T.A.A. Folha Teórica, dia 20090414

Complementos sobre ordenação – I

1. **Ordenação estável**

   Um algoritmo de ordenação diz-se estável se $\{v[i] = v[j] \wedge i < j\} \Rightarrow i' < j'$, onde i' e j' são as posições onde são colocados os elementos que no início estão nas posições de índices i e j, respectivamente.

   - Explique através de um exemplo prático o interesse de uma ordenação ser estável.
   - Escreva o algoritmo da ordenação pela selecção do mínimo por forma a que seja estável.
   - Escreva o algoritmo da ordenação pela selecção do mínimo por forma a que não seja estável.
   - Descreva um processo de modificar um algoritmo de ordenação eventualmente não estável por forma a que passe a sê-lo.

2. **Ordenar sem comparar**

   Suponha que os valores a ordenar são inteiros positivos, não muito grandes e sem repetições. Descreva um algoritmo que os ordene sem efectuar qualquer comparação entre os elementos do vector; nenhum operador de relação ($<$, $<=$, etc.) deve existir no seu algoritmo (mas pode usar ciclos `for` ou `while`)!

3. **A mesma ordem exacta**

   Mostre que, sendo n e m parâmetros positivos, $\Theta(n + m) = \Theta(\max\{n, m\})$.

4. **Uma representação de conjuntos**

   Suponha que um sub-conjunto A de $\{1, 2, \ldots, n\}$ é representado por um vector a constituído por n valores booleanos, sendo $a[x] =$ `True` sse $x \in A$.

   (a) Com $n = 10$, qual a representação de $\{2, 5, 6\}$?

   (b) Escreve uma função `inters(a,b)` que retorna a representação da intersecção dos conjuntos representados por `a` e `b`.

   (c) Escreve uma função `sdiff(a,b)` que retorna a representação da diferença simétrica[1] dos conjuntos representados por `a` e `b`.

   (d) Quais podem ser na prática os inconvenientes desta representação de conjuntos?

[1] A diferença simétrica dos conjuntos A e B é o conjunto dos elementos que pertencem a A ou a B, mas não a ambos, $(A \cup B) \setminus A \cap B\}$

5. **Um algoritmo de ordenação**

Considere o algoritmo seguinte

```
1   for i=1 to u:  c[i] = 0
2   // incrementa c[i] se v[i] existe
3   for i=1 to n:  c[v[i]] = c[v[i]]+1
4   k=0
5   for i=1 to u:   // percorre c
6     for j=1 to c[i]:
7       v[k]=i
8       k=k+1
```

(a) Execute mentalmente o algoritmo com os valores `n=5`, `v[]=[3,5,2,2,3]` (índices contados a partir de 1) e `u=6`, indicando os valores finais dos vectores `v` e `c`.

(b) Analise a ordem de grandeza do tempo de execução deste algoritmo, considerando as contribuições das linhas: `1`, `3`, `4`, `5-8`. Com vista a esta última contribuição, indique o número de vezes que (i) o teste do `for` da linha `5` (`i<=u?`) é efectuado, (ii) o teste do `for` da linha `6` é efectuado e (iii) as linhas `7-8` são executadas.