

**SEGUNDA PROVA DE PRINCÍPIOS DE DESENVOLVIMENTO DE
ALGORITMOS
BE/BM/BMC, 2o. SEMESTRE DE 2006**

Instruções:

1. Não destaque as folhas do caderno de soluções.
2. A prova pode ser feita a lápis. Cuidado com a legibilidade.
3. Não é permitido o uso de folhas avulsas para rascunho.
4. Não é necessário apagar rascunhos no caderno de soluções.
5. Asserções imprecisas valem pouco. Justifique suas asserções (dentro do razoável!).

1. [4 pontos] Nesta questão, supomos que temos

```
typedef struct node *link;  
struct node { int item; link next; };
```

Ademais, supomos que nossas listas não tem cabeça nem cauda. Considere a função recursiva

```
link delete(link x, int v)  
{ if (x == NULL) return NULL;  
  if (x->item == v) { link t = x->next; free(x); return t; }  
  x->next = delete(x->next, v);  
  return x;  
}
```

- (i) Suponha que x aponta para a primeira célula de uma lista cujas células contêm os inteiros 22, 77, 11, 88, 22, 88, 11, nesta ordem. Diga exatamente o efeito da chamada $x = \text{delete}(x, 11)$ (não é necessário “simular” a execução da função; basta você descrever o resultado precisamente), mas justifique como você chegou em sua resposta.
- (ii) Altere a função `delete()` levemente para que ela remova todas as células com item v .
- (iii) Escreva uma função recursiva de protótipo

```
int count(link x, int v)
```

que, ao ser chamado com x apontando para a primeira célula de uma lista ligada e v um inteiro, devolve o número de células na lista que tem item igual a v .

2. [4 pontos] Considere a seguinte implementação do quicksort, de Kernighan e Pike (“The Practice of Programming”, Addison Wesley Longman, 1999):

```
void swap(int v[], int i, int j)
{ int temp = v[i]; v[i] = v[j]; v[j] = temp; }

void quicksort(int v[], int n)
{   int i, last;
    if (n <= 1) return; /* nothing to do */
    last = 0;
    for (i = 1; i < n; i++)
        if (v[i] < v[0])                /* v[0] is the pivot */
            swap(v, ++last, i);
    swap(v, 0, last);                    /* restore pivot */
    quicksort(v, last);                  /* recursively sort */
    quicksort(v + last + 1, n - last - 1); /* each part */
}
```

Lembre-se também do `partition()` que conhecemos:

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

- (i) Note que o algoritmo de partição de Kernighan e Pike é diferente do algoritmo em `partition()`. Dê uma entrada para a qual o algoritmo de Kernighan e Pike escreve valores no vetor `v` pelo menos $2n$ vezes para particionar um vetor com n itens (note que `swap()` escreve duas vezes em `v`).
- (ii) Dê uma cota superior para o número de vezes que `partition()` escreve valores no vetor sendo particionado `a`.
- (iii) Sejam $C_1(n)$ e $C_2(n)$ o número de comparações entre itens que fazem os dois algoritmos de partição acima, para vetores com n entradas. Dê boas cotas superiores para $C_1(n)$ e $C_2(n)$.

3. [4 pontos] Considere o algoritmo de intercalação visto em sala abaixo.

```

Item aux[maxN];
void merge(Item a[], int l, int m, int r)
{
    int i, j, k;
    for (i = m+1; i > l; i--) aux[i-1] = a[i-1];
    for (j = m; j < r; j++) aux[r+m-j] = a[j+1];
    for (k = l; k <= r; k++)
        if (less(aux[i], aux[j]))
            a[k] = aux[i++];
        else
            a[k] = aux[j--];
}

```

- (i) Seja $N = r - l + 1$. Quantas comparações `less(aux[i], aux[j])` são feitas em uma chamada de `merge(a, l, m, r)`?
- (ii) Seja $C(N)$ o número de comparações feitas por mergesort para ordenar um vetor com N elementos. Prove então que

$$C(1) = 0 \quad (1)$$

e

$$C(N) = C(\lfloor N/2 \rfloor) + C(\lceil N/2 \rceil) + N \quad \text{para } N > 1. \quad (2)$$

- (iii) Faça uma tabela para $C(N)$ para valores pequenos de N .
- (iv) Podemos provar que

$$C(N) = N \lceil \log_2 N \rceil + N - 2^{\lceil \log_2 N \rceil}. \quad (3)$$

Em vez de provar (3) em geral, prove que esta fórmula vale se $N = 2^k$ para um inteiro $k \geq$

0. [*Sugestão.* Use indução em k .]