

Projeto e Análise de Algoritmos

a)

```
1 import time
2
3
4 def num_pares(l):
5     inicio = time.time()
6     time.sleep(0.5)
7     pares = []
8     for i in l:
9         if i % 2 == 0:
10            pares.append(i)
11     fim = time.time()
12     print('{0:.0f}'.format((fim - inicio)*10000.0))
13
14
15 num_pares(range(int(10E5)))
16 num_pares(range(int(10E6)))
17 num_pares(range(int(50E6)))
18 num_pares(range(int(10E7)))
19 num_pares(range(int(50E7)))
20
```

Run: main

/Users/victorordozgoite/PycharmProjects/HelloWorld/venv/bin/python /Users/victorordozgoite/PycharmProjects/HelloWorld/main.py

5834
9353
24936
45021
203492

Process finished with exit code 0

Algoritmo 1

10E5 5834
10E6 9353
50E6 24936
10E7 45021
50E7 203492

```
1 import time
2
3
4 def num_paresL(l):
5     inicio = time.time()
6     time.sleep(0.5)
7     pares = list(filter(lambda valor: valor % 2 == 0, l))
8     fim = time.time()
9     print('{0:.0f}'.format((fim - inicio)*10000.0))
10
11
12 num_paresL(range(int(10E5)))
13 num_paresL(range(int(10E6)))
14 num_paresL(range(int(50E6)))
15 num_paresL(range(int(10E7)))
16 num_paresL(range(int(50E7)))
17
18
```

Run: main

/Users/victorordozgoite/PycharmProjects/HelloWorld/venv/bin/python /Users/victorordozgoite/PycharmProjects/HelloWorld/main.py

6009
10916
32438
60870
286909

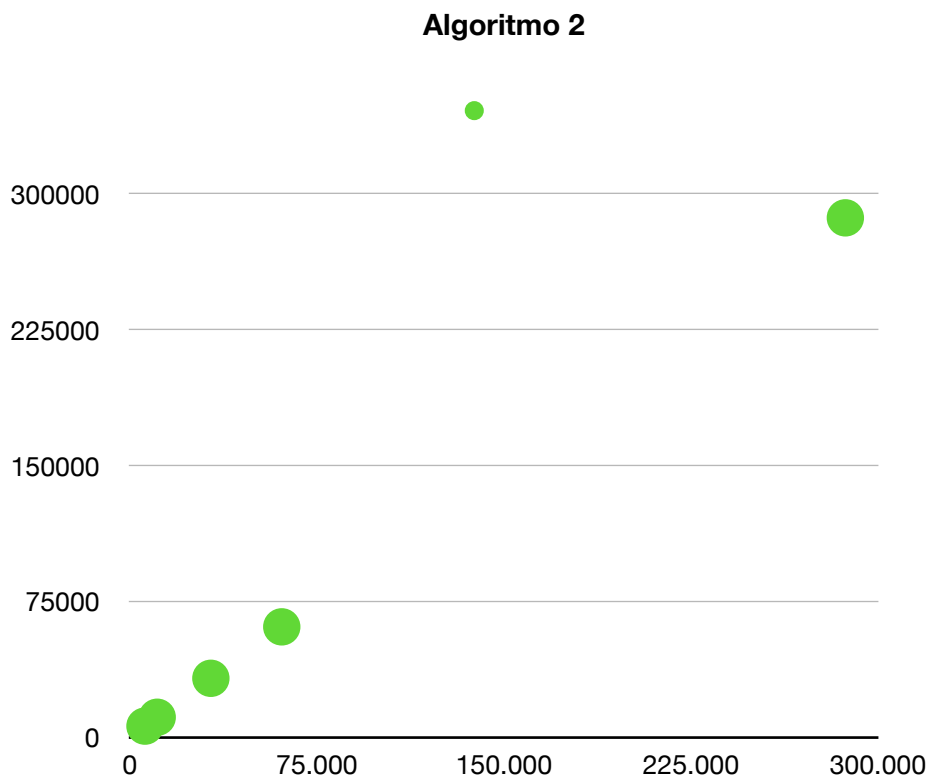
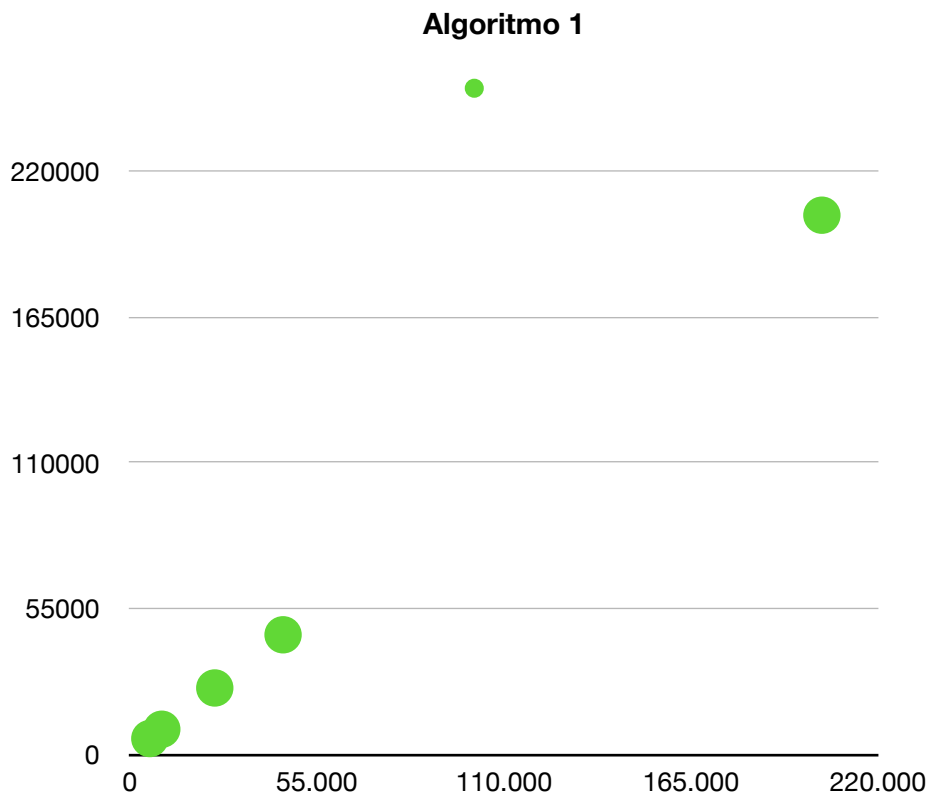
Process finished with exit code 0

Algoritmo 2

10E5 6009
10E6 10916
50E6 32438
10E7 60870
50E7 286909

b) A segunda função demora mais tempo para resolver o problema.

c)



d)

Algoritmo 1

```
def numpares(l):  
    inicio = time.time() # 1  
    time.sleep(0.5) # 1  
    pares = [] # 1  
    for i in l: # n+1  
        if i % 2 == 0: # n  
            pares.append(i) # n  
    fim = time.time() # 1  
    print('{0:.0f}'.format((fim - inicio)*10000.0)) # 1
```

$$T(n) = 1 + 1 + 1 + (n + 1) + n + n + 1 + 1$$

$$T(n) = 3n + 6$$

Algoritmo 2

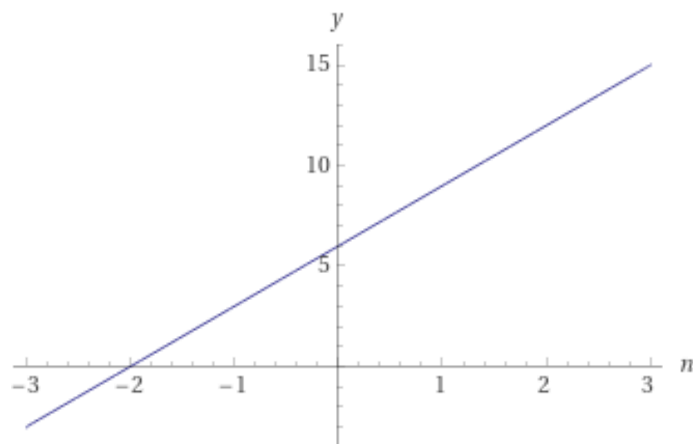
```
def numparesL(l):  
    inicio = time.time() # 1  
    time.sleep(0.5) # 1  
    pares = [] # 1  
    pares = list(filter(lambda valor: valor % 2 == 0, l)) # n  
    fim = time.time() # 1  
    print('{0:.0f}'.format((fim - inicio)*10000.0)) # 1
```

$$T(n) = 1 + 1 + 1 + n + 1 + 1$$

$$T(n) = n + 5$$

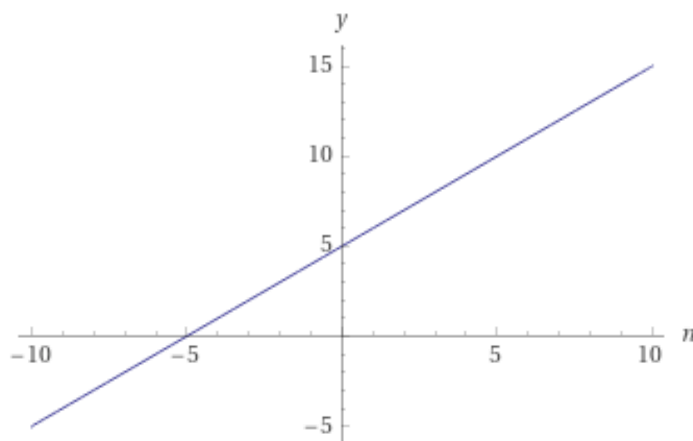
e)

Algoritmo 1



$$T(n) = 3n + 6$$

Algoritmo 2



$$T(n) = n + 5$$

Comparando esses gráficos com os gráficos obtidos na questão “c”, podemos concluir que na Análise Experimental o tempo de execução cresce linearmente em função do tamanho da entrada. Dessa mesma forma se comporta a quantidade de instruções a serem executadas na Análise Teórica, aumentado linearmente à medida que aumenta o tamanho do parâmetro n de entrada.