Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Algoritmos e Estruturas de Dados
Semestre de Verão 2018/19
Primeira série de problemas
Parte 1

Observações:

- Data de entrega: **4 de Abril de 2019**.
- No contexto desta série, o triplo *(v, l, r)* representa o *subarray* do *array* v, compreendido entre os índices l e r inclusivé.

1 Algoritmos Elementares

1. Realize o método estático

   ```
   public static int indexOfSmallest(int[] v, int l, int r)
   ```

   que dado um *sub-array* (v, l, r) ordenado de modo estritamente crescente, cujos elementos foram deslocados circularmente um determinado número de posições para a direita, retorna o índice onde se encontra o menor elemento do *sub-array*. Caso não existam elementos no *sub-array*, o método deverá retornar -1. Por exemplo, o *sub-array* $(v, 0, 5)$, em que $v = \{30, 36, 10, 12, 22, 23\}$, é um *sub-array* ordenado em que os elementos foram deslocados circularmente 2 posições para a direita. Neste caso, o algoritmo deverá retornar 2.

2. Realize o método estático

   ```
   public static int numberOfDistinct(int[] v1, int[] v2)
   ```

   que retorna o número de inteiros distintos que existem simultâneamente em ambos os *arrays* `v1` e `v2` ordenados de modo crescente. Por exemplo, caso `v1` contenha `[1, 1, 2, 2, 3, 3, 4, 4, 5, 5]` e `v2` contenha `[2, 2, 2, 5, 5]` o resultado da avaliação do método deve ser 2 (correspondente aos inteiros 2 e 5 que ocorrem simultaneamente em ambos os *arrays*).

3. Realize o método estático

   ```
   public static int[] squaresSorted(int[] v)
   ```

   que dado o *array* `v` ordenado de modo crescente, retorna um novo *array*, ordenado de modo crescente, composto pelo quadrado dos inteiros presentes em `v`. Note que os inteiros podem ser negativos.

4. Realize o método estático

   ```
   public static int countCommonPoints(Point[] a, Point[] b)
   ```

   que dados dois *arrays* `a` e `b` de pontos no plano, determina o número de pontos em comum entre os dois *arrays*. Assuma que cada um dos *arrays* não contém pontos repetidos. Considere que a classe `Ponto` é definida por:

   ```
   public class Ponto{
      public int x;
      public int y;
   }
   ```

   Por exemplo, os *arrays* $[\{5,3\},\{2,7\},\{5,4\},\{3,6\}]$ e $[\{3,6\},\{2,6\},\{5,3\},\{3,5\}]$ contém 2 pontos em comum.

5. Realize o método estático

   ```
   public static int lessFrequent(int[] v)
   ```

   que dado um *array* `v` de elementos, retorna o elemento, presente em `v`, com o menor número de ocorrências. Assuma que os elementos presentes em `v` pertencem a um intervalo $[min, max]$ em que $max - min + 1 \leq v.length$. Em caso de empate, retorna o primeiro elemento encontrado. Indique a complexidade da sua solução.

2 Análise de desempenho

1. Considere o algoritmo `xpto`.

```
public static long xpto(long x, long n){
   if (n == 0) return 1;
   if (n % 2 == 0) return xpto(x, n/2) * xpto(x, n/2);
   return xpto(x, n/2) * xpto(x, n/2) * x;
}
```

   1.1. Indique a complexidade do mesmo, no pior caso, em função de `n`. Indique a equação de recorrência e resolva-a.

   1.2. Será possível diminuir a complexidade deste método, mantendo a mesma funcionalidade? Justifique.

2. Considere o algoritmo `method`, que recebe como parâmetro dois inteiros `n` e `m`.

```
method(n, m) {
if (n / m = 0)
     return 0
   return 1 + method (n / m, m)
```

   2.1. Considerando que m=2, indique justificando, a complexidade de `method` em função de n.

   2.2. Indique justificando, a complexidade de `method`.

3. (1.5) Considere o algoritmo `xpto`. Indique a complexidade do mesmo em função de $n = r - l + 1$. Indique a equação de recorrência e resolva-a. Assuma que a operação `aux` tem complexidade $O(n)$.

```
static int xpto(int[] a, int l, int r) {
  if(r==l)  return a[l];
  int m=l+(r-l)/2;
  int lsub = xpto(a,l,m);
  int rsub = xpto(a,m+1,r);
  if(lsub == rsub) return lsub;
  if(aux(a,l,r,lsub)>=m) return lsub;
  if(aux(a,l,r,rsub)>=m) return rsub;
  return -1;
 }
```

ISEL, 14 de Março de 2018.