Trabalho 1 - Análise de Algoritmos (INF 1721)

Grupo:
Emanuel Lima Umbelino – 1713090
Yuri Marques Strack – 1710868

1. Código da função:

```
def LinearSelection(A,k):
	if (len(A) <= 1):
		return A[0]
	# divide em grupos de 5
	groups = []
	i = 0
	while i < len(A):
		g = []
		for c in range(5):
			g += [A[i]]
			i += 1
			if (i >= len(A)):
				break
		groups += [g]
	# pega a mediana de cada grupo
	M = []
	for g in groups:
		BubbleSort(g)
		M += [g[len(g)//2]]
	# pega a mediana das medianas
	m = LinearSelection(M,len(M)//2)
	# faz duas listas ao redor da mediana e conta quantos valores iguais a mediana
	L = []
	R = []
	equals = 0
	for i in range(len(A)):
		if (A[i] < m):
			L += [A[i]]
		elif (A[i] > m):
			R += [A[i]]
		else:
			equals += 1
	# se o que eu quero estiver na direita ou na esquerda, recursao pra la
	# se tiver entre os repetidos achei
	if (len(L) >= k):
		return LinearSelection(L,k)
	elif (len(L) + equals >= k):
		return m
	else:
		return LinearSelection(R,k-len(L)-equals)
```

2. Análise do código:

O código satisfaz a equação de recorrência T(n) <= **cst*n** + T(n/5) + T(7*n/10) devido as seguintes partes do algoritmo:

```
while i < len(A):
        g = []
        for c in range(5):
                g += [A[i]]
                i += 1
                if (i >= len(A)):
                        break
        groups += [g]
```

Explicação: O while é executado n vezes (tamanho da lista A) enquanto que a parte interna possui valor constante. Totalizando **cst*n**.

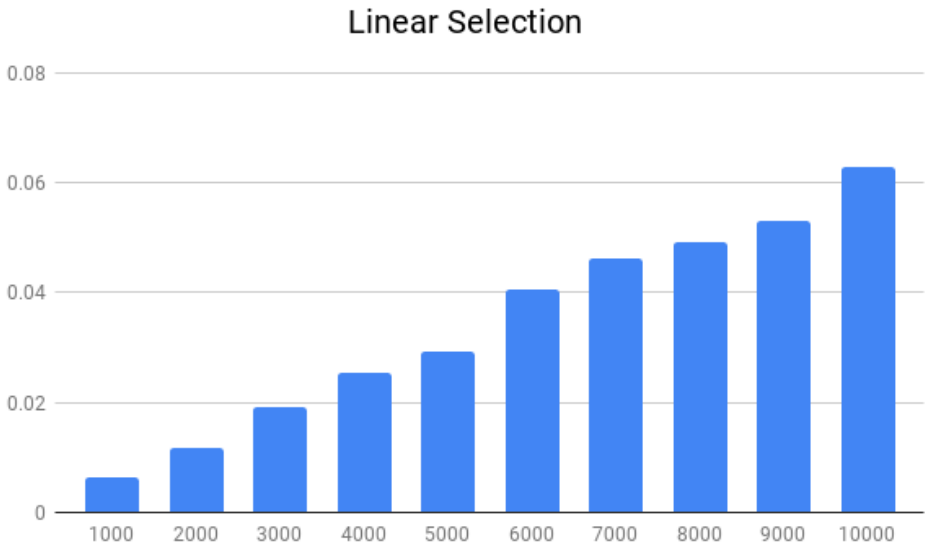
```
for i in range(len(A)):
        if (A[i] < m):
                L += [A[i]]
        elif (A[i] > m):
                R += [A[i]]
        else:
                equals += 1
```

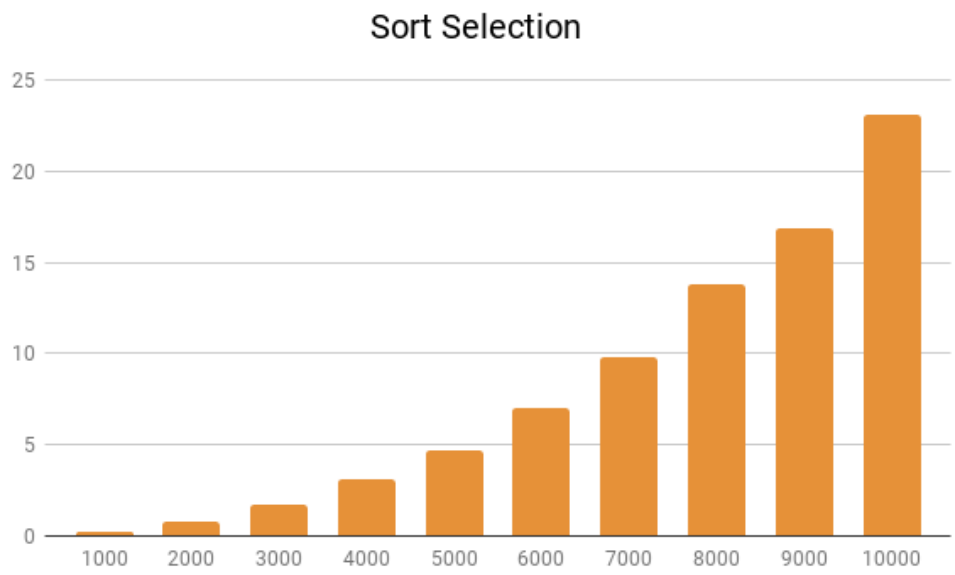
Explicação: A parte interna ao "for" possui valor constante e o loop é executado no tamanho da lista (n). Totalizando **cst*n**.

Obs: O bubbleSort usado para ordenar os grupos possui valor costante (O(5)) pois os grupos tem 5 elementos.

Dessa forma, cst*n + cst*n = **cst*n**.

3. Gráficos comparativos (eixo x = tamanho da entrada, y = media do tempo de execução em segundos):





Pela observação dos gráficos é possível perceber que o LinearSelection aparenta estar crescendo de forma linear, respeitando sua complexidade O(n). Enquanto que o SortSelection aparenta estar crescendo de forma bem mais rápida, similar a n^2, respeitando sua complexidade O(n^2).