MAC338 **PROVA 3** *4/7/2000*

Duração da prova: 2 horas

1. [3 pontos]

(i) Prove que $n^\alpha = O(n^\beta)$ se e só se $\alpha \leq \beta$.

(ii) Determine uma condição necessária e suficiente para os pares (α_1, α_2) e (β_1, β_2) para que tenhamos $n^{\alpha_1}(\log n)^{\alpha_2} = O\left((n^{\beta_1}(\log n)^{\beta_2})\right)$.

(iii) Sabe-se que o k-ésimo primo p_k é tal que $p_k = \Theta(k \log k)$. Seja $\pi(n)$ o número de números primos p com $1 < p \leq n$. Deduza que $\pi(n) = \Theta(n/\log n)$.

2. [3 pontos] O *Problema da Seleção* $s(n, k, \mathbf{x})$ tem como entrada inteiros $1 \leq k \leq n$ e uma seqüência de inteiros $\mathbf{x} = (x_1, \ldots, x_n)$. Neste problema, queremos encontrar um x tal que

$$\left|\{i\colon x_i \leq x\}\right| \geq k \quad \text{ e } \quad \left|\{i\colon x_i \geq x\}\right| \geq n - k + 1.$$

Por exemplo, o problema $s(n, 1, \mathbf{x})$ consiste na determinação do menor elemento em $\mathbf{x}$ e o problema $s(n, \lfloor n/2 \rfloor, \mathbf{x})$ consiste na determinação da mediana dos elementos em $\mathbf{x}$.

(i) Descreva um algoritmo para resolver $s(n, k, \mathbf{x})$. O seu algoritmo deve ter complexidade de pior caso $O(n \log n)$.

(ii) Suponha agora que os inteiros que ocorrem em $\mathbf{x}$ são todos distintos e que formam um conjunto (desconhecido) X. Suponha que queremos resolver $s(n, k, \mathbf{x})$, onde $\mathbf{x}$ é uma dentre todas as $n!$ possíveis permutações de X, com todas estas permutações equiprováveis. Descreva um algoritmo para resolver $s(n, k, \mathbf{x})$ nestas condições que tem complexidade de caso médio $O(n)$.

(iii) Argumente que o seu algoritmo em (ii) de fato tem complexidade de caso médio $O(n)$. Para sua resposta estar completa, você precisa fazer uma análise quantitativa de seu algoritmo.

3. [3 pontos] Nesta questão, estamos interessados em grafos orientados G. Suponha que represetamos nossos grafos por matrizes de adjacência $A(G) = (a_{ij})$, onde colocamos $a_{ij} = 1$ se o arco (i, j) está presente em G e $a_{ij} = 0$ caso contrário. Considere agora o seguinte trecho de código:

```
for (y = 0; y < n; y++)
  for (x = 0; x < n; x++)
    if (a[x][y])
      for (j = 0; j < n; j++)
        if (a[y][j]) a[x][j] = 1;
```

(Os vértices de G são $0, \ldots, n-1$.)

(i) Dê um limitante superior para o número de vezes que a atribuição `a[x][j]=1;` é executada.

(ii) Descreva o efeito do trecho de código acima (para tanto, você poderia considerar o grafo que a nova matriz representa.)

4. [3 pontos] Considere a função abaixo, que é utilizada na manipulação de heaps.

```
void fixDown(Item a[], int k, int N)
  { int j;
    while (2*k <= N)
      { j = 2*k;
        if (j < N && less(a[j], a[j+1])) j++;
        if (!less(a[k], a[j])) break;
        exch(a[k], a[j]); k = j;
      }
  }
```

(i) Em que situação usamos `fixDown()`? Descreva cuidadosamente as hipóteses sobre o heap no momento em que chamamos esta função e explicite qual é o efeito da execução de `fixDown()`.

(*ii*) O que se entende pela *altura* de um elemento em uma árvore? Não deixe de dar um exemplo ilustrativo. Lembre-se de que folhas têm altura 0.

(*iii*) Dê um limitante superior para o número de vezes que executamos `exch()` em função da altura do elemento apontado por `k` no momento da chamada de `fixDown()`.

(*iv*) Relacione o tempo de construção de um heap a partir de um vetor arbitrário com n elementos com a soma das alturas dos nós da árvore binária completa com n nós.

(*v*) Seja $f(n)$ a soma das alturas dos nós da árvore binária completa com n nós. Mostre que $f(n)$ satisfaz a recorrência $f(1) = 0$ e $f(n) = f(\lfloor n/2 \rfloor) + \lfloor n/2 \rfloor$ para $n \geq 2$.

(*vi*) Mostre que $f(n) = n - b(n)$, onde $b(n)$ é o número de bits 1 na expansão binária de n. Deduza que podemos construir um heap a partir de um vetor arbitrário com n elementos executando no máximo n trocas de elementos do vetor (execuções de `exch()`).