MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação
Primeiro semestre de 2001

Prova 1

1. (Valor: 1,5 pontos)

   (a) Seja $f : \mathbb{N} \to \mathbb{N}$. Dê a definição formal de $O(f(n))$ e $\Omega(f(n))$.

   (b) Sejam $f : \mathbb{N} \to \mathbb{N}$ e $g : \mathbb{N} \to \mathbb{N}$. Prove ou disprove que se $g(n) = O(f(n))$ então $\log g(n) = O(\log f(n))$.

2. (Valor: 1,5 pontos) Qual é a complexidade de tempo T(n) do seguinte trecho de programa? Para simplificar, você pode supor que $n = 2^k$ para algum inteiro não-negativo k. Justifique a sua resposta.

```
1.   for (i=0; i<n; i++) {
2.     j=n;
3.     while (j>1) {
4.       /* corpo do while, que consome, digamos, tempo O(1) */
5.       j=j/2;
6.     }
7.   }
```

3. (Valor: 2,0 pontos) Considere a variante do `mergesort` que ordena no lugar.

```
void mergesort_no_lugar(int v[], int ini, int fim) {
  int meio;
  if (ini<fim) {
    meio = (ini+fim)/2;
    mergesort_no_lugar(v,ini,meio);
    mergesort_no_lugar(v,meio+1,fim);
    intercala_no_lugar(v,ini,meio,fim);
  }
}
```

   É possível implementar o `intercala_no_lugar(v,ini,meio,fim)` de forma que ele execute no pior caso em tempo $\Theta(n^2)$, onde n é o número de elementos no vetor `v` entre `ini` e `fim`, ou seja, $n = \texttt{fim} - \texttt{ini} + 1$.

   Seja $T(n)$ a complexidade de tempo no pior caso do algoritmo `mergesort_no_lugar`. Escreva a recorrência obtida do algoritmo para $T(n)$ (seja preciso: se necessário, use $\lceil\ \rceil$ e $\lfloor\ \rfloor$).

   Para simplificar, suponha que $n = 2^k$, para algum inteiro não-negativo k. Para estes valores de n, deduza da recorrência a ordem de grandeza de $T(n)$ (ou seja, deduza que $T(n) = \Theta(f(n))$, para uma função $f(n)$ tão simples quanto possível). Justifique a sua resposta.

4. (Valor: 2,0 pontos) Suponha que um vetor $v[0..n-1]$ está organizado como um *heap* decrescente (portanto $v[0]$ é um elemento máximo). Considere o seguinte problema:

   dado um índice i, remover $v[i]$ e transformar o vetor restante em um *heap* $v[0..n-2]$.

   Descreva um algoritmo que resolva o problema em tempo $O(\log n)$. Justifique a sua resposta.

5. (Valor: 2,0 pontos) Descreva um algoritmo que ordena em tempo $O(n)$ um vetor $v[1..n]$ com inteiros entre 1 e n^2. Justifique a sua resposta.

6. (Valor: 2,0 pontos) Mostre que qualquer algoritmo de busca baseado em comparações faz no pior caso pelo menos $\lceil \log n \rceil$ comparações. Um algoritmo de busca é um algoritmo que determina uma posição em que um elemento x aparece em um dado vetor $v[0..n-1]$. Um algoritmo de busca baseia-se em comparações se o seu fluxo de controle depende apenas de comparações envolvendo elementos do vetor e x.