Lista de Exercícios 1

Universidade Federal do Ceará - Campus Quixadá
Projeto e Ana lise de Algoritmo — QXD0041 – 2023.2 Prof.
Fabio Dias

Complexidade de Algoritmos

1. Desenvolva um algoritmo para determinar se um vetor de tamanho *n* está ordenado. Depois faça a análise de complexidade desse seu algoritmo em relação o ao melhor e pior caso, se houver.

Seja um vetor de inteiros V de tamanho n.
Seja $i = 1$ (atribuição) - C_1
Enquanto $i < n$, faça: (comparação) - $C_2 \cdot n$
{
se $V[i-1] > V[i]$, faça: (comparação com subtração) – $C_3 \cdot n-1$
retorne false (retorno) - C_4
$i += 1$ (incremento) - $C_5 \cdot n-1$
}
retorne true. (retorno) - C_4

O pior caso: o vetor está ordenado.

$$C_1 + \underbrace{n \cdot (C_2 + C_3 + C_5) - C_3 - C_5}_{= C} + C_4$$

$$C_1 + n \cdot C + C_4$$

Desconsiderando as constantes C_1 e C_4, temos que no pior caso, a complexidade desse algoritmo cresce linearmente a medida que n cresce.

$\underline{n \cdot c}$

Complexidade $O(n)$

O melhor caso: o vetor não está ordenado.

Nesse caso existe um $i < n$ em que o valor de V na posição $i-1$ é maior que o valor de V na posição i. Assim, a complexidade aumenta conforme maior o valor de i.

Complexidade $O(1)$ → Com $i = 1$

2. Desenvolva um algoritmo para inserir um elemento em um vetor **ordenado** contendo *n* elementos, de tal modo que após a inserção o ele continue ordenado. Suponha que o vetor tenha capacidade infinita, ou seja, o tamanho do vetor seja *m* >>> *n*, ou seja, ele nunca ficara cheio, sempre terá um espaço no vetor para adicionar um novo elemento. Desconsidere o caso do vetor esta´ vazio ($n = 0$). Depois determine a complexidade desse seu algoritmo em relação o ao melhor e pior caso, se houver.

Seja um vetor V de tamano n
seja inteiro x.
seja $i = 0$ (atribuição) - c_1

Enquanto $V[i] <= x$ e $i < n$, faça: (comparação) - c_2
    $i{+}{+}$ (incremento) - c_3

Enquanto $i < n$, faça: (Comparação) - c_4
{
    int aux = $V[i]$ (atribuição) - c_1
    $V[i] = x$ (atribuição) - c_1
    $x = aux$ (atribuição) - c_1
    $i{+}{+}$ (incremento) - c_3
}
$V[i] = x$ (atribuição) - c_1

No pior caso, o elemento a ser inserido ocupa a primeira posição do vetor. Assim o número de operações é:

$$c_1 + c_2 + n\underbrace{(c_4 + 3 \cdot c_1 + c_3)}_{t} + c_4 + c_1$$

$n \cdot t + 2c_1 + c_2 + c_4 \rightarrow$ O número de operações cresce linearmente à n.

Complexidade $O(n)$

No melhor caso o elemento a ser inserido ocupa a ultima posição, assim o algritmo executa:

$$c_1 + n(c_2 + c_3) + c_2 + c_1$$

$n \cdot t + 2c_1 + c_2 \rightarrow$ Cresce conforme n cresce.

Complexidade $O(n)$

3. Desenvolva um algoritmo para resolver o seguinte problema: dado um vetor ordenado com *n* nu meros inteiros positivos e um outro nu mero inteiro positivo *x*, encontrar dois elementos do vetor cuja a soma ´e igual a *x*. Faça a análise de complexidade do seu algoritmo no pior e melhor caso, se houver.

Seja um vetor ordenado V de tamanho n
Seja um inteiro X
seja i = n (atribuição) - c_1

Enquanto V[i] > x e i => 0, faça: (comparação) - c_2
  i-- (decremento) - c_5
seja j=0 (atribuição) - c_1

Enquanto j < i, faça: (comparação) - c_3
{
  se V[j] + V[i] < x, faça (soma e comparação) - c_4
    j++ (incremento) - c_5
  se V[j] + V[i] > x, faça (soma e comparação) - c_4
    i-- (decremento) - c_5
  se não :
    retorne V[j], V[i] (retorno) - c_6
}
return NULL (retorno) - c_6

4. Desenvolva uma versa a recursiva da questão o anterior.