MAC 5711 - Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2004

Lista 4

Devolução dos exercícios 2, 6, 10 e 11: 29 de abril.

1. **(Exercício 9.1-1/8.1-1 do CLR/CLRS)** Qual é a menor profundidade possível de uma folha em uma árvore de decisão para um algoritmo de ordenação? Qual é a informação que você conclui sobre algoritmo relacionada a este número?

   Resolução: Considere a árvore de decisão de um algoritmo de ordenação baseado em comparações para uma entrada de tamanho n. Lembre-se que cada folha está associada a uma permutação da seüência de entrada: a permutação que ordena aquela seqüência. Respondendo à última pergunta, a menor profundidade de uma folha corresponde exatamente a uma instância de melhor caso do algoritmo.

   A menor profundidade de uma folha em tal árvore é $n-1$. Primeiro observe que há um algoritmo cuja folha de profundidade mínima está a profundidade exatamente $n-1$: o insertion sort. Agora, vamos mostrar por contradição que a profundidade de uma folha que está à menor profundidade é pelo menos $n-1$.

   Suponha que existe uma folha f com profundidade menor que $n-1$. Considere o grafo cujo conjunto de vértices é formado pelos n elementos da seqüência de entrada e há uma aresta entre quaisquer dois deles se eles foram comparados no caminho da raiz até a folha f. Como a profundidade de f é menor que $n-1$, tal grafo tem menos que $n-1$ arestas e portanto é desconexo. Daí podemos derivar duas seqüências de entrada de tamanho n que não são ordenadas por uma mesma permutação, porém tais que as duas seguem o mesmo caminho da árvore, levando à folha f. Isso é uma contradição, já que a folha f está associada a uma única permutação da seqüência de entrada. Para pelo menos uma das duas seqüências, essa permutação não produz a versão ordenada da seqüência de entrada.

2. **(Exercício 9.2-5 do CLR)** Descreva um algoritmo que, dados n inteiros no intervalo 1 até k, preprocesse sua entrada e então responda em $O(1)$ qualquer consulta sobre quantos dos n inteiros dados caem em um intervalo $[a\,..\,b]$. O preprocessamento efetuado pelo seu algoritmo deve executar em tempo $O(n+k)$.

   Resolução: O pré-processamento consiste em contruir um vetor de contadores como o do *counting sort*. Mais precisamente, o pré-processamento consiste no seguinte:

```
int preprocessamento(int v[], int n, int k, int cont[]) {
  int i;

  for (i=0; i<=k; i++)
    cont[i] = 0;
  for (i=0; i<n; i++)
    cont[v[i]]++;
  for (i=2; i<=n; i++)
    cont[i] += cont[i-1];
}
```

   Depois de executada tal função (uma única vez, no início do processamento), pode-se determinar quantos inteiros da seqüência dada em $v[0\,..\,n-1]$ caem num intervalo arbitrário $[a\,..\,b]$ bastando para isso calcular a diferença `cont[b]-cont[a-1]`. Cada consulta deste tipo claramente consome tempo $O(1)$.

3. **(Exercício 10.3-5 do CLR)** Dado um algoritmo linear "caixa-preta" para encontrar uma mediana de um vetor, dê um algoritmo simples, linear, que resolve o problema do k-ésimo mínimo.

   Resolução: O algoritmo ficaria:

```
int kesimominimo (int v[], int ini, int fim, int k) {
  int m, q;

  if (ini == fim)
    return v[ini];
  else {
    m = mediana(v, ini, fim);
    q = particionacompivo(v, ini, fim, m);
    if (k <= q-ini+1)
      return kesimominimo(v, ini, q, k);
    else
      return kesimominimo(v, q+1, fim, k-(q-ini+1));
  }
}
```

   A recorrência que descreve o tempo consumido por este algoritmo é $T(n) = T(\lceil n/2 \rceil) + \Theta(n)$, cuja solução é linear. De fato, para n potência de 2, por substituição, obtemos:

$$T(n) \;=\; T(n/2)+n \;=\; T(n/4)+n/2+n \;=\; T(n/8)+n/4+n/2+n \;=\; \cdots \;=\; T(1)+\sum_{i=0}^{k-1} n/2^i,$$

   onde $k = \log n$. Como $T(1) = 1$ e $\sum_{i=0}^{k-1} 1/2^i \leq 2$, temos que $T(n) \leq 2n+1 = O(n)$. Evidentemente $T(n) = \Omega(n)$ também, já que o particiona consome tempo $\Theta(n)$. Ou seja, a complexidade do algoritmo acima é $\Theta(n)$.

4. **(Exercício 10.3-8 do CLR)** Sejam $X[1..n]$ e $Y[1..n]$ dois vetores, cada um contendo n números ordenados. Dê um algoritmo $O(\log n)$ para encontrar a mediana de todos os $2n$ elementos nos vetores X e Y.

   Resolução: A resolução deste está postergada até vocês entregarem o algoritmo de vocês no panda.