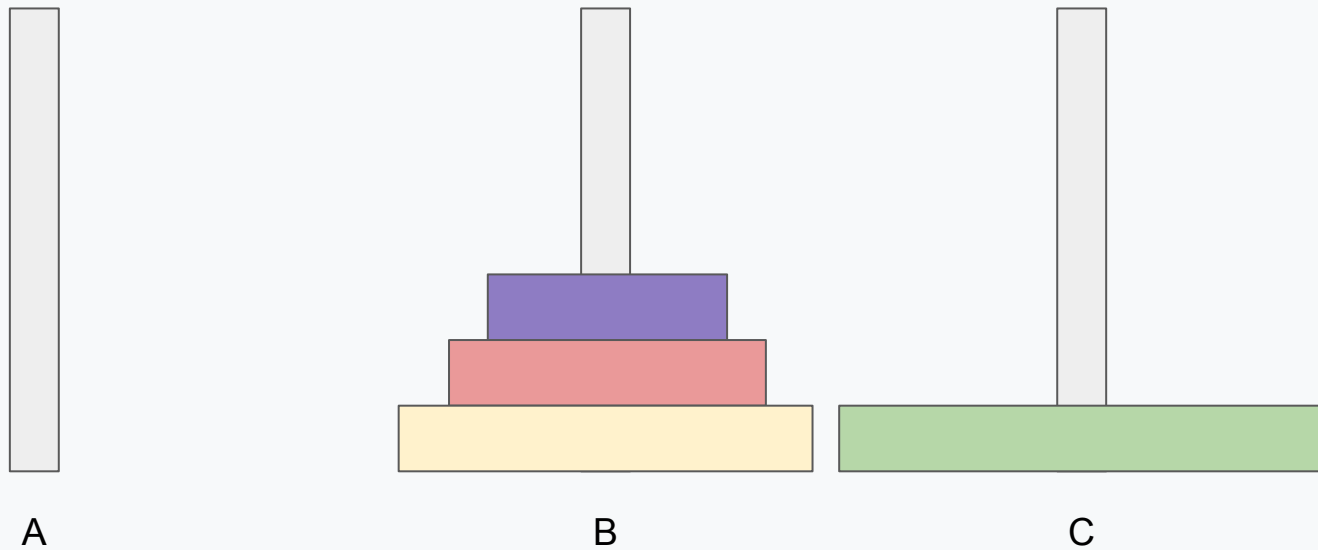
B



C

4. Hanoi (4)

Pas 8

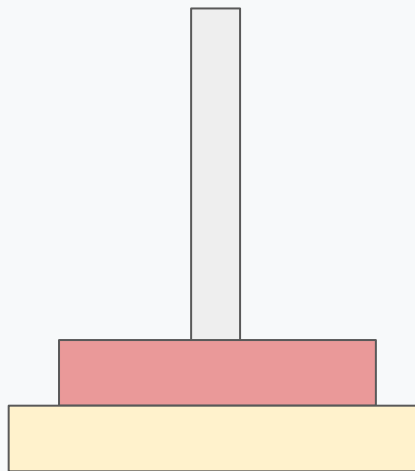


4. Hanoi (4)

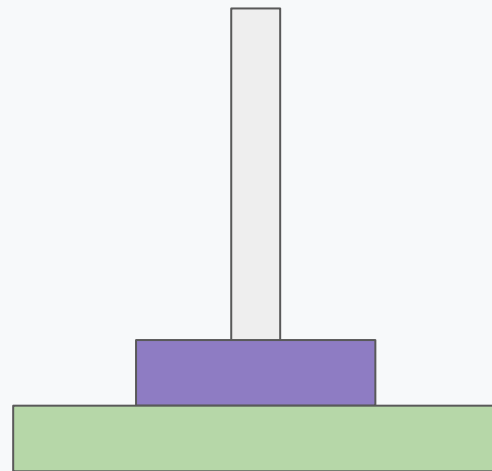
Pas 9



A



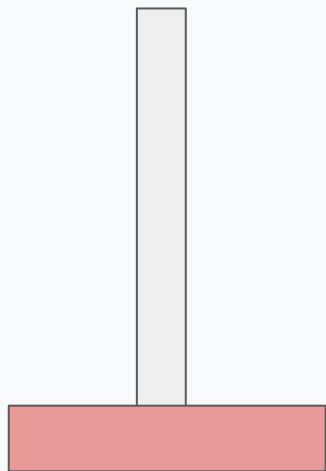
B



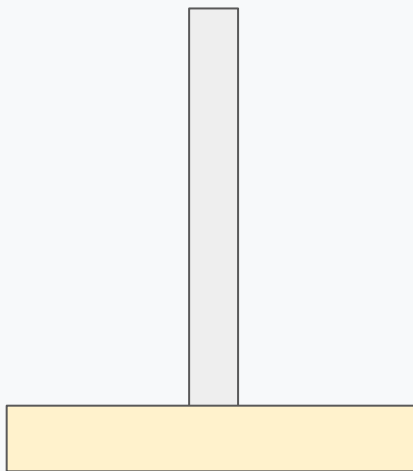
C

4. Hanoi (4)

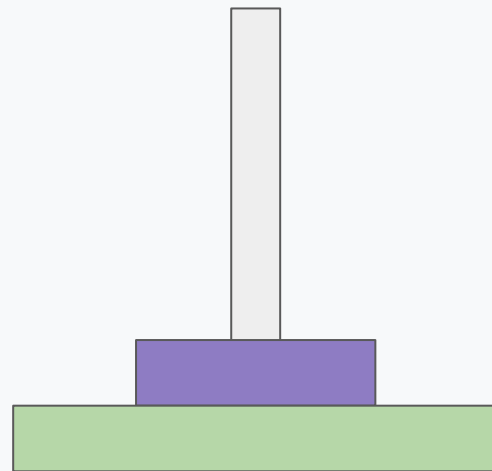
Pas 10



A



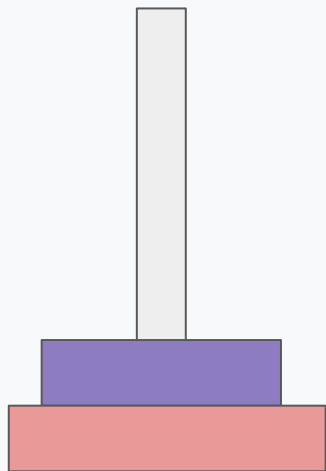
B



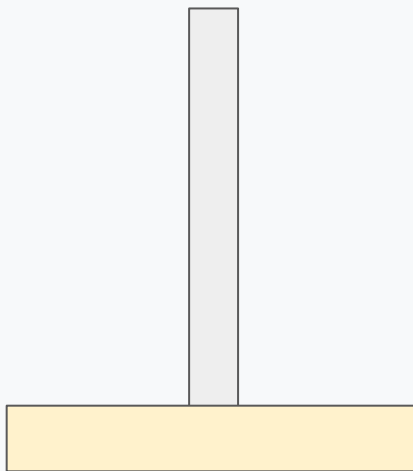
C

4. Hanoi (4)

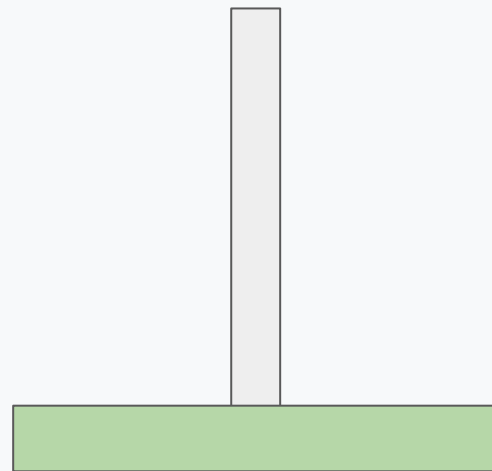
Pas 11



A



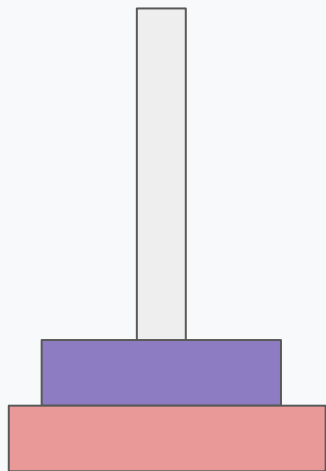
B



C

4. Hanoi (4)

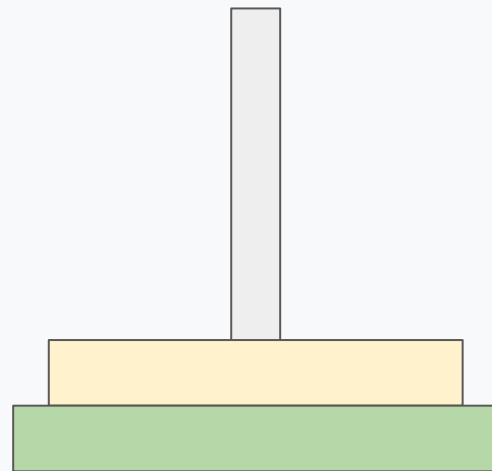
Pas 12



A



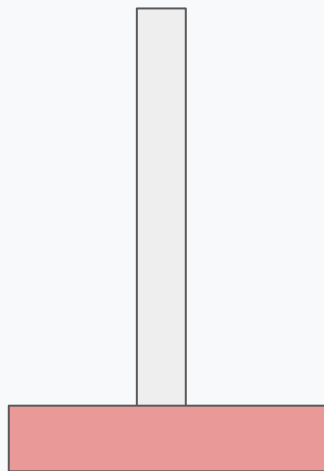
B



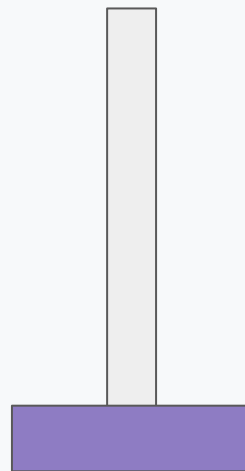
C

4. Hanoi (4)

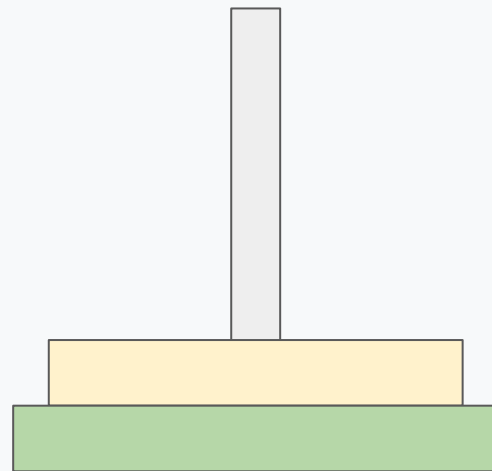
Pas 13



A



B



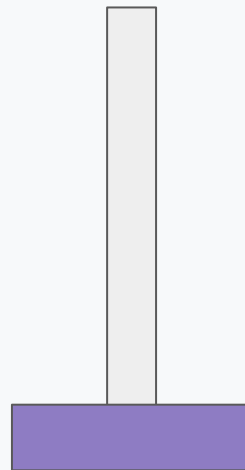
C

4. Hanoi (4)

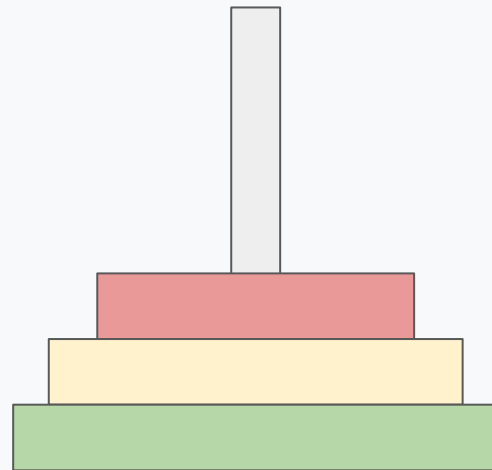
Pas 14



A



B



C

4. Hanoi (4)

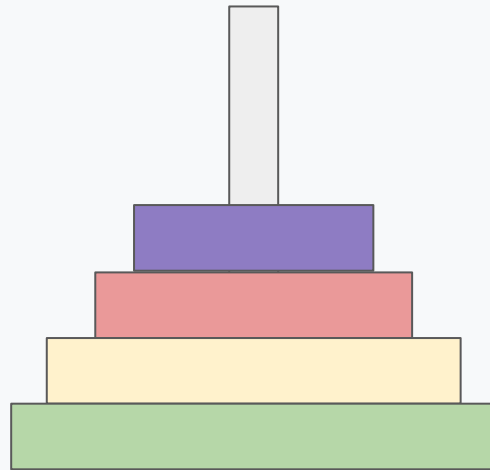
Pas 15



A



B



C

5. Bunny Ears

We have a number of bunnies and each bunny has two big floppy ears. We want to compute the total number of ears across all the bunnies recursively (without loops or multiplication).

$$0 \Rightarrow 0$$

$$1 \Rightarrow 2$$

$$2 \Rightarrow 4$$

$$3 \Rightarrow 6$$

5. Bunny Ears

The recursive solution :

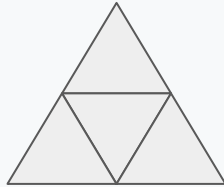
- 1) Check if $n = 1$. If true return 2.
- 2) If false, add 2 to the function with $n-1$

6. Triangle

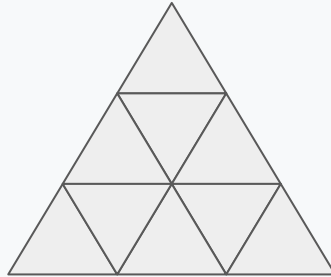
We have triangle made of blocks. The top most row has 1 block, the next row down has 3 blocks, the next row has 5 blocks, and so on. Compute recursively the total number of blocks in such a triangle with the given number of rows.



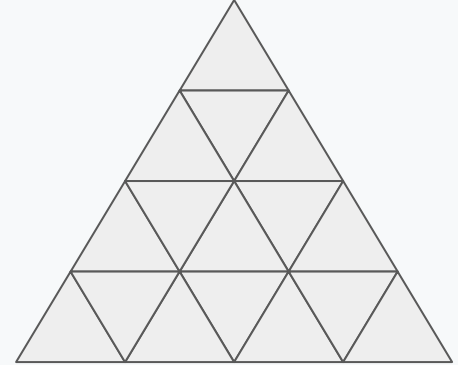
1



4



9



16

6. Triangle

Recursive Solution:

1. Loop for each row
2. For each row recursively add 2 for each number of rows.
3. Add the number of triangles in each row to the total.

7. Chocolate and wrappers

Given the following three values, the task is to find the total number of maximum chocolates you can eat.

1. money: Money you have to buy chocolates
2. price: Price of a chocolate
3. wrap: Number of wrappers to be returned for getting one extra chocolate.

7. Chocolate and wrappers

Input: money = 16, price = 2, wrap = 2

Output: 15

Price of a chocolate is 2. You can buy 8 chocolates from amount 16. You can return 8 wrappers back and get 4 more chocolates. Then you can return 4 wrappers and get 2 more chocolates. Finally you can return 2 wrappers to get 1 more chocolate.

Input: money = 15, price = 1, wrap = 3

Output: 22

We buy and eat 15 chocolates. We return 15 wrappers and get 5 more chocolates. We return 3 wrappers, get 1 chocolate and eat it (keep 2 wrappers). Now we have 3 wrappers. Return 3 and get 1 more chocolate.

So total chocolates = $15 + 5 + 1 + 1$

Input: money = 20, price = 3, wrap = 5

Output: 7

7. Chocolate and wrappers

Solution:

1. Calculate the max initial amount you can buy with money.
2. Recursively calculate the amount of chocolates you can get with wrappers (updating with number of wrappers with the number of chocolates bought + modulo of initial chocolates and wrappers needed)

8. Print matrix in spiral order

Given an $M \times N$ integer matrix, print it in spiral order.

Input:

```
[ 1  2  3  4  5 ]  
[ 16 17 18 19 6 ]  
[ 15 24 25 20 7 ]  
[ 14 23 22 21 8 ]  
[ 13 12 11 10 9 ]
```

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

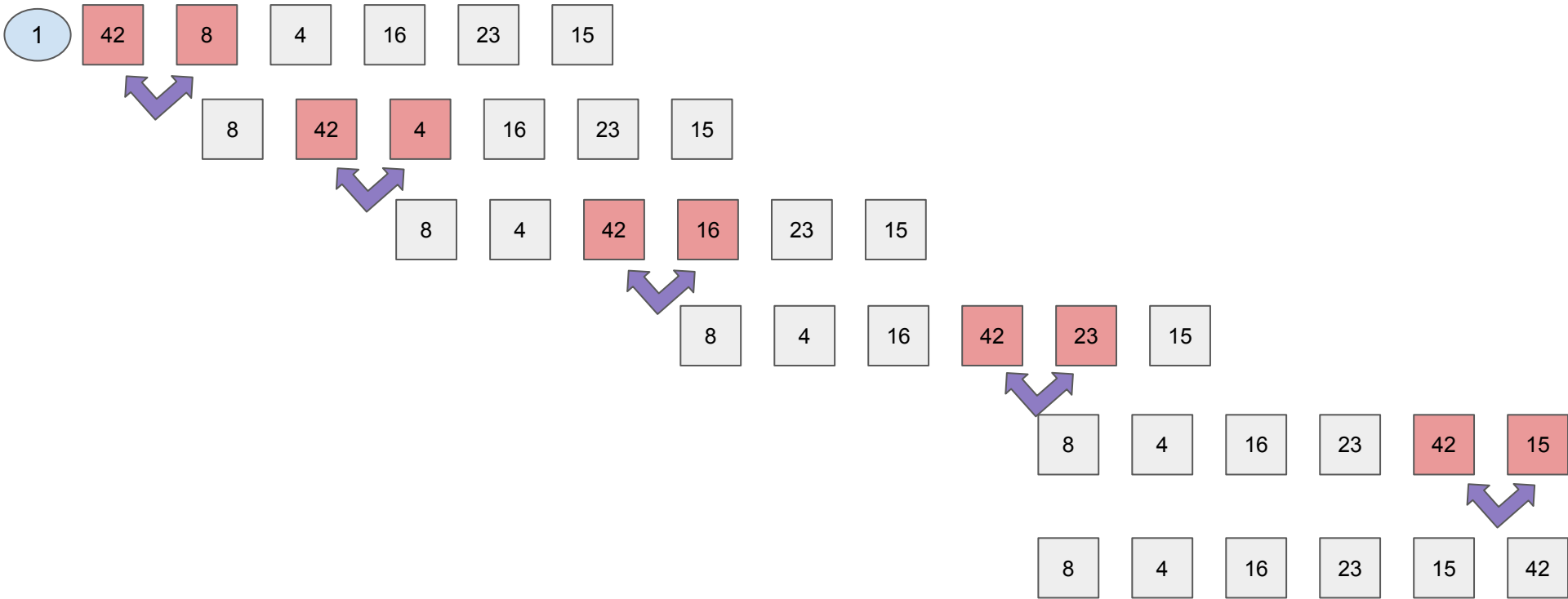
8. Print matrix in spiral order

The idea is to read elements from the given matrix and print the matrix in spiral order. Four loops are used to maintain the spiral order, each for the top, right, bottom, and left corner of the matrix.

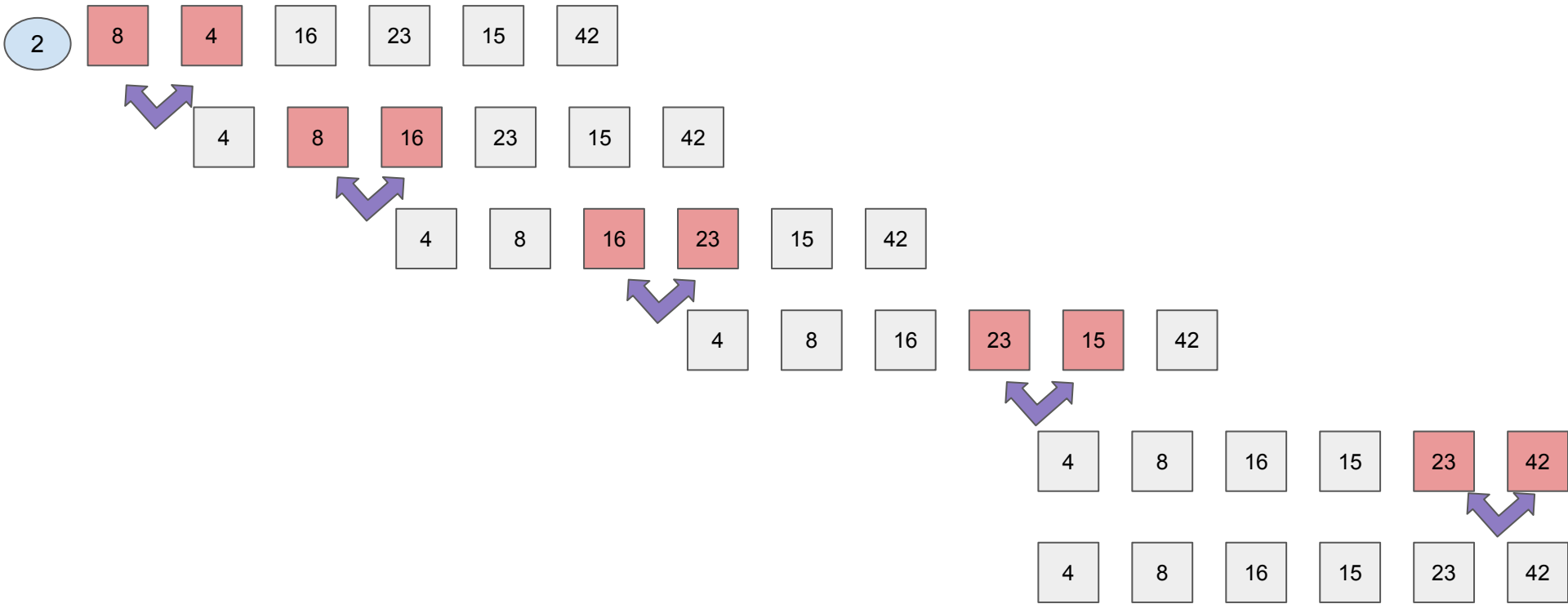
DS-ALGS - Lab 3

Sorting algorithms

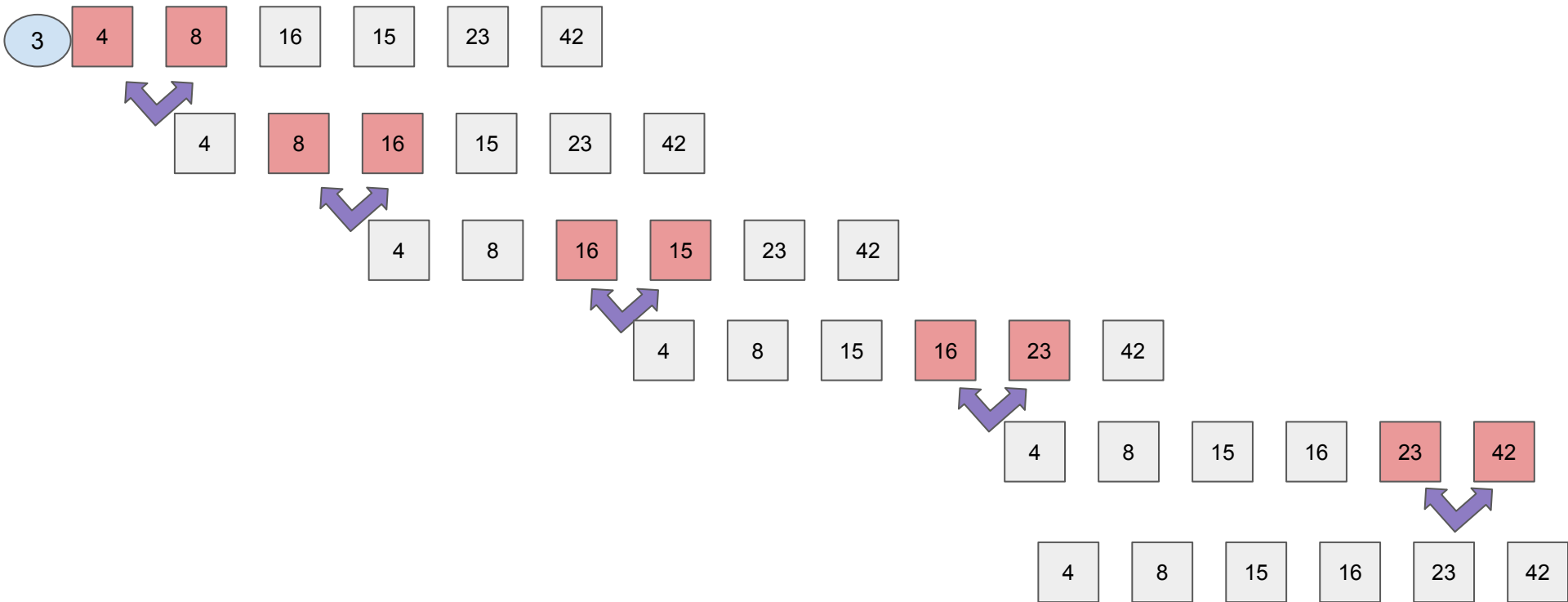
Bubble Sort



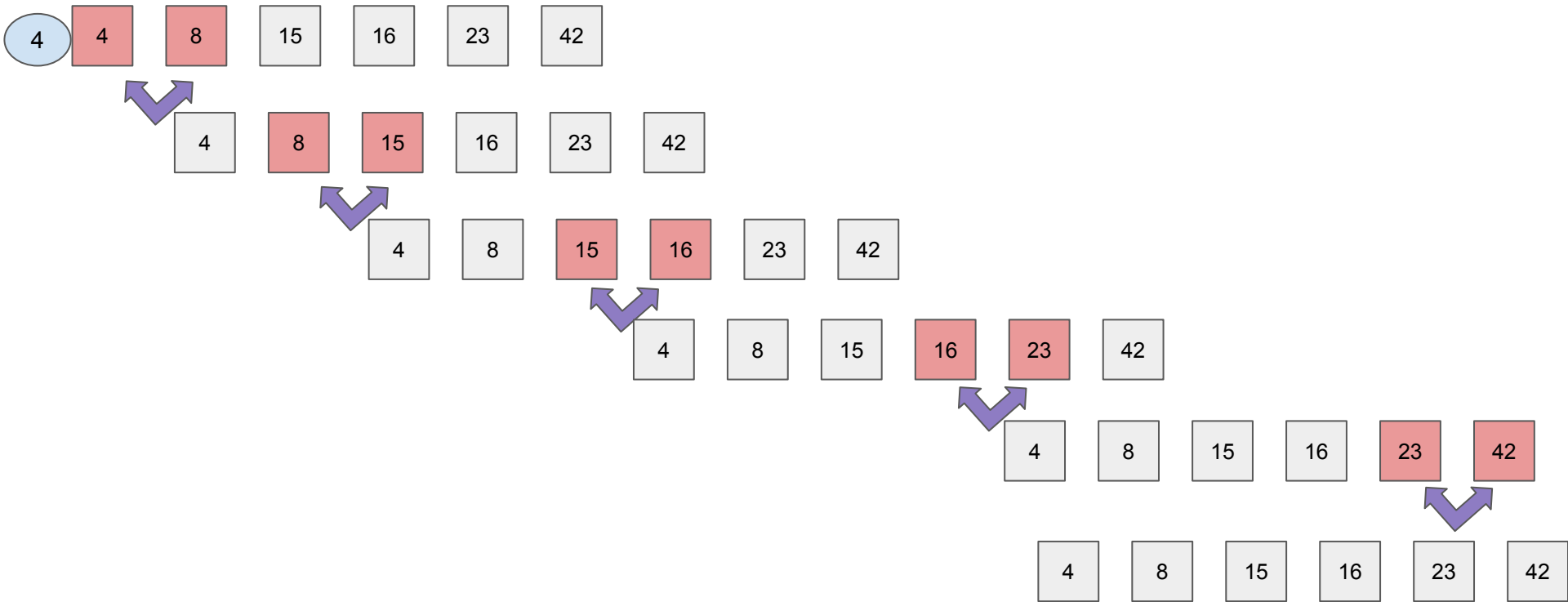
Bubble Sort



Bubble Sort



Bubble Sort - no swap loop



Bubble Sort

1

(**42**, 8, 4, 16, 23, 15) → (8, **42**, 4, 16, 23, 15),
(8, **42**, 4, 16, 23, 15) → (8, 4, **42**, 16, 23, 15),
(8, 4, **42**, 16, 23, 15) → (8, 4, 16, **42**, 23, 15),
(8, 4, 16, **42**, 23, 15) → (8, 4, 16, 23, **42**, 15),
(8, 4, 16, 23, **42**, 15) → (8, 4, 16, 23, 15, **42**).

2

(8, 4, 16, 23, 15, 42) → (4, 8, 16, 23, 15, 42),
(4, 8, **16**, 23, 15, 42) → (4, 8, 16, 23, 15, 42),
(4, 8, **16**, 23, 15, 42) → (4, 8, 16, 23, 15, 42),
(4, 8, 16, **23**, 15, 42) → (4, 8, 16, 15, **23**, 42),
(4, 8, 16, 15, **23**, 42) → (4, 8, 16, 15, 23, **42**).

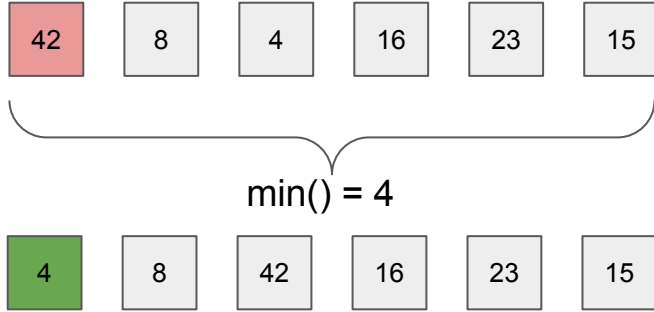
3

(4, 8, 16, 15, 23, 42) → (4, 8, 16, 15, 23, 42),
(4, 8, **16**, 15, 23, 42) → (4, 8, 16, 15, 23, 42),
(4, 8, **16**, 15, 23, 42) → (4, 8, 15, **16**, 23, 42),
(4, 8, 15, **16**, 23, 42) → (4, 8, 15, 16, 23, 42),
(4, 8, 15, 16, **23**, 42) → (4, 8, 15, 16, 23, 42).

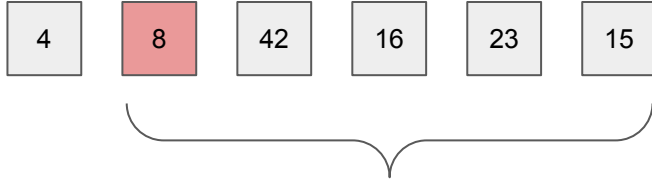
4

(4, 8, 15, 16, 23, 42) → (4, 8, 15, 16, 23, 42),
(4, 8, **15**, 16, 23, 42) → (4, 8, 15, 16, 23, 42),
(4, 8, **15**, 16, 23, 42) → (4, 8, 15, 16, 23, 42),
(4, 8, 15, **16**, 23, 42) → (4, 8, 15, 16, 23, 42),
(4, 8, 15, 16, **23**, 42) → (4, 8, 15, 16, 23, 42).

Selection Sort



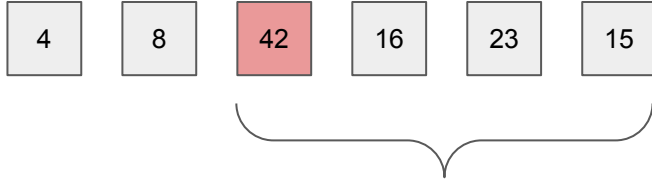
Selection Sort



$\text{min()} = 8$



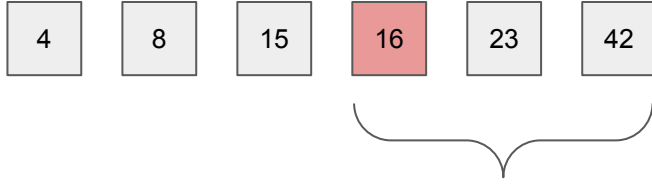
Selection Sort



$\text{min()} = 15$



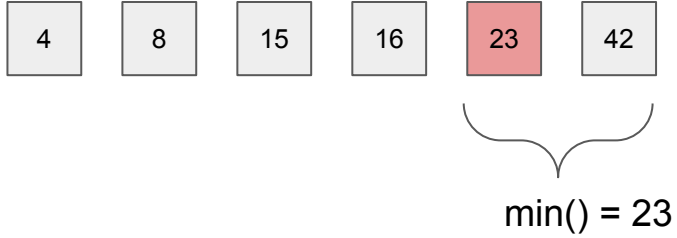
Selection Sort



$\text{min()} = 16$



Selection Sort



Selection Sort

(42, 8, 4, 16, 23, 15) => min(42, 8, **4**, 16, 23, 15) => (**4**, 8, **42**, 16, 23, 15)

(4, 8, 42, 16, 23, 15) => min(**8**, 42, 16, 23, 15) => (4, **8**, 42, 16, 23, 15)

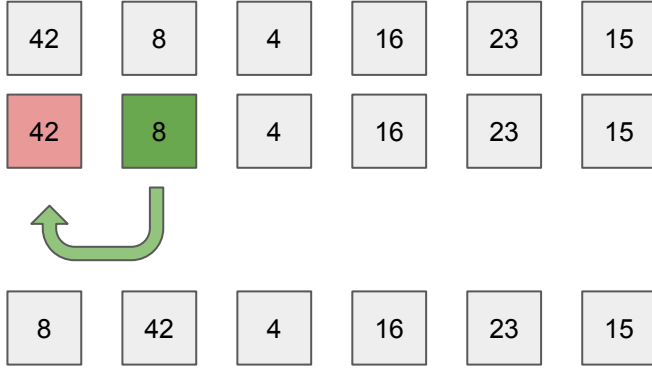
(4, 8, 42, 16, 23, 15) => min(42, 16, 23, **15**) => (4, 8, **15**, 16, 23, **42**)

(4, 8, 15, 16, 23, 42) => min(**16**, 23, 42) => (4, 8, 15, **16**, 23, 42)

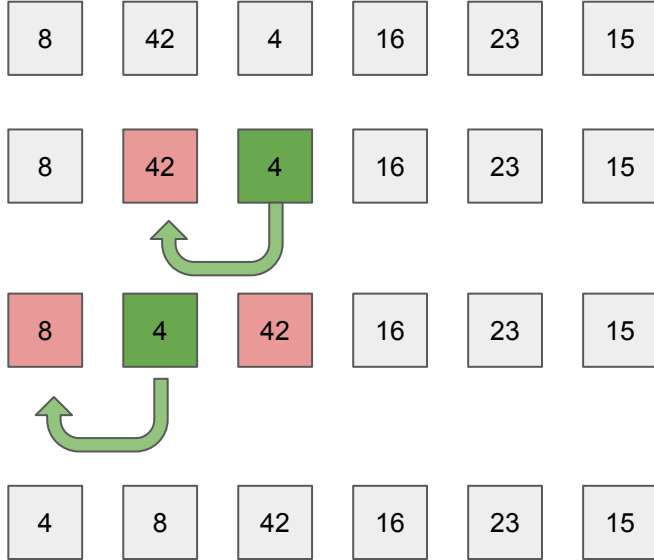
(4, 8, 15, 16, 23, 42) => min(**23**, 42) => (4, 8, 15, 16, **23**, 42)

(4, 8, 15, 16, 23, 42)

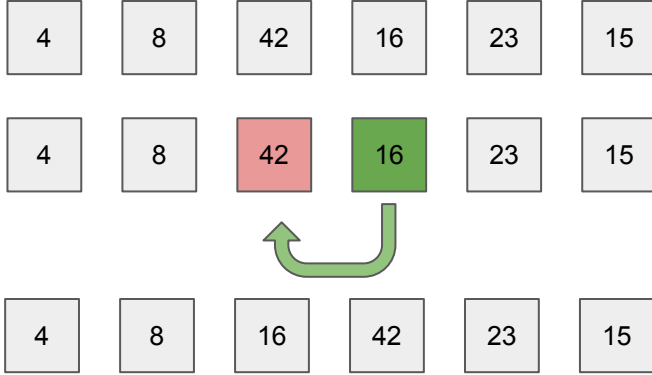
Insertion Sort



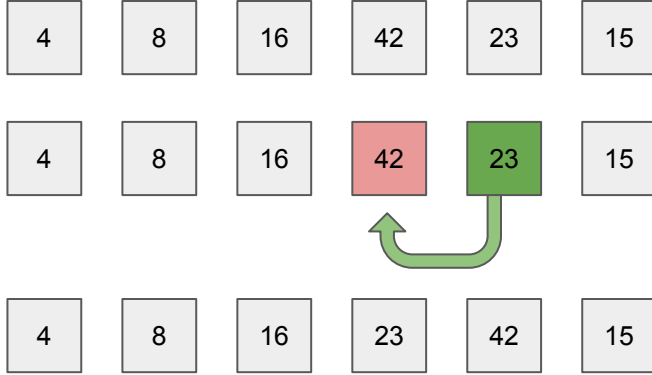
Insertion Sort



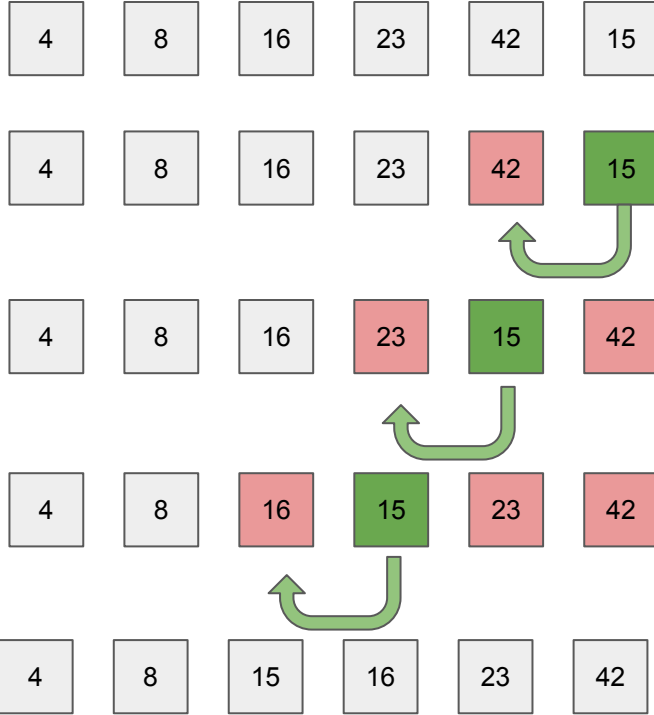
Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort

(42, **8**, 4, 16, 23, 15) -> (**42**, **8**, 4, 16, 23, 15) -> (**8**, 42, 4, 16, 23, 15),

(8, 42, **4**, 16, 23, 15) -> (8, **42**, **4**, 16, 23, 15) -> (**8**, **4**, 42, 16, 23, 15) -> (**4**, 8, 42, 16, 23, 15)

(4, 8, 42, **16**, 23, 15) -> (4, 8, **42**, **16**, 23, 15) -> (4, 8, **16**, 42, 23, 15)

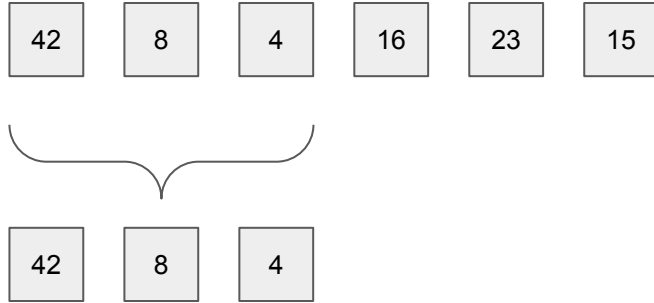
(4, 8, 16, 42, **23**, 15) -> (4, 8, 16, **42**, **23**, 15) -> (4, 8, 16, **23**, 42, 15)

(4, 8, 16, 23, 42, **15**) -> (4, 8, 16, 23, **42**, **15**) -> (4, 8, 16, **23**, **15**, 42) -> (4, 8, **16**, **15**, 23, 42) -> (4, 8, **15**, 16, 23, 42)

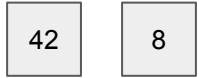
Merge Sort



Merge Sort



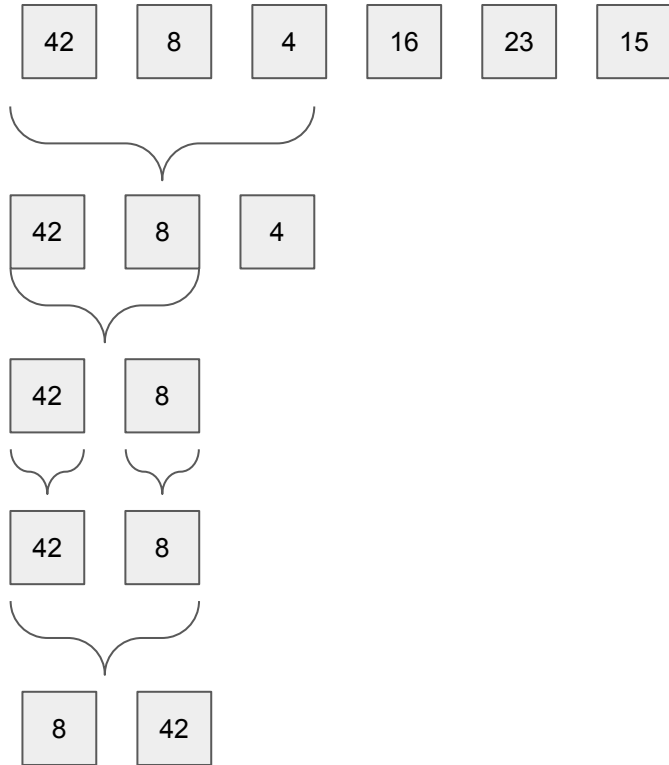
Merge Sort



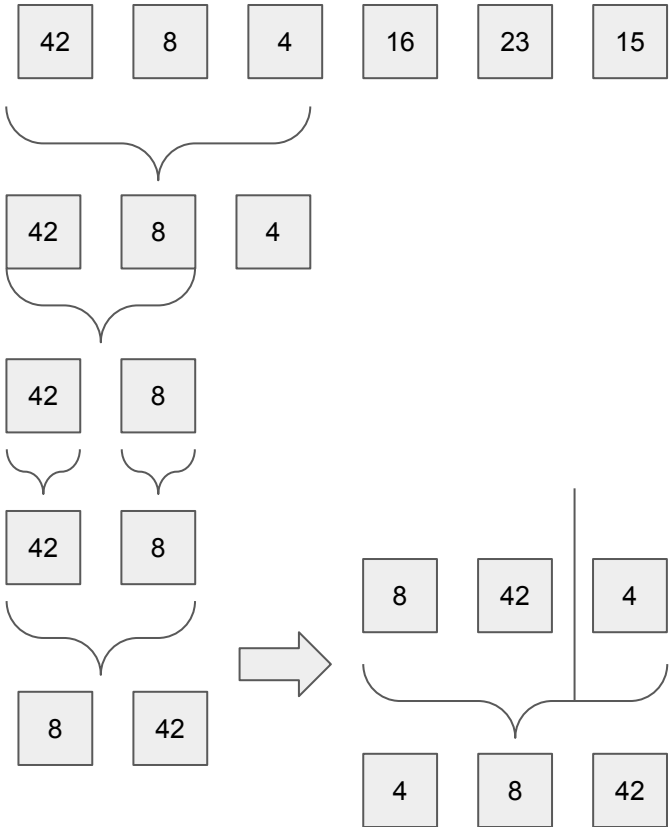
Merge Sort



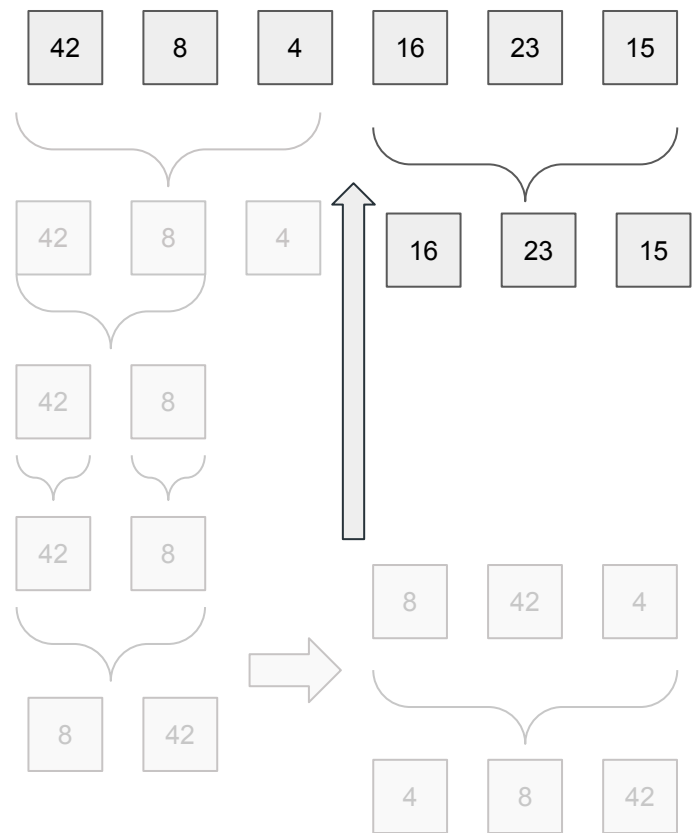
Merge Sort



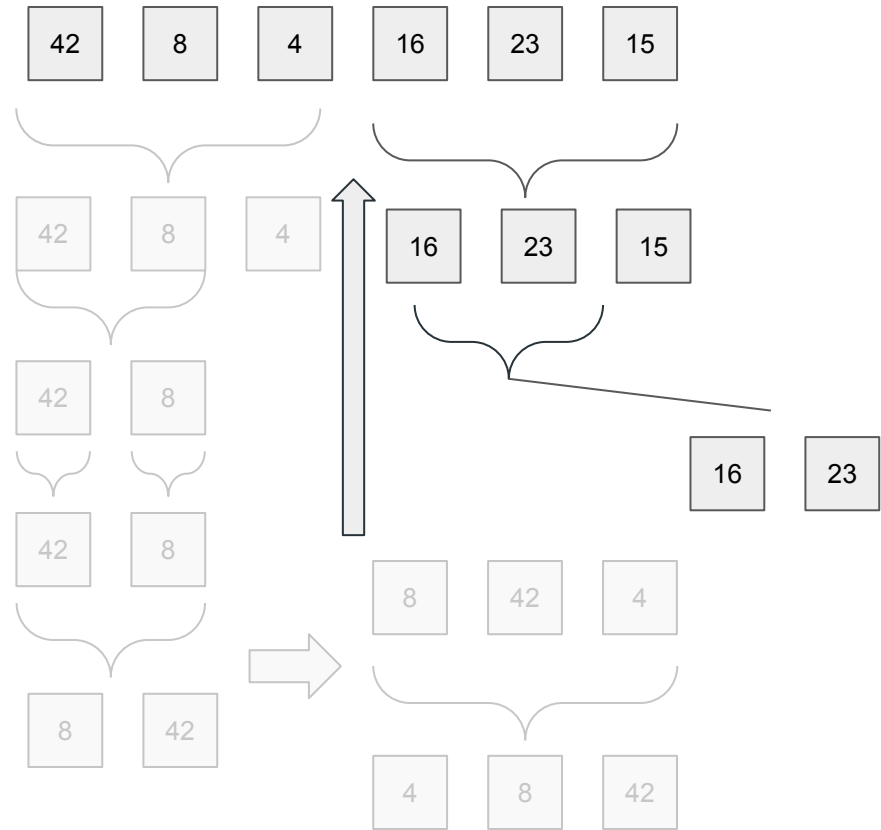
Merge Sort



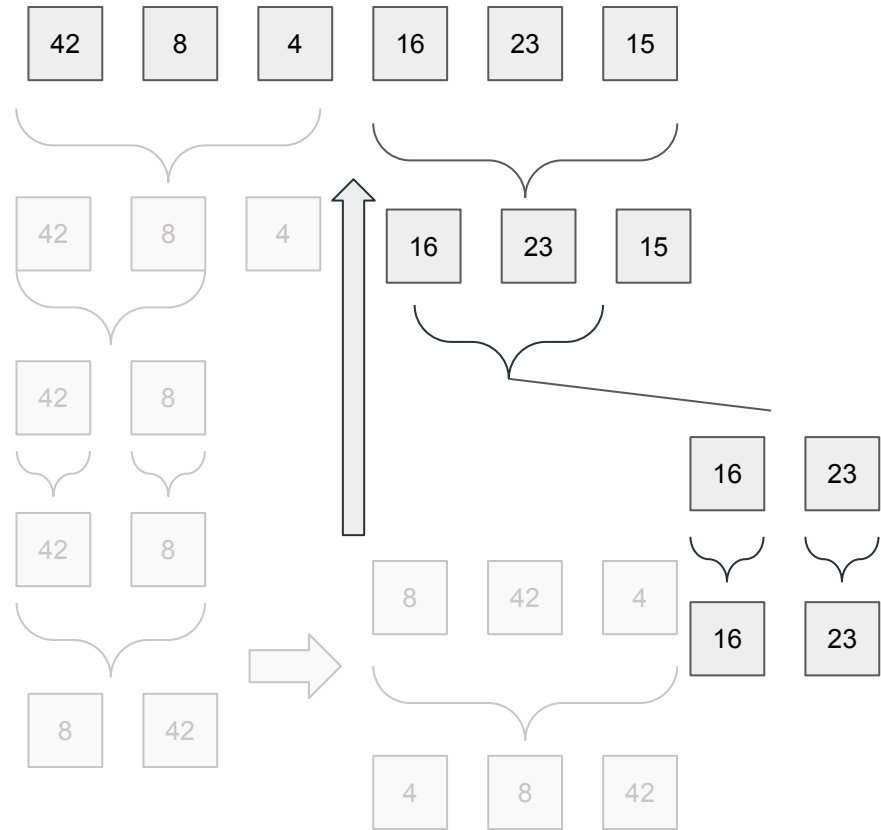
Merge Sort



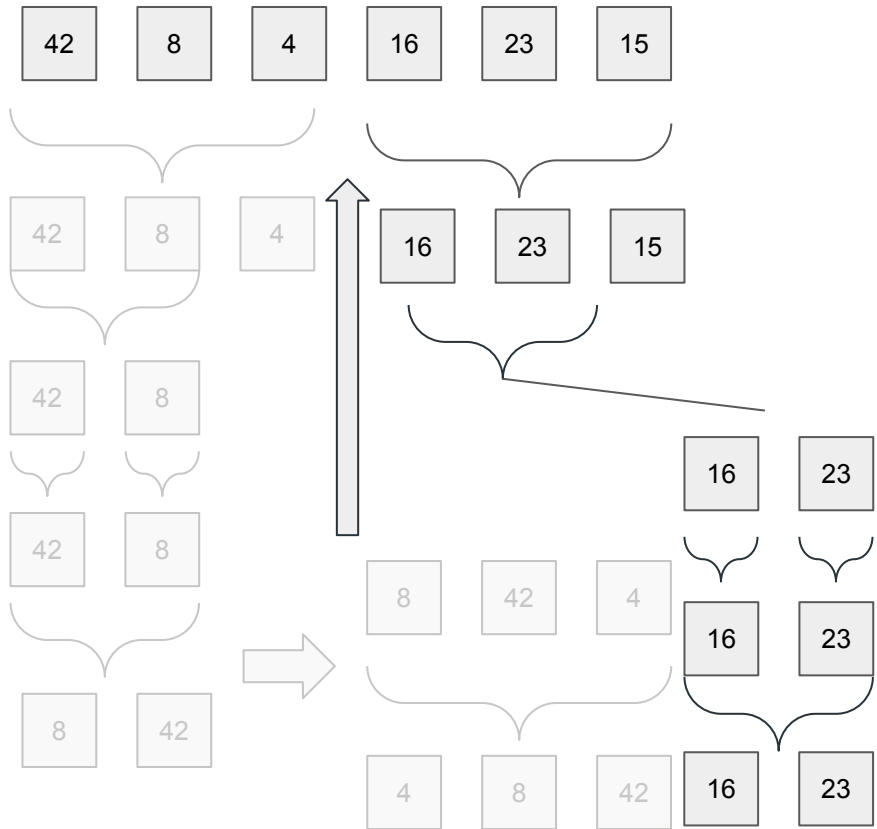
Merge Sort



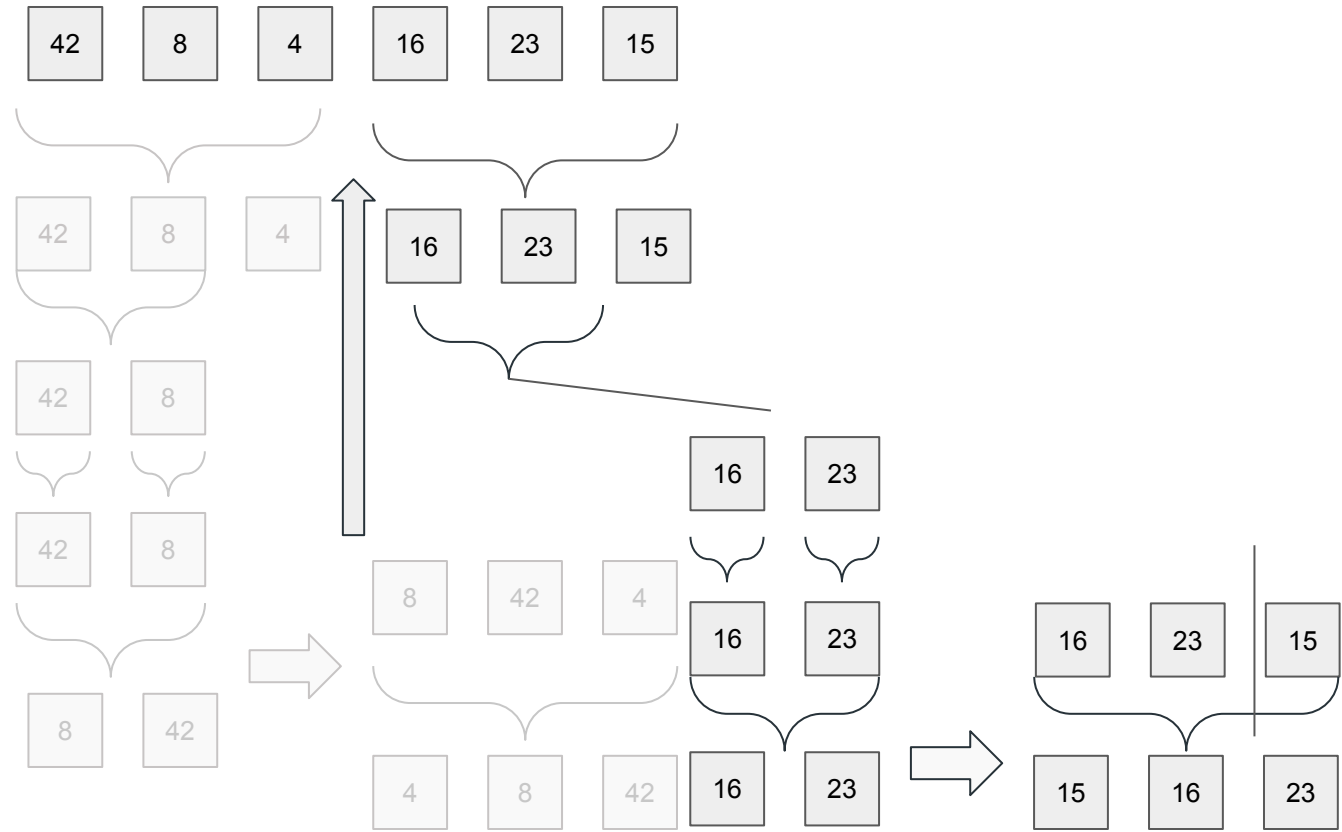
Merge Sort



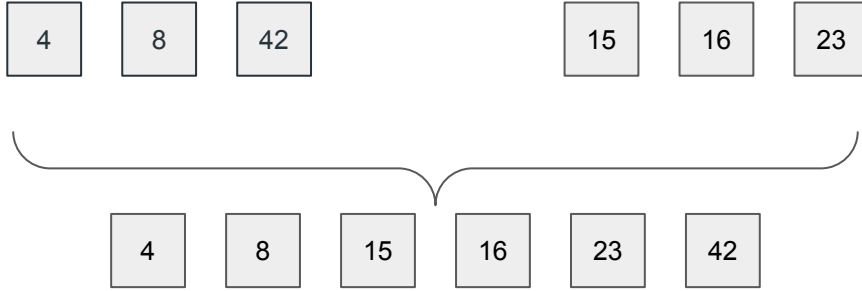
Merge Sort



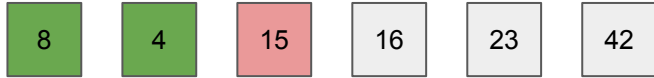
Merge Sort



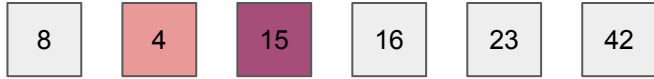
Merge Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



New Problems

- <https://codeforces.com/problemset/problem/1926/B>
- <https://codeforces.com/problemset/problem/1907/A>
- <https://codeforces.com/problemset/problem/1883/A>

<https://codeforces.com/problemset?tags=-800>

The Game!

Two players play a game.

Initially there are n integers a_1, a_2, \dots, a_n written on the board. Each turn a player selects one number and erases it from the board. This continues until there is only one number left on the board, i. e. $n-1$ turns are made. The first player makes the first move, then players alternate turns.

The first player wants to minimize the last number that would be left on the board, while the second player wants to maximize it.

You want to know what number will be left on the board after $n-1$ turns if both players make optimal moves.

The Game!

2 1 3

The first player erases 3

The second erases 1.

2 is left on the board.

Kill the enemy

There is an Agent and he has n weapons. The i -th weapon has a damage value a_i , and the Agent will face an enemy whose health value is H .

The Agent will perform one or more moves until the enemy dies.

In one move, he will choose a weapon and decrease the enemy's health by its damage value. The enemy will die when his health will become less than or equal to 0. However, not everything is so easy: the Agent can't choose the same weapon for 2 times in a row.

What is the minimum number of times that the Agent will need to use the weapons to kill the enemy?

Kill the enemy

Input:

2 4 (2- number of weapons, 4 - enemy health)

3 7 (3 - damage of first weapon, 7 - damage of second weapon)

Output: 1

The Agent can use the second weapon, making health value of the enemy equal to $4 - 7 = -3$. $-3 \leq 0$, so the enemy is dead, and using weapon 1 time was enough.

Kill the enemy

Input:

2 6 (2- number of weapons, 6 - enemy health)

4 2 (4 - damage of first weapon, 2 - damage of second weapon)

Output: 2

The Agent can use the first weapon first, and then the second one. After this, the health of enemy will drop to $6-4-2=0$, meaning he would be killed after using weapons 2 times.

Kill the enemy

Input:

3 11 (3- number of weapons, 11 - enemy health)

2 1 7 (2 - damage of first weapon, 1 - damage of second weapon, 7 - damage of third weapon)

Output: 3

The Agent can use the weapons in order (third, first, third), decreasing the health value of enemy to $11 - 7 - 2 - 7 = -5$ after using the weapons 3 times. Note that we can't kill the enemy by using the third weapon twice, as even though $11 - 7 - 7 < 0$, it's not allowed to use the same weapon twice in a row.

Barrels

You have n barrels lined up in a row, numbered from left to right from one. Initially, the i -th barrel contains a_i liters of water.

You can pour water from one barrel to another. In one act of pouring, you can choose two different barrels x and y (the x -th barrel shouldn't be empty) and pour any possible amount of water from barrel x to barrel y (possibly, all water). You may assume that barrels have infinite capacity, so you can pour any amount of water in each of them.

Calculate the maximum possible difference between the maximum and the minimum amount of water in the barrels, if you can pour water at most k times.

Some examples:

if you have four barrels, each containing 5 liters of water, and $k=1$, you may pour 5 liters from the second barrel into the fourth, so the amounts of water in the barrels are $[5, 0, 5, 10]$, and the difference between the maximum and the minimum is 10;

if all barrels are empty, you can't make any operation, so the difference between the maximum and the minimum amount is still 0

Barrels

Input:

4 1 (4 barrels, 1 move)

5 5 5 5 (capacity of each of the 4 barrels)

Output: 10

Barrels

Input:

4 1 (4 barrels, 1 move)

4 5 6 7 (capacity of each of the 4 barrels)

Output: 13

Barrels

Input:

3 2 (3 barrels, 2 moves)

0 0 0 (capacity of each of the 3 barrels)

Output: 0

ALP Laborator 4

Algoritmi de sortare

Probleme L3 - The Game!

Exemplu:

1 4 19 14 42 0 34 23

4 6 7 x 1 2 3 5

8 numere (n), 7 ($n-1$) miscari totale.

Primul jucator face 4 miscari $(n-1)/2 + 1$, al doilea - 3 $(n-1)/2$

Probleme L3 - The Game!

Daca lista ar fi sortata:

0 1 4 14 19 23 34 42

2 4 6 x 7 5 3 1

Si daca incepem indexarea de la 1:

Jucatorul 1 ar putea face numarul minim sa fie cel de pe pozitia $(n-1)/2+1$ (4) DE LA CAPAT,

Solutia fiind urmatorul numar DE LA CAPAT, adica numarul de pe pozitia $(n-1)/2+2$, adica pozitia 5 DE LA CAPAT.

Jucatorul 2 ar putea face numarul maxim sa fie cel de pe pozitia $(n-1)/2$, adica 3

Solutia ar fi numarul de pe pozitia $(n-1)/2 + 1$, adica pozitia 4 DE LA INCEPUT.

Pozitia 5 de la capat = Pozitia 4 de la inceput [14]

Solutia este pentru 1-indexed. Majoritatea limbajelor de programare sunt 0-indexed.

Probleme L3 - The Game!

Exemplu:

1 4 19 14 42 0 34

4 6 5 x 1 2 3

7 numere (n), 6 ($n-1$) mutari totale. Primul jucator face $3(n-1)/2$ miscari, al doilea - $(n-1)/2$.

Probleme L3 - The Game!

Daca lista ar fi sortata:

0 1 4 14 19 34 42

2 4 6 x 5 3 1

Si daca incepem indexarea de la 1:

Jucatorul 1 ar putea face numarul minim sa fie cel de pe pozitia $(n-1)/2$ (3) DE LA CAPAT,

Solutia fiind urmatorul numar DE LA CAPAT, adica numarul de pe pozitia $(n-1)/2 + 1$, adica pozitia 4 DE LA CAPAT.

Jucatorul 2 ar putea face numarul maxim sa fie cel de pe pozitia $(n-1)/2$, adica 3

Solutia ar fi numarul de pe pozitia $(n-1)/2 + 1$, adica pozitia 4 DE LA INCEPUT.

Pozitia 4 de la capat = Pozitia 4 de la inceput = [14]

Solutia este pentru 1-indexed. Majoritatea limbajelor de programare sunt 0-indexed.

Probleme L3 - Kill the enemy

Cea mai optima solutie ar fi sa folosim alternat doua cele mai mari arme.

Fie i cea mai mare arma si j a doua cea mai mare arma.

In primul caz avem $7(i)$ si $3(j)$

Daca viata este $1, 2 \dots 7$ ar trebui sa atacam cu o singura arma - i , $7 \Leftrightarrow 1$ data

Daca viata ar fi $8, 9$ sau 10 ar trebui sa atacam cu ambele arme. $[7, 3]$, $\Leftrightarrow 2$ ori.

Daca viata ar fi $11, 12 \dots 17$ ar trebui sa atacam cu $[7,3]$, $7 \Leftrightarrow 3$ ori.

Daca viata ar fi $18, 19$ sau 20 ar trebui sa atacam de $[7, 3]$, $[7, 3]$ $\Leftrightarrow 4$ ori

Daca viata ar fi $21, 22 \dots 27$ ar trebui sa atacam cu $[7,3]$, $[7,3]$, 7 , $\Leftrightarrow 5$ ori.

Daca viata ar fi $28, 29$ sau 30 ar trebui sa atacam de $[7, 3]$, $[7, 3]$, $[7,3]$ $\Leftrightarrow 6$ ori

Probleme L3 - Kill the enemy

Ca regula generala, observam ca atacam de exact $2 * H / (i+j)$ ori,

Daca H se imparte exact la $(i+j)$ $10 (\Leftrightarrow 2)$, $20 (\Leftrightarrow 4)$, $30 (\Leftrightarrow 6)$.

Daca impartirea are un rest mai adaugam 1 sau 2 atacuri in functie de cata viata ramane:

Daca mai ramane $\leq i$ viata, mai atacam o data dupa, deci $2 * H / (i+j) + 1$

Altfel mai atacam de doua ori dupa $2 * H / (i+j)$, deci $2 * H / (i+j) + 2$

Probleme L3 - Barrels

Solutia ar fi sa luam $k+1$ cele mai mari butoaie si sa turnam toata apa in unul din ele.

Ex:

Barrels = 4 5 6 7

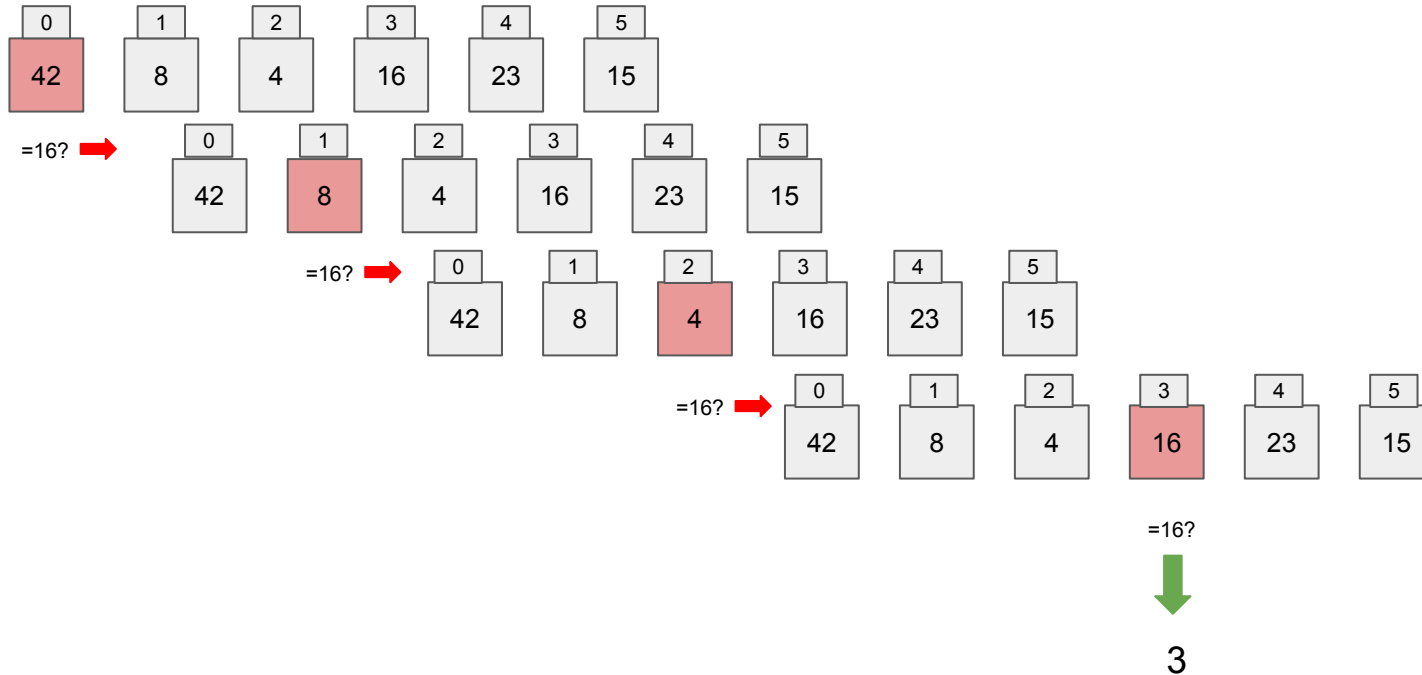
$K = 1$

Deci luam butoaiele 6 si 7, si dintr-o miscare mutam din 6 in 7 sau invers.

Practic, solutia ar fi suma a $k+1$ cele mai mari butoaie.

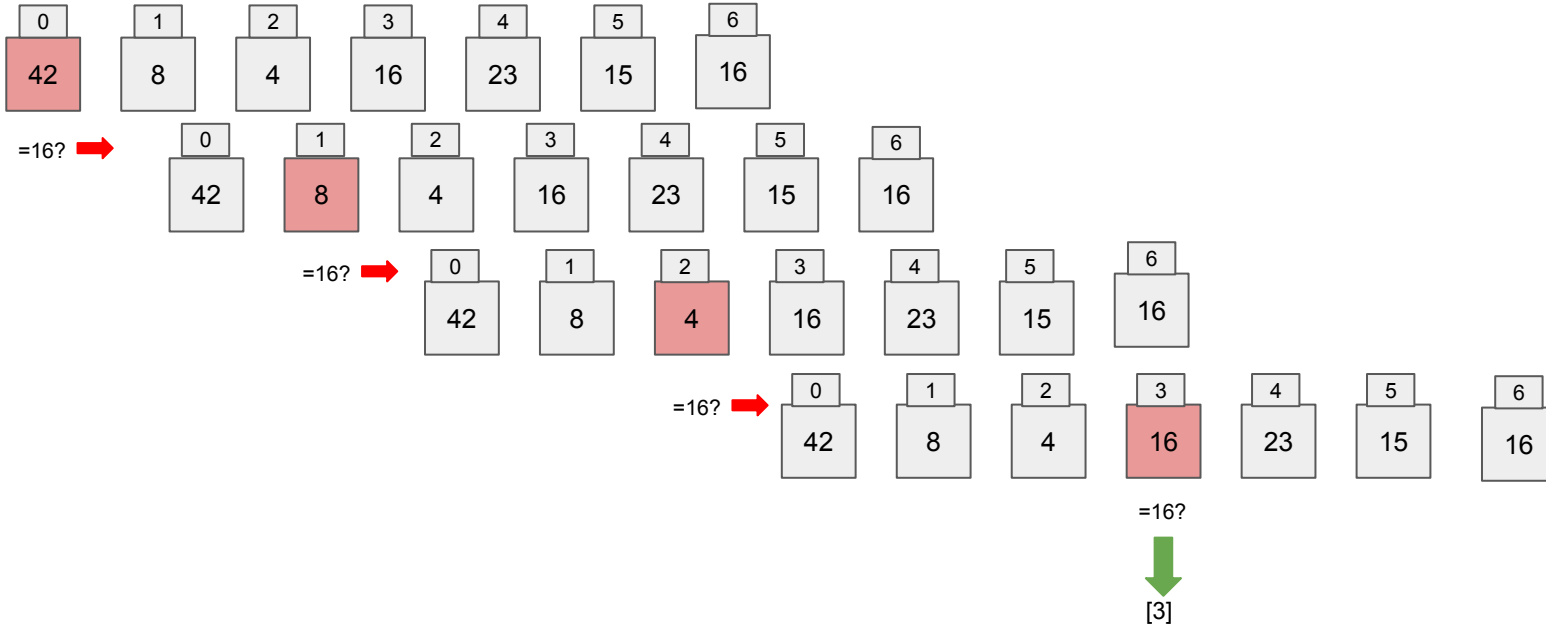
Cautare secventiala

Item = 16

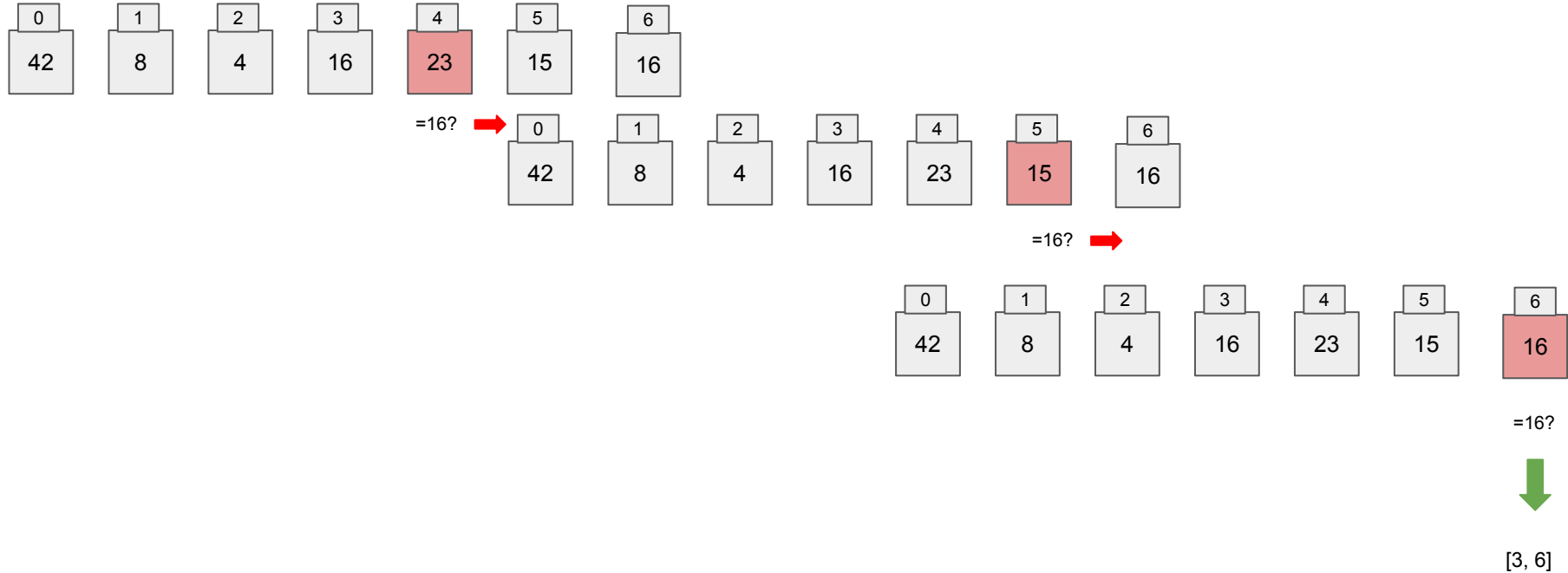


Cautare secventiala multipla

Item = 16

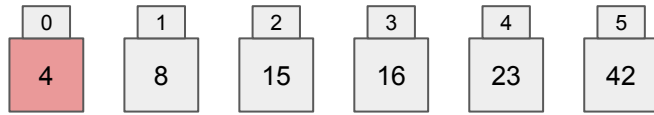


Cautare secventiala multipla



Cautare secventiala in liste ordonate

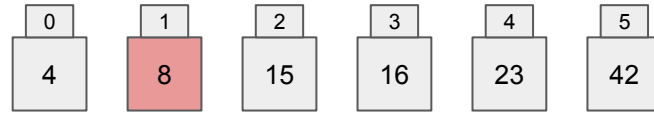
Item = 15



=15?



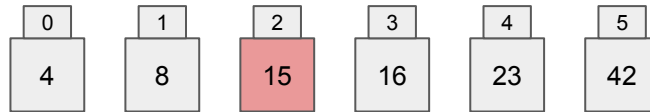
>15?



=15?



>15?



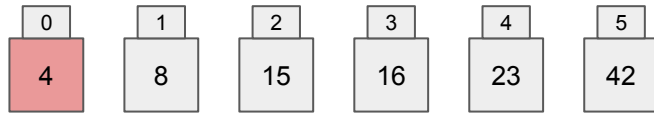
=15?



2

Cautare secventiala in liste ordonate

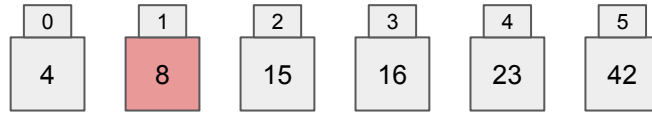
Item = 13



=13?



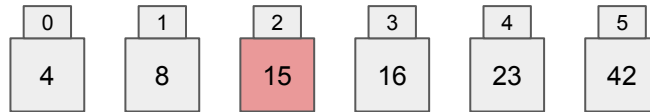
>13?



=13?



>13?



=13?



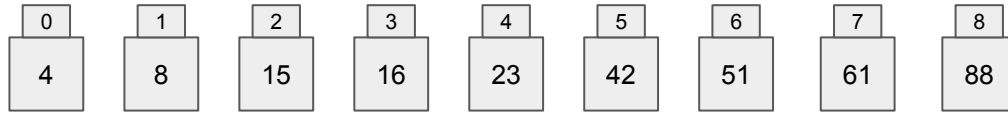
>13?



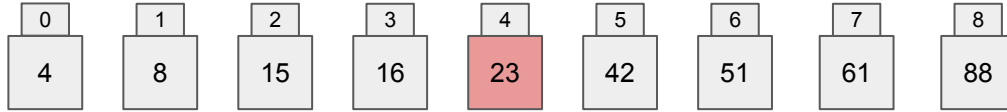
break; -1

Binary Search

Item = 61



Middle = $0 + (8-0)/2 = 4$

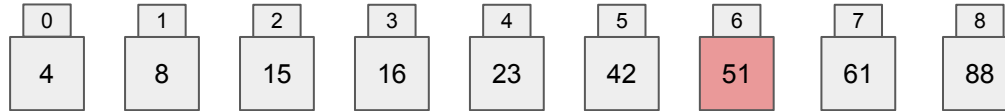


= 61?

> ? left = 5



Middle = $5 + (8-5)/2 = 5 + 1 = 6$



= 61?

> ? left = 7



Binary Search

$$\text{Middle} = 7 + (8-7)/2 = 7$$

0	1	2	3	4	5	6	7	8
4	8	15	16	23	42	51	61	88

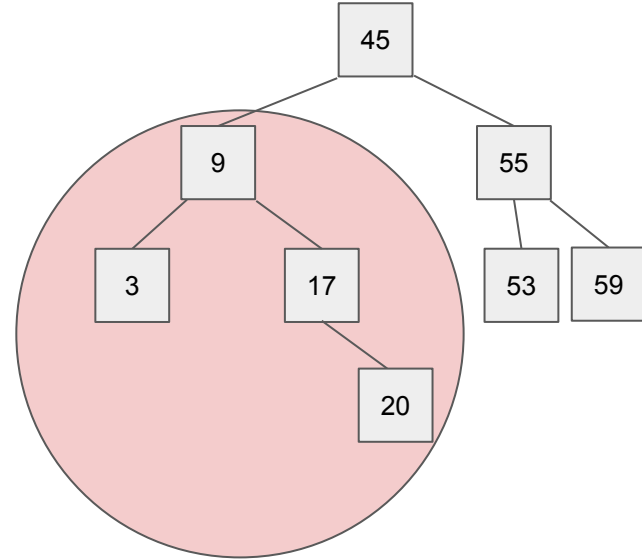
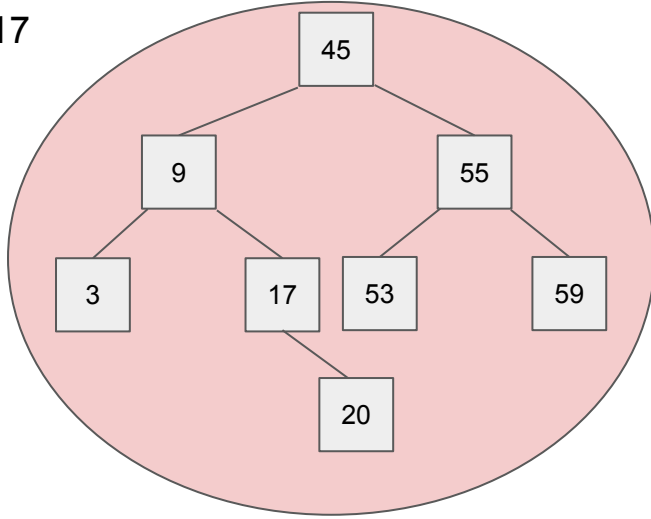
= 61?



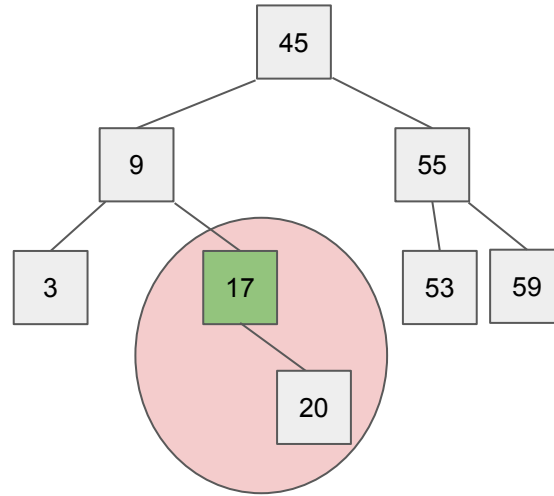
7

Binary Search Tree

17



Binary Search Tree



Probleme L1 - Angry Professor

Solutia 1 $O(n)$. Traversam toata lista de studenti si ii numaram pe cei care au venit la timp si comparam totalul cu minimul acceptat.

Solutia 2 $O(n)$. Filtram lista cu cei acceptati si numaram totalul elementelor din acea lista.

Solutia 3. $O(n \log n)$. Sortam lista si verificam daca elementul pe indexul [minim acceptat -1] este mai mare decat 0 (studentul este intarziat)

Probleme L1 - Max Product in Array

Solutia 1 $O(n^2)$. Iteram prin toate elementele de 2 ori si le inmultim pe toate intre ele, memorand si actualizand cel mai mare numar obtinut.

Solutia 2 $O(n \log n)$. Sortam lista si verificam care dintre produse este mai mare: $\text{index}[0] * \text{index}[1]$ sau $\text{index}[\text{length}-1] * \text{index}[\text{length}-2]$

Solutia 3 $O(n)$. Traversam o singura data toata lista si actualizam 4 numere: 2 cele mai mari si 2 cele mai mici. La final returnam produsul cel mai mare dintre 2 cele mai mici sau 2 cele mai mari numere.

Probleme L1 - Pair with given sum

Solutia 1 $O(n^2)$ aka Brute Force. Iteram prin toata lista de 2 ori, adunam numerele si daca suma este potrivita, afisam perechea.

Solutia 2 $O(n \log n)$. Sortam lista, dupa care iteram si calculam prin ea actualizand indecsii de traversare.

Solutia 3 $O(n)$. Iteram o singura data prin lista si salvam numerele intr-un dictionar (Map), verificand daca diferenta dintre suma si iteratorul curent exista in acel dictionar.

Probleme L1 - DoorsAndKeys

Solutia consta in definirea unor liste “paralele” pentru usi si chei, dupa care in sirul de caractere de test se va compara indexul pentru fiecare element din cele doua liste. Daca cel din lista de chei este mai mic decat cel din lista de usi pentru toate cele trei elemente, afisam “DA”, altfel - “NU”.

Probleme L1 - Not Shading

Exista 4 variante posibile.

Toate elementele sunt W, deci returnam -1.

Elementul este deja B, deci returnam 0.

Coloana sau randul pe care se afla elementul are o valoare B, deci returnam 1.

Coloana si randul nu contin o valoare B, insa o astfel de valoare exista in intreaga matrice, deci returnam 2.

Another Game

You are playing a very popular computer game. The next level consists of n consecutive locations, numbered from 1 to n , each of them containing either land or water. It is known that the first and last locations contain land, and for completing the level you have to move from the first location to the last. Also, if you become inside a location with water, you will die, so you can only move between locations with land.

You can jump between adjacent locations for free, as well as no more than once jump from any location with land i to any location with land $i+x$, spending x coins ($x \geq 0$).

Your task is to spend the minimum possible number of coins to move from the first location to the last one.

Another Game

1 0 1 0 1

The only way to move from the first location to the last one is to jump between them, which will cost 4 coins.

1 0 1 1

You can jump from the first location to the third for 2 coins, and then jump to the fourth location for free, so the answer is 2.

Consecutive sum riddle

Tom Marvollo has a riddle for you and if you manage to solve it, he will give you a Butterbeer.

You are given an integer n . You need to find two integers l and r such that $-10^{18} \leq l < r \leq 10^{18}$ and $l + (l+1) + \dots + (r-1) + r = n$.

1

$$0+1=1. \Rightarrow 0, 1$$

2

$$(-1)+0+1+2=2. \Rightarrow -1, 2$$

Consecutive sum riddle

6

$$1+2+3=6. \Rightarrow 1,3$$

100

$$18+19+20+21+22=100. \Rightarrow 18, 22$$

25

$$(-2)+(-1)+0+1+2+3+4+5+6+7=25. \Rightarrow -2, 7$$

H-index

Jorge is a physicist who has published many research papers and wants to know how much impact they have had in the academic community. To measure impact, he has developed a metric, H-index, to score each of his papers based on the number of times it has been cited by other papers. Specifically, the H-index score of a researcher is the largest integer H such that the researcher has H papers with at least H citations each.

Jorge has written N papers in his lifetime. The i -th paper has C_i citations. Each paper was written sequentially in the order provided, and the number of citations that each paper has will never change. Please help Jorge determine his H-index score after each paper he wrote.

H-index

3 papers => 5 1 2 (citations)

After the 1st paper, Jorge's H-index score is 1, since he has 1 paper with at least 1 citation.

After the 2nd paper, Jorge's H-index score is still 1.

After the 3rd paper, Jorge's H-index score is 2, since he has 2 papers with at least 2 citations (the 1st and 3rd papers).

H-index

6 papers => 1 3 3 2 2 15 (citations)

After the 1st paper, Jorge's H-index score is 1, since he has 1 paper with at least 1 citation.

After the 2nd paper, Jorge's H-index score is still 1.

After the 3rd paper, Jorge's H-index score is 2, since he has 2 papers with at least 2 citations (the 2nd and 3rd papers).

After the 4th paper, Jorge's H-index score is still 2.

After the 5th paper, Jorge's H-index score is still 2.

After the 6th paper, Jorge's H-index score is 3, since he has 3 papers with at least 3 citations (the 2nd, 3rd and 6th papers).

ALP Lab 5

Backtracking




































Starting from a **starting point**, create a **candidate solution** iteratively/recursively, checking its **validity** and then check if its a **final solution**.

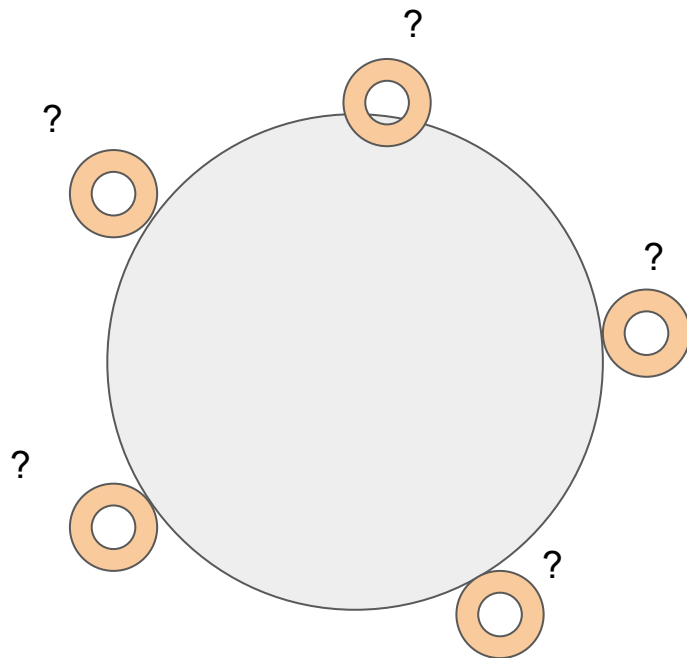
Three important sequences:

- Method called iteratively/recursively;
- Validation of the candidate solution;
- Verification of the final solution.

Displaying the solution.

1. Party

		x1	x2	x3	x4	x5
						
x1						
x2						
x3						
x4						
x5						



1. Party

The process of finding the solution:

Sequentially, we seat each person at the table.

If the person's seat is valid,

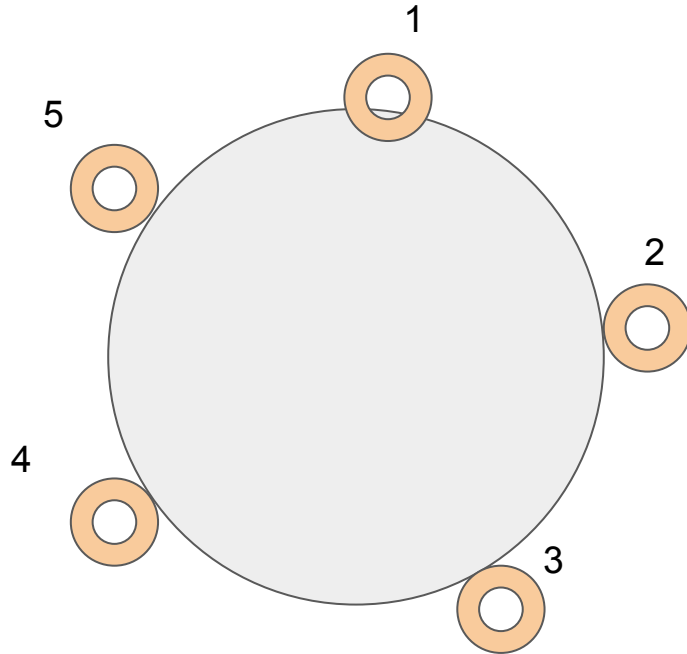
We check if the seating arrangement is final:

If yes, we display the solution; if no, we continue with the next person.

1. Party

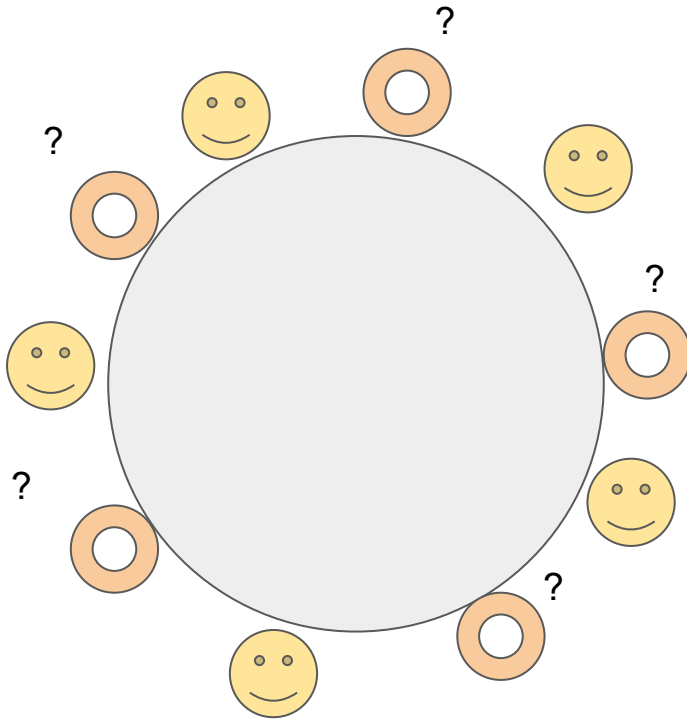
Is solution?

$N = 5$

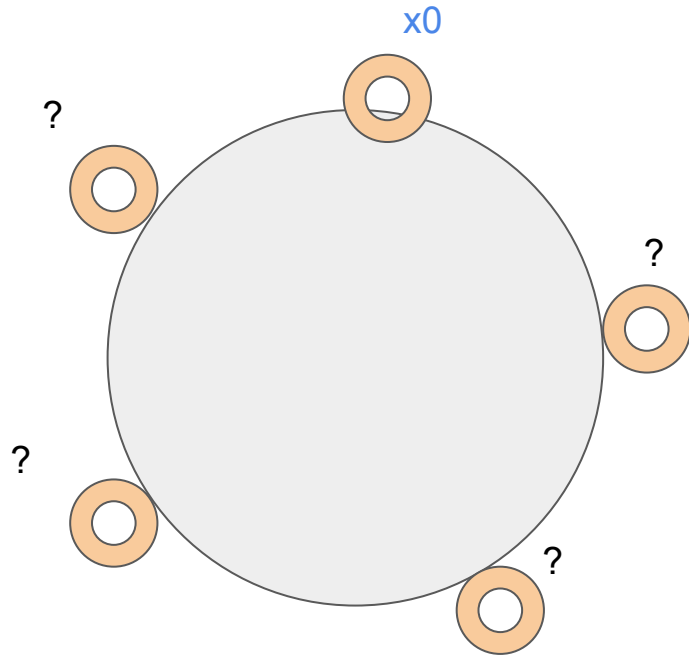


1. Party

Is Valid?



1 Party



Is valid?

True

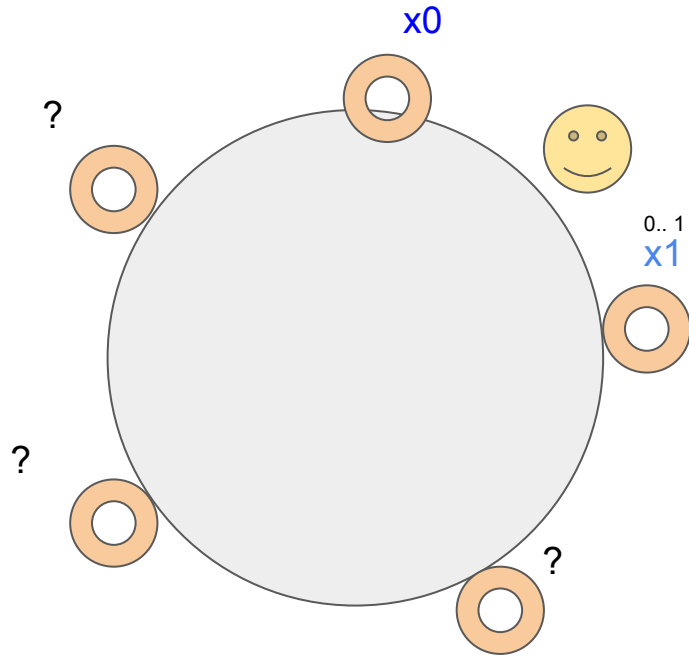


Is solution?

False



1 Party



Is valid?

True

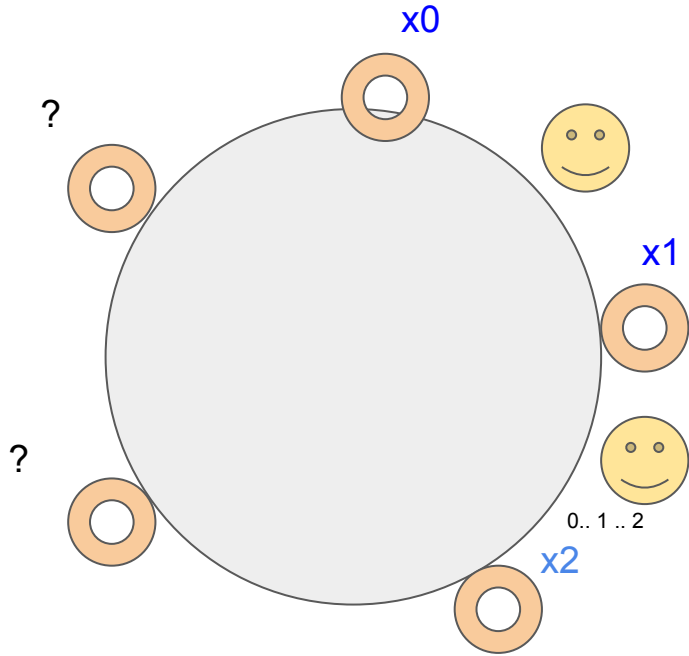


Is solution?

False



1 Party



Is valid?

True

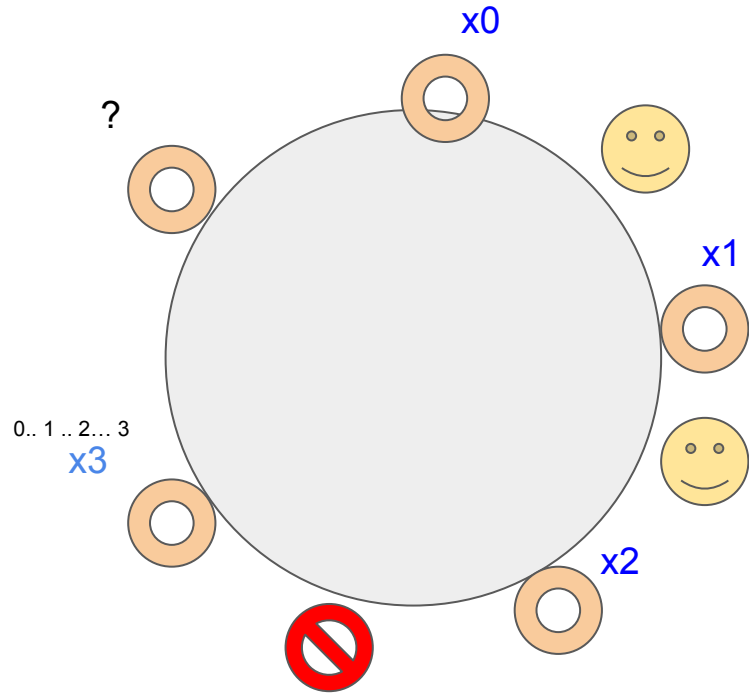


Is solution?

False



1 Party



Is valid?

False

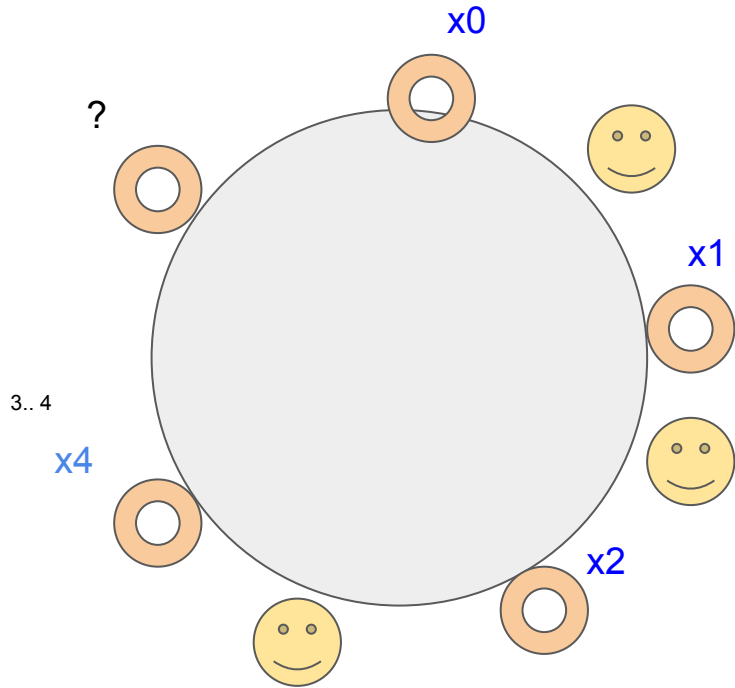


Is solution?

False



1 Party



Is valid?

True

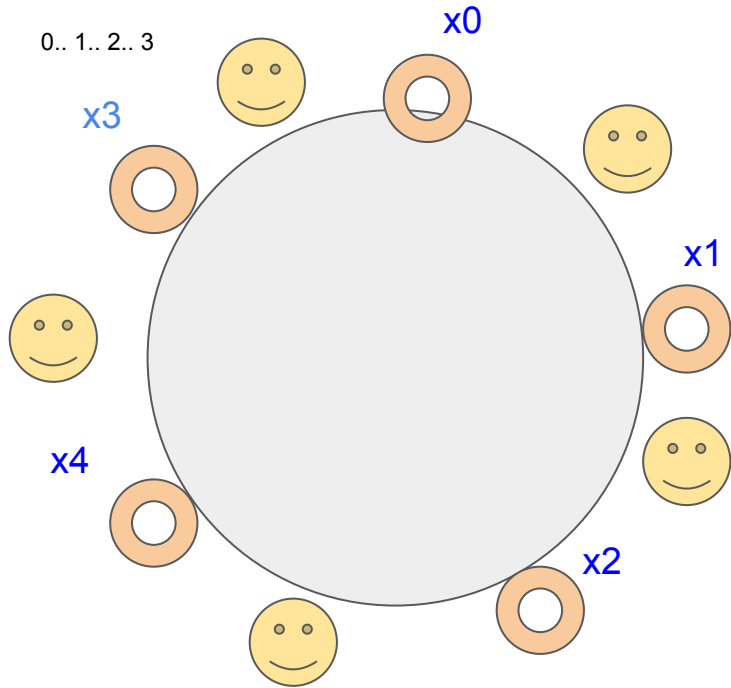


Is solution?

False



1 Party



Is valid?

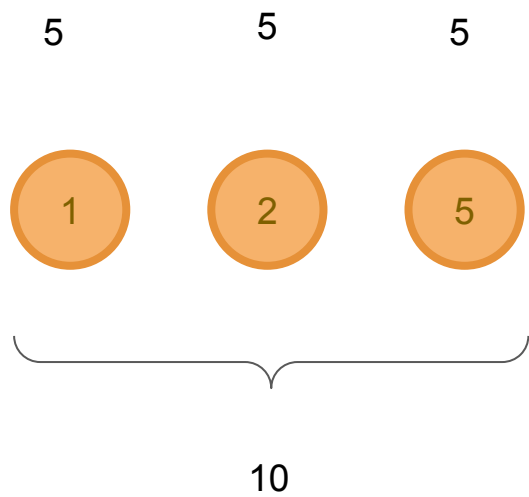
True



Is solution?

True

2 Coins



2. Coins

Is valid?

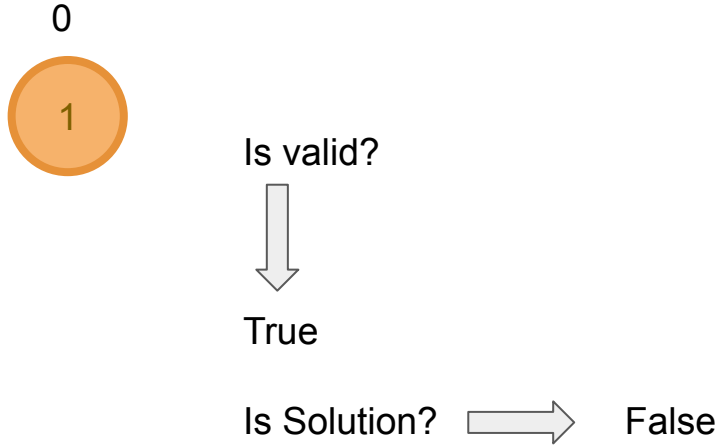
The sum is \leq than the set amount and the max coins for current coin type is not exceeded.

2. Coins

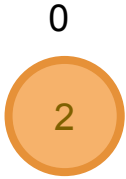
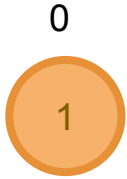
Is solution?

The sum is exactly the requested amount.

2. Coins



2. Coins



Is valid?



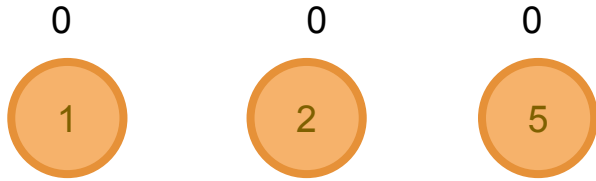
True

Is Solution?



False

2. Coins



Is valid?



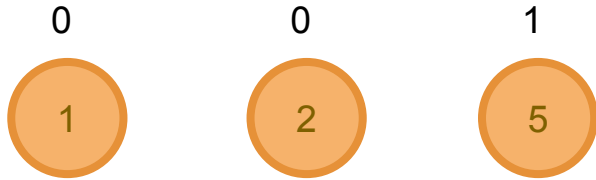
True

Is Solution?



False

2. Coins



Is valid?



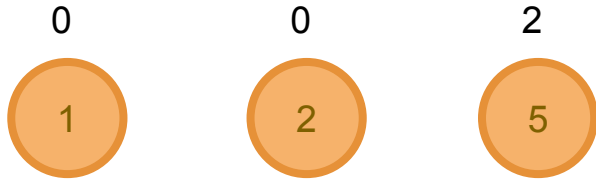
True

Is Solution?



False

2. Coins



Is valid?



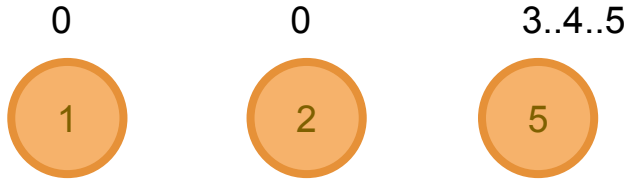
True

Is Solution?



True

2. Coins

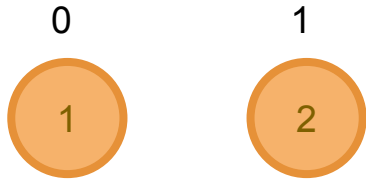


Is valid?



False

2. Coins



Is valid?



True

Is Solution?



False

2. Coins



Is valid?



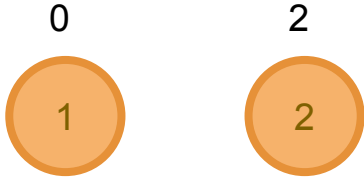
True

Is Solution?



False

2. Coins



Is valid?



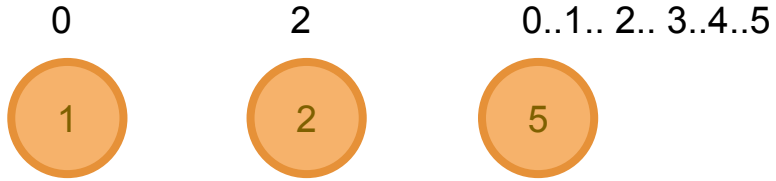
True

Is Solution?



False

2. Coins



Is valid?



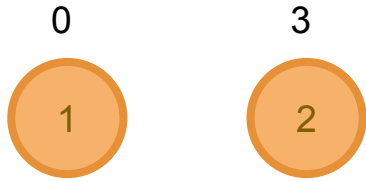
True

Is Solution?



False

2. Coins



Is valid?



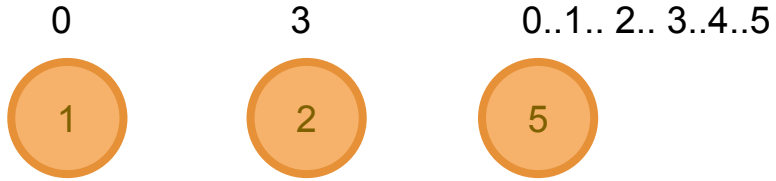
True

Is Solution?



False

2. Coins



Is valid?



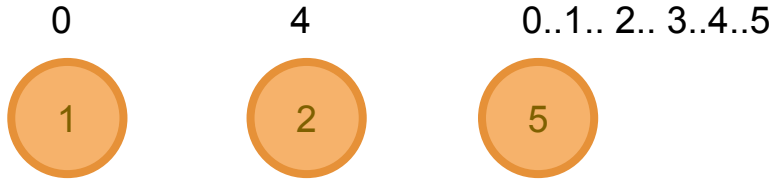
True

Is Solution?



False

2. Coins



Is valid?



True

Is Solution?



False

2. Coins



Is valid?



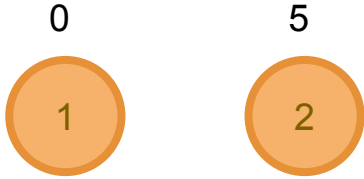
True

Is Solution?



False

2. Coins



Is valid?



True

Is Solution?



True

3. Chess Queens

-	Q	-	-
-	-	-	Q
Q	-	-	-
-	-	Q	-

N= 4 queens
NxN = 4x4 grid

3. Chess Queens

Is valid?

A new queen is not on the same row, column or diagonal.

Is solution?

All N queens are placed.

3. Chess Queens

Q	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-

3. Chess Queens

.....

Q	-	-	-
-	-	Q	-
-	-	-	-
-	-	-	-

3. Chess Queens

.....

Q	-	-	-
-	-	Q	-
-	-	-	-
-	Q	-	-



Cannot put a queen anywhere here



3. Chess Queens

Q	-	-	-
-	-	-	Q
-	-	-	-
-	-	-	-

3. Chess Queens

.....

Q	-	-	-
-	-	-	Q
-	Q	-	-
-	-	-	-

3. Chess Queens

.....

Q	-	-	-
-	-	-	Q
-	Q	-	-
-	-	-	-



Cannot put a queen anywhere here

3. Chess Queens

-	Q	-	-
-	-	-	-
-	-	-	-
-	-	-	-

3. Chess Queens

.....

-	Q	-	-
-	-	-	Q
-	-	-	-
-	-	-	-

3. Chess Queens

-	Q	-	-
-	-	-	Q
Q	-	-	-
-	-	-	-

3. Chess Queens

.....

-	Q	-	-
-	-	-	Q
Q	-	-	-
-	-	Q	-



Is solution = True

4. Sudoku

3		6	5		8	4		
5	2							
	8	7					3	1
		3		1			8	
9			8	6	3			5
	5			9		6		
1	3					2	5	
							7	4
		5	2		6	3		

4. Sudoku

Is valid?

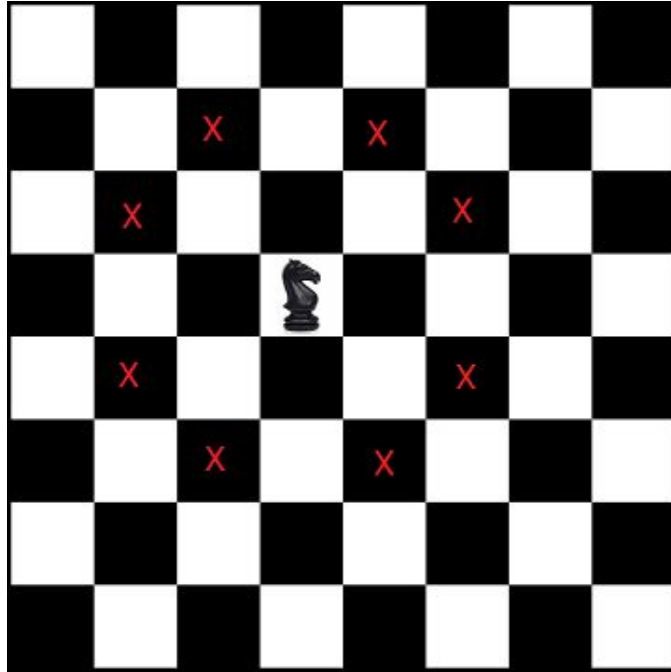
- Values in a row are [1-9];
- Values in a column are [1-9];
- Values in a 3x3 square are [1-9].

Is Solution?

- There are no more values to fill

5. Knight Tours

Given a chessboard, print all sequences of moves of a knight on a chessboard such that the knight visits every square only once.



5. Knight Tours

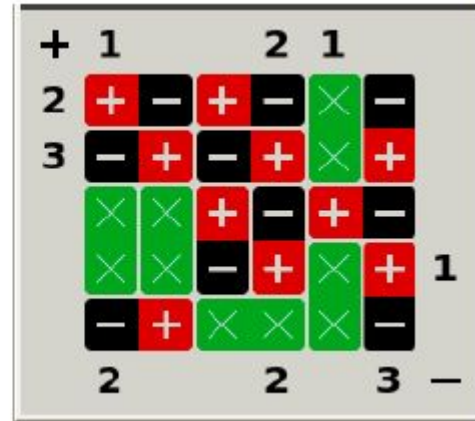
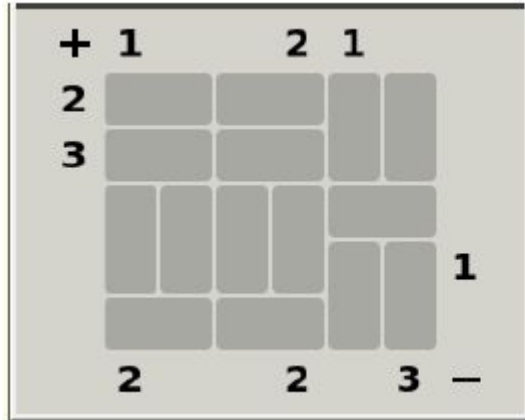
Is Valid?

- The destination exists on the board

Is Solution?

- All squares are visited.

6. Magnet Puzzle



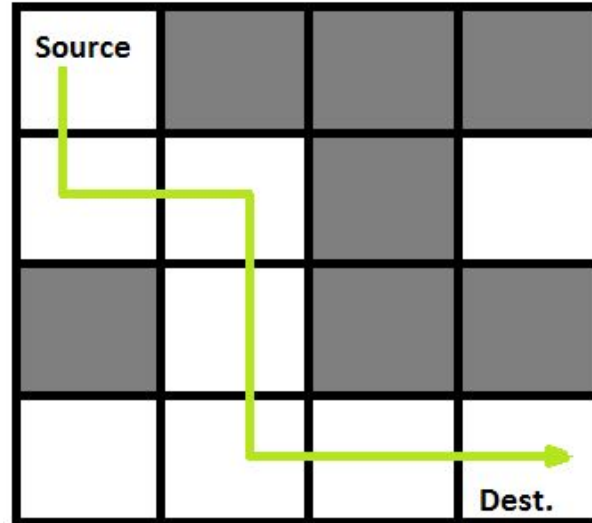
6. Magnet Puzzle

The puzzle game Magnets involves placing a set of domino-shaped magnets (or electrets or other polarized objects) in a subset of slots on a board so as to satisfy a set of constraints.

Each slot contains either a blank entry (indicated by 'x's), or a “magnet” with a positive and negative end. The numbers along the left and top sides show the numbers of '+' squares in particular rows or columns. Those along the right and bottom show the number of '-' signs in particular rows or columns. Rows and columns without a number at one or both ends are unconstrained as to the number of '+' or '-' signs, depending on which number is not present. In addition to fulfilling these numerical constraints, a puzzle solution must also satisfy the constraint that no two orthogonally touching squares may have the same sign (diagonally joined squares are not constrained).

7. Rat maze

A Maze is given as $N \times N$ binary matrix of blocks where source block is the upper left most block i.e., `maze[0][0]` and destination block is lower rightmost block i.e., `maze[N-1][N-1]`. A rat starts from source and has to reach the destination. The rat can move only in two directions: forward and down.



8. Word Matrix

Given an $M \times N$ boggle board, find a list of all possible words that can be formed by a sequence of adjacent characters on the board.

Consider the following the traditional 4×4 boggle board. If the input dictionary is [START, NOTE, SAND, STONED], the valid words are [NOTE, SAND, STONED].

M	S	E	F
R	A	T	D
L	O	N	E
K	A	F	B

4 x 4 boggle board

Problems

Given an array of values, print out all permutations of that array.