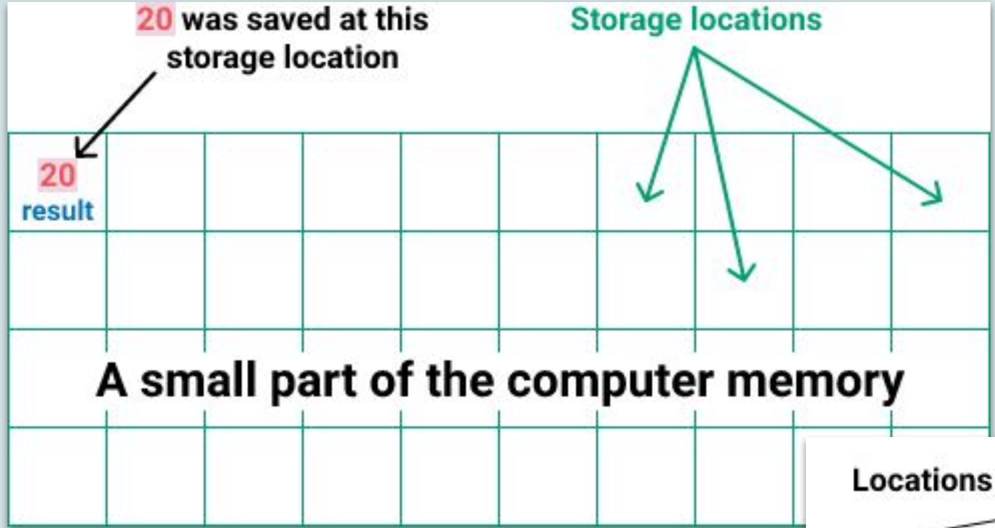
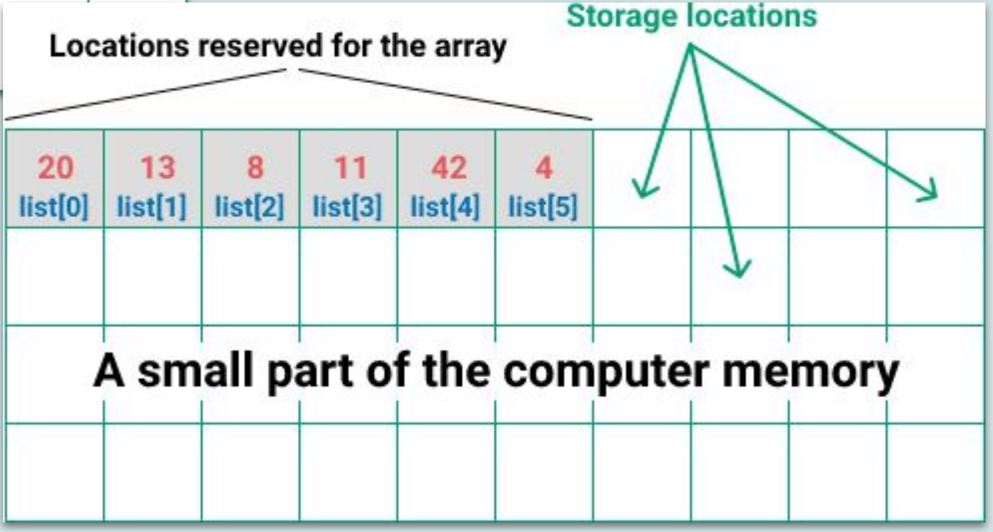
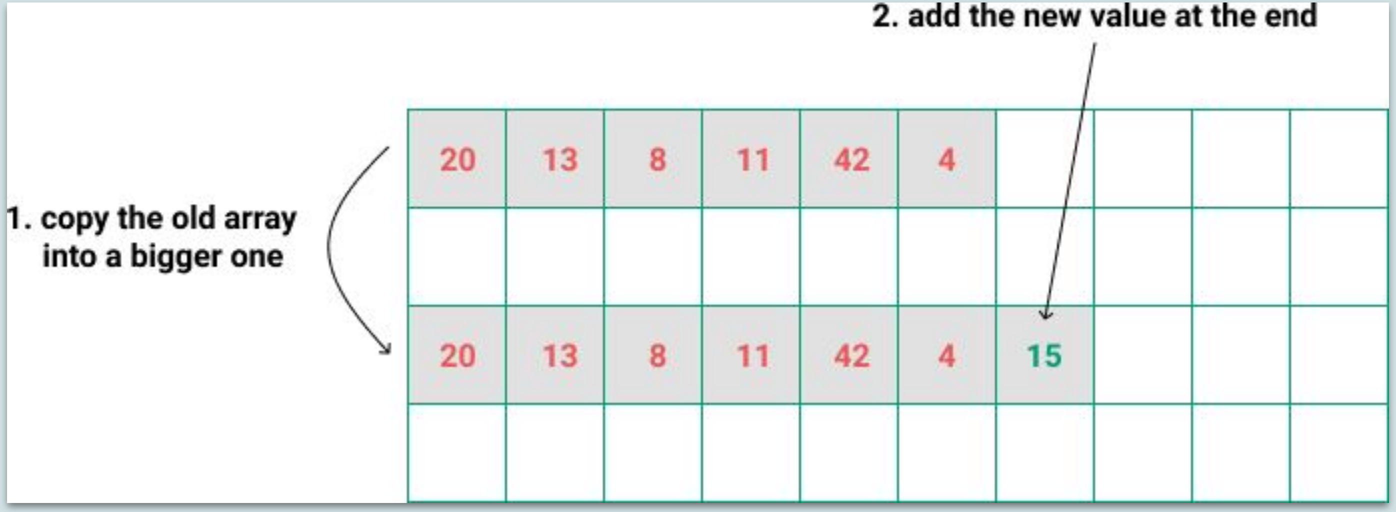
Algorithm Complexity I

Case Study

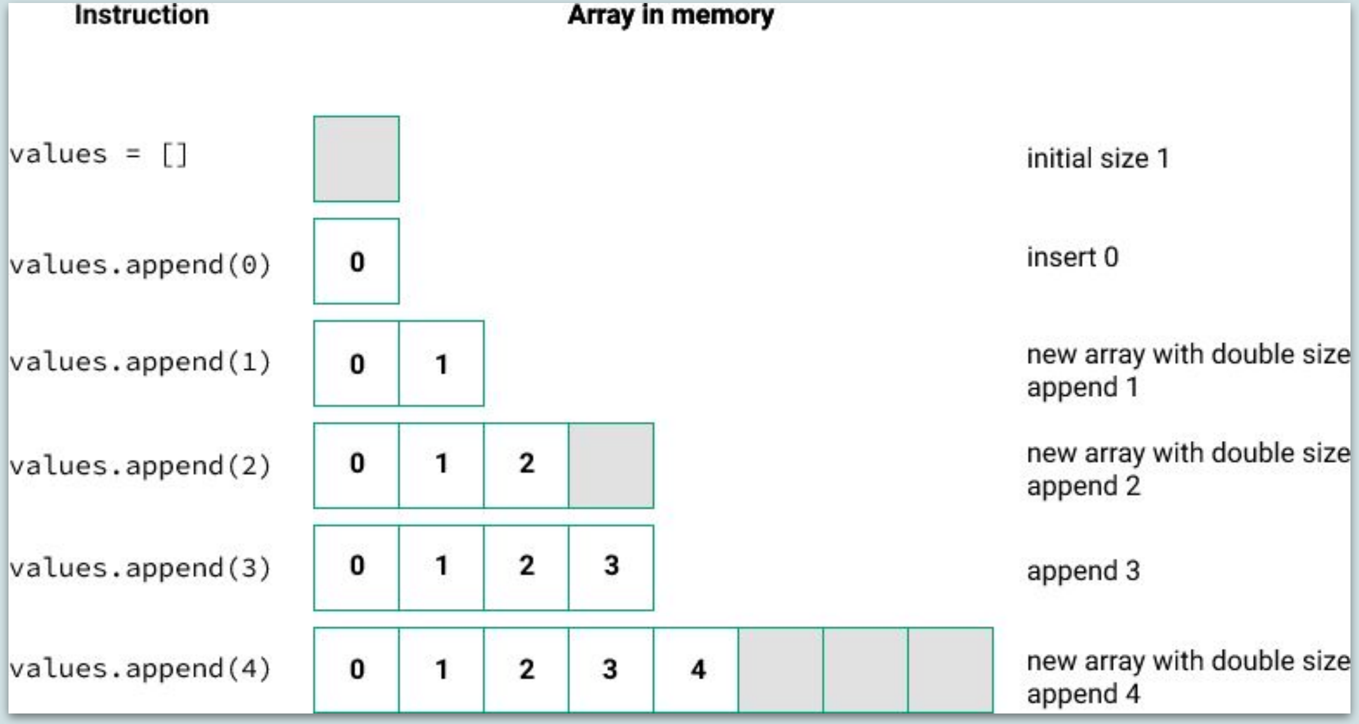
ivanovitch.silva@ufrn.br
@ivanovitchm


20 was saved at this storage location
Storage locations
20
result
A small part of the computer memory

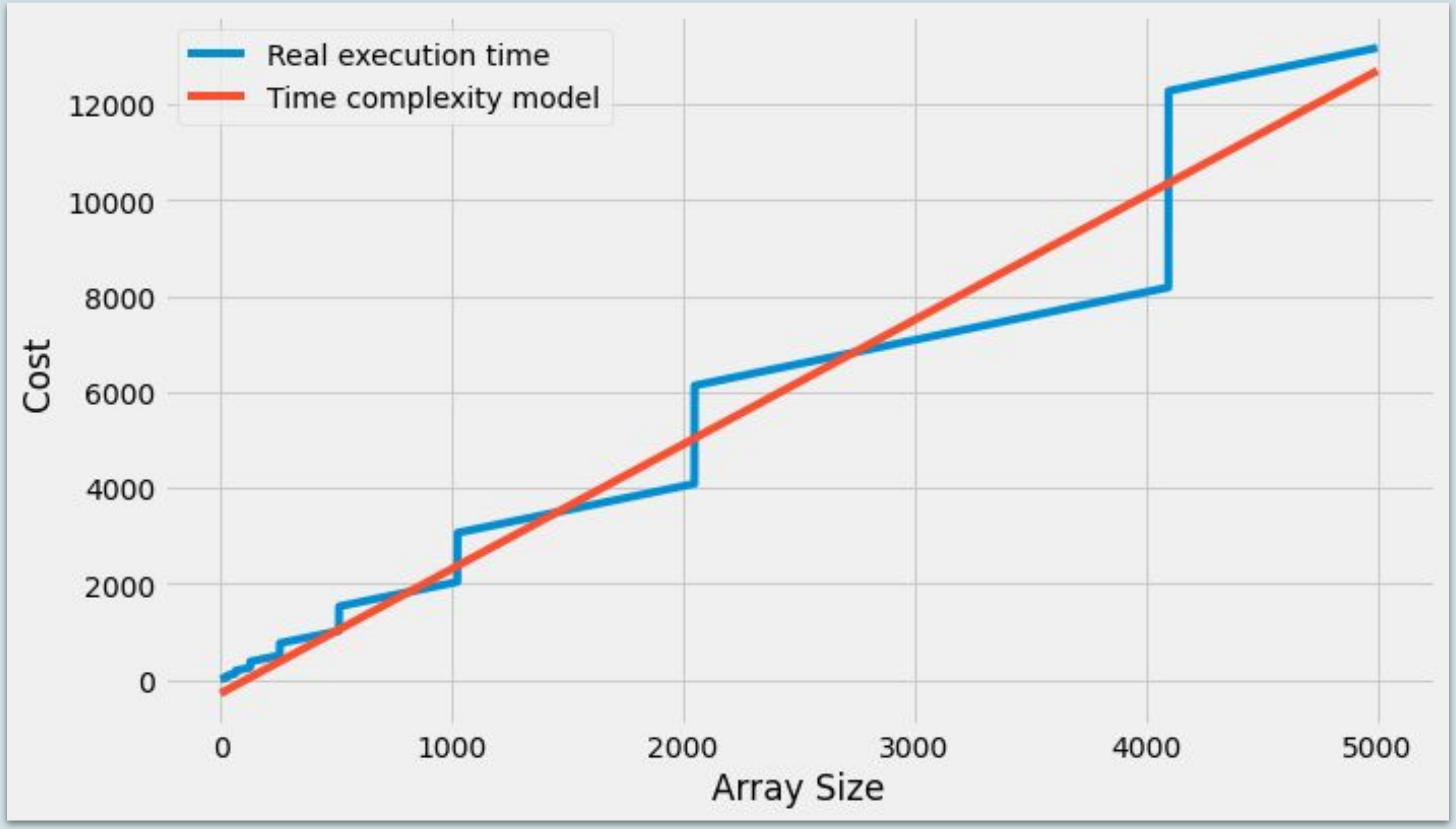

Locations reserved for the array
Storage locations
20
list[0]
13
list[1]
8
list[2]
11
list[3]
42
list[4]
4
list[5]
A small part of the computer memory


2. add the new value at the end
1. copy the old array into a bigger one
20
13
8
11
42
4
20
13
8
11
42
4
15

```python
values = []
values.append(0)
values.append(1)
values.append(2)
values.append(3)
values.append(4)
```



Instruction	Array in memory	
`values = []`	▨	initial size 1
`values.append(0)`	0	insert 0
`values.append(1)`	0 1	new array with double size append 1
`values.append(2)`	0 1 2 ▨	new array with double size append 2
`values.append(3)`	0 1 2 3	append 3
`values.append(4)`	0 1 2 3 4 ▨ ▨ ▨	new array with double size append 4

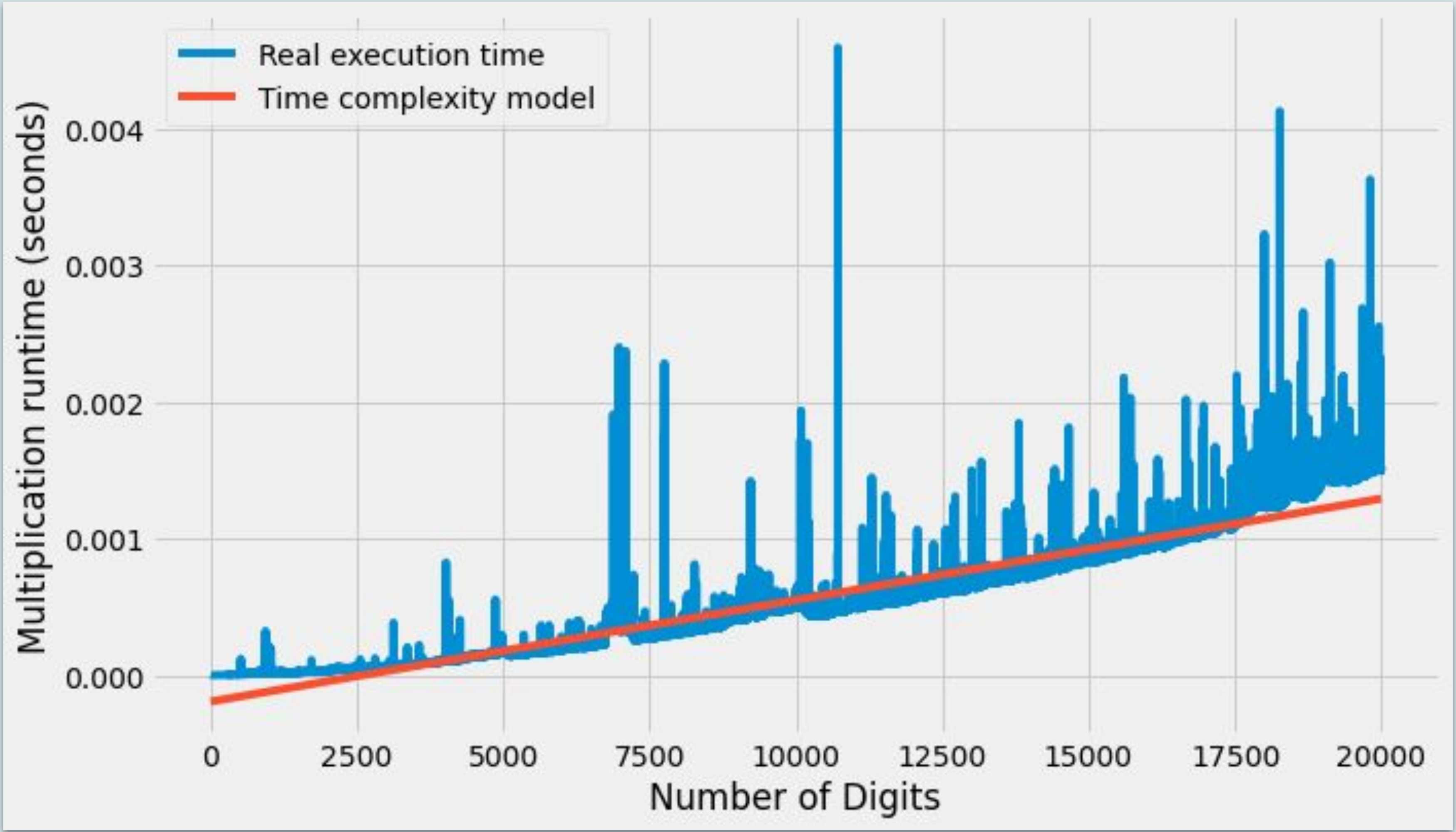

Real execution time
Time complexity model
Cost
Array Size
0
2000
4000
6000
8000
10000
12000
0
1000
2000
3000
4000
5000

Method	Description	Complexity
len()	Get the length of the list	O(1)
append()	Add an element to the list	O(1)
pop()	Retrieve and remove the last element of the list	O(1)
remove(x)	Remove the first ocurrence of x (if it exists)	O(N)
insert(i)	Insert the element at index i	O(N)

```python
def add_with_append(N):
    values = []
    for i in range(N):
        values.append(i)
    return values

def add_with_insert(N):
    values = []
    for i in range(N):
        values.insert(0, i)
    return values
```

```
0.012230873107910156
0.7769486904144287
```

```python
start = time.time()
add_with_append(50000)
end = time.time()
time_append = end - start

start = time.time()
add_with_insert(50000)
end = time.time()
time_insert = end - start

print(time_append)
print(time_insert)
```


Real execution time
Time complexity model
Multiplication runtime (seconds)
0.004
0.003
0.002
0.001
0.000
0
2500
5000
7500
10000
12500
15000
17500
20000
Number of Digits