Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Algoritmos e Estruturas de Dados
Semestre de Verão 2016/17
Primeira série de problemas

Observações:

- Data de entrega: **26 de Abril de 2017**.
- 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

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

   que dado o *array* `v` retorna dois elementos desse *array* cuja soma seja o mais próximo de zero possível (ou zero, caso existam). Esses dois elementos são retornados através de um *array* com duas posições. Caso não existam pelo menos dois elementos nesse *array*, deve ser retornado `null`. Indique, justificando, a complexidade do método.

2. Realize o método

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

   que, dado o *subarray* (v, l, r), retorna um *array* contendo os índices que delimitam o início e o fim da maior subsequência crescente de elementos contíguos, presentes no array v. Em caso de empate, retorna os índices da primeira subsequência encontrada. Por exemplo, se $v = \{30, 36, 10, 12, 22, 23\}$, $l = 0$ e $r = 5$, o *array* a retornar devera ser $\{2, 5\}$.

3. Realize o método

   ```
   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

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

   que retorna a mediana dos elementos do *sub-array* (v, l, r). A *mediana* é o valor que separa a metade maior e a metade menor de um conjunto de dados. No entanto, se a dimensão do conjunto for par, a mediana é definida como a média dos dois valores do meio. Valorizam-se soluções $\Omega(\log n)$.

5. Realize o metodo

   ```
   public static int greatestOccurrence(int[] v, int min, int max)
   ```

   que dado um *array* `v` em que os elementos pertencem ao intervalo $[min, max]$, e que $min, max \in \mathbb{N}$, retorna o elemento nele presente que tenha o maior número de ocorrências. Em caso de empate, retorna o primeiro encontrado. Considere que $max - min + 1 \leq v.length$. A solução deve ter complexidade $O(n)$, em que $n = v.length$.

2 Análise de desempenho

1. Considere os algoritmos de ordenação estudados.

   1.1. Indique um algoritmo cujo custo, qualquer que sejam os valores presentes na sequência a ordenar, pertence a $O(n^2)$ mas não pertence a $\Theta(n^2)$.

   1.2. Indique um algoritmo cujo custo, quaisquer que sejam os valores presentes na sequência a ordenar, pertence a $O(n \log n)$ mas nao pertence a $\Theta(n \log n)$.

2. Considere o seguinte excerto de código. Assuma que `a` e `b` são *arrays* de inteiros e que $n \geq 0$ e $k \geq 0$.

```
int k=a.length;
int n=b.length;
System.out.println(xpto(a,b,n));
...
```

   em que

```
public static int xpto(int[] a, int[] b, int n){
  if(n>0){
    if(binarySearch(a,b[--n])!=-1)
        return 1 + xpto(a, b, n);
       /*o algoritmo pesquisa binária retorna -1 se o elemento não estiver presente em a*/
  }
  return 0;
}
```

   Indique, justificando, a complexidade deste excerto de código.

3. Considere o algoritmo `xpto`, que recebe como parâmetro dois inteiros n e m.

```
public static int xpto( int n, int m ){
  if ( n/m == 0 ) return 0;
  return 1 + xpto ( n/m , m);
}
```

   3.1. Considerando que $m = 2$, indique, justificando, a complexidade de `xpto` em função de n.

   3.2. Indique, justificando, a complexidade de `xpto`.

4. Indique, justificando, como implementaria, de modo estável, um algoritmo de ordenação que em a cada i-ésima iteração seleciona o i-ésimo menor elemento dos restantes ainda não selecionados. Analise a sua solução quanto ao tempo e ao espaço.

3 Problema: Mediana incremental

Considere que se pretende realizar uma aplicação que permita actualizar e obter a qualquer momento a mediana de um conjunto de inteiros, obtidos a partir do (*input stream*) ao longo do tempo. Assuma que, antes de iniciar a aplicação esse conjunto está vazio. **As operações de atualização deverá pertencer a** $O(log\ n)$ **e a obtenção da mediana deverá pertencer a** $O(1)$.

Parâmetros de execução

Para iniciar a execução da aplicação a desenvolver, terá de se executar:

```
java median
```

que inicializa o conjunto vazio de inteiros. **Durante a sua execução**, a aplicação processa os seguintes comandos:

- `updateSet i`
  que insere o elemento i no conjunto e actualiza a mediana do conjunto.o.

- `getMedian`
  que retorna a mediana dos elementos pertencentes ao conjunto até ao momento.
- `e`
  que termina a aplicação.

Avaliação Experimental

Realize uma avaliação experimental do(s) algoritmo(s) desenvolvido(s) para a resolução deste problema. Apresente os resultados graficamente, utilizando uma escala adequada.

ISEL, 16 de Abril de 2017.