Armando Matos, 2008/2009

Tópicos Avançados em Algoritmos - Folha de exercícios 05

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

2. Suponha que a ordenação do vector `v[1..n]` resulta no vector `w[1..n]`, sendo, para `i`=1, 2,... `n`, o elemento `v[i]` movido para o índice `m(i)` do vector `w`, isto é, para `w[m[i]]`. Em termos desta formalização, defina ordenação estável.

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

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

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

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

4. Mostre que o algoritmo seguinte ordena correctamente os valores do vector `v[]`.

```
 Algoritmo de ordenação por contagem
 Vector a ordenar: v[1..n]
 O resultado fica no vector w[1..n]

1   for i=1 to u:  c[i] = 0
2   for i=1 to n:  c[v[i]] = c[v[i]]+1
3   for i=2 to n:  c[i] = c[i]+c[i-1]
4   for i=n downto 1:
5     w[c[v[i]]] = v[i]
6     c[v[i]] = c[v[i]] -1
```

   Na sua justificação os seguintes passos podem ser úteis:

   - Seguir o algoritmo para um pequeno exemplo (com elementos repetidos).
   - Indicar a função das variáveis usadas, incluindo a do vector `c[]`; note-se que o significado da variável `i` depende da linha em consideração.
   - Descrever os efeitos e o conteúdo dos vectores `v[]`, `c[]`, e `w[]` após cada uma das linhas 1, 2, 3, 4, 5 e 6.

5. Mostre que o algoritmo do problema anterior é estável.

[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, $(A \cup B) \setminus A \cap B\}$

6. Explique e analise o programa apresentado onde n é o número de elementos do vector `lista` a ordenar e m é o número (máximo) de símbolos de cada elemento. Supõe-se que as operações elementares das filas são $O(1)$ (o que na linguagem python é duvidoso...); mostre que a ordem de grandeza do algoritmo é $\Theta(mn)$.

```
   # comprimento do alfabeto
   nalfa=256

   # radix sort; (menos significativo primeiro)
    def radixsort(lista):
1       m=max(map(len,lista))      # o maior comprimento de um string
2       balde=[0]*nalfa            # vector com nalfa zeros: [0,0,...,0]
3       for i in range(0,nalfa):
4           balde[i] = Fila()      # uma fila por índice
5       for i in range(m-1,-1,-1): # dos menos para os mais significativos...
6           rsort(lista,i,m,balde) # ... ordena a lista
7       return lista

   #-- ordena segundo simbolo de ordem d
   def rsort(lista,d,m,balde):
A       for x in lista:
B           # x pode ter comprimento < d!
C           if d<len(x):
D               indice=ord(x[d])
E           else:
F               indice=0
G           balde[indice].entra(x)
H       ilista=0
I       for d in range(nalfa):
J           while not balde[d].vazia():
K               lista[ilista]=balde[d].sai()
L               ilista += 1
M       return lista

   class Fila:
        def __init__(self):
            self.fila=[]
        def vazia(self):
            return self.fila==[]
        def entra(self,x):          # entra um elemento para a fila (trás)
            self.fila.append(x)
        def sai(self):              # sai um elemento da frente da fila
            return self.fila.pop(0)

   #-----Exemplo de uso ---------------------------------------------
   >>> a=["bbc","abc","acb","ccb","z","za","ccb"]
   >>> print radixsort(a)
       ['abc', 'acb', 'bbc', 'ccb', 'ccb', 'z', 'za']

   >>> b=["b","a","c"]
   >>> print radixsort(b)
       ['a', 'b', 'c']
```