UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA

***Departamento de Engenharia Informática***

# Trabalho nº 3 de Algoritmos e Estruturas de Dados

## 2012-2013 – 2º Semestre

**Submissão Mooshak Tarefas A e B:**
21 de Abril de 2013 23:00.

**Submissão Mooshak Tarefas C:**
29 de Abril de 2013 23:00.

**Entrega Tarefa D:** 3 de Maio de 2013 17:00
no cacifo do docente das aulas teóricas.

*Nota Importante: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional licenciado. Qualquer tentativa de fraude pode levar a anulação da componente prática tanto do facilitador como do prevaricador.*

*A utilização de código disponível na net, relativo a algoritmos de ordenamento, desde que não infrinja direitos de autor é permitido neste trabalho. É assumido que este código foi compreendido com nível de detalhe idêntico ao esperado se tivesse sido desenvolvido pelo aluno. A explicação do código é matéria de avaliação desta ficha.*

***Só são aceites para defesa trabalhos para os quais os alunos tenham entregue o relatório dentro dos prazos definidos.***

**Objectivos:**

Estudo de algoritmos de ordenamento incluídos no plano da Disciplina.

**Tarefas**

- Tarefa A – Ordenamento por Inserção
- Tarefa B – Ordenamento à escolha do aluno
- Tarefa C – Ordenamento à escolha do aluno
- Tarefa D – Relatório

---

A escolha de um algoritmo de ordenamento representa um compromisso entre eficiência temporal, espaço de memória ocupada e complexidade do algoritmo, não havendo um que seja universalmente superior a todos os outros.

Na escolha do algoritmo a adoptar é importante ter em atenção a utilização específica que este vai ter, nomeadamente a dimensão dos ficheiros a ordenar e o número de vezes que o programa vai ser utilizado.

---

A empresa NovasRotas Lda dedica-se ao desenvolvimento de software para gestão de frotas de veículos. O software utilizado envolve a existência de equipamento de localização a bordo das viaturas, das quais são recebidos registos com os seguintes campos:

- Identificador do registo ("integer")
- Identificador da viatura ("integer")
- "timestamp" (AAAA-MM-DD HH:MM:SS)
- latitude ("float")
- longitude ("float")
- cidade ("string")

A seguir apresenta-se uma amostra dos registos recebidos, em que a primeira linha contém o nome das colunas (os ficheiros de entrada de teste não contêm a linha de cabeçalho):

```
id,id_vehicle,date,latitude,longitude,city
1245619,879,2009-12-09 00:00:47,38.832,-9.09474,Lisbon
1245618,320,2009-12-09 00:00:53,38.7401,-9.22841,Lisbon
1245617,720,2009-12-09 00:00:55,38.7206,-9.12267,Lisbon
```

Nas tarefas de ordenamento:

- Para as medições de tempo deve adoptar soluções que minimizem a contaminação do tempo de execução pelo tempo de leitura e escrita dos dados, medindo dentro do possível só os tempos de ordenamento. Se em alguma situação tal não for possível deve ser identificada.
- Na tarefa C não se exige que o ordenamento seja estável.
- Não se deve preocupar com a existência de registos duplicados.
- Considerando que vai ter uma chave principal e uma secundárias para ordenamento, o método que usar para combinar estes dois ordenamentos deve ser consistente ao longo das várias tarefas.

**Tarefa A :: ordenamento por inserção**

- Ordene os registos por ordem crescente do "identificador da viatura" (chave principal) seguido de ordem crescente do timestamp (chave secundária).
- Ordene os registos por distância crescente ao ponto de origem[1][2] .
- Teste no mooshak.

**Tarefa B :: ordenamento eficiente**

- Decida sobre o algoritmo temporalmente mais eficiente para organizar a amostra 6 por ordem crescente do "identificador da viatura" (chave principal) seguido de ordem crescente do timestamp (chave secundária).
- Teste no mooshak.
- Teste no ambiente de desenvolvimento com a amostra que lhe vai ser disponibilizada.

---

[1] Ponto de origem dado pelas coordenadas 38.750, -9.110 .

[2] Para cálculo da distância entre dois pontos usar o algoritmo com a fórmula haversine, descrito em http://stackoverflow.com/questions/365826/calculate-distance-between-2-gps-coordinates

**Tarefa C :: ordenamento eficiente**

- Decida sobre o algoritmo temporalmente mais eficiente para organizar a amostra 6 por distância crescente ao ponto de origem.
- Teste no mooshak.
- Teste no ambiente de desenvolvimento com a amostra que lhe vai ser disponibilizada.

**Tarefa D :: relatório**

O relatório deve compreender três partes:

1. Avaliação Experimental
*Medições de desempenho na forma de gráfico para os algoritmos considerados nas tarefas A a C, utilizando a amostra 6, a 1%, 20%, 40%, 60%, 80%, 100% dos dados desta amostra*[3].

2. Tomada de Decisão 1 (max. 1000 caracteres)
*Justificação da decisão tomada na escolha do algoritmo de ordenamento por "identificador da viatura" e "timestamp".*

3. Tomada de Decisão 2 (max. 1000 caracteres)
*Justificação da decisão tomada na escolha do algoritmo de ordenamento por "distância ao ponto de origem".*

## Referências


Introduction to Algorithms, 2nd Edition
Thomas H. Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein
The MIT Press, 2001, 2005

Algorithms in C, Third Edition
Parts 1-4
Robert Sedgewick
Addison-Wesley

Materiais da disciplina de AED (2009-2010)


## Testes no Mooshak

**Input**

O *input* é composto por uma linha com o número de elementos a ordenar. Seguido das linhas com os registos de seguimento dos veículos.

[3] Para o algoritmo de ordenamento por inserção, caso os tempos de execução se tornem demasiado longos, terminar as medições de tempo com uma porção da amostra que julgue razoável.

**Deve ainda ter em atenção, nas opções tomadas, de que se pretende que estes algoritmos manipulem ficheiros com um número de registos da ordem de $10^6$, com quantidades consideráveis de chaves repetidas, com comprimento da chaves variável e em que se privilegia a rapidez de processamento relativamente à quantidade de memória ocupada.**

**Output**

O output deverá compreender um conjunto de linhas, cada linha com a chave/chaves em causa por ordem crescente e mantendo a informação original.

**Nota: tanto a última linha de input como de output terminam com o carácter de mudança de linha.**

Exemplo de *input*

```
4[\n]
1245619,879,2009-12-09 00:00:47,38.832,-9.09474,Lisbon[\n]
1245618,320,2009-12-09 00:00:53,38.7401,-9.22841,Lisbon[\n]
1245617,720,2009-12-09 00:00:55,38.7206,-9.12267,Lisbon[\n]
1245618,320,2009-12-09 00:00:59,38.7402,-9.22849,Lisbon[\n]
```

Exemplo de *output (ordenamento por identificador da viatura e timestamp)*

```
1245618,320,2009-12-09 00:00:53,38.7401,-9.22841,Lisbon[\n]
1245618,320,2009-12-09 00:00:59,38.7402,-9.22849,Lisbon[\n]
1245617,720,2009-12-09 00:00:55,38.7206,-9.12267,Lisbon[\n]
1245619,879,2009-12-09 00:00:47,38.832,-9.09474,Lisbon[\n]
```

Exemplo de *output (ordenamento por distância ao ponto de referência)*

```
1245617,720,2009-12-09 00:00:55,38.7206,-9.12267,Lisbon[\n]
1245619,879,2009-12-09 00:00:47,38.832,-9.09474,Lisbon[\n]
1245618,320,2009-12-09 00:00:53,38.7401,-9.22841,Lisbon[\n]
1245618,320,2009-12-09 00:00:59,38.7402,-9.22849,Lisbon[\n]
```

Bom trabalho,

Carlos Lisboa Bento