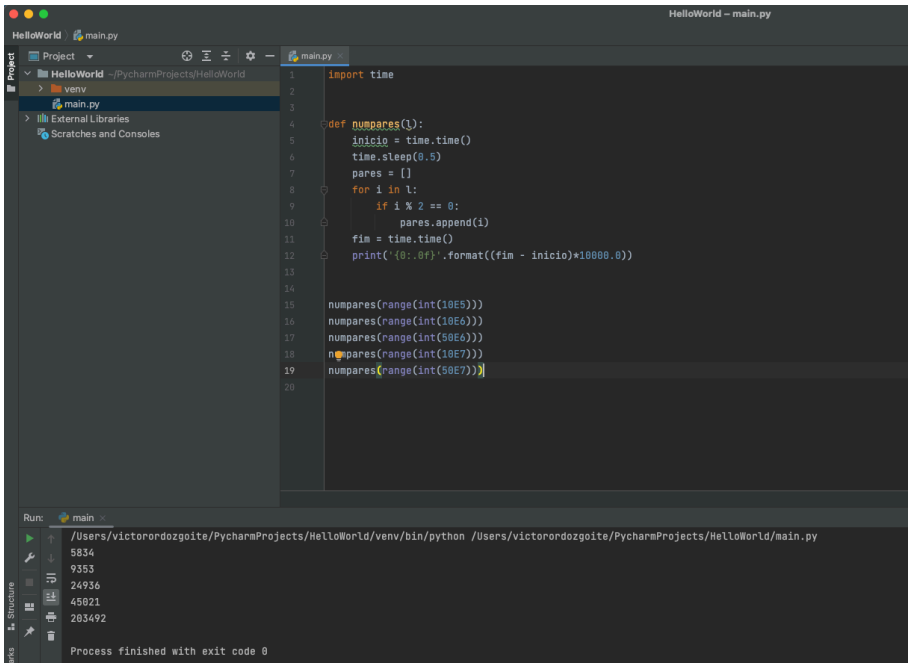
Aluno: Victor Rafael Ordozgoite
Curso: Engenharia da Computação

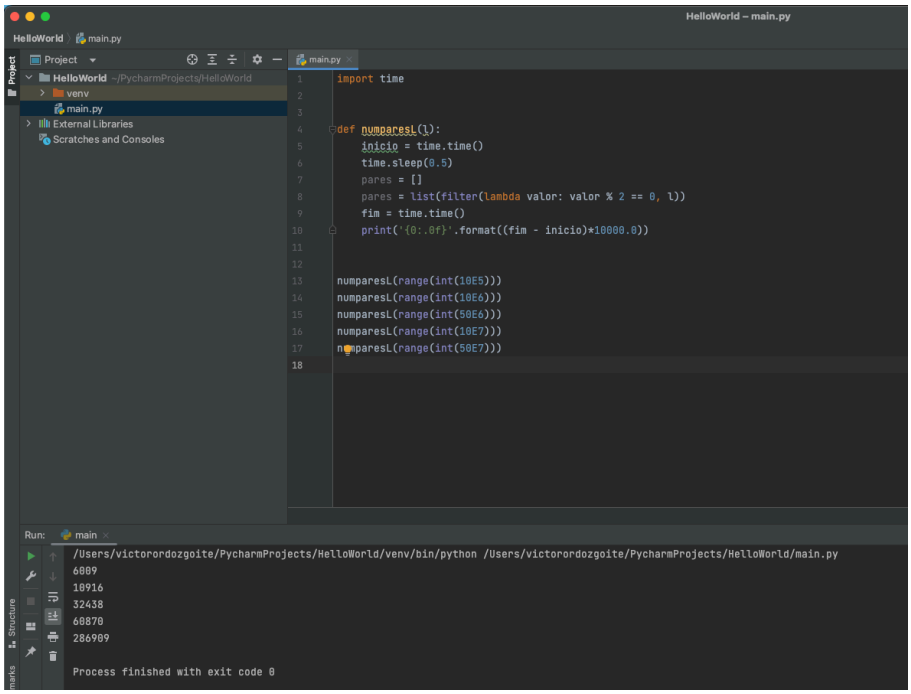
Projeto e Análise de Algoritmos

a)



Algoritmo 1

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

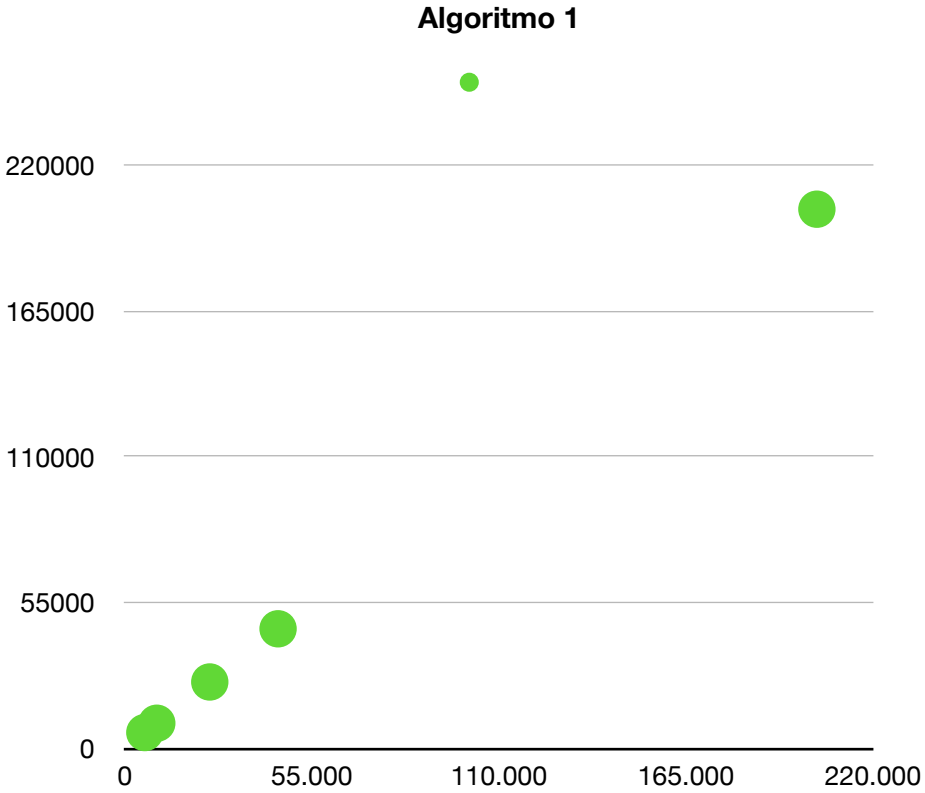


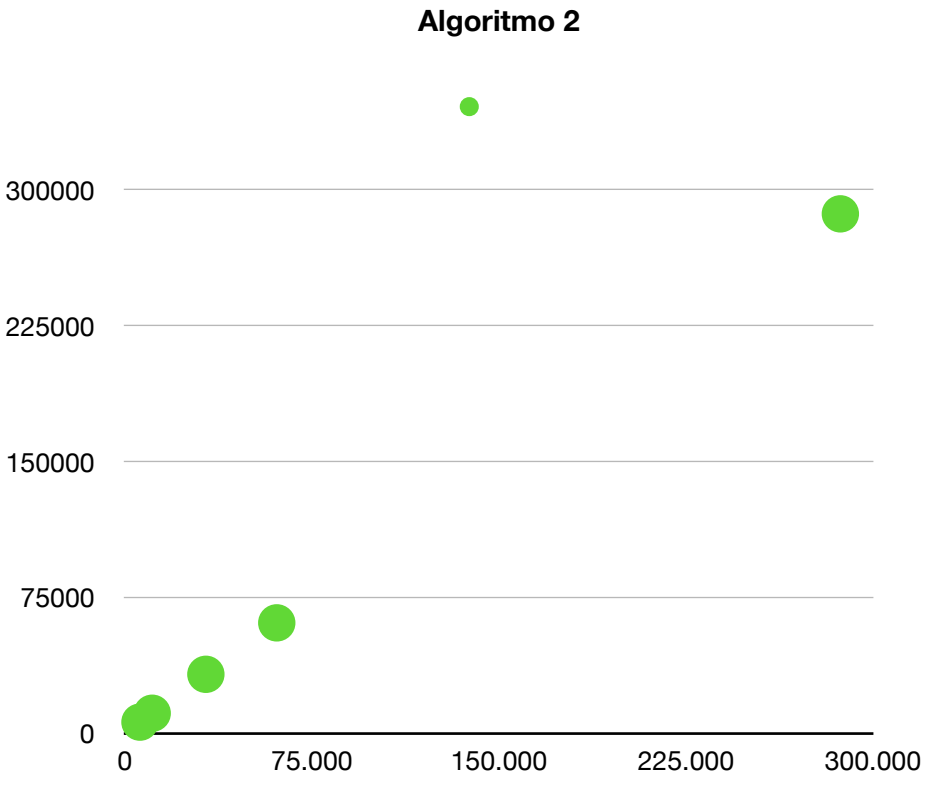
Algoritmo 2

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

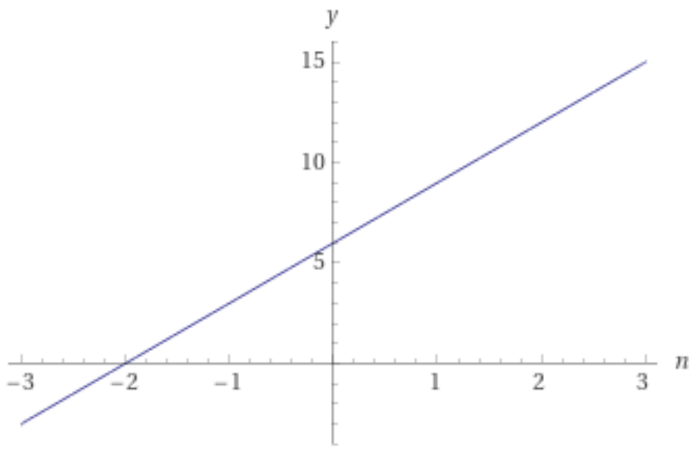
```
def numparesL(l):
    inicio = time.time()                                    # 1
    time.sleep(0.5)                                         # 1
    pares = []                                              # 1
    pares = list(filter(lambda valor: valor % 2 == 0, l))   # n
    fim = time.time() 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

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

T(n) = n + 5

e)

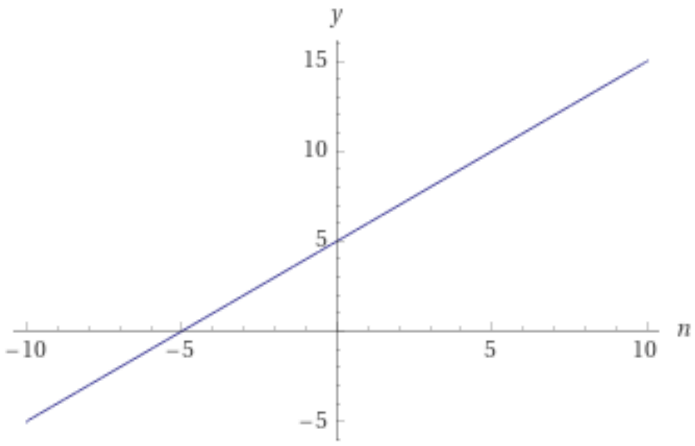
Algoritmo 1



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.