AED, 2017/2018
Aula prática 1

Prazos

- **Pontuação máxima:** 15 pontos
- **Prazo de entrega:** 16/Fev/2018, 23:55
- **Entrega:** pela tutoria electrónica.

  Devem submeter um único ficheiro ZIP com o seguinte nome: `PL<d>-g<X>.zip` onde d é o número da vossa turma prática e X é o vosso número de grupo (que vos será atribuído em breve). Ou seja, o grupo 5 da turma PL2 terá de submeter o ficheiro com o nome `PL2-g5.zip`.

  - O ficheiro zip deverá conter um ficheiro com o nome `relatorio.pdf` com as respostas às questões solicitadas.
  - O ficheiro zip também deverá incluir o código desenvolvido.
  - Não serão aceites submissões que não respeitem estas regras.

Exercício

O *Insertion Sort* é um algoritmo de ordenação que funciona de modo análogo ao modo como muitos de nós ordenamos uma mão de cartas: mantemos as cartas na mão sempre ordenadas, e, ao recolhermos mais uma carta, inserimo-la na posição correcta para que as cartas na mão continuem ordenadas.

Apresenta-se seguida o pseudocódigo do algoritmo ao estilo que vão encontrar no livro recomendado da disciplina.

INSERTION-SORT(A)

```
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

No pseudocódigo acima apresentado os números de linha que aparecem à esquerda são meramente indicativos e ajudam-nos quando nos queremos referir a uma determinada linha do código. Note que o array A começa na posição 1, não na posição 0 como em C e Java. Esta é uma convenção que é adoptada no livro.

1. Tente compreender o algoritmo de modo a ficar convencido que de facto ele ordena o array A por ordem crescente. Se não ficar totalmente convencido, execute o algoritmo à mão, passo a passo, usando papel e lápis, no seguinte input de 5 elementos: [6, 2, 7, 8, 3].

2. Implemente o algoritmo INSERTION-SORT em Java e certifique-se que a sua implementação está correcta.

   (Ao traduzir o pseudocódigo para Java, há duas maneiras de lidar com a questão dos arrays começarem na posição 1 em vez da posição 0. A primeira é alterar a manipulação dos indíces do array de forma apropriada (é fácil cometermos erros se não o fizermos de forma cuidadosa). A segunda é reservar mais uma posição do array e ignorar a posição 0. Isto é, se precisar de ordenar um array A com n elementos, declare o array em Java com $n+1$ elementos e o seu array fica $A[0\,..\,n]$ mas $A[0]$ nunca é usado.)

3. Teste o algoritmo em inputs gerados aleatoriamente de tamanho $2^{10} = 1024$, 2^{11}, 2^{12}, ..., $2^{20} = 1048576$, e meça o tempo de execução do algoritmo. Cada elemento do array de input deverá ser um número inteiro aleatório no intervalo $[1\,..\,100000]$. (Não é necessário ir até à dimensão 2^{20} caso o algoritmo se torne demasiado lento no seu computador. Pode terminar os testes quando as execuções do algoritmo começarem a exceder 5 minutos.)

   Para o auxiliar, junta-se código Java para lançar n dados. O resultado de um lançamento de um dado é um número aleatório entre 1 e 6. Primeiro devemos importar a biblioteca,

   ```
   import java.util.Random;
   ```

   depois podemos fazer algo deste estilo,

   ```
   Random rand = new Random();
   for( int i=0; i<n; i++ )
      System.out.println( 1 + rand.nextInt(6) );
   ```

   Para medir o tempo de execução faça uso de `System.currentTimeMillis()` que dá o tempo actual em milisegundos. Meça exclusivamente o tempo gasto na ordenação. Algo deste género,

   ```
   Initialize(A);
   long start = System.currentTimeMillis();
   InsertionSort(A);
   long end = System.currentTimeMillis();
   System.out.println("Sorting time is " + (end-start)/1000.0 + " secs");
   ```

4. Faça um gráfico com os resultados obtidos (dimensão do array no eixo do x, tempo de execução no eixo do y). O que pode concluir?

5. Altere o código de INSERTION-SORT de tal forma que para além de ordenar o array de input, o algoritmo retorne o número de vezes que a instrução $A[i+1] = A[i]$ é executada (linha 6 do pseudocódigo).

6. Faça um novo gráfico com os resultados obtidos (dimensão do array no eixo do x, número de execuções de $A[i+1] = A[i]$ no eixo do y). O que pode concluir?

7. Indique uma estimativa do tempo que o seu algoritmo demoraria a ordenar um array de 10000000 (10 milhões) de inteiros gerados aleatoriamente. Justifique a resposta.

8. Repita todo o exercício quando o input não é gerado de forma aleatória mas sim quando corresponde ao melhor caso. Isto é, quando o input é de tal forma que dá o mínimo de trabalho possível ao algoritmo. Não é difícil de ver que o melhor caso neste exemplo ocorre quando o input está ordenado por ordem crescente logo à partida.

9. Repita a alínea 8 quando o input corresponde ao pior caso. Também não é difícil de ver que o pior caso ocorre quando o input está ordenado por ordem decrescente.