MAC338 **PROVA REC** *21/8/2000*

Duração da prova: 2 horas e 30 minutos

1. [2 pontos] Lembre que podemos escrever $O(f(n))$ no lugar de qualquer função $g(n)$ para a qual existam $C > 0$ e $n_0 \geq 1$ tais que

$$|g(n)| \leq Cf(n) \qquad \text{para todo } n \geq n_0.$$

(*i*) Considere as funções $f(n) = (\log n)^A$ e $g(n) = n^\varepsilon$. Em quais casos (em termos dos valores de ε e A), temos $g(n) = O(f(n))$? Justifique sua resposta.

(*ii*) Determine uma condição necessária e suficiente para as triplas $(\alpha_1, \alpha_2, \alpha_3)$ e $(\beta_1, \beta_2, \beta_3)$ para que tenhamos $n^{\alpha_1}(\log n)^{\alpha_2}(\log\log n)^{\alpha_3} = O\big((n^{\beta_1}(\log n)^{\beta_2}(\log\log n)^{\beta_3})\big)$. Neste item, você deve enunciar a condição claramente.

(*iii*) Justifique sua resposta ao item (*ii*) acima.

2. [3 pontos] Seja G um grafo orientado. Defina o grafo $G' = (V', E')$ como sendo o seguinte grafo orientado: V' é o conjunto dos componentes fortemente conexos de G. Formalmente,

$$V' = \{[x] : x \in V\},$$

onde $[x]$ denota o componente fortemente conexo de x em G. O arco $([x], [y])$ existe em G' se e só se existe um arco $(x', y') \in E(G)$ com $x' \in [x]$ e $y' \in [y]$.

(*i*) Mostre que G' é necessariamente um grafo orientado acíclico, isto é, não contém circuitos orientados.

(*ii*) Descreva o algoritmo de Kosaraju e Sharir, visto durante o semestre, para determinar os componentes fortemente conexos de um grafo orientado.

(*iii*) Com base em sua resposta ao item (*ii*) acima, deduza em que ordem o algoritmo descobre os componentes fortemente conexos do grafo dado. Intuitivamente, podemos dizer que o algoritmo 'descobre' os vértices do grafo G' do item (*i*) em uma certa ordem. É verdade que estes vértices são descobertos em uma ordem topológica de G'? (As fontes são descobertas primeiro, depois os vértices 'intermediários', e por fim os sorvedouros.) Justifique sua resposta.

3. [2 pontos] Podemos representar a estrutura das chamadas recursivas do mergesort através de uma árvore binária, que é "quase balanceada". Seja $h(n)$ a altura desta árvore quando a entrada tem n elementos. Ademais, seja $k(n)$ a profundidade mínima de uma folha em tal árvore. Mostre que temos, para todo $n > 0$,

(*i*) $h(n) = \lceil \log_2 n \rceil$,

(*ii*) $k(n) = \lfloor \log_2 n \rfloor$.

(*iii*) Deduza que esta árvore é balanceada se n é uma potência de 2.

4. [3 pontos] Considere a seguinte função de partição de Kernighan e Pike, dado no livro *The Practice of Programming*.

```
int partition(Item a[], int l, int r)
  { int i, last; Item v = a[r];
    last = r;
    for (i=r-1; i>=l; i--) {
      if (less(v, a[i]))
        { --last; exch(a[i], a[last]); }
    }
    exch(a[last], a[r]);
    return last;
  }
```

Nesta questão, você deve supor que `a[]` contém os elementos $\{0, \ldots, n-1\}$, em alguma ordem.

(*i*) Quantas vezes a troca `exch(a[i], a[last])` é executada quando $a[r] = k$?

(*ii*) Suponha agora que a entrada `a[]` contém uma permutação de $\{0,\ldots,n-1\}$, com todas as permutações equiprováveis. Mostre que o número esperado de trocas `exch(a[i], a[last])` que serão executadas é $(n-1)/2$.

Considere agora a função de partição de Sedgewick abaixo.

```
int partition(Item a[], int l, int r)
  { int i = l-1, j = r; Item v = a[r];
    for (;;)
      { while (less(a[++i], v)) ;
        while (less(v, a[--j])) if (j == l) break;
        if (i >= j) break;
        exch(a[i], a[j]);
      }
    exch(a[i], a[r]);
    return i;
  }
```

(*i*) Qual é o número total de comparaçãos que fazemos dos elementos de `a[]` com `v`? (Isto é, número de comparações `less(a[++i], v)` mais o número de comparações `less(v, a[--j])`.)

(*ii*) Suponha agora que $\mathtt{a[r]} = k$. Seja q o número índices i $(0 \le i < k)$ para os quais temos $\mathtt{a}[i] \ge k$. Quantas vezes executamos a troca `exch(a[i], a[j])`?

(*iii*) Suponha agora que a entrada `a[]` contém uma permutação de $\{0,\ldots,n-1\}$, com todas as permutações equiprováveis. Qual é o valor esperado de q do item (*ii*), *supondo que* $\mathtt{a[r]} = k$?

(*iv*) Determine o número esperado de execuções da troca `exch(a[i], a[j])`, supondo novamente que a entrada `a[]` contém uma permutação de $\{0,\ldots,n-1\}$, com todas as permutações equiprováveis.