CI/UFPB
Disciplina: Estrutura de Dados e Complexidade de Algoritmos
Professor: Gilberto Farias
Aluno: Pablo Herivelton Ramos Goes
Matricula: 20201006605

Análise do algoritmo Insertion Sort

def insertionSort(lista):	custo	vezes
for i in range(1, len(lista)):	c_1	n
chave = lista[i]	c_2	n-1
k = i	c_3	n-1
while k > 0 and chave < lista[k - 1]:	c_4	$\sum_{i=1}^{n}$ ti
lista[k] = lista[k - 1]	c_5	$\sum_{i=1}^{n}(\text{ti} - 1)$
k -= 1	c_6	$\sum_{i=1}^{n}(\text{ti} - 1)$
lista[k] = chave	c_7	n-1
return lista		

$T(n) = c_1n + c_2(n-1) + c_3(n-1) + c_4\sum_{i=1}^{n} \text{ti} + c_5 \sum_{i=1}^{n}(\text{ti} - 1) + c_6 \sum_{i=1}^{n}(\text{ti} - 1) + c_7(n-1)$

Para o melhor caso, onde o array se encontra ordenado, temos que ti = 1, como o array está ordenado, nunca vai ser executado o código dentro do while, em compensação teremos que a linha do while, executará n-1 vezes, então teremos:

$T(n) = c_1n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_5(0) + c_6(0) + c_7(n-1)$, logo
$T(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n – (c_2 + c_3 + c_4 + c_7)$
Chamando $c_1 + c_2 + c_3 + c_4 + c_7$ de a e $c_2 + c_3 + c_4 + c_7$ de b, teremos
$T(n) = an – b$ (MELHOR CASO)

Para o pior caso, onde o array se encontra invertido, temos que ti = i e baseado no T(n) geral do algoritmo Insertion Sort, teremos

$T(n) = c_1n + c_2(n-1) + c_3(n-1) + c_4(n(n+1)/2 -1) + c_5(n(n-1)/2) + c_6(n(n-1)/2) + c_7(n-1)$
$T(n) = (c_4/2 + c_5/2 + c_6/2)n^2 + (c_1 + c_2 + c_3 + c_4/2 + c_5/2 + c_6/2 + c_7)n – (c_2 + c_3 + c_4 + c_7)$
$T(n) = an^2 + bn - c$ (PIOR CASO)

Análise do código do Selection Sort.

def selectionSort(lista):	custo	vezes
for i in range(len(lista)):	c_1	n
menorIndice = i	c_2	n
for j in range(i+1, len(lista)):	c_3	$\sum_{i=0}^{n}$ ti $\sum_{j=0}^{n}$ tj = (n^2+n)/2
if lista[j] < lista[menorIndice]:	c_4	$\sum_{i=0}^{n}$ ti $\sum_{j=0}^{n}$ tj $-$ 1 = (n^2-n)/2
menorIndice = j	c_5	$\sum_{i=0}^{n}$ ti $\sum_{j=0}^{n}$ tj $-$ 1 = (n^2-n)/2
temp = lista[i]	c_6	n
lista[i] = lista[menorIndice]	c_7	n
lista[menorIndice] = temp	c_8	n
return lista		

O selection sort possui a mesma função T(n) para ambos os casos, melhor e pior caso.
Logo a função T(n) será:
$c_1n + c_2n + c_3(n^2+n)/2 + c_4(n^2-n)/2 + c_5(n^2-n)/2 + c_6n + c_7n + c_8n$, isso implica em
$T(n) = (c_1 + c_2 + c_3 + c_6 + c_7 + c_8)n + (c_4 + c_5)\ (n^2-n)/2 + c_3(n^2+n)/2$
$T(n) = a(n) + b((n^2-n)/2) + c((n^2+n)/2)$ (MELHOR E PIOR CASO)