

Algorithm Complexity I

DCA3702

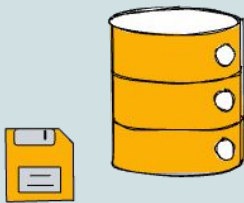
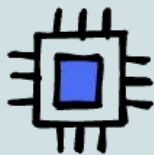
ivanovitch.silva@ufrn.br
@ivanovitchm



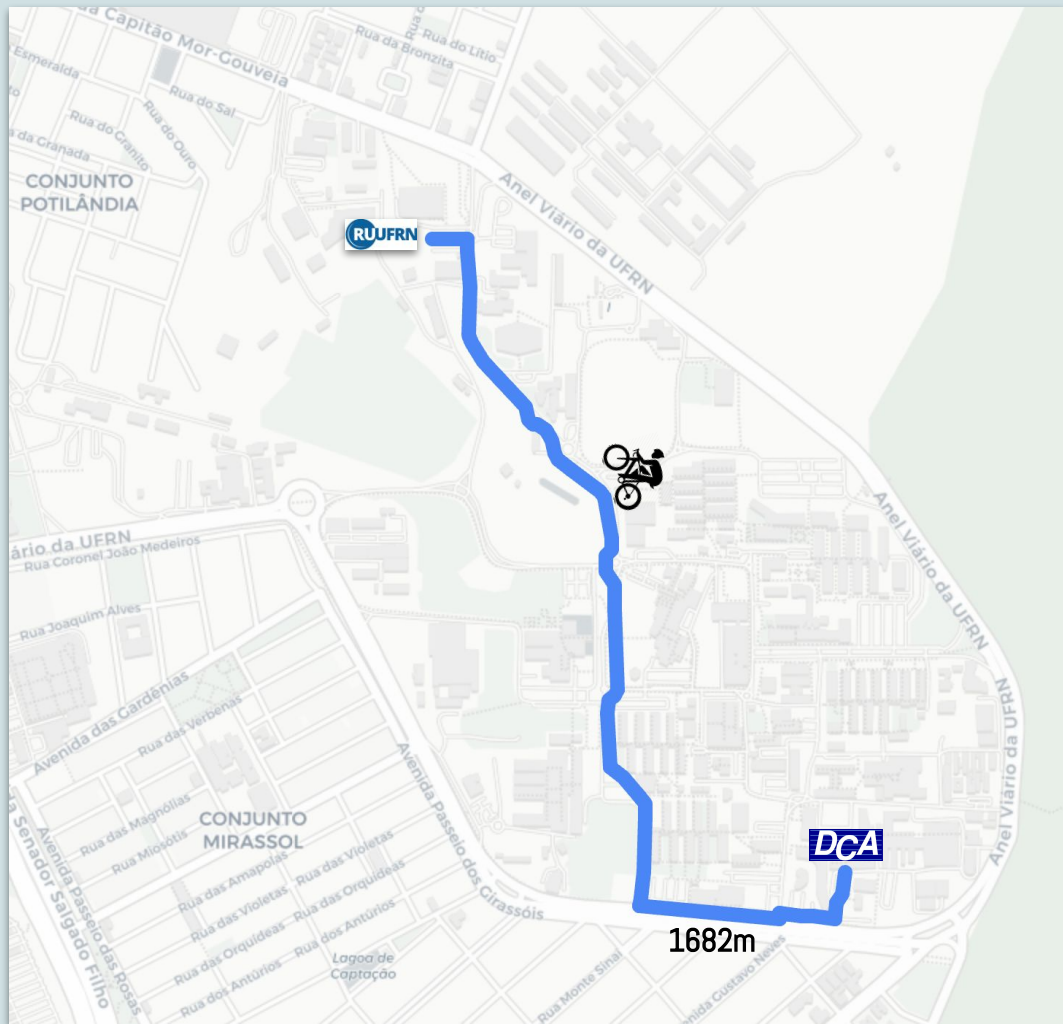
How to compare different algorithms and find the one that will perform **the best** ?



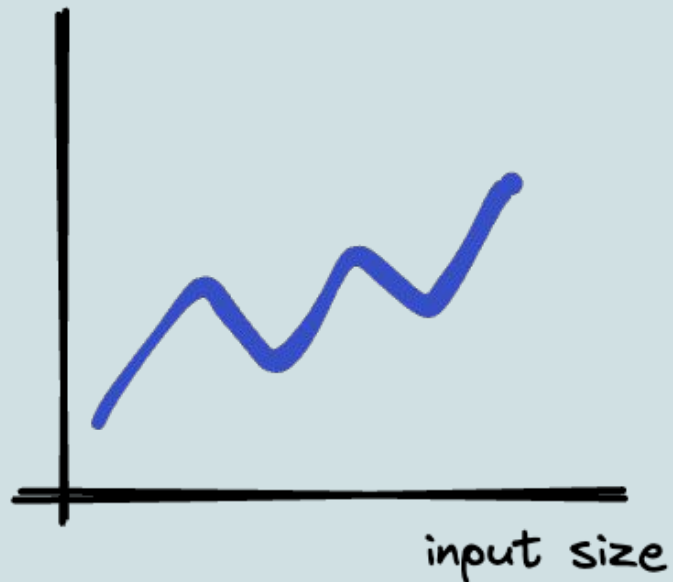
Time



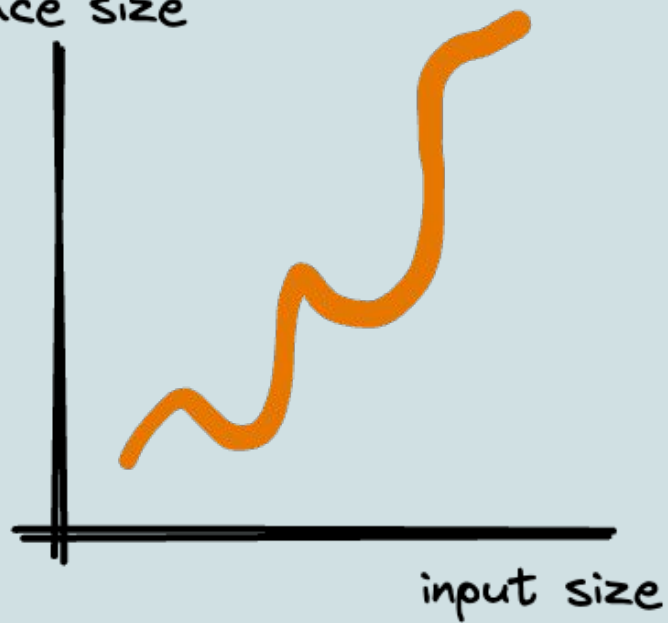
Space



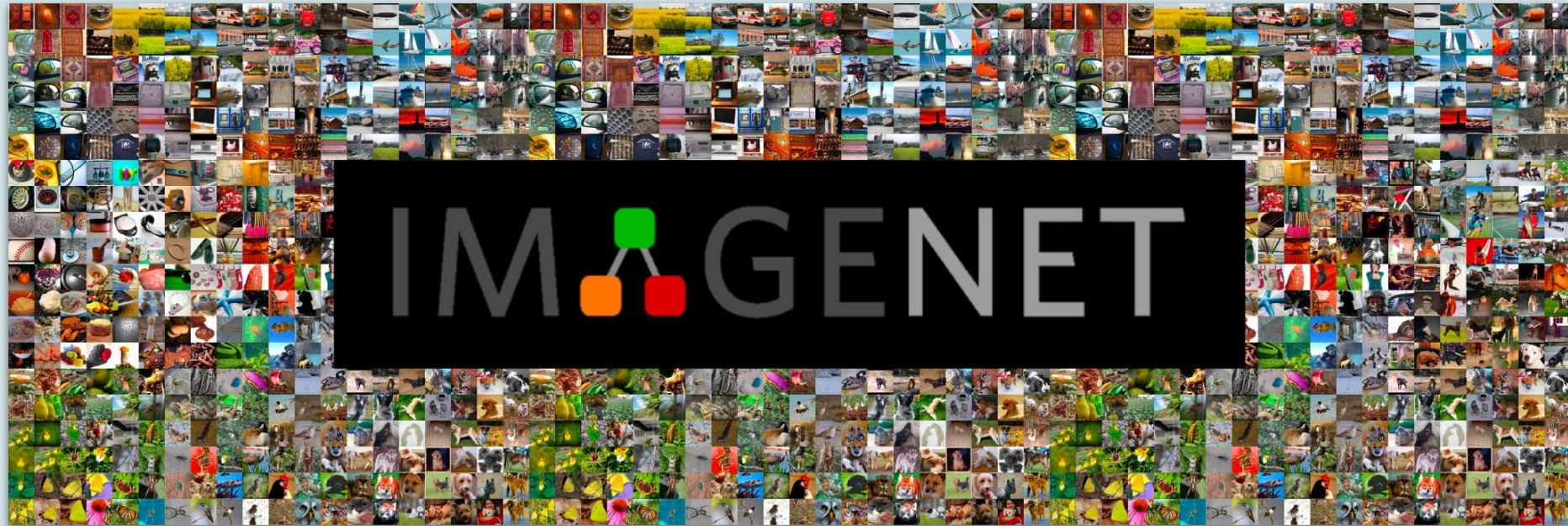
number of
operations



auxiliary
space size

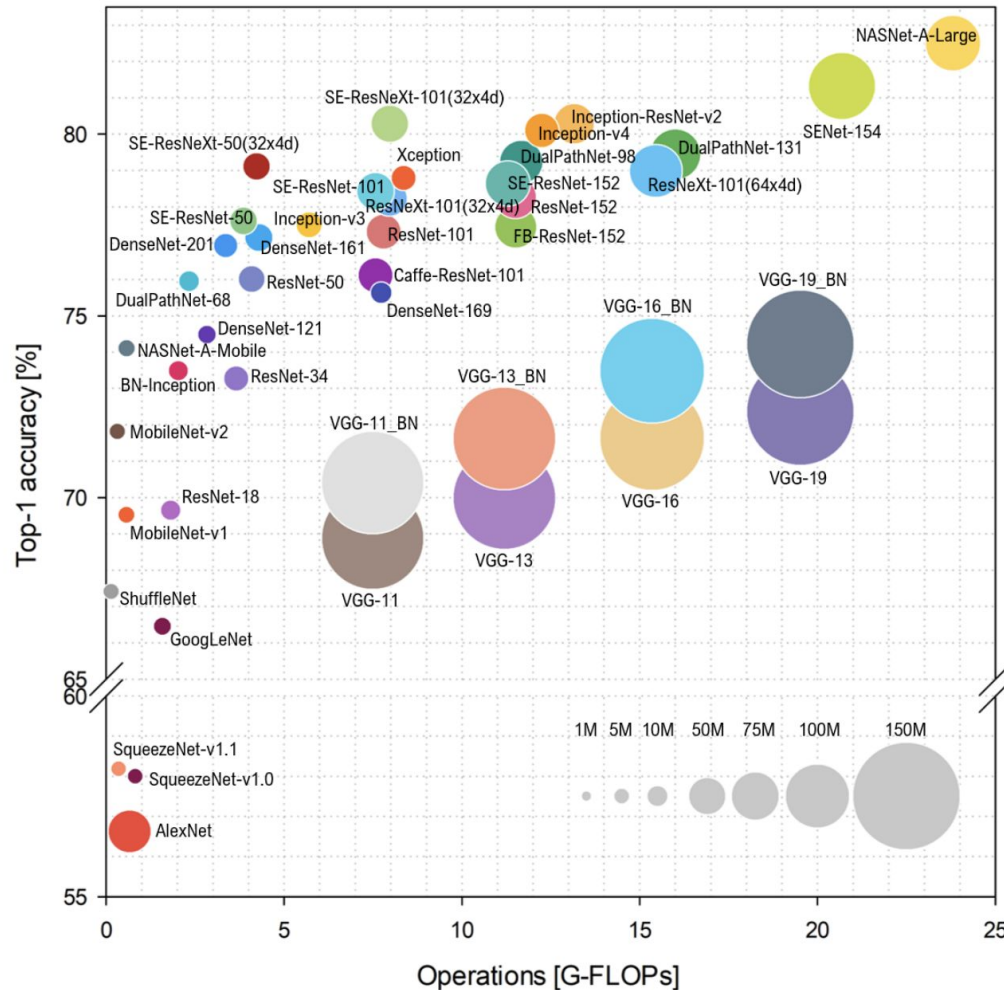


14,197,122 images, 21841 synsets indexed



<https://www.image-net.org/>

Algorithm Complexity



ML Model Evolution ImageNet

Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures". IEEE Access, vol. 6, 2018.

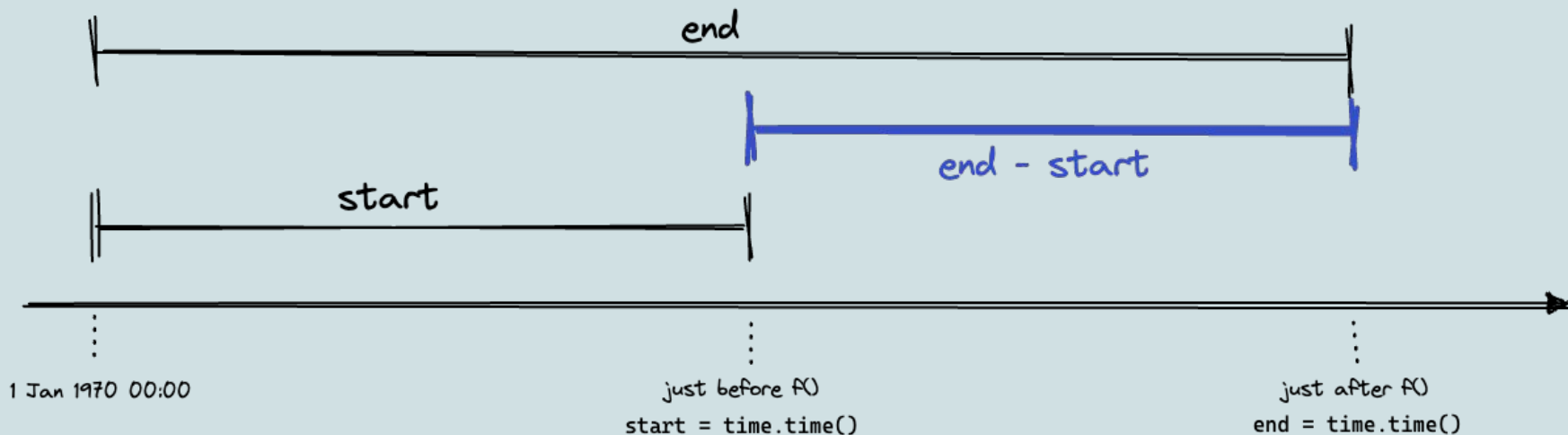


```
def maximum(values):  
    answer = None  
    for value in values:  
        if answer == None or answer < value:  
            answer = value  
    return answer
```

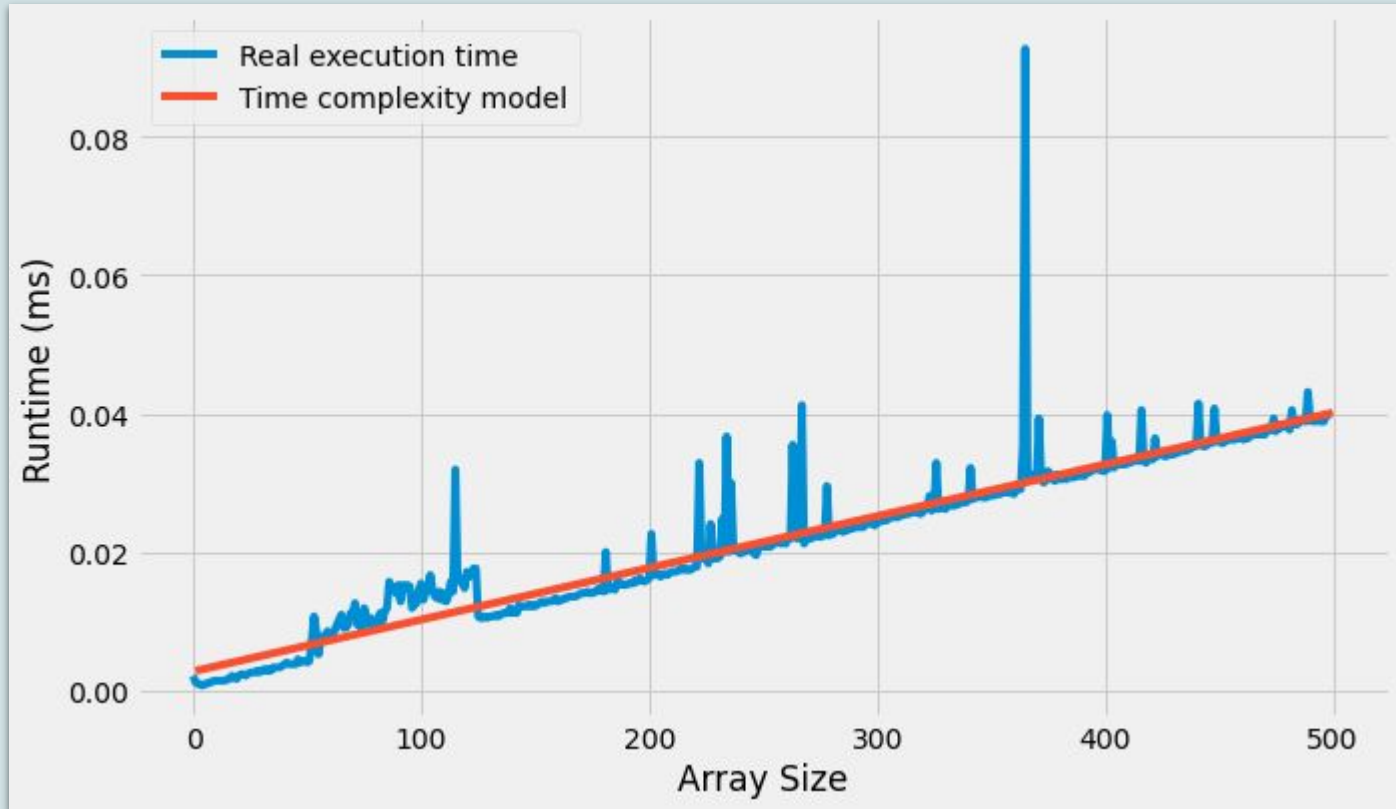
If we double the input, do we double the execution time, do we quadruple it, or something else entirely?

```
import time
start = time.time()
f()
end = time.time()
runtime = end - start
```

How to measure
the execution
times ?



Exec **maximum** (*values*) 500 times change the *values* size





information

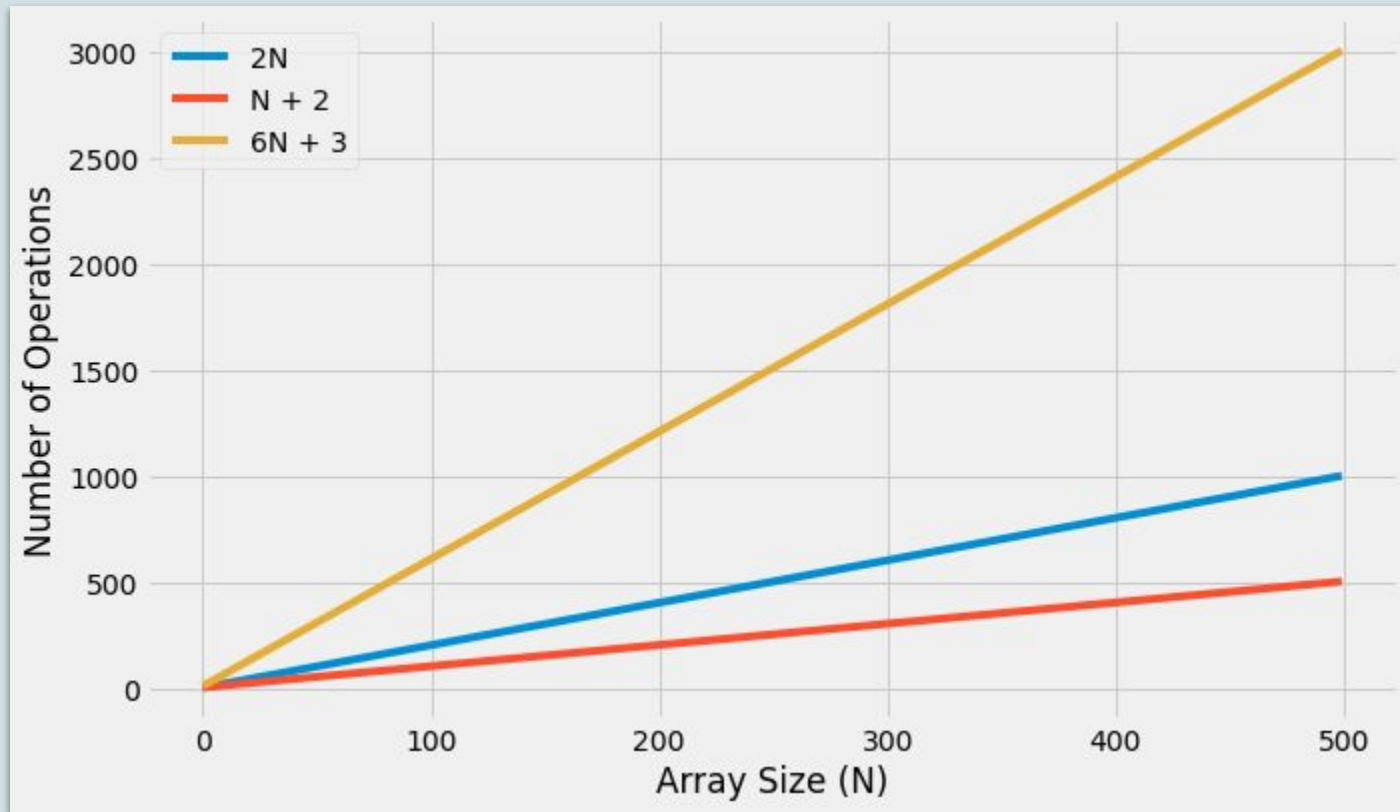
Recall that we don't want to know
the exact execution time, just how fast it is growing

```
def maximum(values):  
    answer = None           # c1    1 time  
    for value in values:    # c2    N times  
        if answer == None or answer < value: # c3    N times  
            answer = value    # c4    N times  
    return answer           # c5    1 time
```

$$C_1 + C_2 \times N + C_3 \times N + C_4 \times N + C_5$$

$$(C_1 + C_5) + N(C_2 + C_3 + C_4)$$

$$\alpha + N\beta$$



Regardless of the values of α and β , the function $\alpha + N\beta$ is a straight line. We call an algorithm whose time complexity is a straight line a **linear time algorithm**.



Best-Case



Average-Case



Worst-Case

```
def maximum(values):  
    answer = None  
    for value in values:  
        if answer == None or answer < value:  
            answer = value  
    return answer
```

How many
times is this
line executed?



Two Number Sum

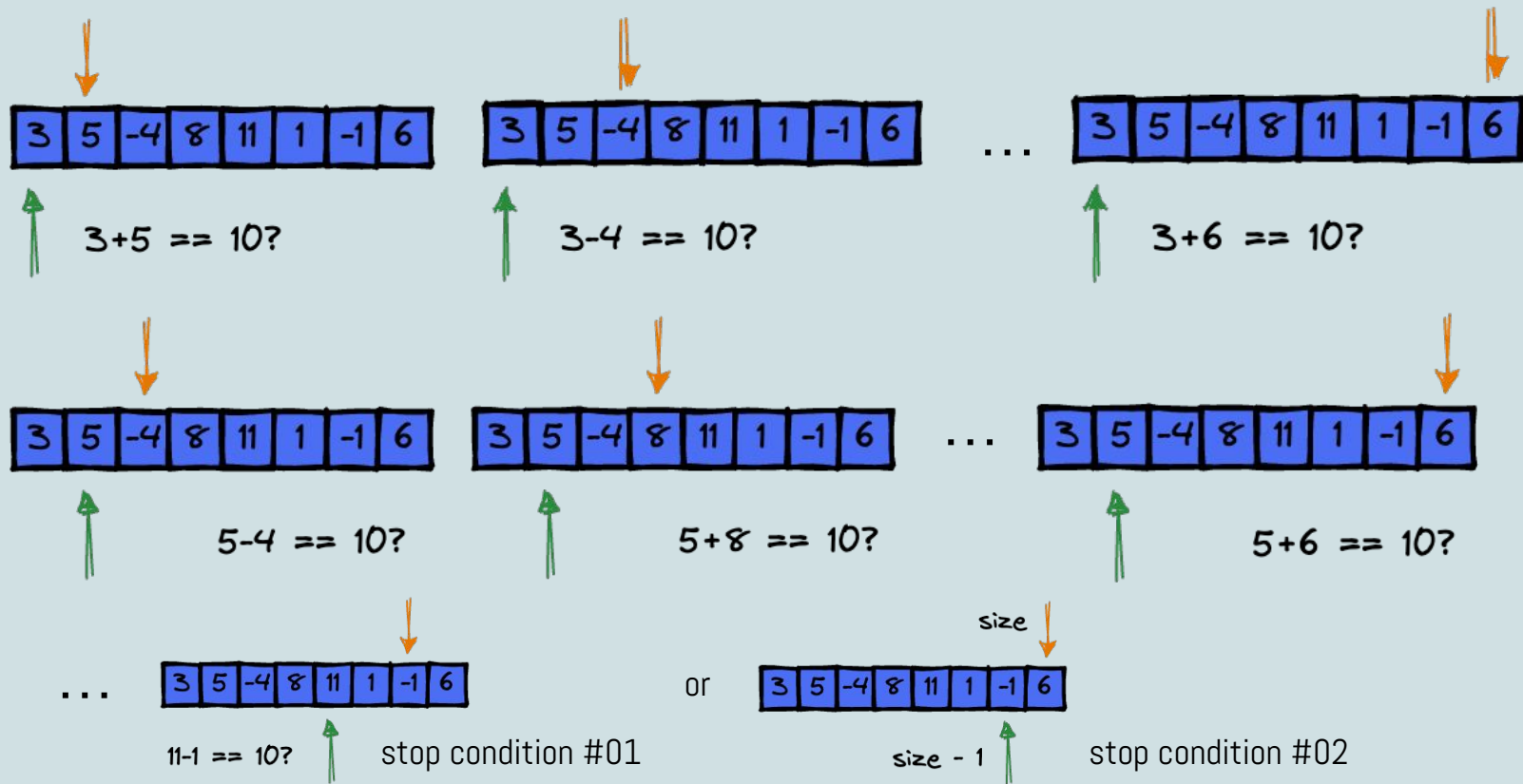
```
# sample input
array = [3,5,-4,8,11,1,-1,6]
targetSum = 10

# sample output
[-1,11] or [11,-1]
```

Write a function that take in a non-empty array of distinct integers and an integer representing a target sum. If any two numbers in the input array sum up to the target sum, the function should return then in an array, in any order. If no two numbers sum up to the target sum, the function should return an empty array.

Note that the target sum has to be obtained by summing two different integers in the array, you can't add a single integer to itself in order to obtain the target sum.

Solution



Solution

$$2(N - 1) + 3(N - 1)^2 + 2$$

$$2(N - 1) + 3(N^2 - 2N + 1) + 2$$

$$3N^2 - 4N + 3$$

```
def twoNumberSum(array, targetSum):  
    for i in range(len(array) - 1):           #C1    N-1  
        firstNum = array[i]                   #C2    N-1  
        for j in range(i+1, len(array)):      #C3    (N-1)2  
            secondNum = array[j]              #C4    (N-1)2  
            if firstNum + secondNum == targetSum: #C5    (N-1)2  
                return [firstNum, secondNum]   #C6    1  
    return []                                  #C7    1
```