Pedro Ailan

201700129431

a) Prove que $n^2 + n + 10$ = Ômega(n lg n). (Provar)

$f(n) = n^2 + n + 10$

$i(n) = n \log n$

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

Sendo assim,

$n^2 + n + 10 >= n \log n$

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

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

b) A afirmação a seguir está correta? $n^2/1000 - 999n$ = Teta(n^2). (Provar)

$n^2/1000 - 999n = \theta (n^2)$ PARA $n_o = 1000$

$f(n) = n^2/1000 - 999n;$

$i(n) = n^2$

$n^2/1000 - 999n = \theta(n^2)$ se, somente se $n^2/1000 - 999n = \theta(n^2) = \Omega(n^2)$

PARA:

$f(n) <= \theta(i)$:

$n^2/1000 - 999n <= n^2$

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

PARA:

$f(n) >= \Omega(i)$

$n^2/1000 - 999n >= n^2$

$n/1000 - 999n >= n$

FALSO para todo $n >= 1000$

Como $\theta(n^2) \neq \Omega(n^2)$, $n^2/1000 - 999n = \theta(n^2)$ 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)

para j crescendo de 2 até n	$= n$
x := A[j]	$= n-1$
i := j−1	$= n-1$
enquanto i > 0 e A[i] > x	$\leq 2+3+...+n$
A[i+1] := A[i]	$\leq 1+2+3+...+n-1$
i := i−1	$\leq 1+2+3+...+n-1$
A[i+1] := x	$= n-1$

A estimativa de para o tempo no pior caso

$$T(n) \leq (3/2)n^2 + (7/2)n - 4.$$

Para valores grandes de n, o n^2 é muito maior que $(7/2)n$, sendo assim, n^2 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: $O(n^2)$ unidades de tempo.

No melhor caso, os vetores que correspondem são:

```
para i = 2 até n, incrementando faça
atual = A[i]
j = i − 1
enquanto j > 0 e A[j] > atual faça
A[j + 1] = A[j]
j = j – 1
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 $n-1 = \Theta(n)$ 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 $2 + \cdots + n = n(n+1)/2 - 1 = \Theta(n^2)$ execuções. Esse caso ocorre quando A está em ordem decrescente.

Diante disso, podemos concluir que Ordenação por Inserção no melhor caso consome: <u>O(n)</u> 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.