Pedro Ailan

201700129431

a) Prove que n² + n + 10 = Ômega(n lg n). (Provar)

f(n) = n² + n + 10

i(n) = n log n

Existe um c e um n tal que f(n) >= c i(n) para qualquer n >= 0

Sendo assim,

n² + n + 10 >= n log n

n + 1 + 10/n >= log (n)

n >= log (n) - 10/n - 1

b) A afirmação a seguir está correta? n²/1000 − 999 n = Teta(n²). (Provar)

n²/1000 - 999n = θ (n²) PARA nₒ = 1000

f(n) = n²/1000 – 999n;

i(n) = n²

n²/1000 - 999n = θ(n²) se, somente se n²/1000 – 999n = θ(n²) = Ω(n²)

PARA:

f(n) <= θ(i) :

n²/1000 – 999n <= n²

n/1000 – 999 <= n , verdadeiro para todo n >= 0

PARA:

f(n) >= Ω(i)

n²/1000 – 999n >= n²

n/1000 – 999n >= n

FALSO para todo n >= 1000

Como θ(n²) ≠ Ω(n²), n²/1000 – 999n = θ(n²) é falso.

c) Qual o consumo de tempo de Ordenação-por-Inserção no melhor caso? (Suponha que cada execução de cada linha do algoritmo consome 1 unidade de tempo.) Como são os vetores que correspondem ao melhor caso?

O consumo de tempo vai depender das variantes:

1. Do vetor a[1..n]
2. Da máquina que está executando o algoritmo.

Se utilizarmos uma máquina 2 vezes mais lenta o consumo será multiplicado por 2. Se utilizarmos uma máquina 2 vezes mais rápida o consumo será multiplicado por ½ . A partir disso podemos ignorar a contribuição da máquina.

Ordenação-por-Inserção (A, n)

1. para j crescendo de 2 até n = n

2. x := A[j] = n−1

3. i := j−1 = n−1

4. enquanto i > 0 e A[i] > x ≤ 2 + 3 + ... + n

5. A[i+1] := A[i] ≤ 1 + 2 + 3 + ... + n−1

6. i := i−1 ≤ 1 + 2 + 3 + ... + n−1

7. A[i+1] := x = n−1

A estimativa de para o tempo no pior caso

T(n) ≤ (3/2)n² + (7/2)n − 4.

Para valores grandes de n, o n² é muito maior que (7/2)n, sendo assim, n² é a única parte sólida, todo o resto depende da máquina, sistema operacional e determinados detalhas de implementação do algoritmo. Logo, é suficiente dizer que o algoritmo consome: O(n²) unidades de tempo.

No melhor caso, os vetores que correspondem são:

1 para i = 2 até n, incrementando faça

2 atual = A[i]

3 j = i − 1

4 enquanto j > 0 e A[j] > atual faça

5 A[j + 1] = A[j]

6 j = j – 1

7 A[j + 1] = atual

No melhor caso, o teste do laço enquanto é executado somente uma vez para cada valor de i do laço para, totalizando n−1 = Θ(n) execuções. Esse **caso ocorre quando A já está ordenado**. No pior caso, o teste do laço enquanto é executado i vezes para cada valor de i do laço para, totalizando 2 + · · · + n = n (n + 1)/2 − 1 = Θ(n²) execuções. Esse caso ocorre quando A está em ordem decrescente.

**Diante disso, podemos concluir que Ordenação por Inserção no melhor caso consome: O(n) unidades de tempo.**

**Referências**

* <https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/insert.html>
* Negri Lintzmayer, C.; Oliveira Mota, G. Análise de Algoritmos e Estruturas de Dados. Versão: 16 de novembro de 2020.