MAC338 **PROVA 2** *20/6/2000*

Duração da prova: 2 horas

1. [3 pontos] Considere a seguinte implementação do algoritmo de ordenação por inserçao.

```
void insertion(Item a[], int l, int r)
  { int i;
    for (i = l+1; i <= r; i++) compexch(a[l], a[i]);
    for (i = l+2; i <= r; i++)
      { int j = i; Item v = a[i];
        while (less(v, a[j-1]))
          { a[j] = a[j-1]; j--; }
        a[j] = v;
      }
  }
```

(i) O primeiro `for` coloca um elemento mínimo do vetor dado na posição `a[l]`. Suponha que omitimos este `for` na implementação acima e que incializamos `i` no segundo `for` com `l+1`. Esta nova implementação funciona? Justifique.

(ii) Explique a vantagem de se colocar um elemento mínimo do vetor dado em `a[l]`.

(iii) A partir deste item, você deve supor que `a[]` contém os elementos $\{0, \ldots, n-1\}$, em alguma ordem. Uma *inversão* no vetor `a[]` é um par (i, j) com $i < j$ e `a`$[i] >$ `a`$[j]$. Suponha que $n = 7$ e que `a`$[i] = 3i \bmod 7$. Determine o número de inversões neste vetor `a[]`.

(iv) Dizemos que um elemento `a`$[i]$ está *envolvido* em q inversões se o numero de pares (j, i) que são inversões em `a[]` é q. Se `a`$[i]$ está envolvido em q inversões, quantas vezes a atribuição `a[j] = a[j-1]` é executada no procedimento acima? Quantas vezes a comparação `less(v, a[j-1])` é feita?

(v) O *deslocamento* de um elemento i $(0 \leq i < n)$ em `a[]` é a quantidade $|j-i|$, onde j é tal que `a`$[j] = i$. Isto é, o deslocamento de um elemento é a diferença entre a posição em que ele se encontra e a posição em que ele ficará ao ordenarmos o vetor. Suponha que o deslocamento de todo elemento i $(0 \leq i < n)$ é de no máximo m em uma dada instância `a[]`. Mostre que o número de inversões em que pode estar envolvido um elemento `a`$[i]$ é de no máximo $2m - 1$.

(vi) Prove que o tempo de execução da chamada `insertion(a,0,n-1)` em uma instância `a[]` como descrito em (v) é $O(mn)$. Em particular, se m é uma constante, ordenamos `a[]` em tempo linear em n.

2. [3 pontos] Considere a seguinte versão iterativa (não-recursiva) do quicksort.

```
#define push2(A, B)  push(B); push(A);
void quicksort(Item a[], int l, int r)
  { int i;
    stackinit(); push2(l, r);
    while (!stackempty())
      { l = pop(); r = pop();
        if (r <= l) continue;
        i = partition(a, l, r);
        if (i-l > r-i) { push2(l, i-1); push2(i+1, r); }
        else { push2(i+1, r); push2(l, i-1); }
      }
  }
```

(i) Ao particionarmos um vetor dado em duas partes para as chamadas recursivas, qual das partes é ordenado primeiro na implementação acima?

(*ii*) Seja $t(n)$ o número máximo de pares de elementos `(l,r)` que a pilha conterá quando executamos `quicksort(a,0,n-1)`. Mostre que $t(0) = t(1) = 1$ e que

$$t(n) \leq \max\big\{ \max\{t(a) + 1,\, t(b)\}\colon\, a + b = n - 1 \text{ e } 0 \leq a \leq b\big\}$$

para todo $n \geq 2$.

(*iii*) Considere a seqüência de números s_n $(n \geq 0)$ dada pela recorrência $s_0 = s_1 = 1$ e $s_n = s_{\lfloor n/2 \rfloor} + 1$ para $n \geq 2$. Faça uma tabela com os valores de s_n para valores pequenos de n.

(*iv*) Baseado na tabela em (*iii*), intua uma fórmula fechada para s_n. Prove esta fórmula por indução em n.

(*v*) Prove por indução em n que $t(n) \leq s_n$ para todo $n \geq 0$.

(*vi*) Suponha agora que trocamos `i-l > r-i` por `i-l <= r-i` no procedimento acima. Dê, para cada n, uma instância de tamanho n para esta variante do procedimento acima que força o uso de mais que uma quantidade logarítmica de espaço para a pilha. Quanto espaço é necessário para a pilha em suas instâncias?

3. [3 pontos] Suponha que temos k listas de inteiros, cada uma delas ordenadas em ordem crescente. Suponha que o número total de inteiros que temos é n; isto é, a soma dos números de elementos nas lista é n. Suponha agora que queremos obter uma *única lista* com estes n inteiros em ordem crescente. Descreva uma estrutura de dados que permita obter esta nova lista em tempo $O(k + n \log k)$. [Não dê os detalhes de implementação; diga, em detalhe, quais estruturas usar e como elas devem ser usadas—uma boa forma de responder esta questão é dar uma descrição de sua solução em palavras e então dar um pseudocódigo.]

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 $\mathtt{a}[\mathtt{r}] = k$. Seja q o número índices i ($0 \leq i < k$) para os quais temos $\mathtt{a}[i] \geq k$. 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* $\mathtt{a}[\mathtt{r}] = k$?

(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. [Este valor é por volta de $n/6$.]