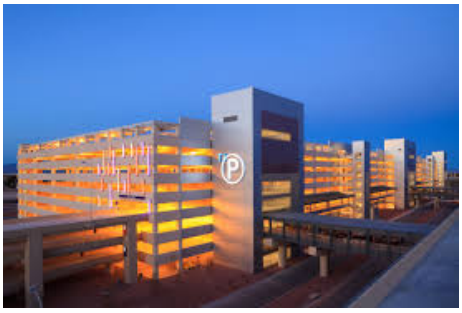
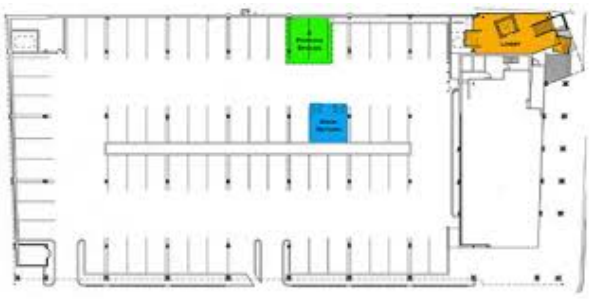
Instituto Superior Técnico

MEEC - Mestrado em Engenharia Eletrotécnica e de Computadores

Algoritmos e Estruturas de Dados

Guia do Projecto



PARQUE AUTOMÁTICO

Versão 1.2 (27/11/2015)

2015/2016

1º Semestre

Conteúdo

Revisões

Versão 1.0 (29 de Outubro de 2015)	Versão inicial
Versão 1.1 (30 de Outubro de 2015)	Adição de restrições
Versão 1.2 (26 de Novembro de 2015)	Clarificada interpretação pretendida para as restrições. Definido procedimento para garagem cheia.

1 Introdução

Pretende-se neste trabalho desenvolver um programa capaz de resolver o problema do PARQUE AUTOMÁTICO, ou seja, de gerir um espaço de estacionamentos com múltiplas entradas e acessos. O objectivo do programa é de, para cada viatura que se apresente a uma das entradas do parque, o programa escolher automaticamente um lugar de estacionamento para essa viatura e ser capaz de indicar o melhor caminho até esse lugar.

O PARQUE AUTOMÁTICO, trabalha sobre uma descrição do parque, possivelmente com múltiplos pisos, em que em cada piso há diversos espaços para estacionamento e outros para circulação. O parque pode estar directamente ligado a múltiplos edifícios, de serviços, lazer, etc, que definem o tipo de acessos possíveis, pelo que ao entrar no parque, o condutor da viatura deve indicar qual o tipo ou número de acesso que pretende utilizar, sendo que a viatura deve ser conduzida e estacionada perto de um acesso desse tipo. O programa deve naturalmente ser bastante geral, para poder ler uma descrição do parque e sobre ela tomar decisões quanto ao estacionamento de viaturas.

A eficiência na gestão do parque mede-se em termos da capacidade existente para estacionar o maior número de viaturas possível, minimizando o percurso da viatura desde a entrada até ao lugar para ela escolhido, bem como o percurso do condutor, desde que estaciona a viatura até ao acesso pretendido.

Na Figura 1 mostra-se uma configuração possível de um parque, sendo indicados (neste caso por ícones com pequenas setas) dois pontos de entrada, dois pontos de acesso (com ícones indicando uma porta), vários lugares de estacionamento válidos (fundo esverdeado) e faixas de circulação. Naturalmente que o programa a desenvolver poderá deparar-se com múltiplas configurações de parques, em que o estacionamento das viaturas será mais simples ou mais complexo. A escolha de um lugar para estacionar uma viatura influencia não apenas o trajecto para o estacionamento, mas também o trajecto do condutor desde o estacionamento até ao acesso, bem como o estacionamento futuro de outras viaturas no parque.

O princípio de funcionamento do programa deve ser o de que para cada viatura que se apresente

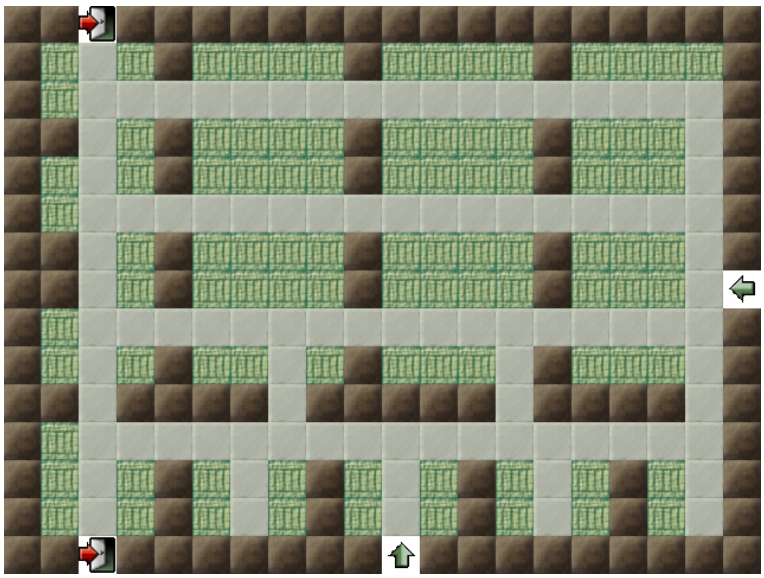


Figura 1: Exemplo de configuração inicial de um parque de estacionamento.

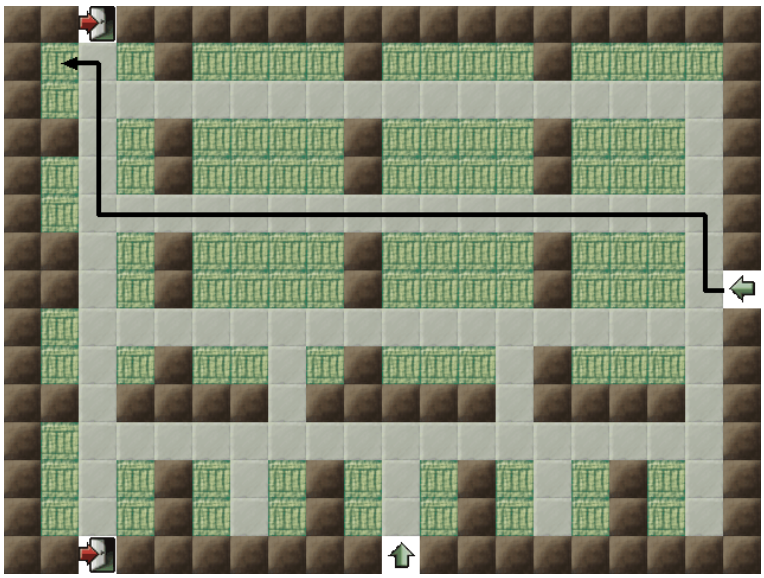


Figura 2: Exemplo de trajecto para estacionamento de uma viatura num parque de estacionamento.

à entrada do parque, é escolhido um lugar para ela e são-lhe dadas indicações, passo a passo, sobre como se dirigir a esse local. Essas indicações, que poderiam surgir em placares indicadores, dizem em cada momento se a viatura pode avançar, virar à esquerda ou à direita ou arrumar num lugar de estacionamento contíguo à sua posição, de novo à esquerda ou à direita. Por exemplo, se uma viatura entrar na entrada lateral à direita e indicar que pretende ir para o acesso no topo à esquerda, a Figura 2 indica um trajecto possível para essa viatura.

O problema do PARQUE AUTOMÁTICO é um problema algo complexo se encarado do ponto de vista computacional. De facto a existência de múltiplas entradas no parque faz com que, dependendo da política de atribuição de lugares ou da temporização das chegadas de veículos a essas entradas, o resultado de uma dada execução possa ser dependente da sequência de escolhas efectuadas para estacionamento das viaturas. Para configurações simples, com poucas viaturas, a solução óptima pode ser encontrada facilmente, mas para configurações mais complexas ou determinados padrões de entrada de viaturas, essa solução pode ser díficil de encontrar, tendo o programa nesse caso de tomar decisões quanto ao lugar de cada veículo. No entanto, do ponto de vista teórico não existe qualquer diferença entre as configurações mais simples e as mais complexas, resolvendo-se ambas de forma semelhante, apenas exigindo as últimas mais recursos computacionais. Em princípio um algoritmo capaz de resolver as configurações mais básicas deverá, se lhe forem dados recursos computacionais suficientes (medidos em tempo de computação e memória disponíveis), ser capaz de resolver qualquer outra configuração. Uma descrição mais detalhada do contexto de utilização do programa a desenvolver, bem como de alguns factores essenciais a ter em consideração no seu desenvolvimento será introduzida na Secção 2.

Na Secção 3 descreve-se o modo de utilização e invocação do programa gestor do parque de estacionamento e os objectivos pretendidos para o mesmo. Descreve-se igualmente o formato de descrição do parque, bem como o formato desejado para os resultados a gerar pelo gestor. Associado ao enunciado do gestor do PARQUE AUTOMÁTICO, o corpo docente da disciplina disponibilizará um programa que permite a verificação e nalguns casos a visualização dos resultados gerados pelo programa gestor. A interface do visualizador é descrita na Secção 4.

Na Secção 5 discutem-se os aspectos básicos relacionados com a avaliação do projecto, descrevendo-se os moldes em que a mesma se irá desenrolar. Na Secção 6 é incluído o **Código de Honestidade Académica** em vigor nesta disciplina e cujos princípios se espera que todos os estudantes sigam de forma escrupulosa. Qualquer estudante que, pela sua conduta directa ou indirecta não respeite o código estipulado será denunciado ao Conselho Pedagógico da escola e serão tomadas todas as medidas necessárias para a instauração de um processo disciplinar. Pensamos que o desrespeito pelo estipulado no código, que todos os alunos implicitamente assumem conhecer e se obrigam a cumprir, constitui uma desonestidade e uma ofensa grave do ponto de vista ético para com o corpo docente e, sobretudo, para com os colegas.

Lembramos igualmente que a verificação de potenciais violações a este código é feita de forma automática com recursos a sofisticados métodos de comparação de código, que envolvem não apenas a comparação directa do código mas também da estrutura do mesmo. Esta verificação é feita com recurso ao *software* disponibilizado em `http://moss.stanford.edu/`.

2 O problema do PARQUE AUTOMÁTICO

O gestor do PARQUE AUTOMÁTICO trabalha sobre uma sequência de planos de dimensão $N \times M$, representando os diversos pisos de estacionamento, no qual estão indicadas as entradas e acessos do parque, bem como os lugares de estacionamento livres ou ocupados. A função do gestor é determinar a sequência de instruções a indicar a cada viatura que se apresente a uma das entradas do parque e que nela pretenda estacionar, de forma a conduzi-la ao lugar pretendido. Através dessas instruções e seguindo estritamente as mesmas, as viaturas podem circular no parque, desde a entrada até um lugar de estacionamento válido que lhes é atribuído. Todas as instruções dadas devem ser consentâneas com as regras de circulação no parque, pelo que uma viatura não pode circular fora das vias para esse efeito criadas, isto é:

- uma viatura não pode circular através das paredes de um parque;
- uma viatura não pode circular sobre lugares de estacionamento livres e, naturalmente;
- uma viatura não pode circular em posições onde estão outras viaturas estacionadas;
- se o parque estiver cheio, qualquer viatura que se apresente a uma das entradas espera aí até que haja lugar.

Na Figura 3 indicam-se alguns trajectos que violam as regras indicadas e que portanto não seriam aceitáveis. A sequência de instruções geradas pelo gestor para conduzir as viaturas ao seu lugar de estacionamento deve satisfazer todas estas regras e deverá ser indicada da forma descrita adiante, na Secção 3.5.

De forma a poder aferir-se da qualidade do trabalho de gestão efectuado, é definida uma métrica que leva em conta o ponto de entrada no parque, o ponto de acesso pretendido, o lugar de estacionamento atribuído, bem como os trajectos da entrada até ao estacionamento e do estacionamento ao acesso. Como da entrada do parque até ao lugar de estacionamento o utente do parque viaja dentro do veículo, e desde o lugar de estacionamento até ao acesso, viaja a pé, o trajecto total é definido como sendo a soma do trajecto da entrada ao estacionamento e o triplo do trajecto entre o estacionamento e o acesso (ou seja o trajecto entre o lugar de estacionamento e o acesso pretendido, é pesado por um factor de 3 face ao trajecto da viatura desde a entrada até ao lugar de estacionamento). Desta forma, o gestor, ao receber a informação de uma viatura que entra e cujo condutor pretende aceder a um dado acesso, deve localizar o acesso respectivo e procurar um lugar de estacionamento que esteja livre e

cuja localização minimiza a soma da distância relativamente ao percurso da entrada até esse lugar, com o triplo da distância entre esse lugar e o acesso pretendido. Estas distâncias, como veremos adiante na descrição do parque, são medidas em termos do número de posições que compõem o trajecto escolhido.

Se uma viatura aceder ao parque estando este completo, essa viatura deve aguardar na entrada o tempo suficiente até que haja um lugar que fique vago, após o que o lugar lhe deve ser atribuído e ela deve prosseguir da forma usual.

Dado tratar-se de uma resolução computacional do problema, pretende-se que o resultado final da gestão do parque seja, para a mesma sequência de veículos, feita com o menor dispêndio de tempo possível. Desta forma, serão valorizados os gestores que encontrem mais rapidamente ou exigindo menores recursos uma mesma solução, ou que consigam obter uma boa solução dentro de limites aceitáveis para os recursos computacionais existentes. Os detalhes da métrica a utilizar para quantificar o trabalho do gestor do parque de estacionamento serão indicados adiante.

Considere a situação indicada na Figura 4. Assuma que neste caso duas viaturas que se apresentam em entradas diferentes, pretendem ambas estacionar perto do mesmo acesso, no caso o acesso em baixo à esquerda. A figura indica uma solução possível para estacionamento das duas viaturas. Note que a solução indicada graficamente é claramente não óptima e é apresentada apenas como exemplo.

3 O Gestor do PARQUE AUTOMÁTICO

O programa a desenvolver deve conseguir ler uma configuração inicial para a configuração do parque, incluindo as suas dimensões, a localização dos lugares de estacionamento e das vias de circulação, o número de pisos, número e localização de entradas e acessos. Deve ainda ser capaz de encontrar lugares de estacionamento para cada viatura que se apresente numa das entradas, de forma a minimizar o percurso de estacionamento e de gerar uma sequência de instruções capazes de conduzir a viatura até esse local.

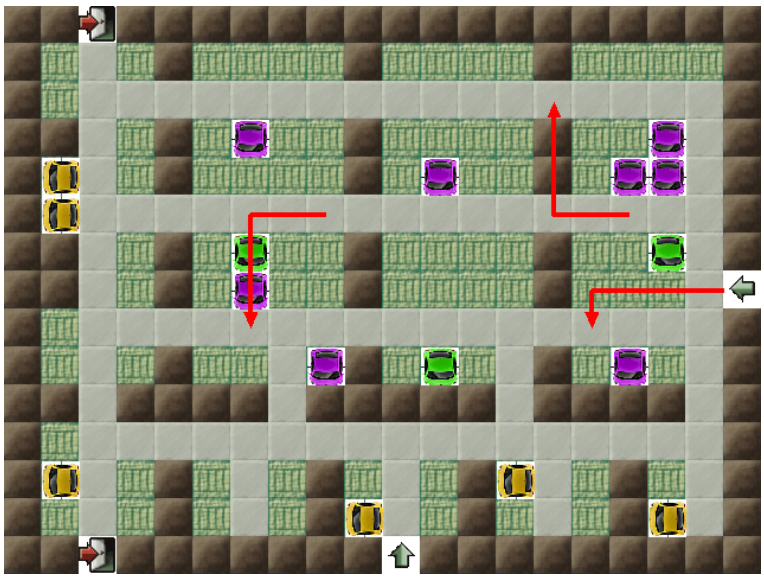


Figura 3: Exemplo de trajectos que violam as regras de circulação num parque de estacionamento. Todos os trajectos indicados a vermelho violam as regras anteriormente enunciadas.

Essa informação, que tem de ser calculada, inclui indicação sobre as vias de circulação a utilizar para chegar ao local pretendido e estacionar, bem como o trajecto desde o estacionamento até ao acesso pretendido. Nas Secções seguintes descreve-se a forma como o programa gestor deve ser invocado, a forma como a configuração do parque pode ser descrita, o formato dos dados de entrada para estacionamento e a sintaxe a utilizar nos dados de saída, incluindo-se aqui as instruções para estacionar uma dada viatura.

3.1 Execução do programa Gestor

O programa gestor da PARQUE AUTOMÁTICO deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./gestor parque.cfg parque.inp [parque.res]
```

onde:

`gestor` designa o nome do ficheiro executável contendo o programa gestor da PARQUE AUTOMÁTICO. Sugere-se que o nome a utilizar seja exactamente o indicado.

`parque.cfg` é o nome do ficheiro contendo a descrição da configuração do parque, de acordo com o formato indicado na Secção 3.2.

`parque.inp` é o nome do ficheiro contendo a informação sobre as viaturas que se apresentam à entrada do parque para ser estacionadas e cujo formato é descrito na Secção 3.3.

`parque.res` é o nome do ficheiro contendo a informação sobre as possíveis restrições à utilização do parque e cujo formato é descrito na Secção 3.4; este argumento pode ou não ser utilizado: se não for assume-se que não há restrições.

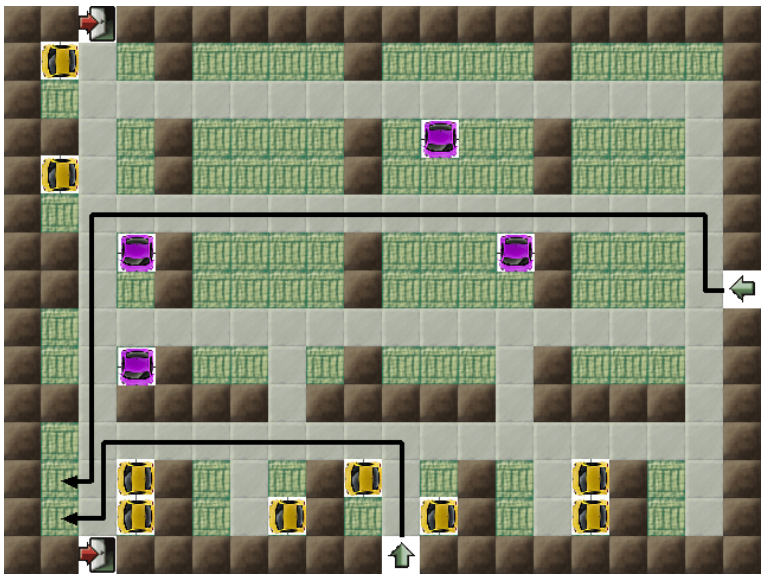


Figura 4: Exemplo de trajectos para estacionamento de duas viaturas, entrando por entradas diferentes num parque de estacionamento.

O 1^{o} argumento na chamada do programa, que designa o nome do ficheiro contendo a descrição da configuração do parque, pode ter qualquer nome ou extensão, nada sendo assumido para esse efeito. Se se pedir a execução do gestor indicando-se o nome de um ficheiro que não exista, o programa deve simplesmente abortar a execução com uma mensagem de erro. Assume-se que os ficheiros contendo a descrição da configuração dos parques estão sempre correctos, pelo que o programa gestor a desenvolver não tem de se preocupar com a detecção de erros na descrição da configuração. Da mesma forma, os 2^{o} e 3^{o} argumentos na chamada da função, que designam respectivamente o nome do ficheiro contendo a informação sobre as entradas de viaturas, e o nome do ficheiro contendo a informação sobre possíveis restrições, pode igualmente ter qualquer nome ou extensão.

As restrições correspondem a limitações temporárias de utilização de lugares, seja por obras ou por qualquer outro tipo de actividade que impeça a normal utilização desses locais.

De seguida descrevem-se com detalhe o formato dos ficheiros indicados.

3.2 Descrição da Configuração do Parque

Nesta secção apresentam-se as regras de descrição das configurações dos parques de estacionamento a utilizar pelo programa a desenvolver nesta disciplina. A descrição de cada configuração deve ser feita num ficheiro, obedecendo às seguintes regras:

- cada ficheiro contém a descrição completa de um único parque de estacionamento.
- a 1^{a} linha do ficheiro deve conter cinco (5) números inteiros, N, M, P, E e S, separados por um espaço. Os três primeiros números descrevem a geometria do parque, indicando respectivamente o número de colunas (logo referentes às coordenadas em **x** ou horizontais), o número de linhas (coordenadas em **y** ou verticais) da configuração do parque e o número de pisos do parque (coordenada **z**). Os restantes dois números indicam o número total de entradas para viaturas em todo o parque e o número total de acessos para peões em todo o parque. Assume-se implicitamente que todos os pisos têm a mesma dimensão, $N \times M$.

- A informação restante consiste na descrição dos vários pisos do parque, de 0 a $P-1$, das suas respectivas entradas e acessos. A descrição de cada piso P_k do parque é feita da forma seguinte:
  - as M $1^{\underline{as}}$ linhas contêm a descrição da configuração do piso P_k, que é feita numa tabela de caracteres de $M \times N$, seguindo a seguinte convenção:

    - o caractere **'@'** representa uma parede intransponível. Não é possível circular sobre ou transpôr uma parede.
    - o caractere **'e'** representa uma entrada para viatura.
    - o caractere **'a'** representa um acesso.
    - o caractere **'.'** representa um lugar vazio. Não é possível circular sobre ou transpôr um lugar vazio, embora seja possível nele estacionar.
    - o caractere **'x'** representa um lugar ocupado e portanto proibido.
    - o caractere **' '** (espaço) indica uma via de circulação.
    - o caractere **'u'** representa uma rampa de circulação que permite a uma viatura aceder ao piso imediatamente acima do piso actual.
    - o caractere **'d'** representa uma rampa de circulação que permite a uma viatura aceder ao piso imediatamente abaixo do piso actual.

Assume-se que todas as vias de circulação, seja no mesmo piso, ou em ligação a pisos acima ou abaixo do actual, são bidireccionais.

Define-se que a posição mais à esquerda, em baixo, no piso 0, é a posição $(0, 0, 0)$ em que a 1ª coordenada se refere à posição em **x** (horizontal), a 2ª coordenada se refere à posição em **y** (vertical), e a 3ª ao piso correspondente. As coordenadas válidas num dado parque vão assim desde o ponto $(0, 0, 0)$ ao ponto $(N - 1, M - 1, P - 1)$.

– após a descrição da configuração de um dado piso segue-se a descrição das entradas desse piso. As entradas do parque no piso P_k são descritas por linhas do tipo:

Exxx xe ye ze -

em que "Exxx" é um identificador para a entrada, "xe", "ye" e "ze" são números inteiros que indicam as coordenadas da entrada e o caractere **'-'** explicita ser esta uma entrada. Naturalmente que, no piso P_k a coordenada "ze" deve ter sempre o valor P_k (ou seja no piso 0 os acessos devem ter sempre coordenada "ze" a zero). Assume-se igualmente que as entradas se situam sempre ao longo da parede exterior do parque. Note-se que pode haver entradas em vários pisos do parque, mas é também possível que alguns pisos não tenham qualquer entrada. O número total de entradas em todos os pisos é o indicado na primeira linha do ficheiro.

– finalmente segue-se a descrição da localização dos acessos do parque no piso P_k. Cada linha é do tipo:

Axxx xs ys zs A

em que "Axxx" é um identificador para o acesso, "xs", "ys" e "zs" são números inteiros representando as coordenadas do acesso e "A", um caractere, indicando o tipo de acesso é pretendido. Naturalmente que, no piso P_k a coordenada "zs" deve ter sempre o valor P_k (ou seja no piso 0 os acessos devem ter sempre coordenada "zs" a zero). O tipo de acesso, definido por um único caractere, pode representar um acesso para um espaço comercial, um cinema, escritório, zona habitacional, etc. Neste projecto são válidos os seguintes tipos de acessos: "C" (cinema), "H" (habitação), "E" (escritórios), "L" (comércio) ou "R" (restauração).

– A descrição de qualquer piso é terminada por uma linha que contém o caractere **'+'** na primeira posição da linha.

A descrição dos vários pisos deve ser consistente, no sentido em que se numa dada posição de um dado piso existe por exemplo um caractere **'d'** que indica uma rampa descendente, então tem necessariamente de existir um piso inferior e nesse piso, na mesma posição, tem de estar um caractere **u**, ou seja uma rampa ascendente.

A título de exemplo, a Figura 5 mostra o conteúdo de um ficheiro contendo a configuração do parque que serviu de exemplo anteriormente e a que corresponde a descrição da Figura 1. Na página da disciplina, na Secção referente ao **Projecto** serão posteriormente colocados alguns ficheiros de exemplo.

A métrica a utilizar para distância, já mencionada acima, é assim o número de posições distintas, (x, y, z) que compõem o percurso da viatura desde a entrada ao lugar de estacionamento, somado com o número de posições distintas, pesado com um factor de 3, no percurso pedestre entre o lugar de

```
20 15 1 2 2
@@a@@@@@@@@@@@@@@@@@@
@. .@....@....@....@
@.                  @
@@ .@....@....@... @
@. .@....@....@... @
@.                  @
@@ .@....@....@... @
@@ .@....@....@... e
@.                  @
@. .@.. .@... @... @
@@ @@@@ @@@@@ @@@@ @
@.                  @
@. .@. .@. .@. .@. @
@. .@. .@. .@. .@. @
@@a@@@@@@@e@@@@@@@@@
E1 19 7 0 -
E2 10 0 0 -
A1  2 0 0 R
A2  2 14 0 C
+
```

Figura 5: Exemplo do ficheiro de configuração inicial do parque de estacionamento correspondente à descrição da Figura 1.

estacionamento e o acesso. Assume-se que a velocidade de uma viatura dentro do parque é constante e que demora uma unidade de tempo a percorrer cada posição do parque, com excepção de posições marcadas com **'u'** ou **'d'** (rampas de acesso vertical) que permitem a mudança de piso e que demoram duas unidades de tempo a percorrer.

3.3 Descrição do formato de entrada de dados

Nesta secção apresentam-se as regras de descrição do formato de entrada de viaturas no parque de estacionamento a utilizar pelo programa a desenvolver nesta disciplina. Para cada parque pode ser definido um ficheiro referente à movimentação de viaturas no parque cujas regras são bastante simples. O ficheiro é constituído por múltiplas linhas em que em cada linha o formato é de um dos três tipos seguintes:

- Entrada de viatura: Vxxx ta T ex ey ez
  em que "Vxxx" é o identificador da viatura, "ta" é um inteiro, representando o instante de tempo em que a viatura se apresenta à entrada do parque, "T", um caractere, que indica o tipo de acesso que é pretendido e a tripla de inteiros, "ex", "ey" e "ez", são as coordenadas "(ex, ey, ez)" da entrada à qual a viatura se apresenta. ~~Naturalmente a~~ A entrada indicada corresponde a uma das entradas do parque, o que não tem de ser verificado., ~~tem de ser correcta, isto é tem de corresponder a uma entrada real do parque e tal deve ser verificado~~. Quanto ao tipo de acesso pedido, o gestor do parque é livre de encaminhar a viatura para qualquer um dos acessos do tipo indicado.

```
Va1    0 C 10 0 0
VAA    2 R 19 7 0
Vbx    3 C 10 0 0
Va1   40 S
```

Figura 6: Exemplo de um ficheiro de entrada para a configuração do parque correspondente à descrição da Figura 1.

- Saída de viatura: Vyyy tb S
  com o mesmo significado que anteriormente mas que indica que no instante "tb" a viatura identificada por "Vyyy" libertou o lugar de estacionamento que ocupava e abandonou o parque (neste caso não é relevante o percurso que foi utilizado para sair do parque, apenas que o estacionamento que ocupava fica disponível a partir do instante "tb").
- Libertação de lugar: Vyyy tb S ex ey ez
  semelhante ao caso anterior mas que indica que a viatura previamente estacionada no local com as coordenadas indicadas, "(ex, ey, ez)", abandonou o parque no instante "tb" (neste caso o identificador da viatura não é relevante tal como não o é o percurso usado pela viatura para sair do parque; este caso serve para libertar lugares de um parque que inicialmente já estava parcialmente cheio, mas cujas viaturas não foram estacionadas pelo gestor).

A título de exemplo, um conjunto de entradas de viaturas possíveis para o parque da Figura 5 seria o indicado na Figura 6. As viaturas aparecem neste ficheiro por ordem crescente de tempo, seja tempo de chegada, seja tempo de saída ou libertação de lugar e devem ser processadas por essa ordem, uma de cada vez. Se uma viatura chegar ao parque quando este está cheio, o processamento possível nesse instante corresponde a colocar a viatura numa fila de espera e indicar no ficheiro de saída a sua chegada (ver Secção 3.5). O programa deve então passar o processamento para a viatura seguinte que aparecer no ficheiro de entrada. Uma viatura parada na fila de espera entra no parque assim que vagar um lugar para onde pode ser enviado.

3.4 Descrição do formato de descrição de restrições

Nesta secção apresentam-se as regras para o formato de descrição das restrições à utilização do parque de estacionamento que devem ser respeitadas pelo programa a desenvolver nesta disciplina. Para cada parque pode ser definido um ficheiro contendo as restrições à utilização de certas posições do parque, restrições essas que são bastante simples. O ficheiro é constituído por múltiplas linhas em que em cada linha o formato é de um dos dois tipos seguintes:

- Piso interdito: R ta tb px
  em que "px" é o piso que fica interdito para estacionamento entre os instantes "ta" e "tb". Se "tb" for "0" isso significa que o piso "px" fica inutilizável por tempo indeterminado a partir de "ta". Note que estando o piso interdito para estacionamento,é no entanto possível circular por ele.
- Posição interdita: R ta tb ex ey ez
  semelhante ao caso anterior mas que indica que a posição com as coordenadas indicadas, "(ex, ey, ez)" passa a estar indisponível entre os instantes "ta" e "tb" (ou a partir de "ta" se "tb" for

```
R   10 20 1 1 0
R   10  0 1 2 0
R   12  0 1 3 0
```

Figura 7: Exemplo de um ficheiro de entrada para a descrição das restrições à utilização do parque correspondente à descrição da Figura 1.

"0"). Note que a posição tanto pode ser um lugar de estacionamento, uma via de circulação ou uma rampa (que também é uma via de circulação).

A título de exemplo, um conjunto de restrições possíveis para o parque da Figura 5 seria o indicado na Figura 7.

O gestor do PARQUE AUTOMÁTICO deve ter em conta as restrições que lhe são indicadas, não devendo fornecer instruções que levem ao estacionamento de uma viatura seja num local que não está disponível por estar restrito seja porque necessita de passar numa posição que esteja restrita.

Uma dada restrição afecta a atribuição de um lugar de estacionamento a uma dada viatura, se no momento em que inicia a sua marcha a restrição estiver activa. Por exemplo no caso da Figura 7, a primeira restrição deve ser tida em conta para viaturas que iniciem a sua marcha dentro da garagem entre os instantes $t = 10$ e $t = 20$, altura a partir da qual deixa de estar em efeito. Já a segunda restrições diz-se activa a partir do instante $t = 10$ inclusivé. A terceira restrição torna-se activa para veículos que iniciem a sua marcha a partir do instante $t = 12$.

3.5 Funções e formato de saída de dados

O resultado da execução do gestor do GARAGEM DE ESTACIONAMENTO resume-se à determinação da menor sequência de posições que definem o percurso entre a entrada e o lugar de estacionamento e daí para o acesso pretendido. O resultado do processamento por parte do gestor deve ser o de arrumar todas as viaturas que apareçam em qualquer das entradas, desde que não excedam a capacidade do parque. Para cada viatura deve determinar-se o local ideal de estacionamento e gerar o percurso ideal que deve ser colocado no ficheiro de saída. Desta forma, o resultado do processamento deve ser:

- a sequência de posições que compõem o trajecto calculado, colocadas num ficheiro de saída, cujo nome deve ser o mesmo do ficheiro de descrição das garagens mas **com extensão** ``.pts''. Este ficheiro deve ser criado e aberto pelo programa gestor.
- cada linha do ficheiro corresponde a uma posicão no parque em que se dá uma mudança de direcção ou de piso, incluindo o ponto de entrada da viatura no parque. Cada linha deve conter o identificador da viatura, "Vxxx" seguido de **quatro** números inteiros, separados por espaços e de um caractere. O primeiro número corresponde ao instante de tempo em que a viatura chega à posição indicada pelas coordenadas dos restantes três números. O caractere que termina a linha pode ser o caractere **'i'**, indicando que a viatura entrou nesse momento no parque, **'m'**, indicando que a viatura está em movimento, o caractere **'e'** que significa que a viatura estacionou, o caractere **'p'** que significa percurso pedestre, após o estacionamento, ou o caractere **'a'** que significa que o lugar ou acesso pretendido foi alcançado. Se o percurso envolver uma mudança de piso (subida ou descida), devem constar da saída os dois pontos correspondentes ao início e fim da subida ou descida de piso.

```
VAA    2 19  7  0 i
VAA    3 18  7  0 m
VAA    4 18  6  0 m
VAA   20  2  6  0 m
VAA   24  2  2  0 m
VAA   25  1  2  0 e
VAA   26  2  2  0 p
VAA   28  2  0  0 a
VAA    2 25 28 32 x
```

Figura 8: Solução do problema do estacionamento de uma viatura que em $t = 2$ se apresenta na entrada $(19, 7, 0)$, cujo destino é um acesso de tipo "R" e que o local escolhido para arrumar o carro é o estacionamento na posição $(1, 2, 0)$.

- após o percurso pedestre chegar ao acesso escolhido, deve ser gerada uma linha adicional de resumo, com o mesmo identificador, e em que os quatro números representam respectivamente o instante inicial de entrada no parque, o instante em que a viatura foi estacionada, o instante em que o condutor chegou ao acesso e o "custo" total acumulado da operação, pesado da forma indicada na Secção 2, seguido do caractere terminador **'x'**.
- Se na entrada estiver a indicação de que uma viatura abandonou o seu lugar de estacionamento, deve ser gerada uma única linha contendo o identificador da viatura, o instante em que abandonou o parque, o lugar que vagou, e o caractere **'s'** (note que em qualquer dos casos, o lugar vago é conhecido, pois ou é indicado no ficheiro de entrada ou, através do identificador, o gestor sabe onde a viatura indicada estava estacionada).

Como exemplo, uma sequência de posições referentes ao parque apresentado nas Figuras 1 e 4 e descrito na Figura 5 é apresentada na Figura 8. Este exemplo assume, como descrito na Figura 6, que no instante $t = 2$ se apresenta uma viatura, identificada por "VAA" na entrada de coordenadas $(19, 7, 0)$ cujo destino é um acesso de tipo "R" e que o local escolhido para arrumar a viatura é o estacionamento na posição $(1, 2, 0)$. É ainda apresentada a linha final, com a pontuação total, apenas para ilustrar o formato pretendido.

Os dados referentes às várias posições que vão sendo percorridas durante o percurso deverão ser enviados para o ficheiro de saída. Para uniformizar o formato de saída, a geração desta informação deverá ser **obrigatoriamente** feita através de uma chamada a uma função em "C" fornecida pelo corpo docente. Esta função tem a seguinte assinatura:

```
int escreve_saida(FILE *fp, char *vid, int tk,
                  int pX, int pY, int pZ, char tm);
```

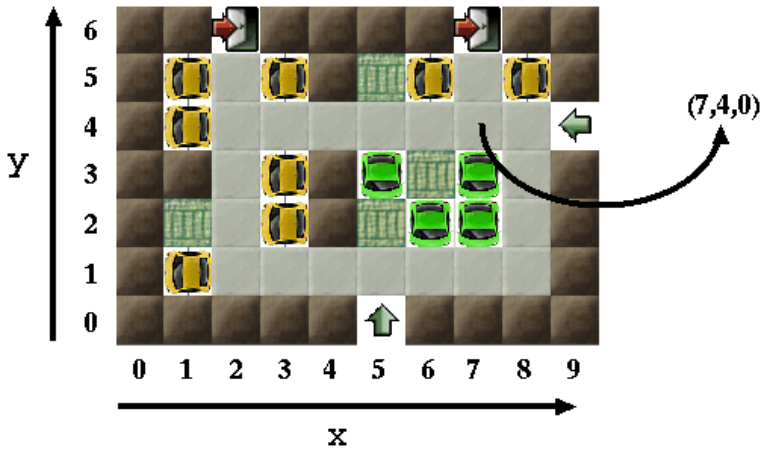
Os argumentos desta função são respectivamente o ponteiro para o descriptor do ficheiro de saída, o id da viatura, o instante de tempo em que ocorre uma mudança de direcção no percurso da mesma, as coordenadas do ponto em que essa mudança ocorre, e o tipo de movimento segundo as possibilidades anteriormente descritas. Em cada chamada, a função retorna o número `0` caso não haja qualquer erro, ou o valor `-1` caso haja algum erro a reportar. Nesse caso é igualmente enviada uma mensagem de erro para `stderr`. A função deve ser chamada logo após cada jogada ter sido escolhida, sendo que as jogadas serão apresentadas no ficheiro de saída, de forma sequencial correspondendo à sequência

escolhida. Quando o programa gestor terminar de processar todas as viaturas, deverá fechar o ficheiro de saída.

O ficheiro `escreve_saida.c`, contendo a função descrita, será disponibilizado em breve na página da disciplina, na secção de **Projecto**.

3.6 Exemplo de execução

Nesta secção apresentamos um exemplo completo de execução do programa gestor do PARQUE AUTOMÁTICO. Considere o pequeno parque ilustrado na Figura 9, cuja descrição no formato descrito na Secção 3.2 é indicada à direita na mesma Figura.




```
10 7 1 2 2
@@a@@@@a@@
@x x@.x x@
@x       e
@@ x@x.x @
@. x@.xx @
@x       @
@@@@@e@@@@
Ea 5 0 0 -
Eb 9 4 0 -
Ac1 7 6 0 C
Ar1 2 6 0 R
+
```

Figura 9: Exemplo ilustrativo um pequeno parque, à esquerda e, à direita, a sua descrição de acordo com o formato descrito na Secção 3.2. Na figura, para clarificação, são indicados os eixos e as coordenadas de um dado ponto como exemplo.

Assuma que o ficheiro de entrada contém a seguinte informação:

```
Va1   0 C 5 0 0
VAA   2 R 9 4 0
Vbb 25 R 5 0 0
VAA 30 S
```

e que o ficheiro de restrições contém a indicação:

```
R 0 0 5 2 0
R 1 0 5 5 0
R 0 6 8 3 0
```

indicando que de facto há dois lugares que estão sempre indisponíveis.

Claramente neste caso o caminho mais curto após a entrada seria ir pela direita, arrumar na posição "(6,3,0)" e prosseguir para o acesso em "(7,6,0)". No entanto esse caminho viola a 3^a restrição pois a posição "(8,3,0)", onde a viatura chegaria no instante "t=6" está inacessível desde o instante "t=0" em que a viatura inicia a sua marcha. Como tal é necessário ir pelo lado oposto e neste caso, o gestor deveria indicar a seguinte sequência de posições:

```
Va1    0   5   0   0 i
Va1    1   5   1   0 m
Va1    4   2   1   0 m
Va1    7   2   4   0 m
Va1   11   6   4   0 m
Va1   12   6   3   0 e
Va1   13   6   4   0 p
Va1   14   7   4   0 p
Va1   16   7   6   0 a
Va1    0  12  16  24 x
VAA    2   9   4   0 i
VAA    9   2   4   0 m
VAA   11   2   2   0 m
VAA   12   1   2   0 e
VAA   13   2   2   0 p
VAA   17   2   6   0 a
VAA    2  12  17  25 x
Vbb   25   5   0   0 i
VAA   30   2   2   0 s
Vbb   31   5   1   0 m
Vbb   34   2   1   0 m
Vbb   35   2   2   0 m
Vbb   36   1   2   0 e
Vbb   37   1   2   0 p
Vbb   41   2   6   0 a
Vbb   25  36  41  26 x
```

Note-se na sequência anterior o tratamento feito à viatura Vbb: quando chega ao parque, no instante $t = 25$, o parque está cheio e ela espera, sendo no entanto a sua chegada indicada na linhas com um 'i' no final. Essa viatura deve então ser colocada numa fila de espera, e o programa passa a processar a viatura seguinte. Quando vagar um lugar no parque, a viatura que estava em espera entra no parque no instante seguinte e é tratada de imediato.

4 Visualizador do gestor do PARQUE AUTOMÁTICO

O corpo docente fornecerá brevemente um programa visualizador que permite simultaneamente visualizar e verificar a saída gerada pelo gestor do PARQUE AUTOMÁTICO. O verificador deverá ser invocado com três argumentos, respectivamente o ficheiro de configuração do parque, o ficheiro de entrada e o ficheiro de saída do programa gestor. Por questões de ordem sobretudo gráfica, o visualizador será limitado para garagens cuja dimensão permita uma visualização eficaz, embora faça verificação de garagens de qualquer dimensão. Do ponto de vista da disciplina, o modo de invocação é assim o seguinte:

```
aed$ pgvis parque.cfg viaturas.inp [parque.res] parque.pts
```

onde:

`pgvis` designa o nome do ficheiro executável contendo o programa visualizador;

`parque.cfg` é o nome do ficheiro contendo a descrição da configuração do parque, de acordo com o formato indicado na Secção 3.2.

`viaturas.inp` é o nome do ficheiro contendo a informação sobre as viaturas que se apresentam à entrada do parque para ser estacionadas e cujo formato é descrito na Secção 3.3.

`parque.res` é o nome do ficheiro contendo a informação sobre as restrições à utilização do parque e cujo formato é descrito na Secção 3.4.

`parque.pts` é o nome do ficheiro contendo a saída do programa gestor e cujo formato é descrito na Secção 3.5.

Se não houver restrições à utilização do parque, a chamada anterior terá apenas 3 argumentos em que o último é o ficheiro contendo a saída do programa.

O visualizador estará instalado nas máquinas do laboratório, sendo que o código fonte será igualmente disponibilizado na página da disciplina, na secção de **Projecto**.

5 Avaliação do Projecto

O projecto de AED está dimensionado para ser feito por **grupos de dois alunos** não se aceitando grupos de outra dimensão. Quando os grupos estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fenix, no grupo de Projecto correspondente. A inscrição do grupo deve ser feita incluindo os ndados de todos os elementos do grupo simultaneamente.

A avaliação do projecto decorre em três ou quatro instantes distintos. O primeiro instante coincide com a submissão electrónica do código do projecto. O segundo instante corresponde à entrega do relatório final em mão aos docentes, entrega essa que ratifica a submissão anteriormente mencionada. É possível aos alunos submeter o projecto e entregar o relatório nos dois dias seguintes a essa data, mas nesses casos aplica-se uma penalização (veja Tabela 1).

Num terceiro instante há uma proposta enviada pelo corpo docente que pode conter uma indicação de uma data para a discussão e defesa do trabalho ou uma proposta de nota para a componente de projecto. Caso os alunos aceitem a nota proposta pelos docentes, a discussão não é necessária e a nota torna-se final. Se, pelo contrário os alunos decidirem recorrer da nota proposta, será marcada uma discussão de recurso em data posterior. Caso haja marcação de uma discussão, ela será obrigatoriamente feita a todos os elementos do grupo, sem prejuízo de as classificações dos componentes do grupo poderem ser distintas. As datas de entrega referentes aos vários passos da avaliação do projecto estão indicadas na Tabela 1.

A submissão electrónica estará disponível em data e condições a indicar posteriormente na página da cadeira e serão aceites trabalhos entregues até ao instante final indicado. Ao contrário de anos anteriores este ano **não haverá qualquer extensão no prazo de entrega**, pelo que os alunos devem organizar o seu tempo de forma a estarem em condições de entregar a versão final na data indicada. O relatório final deverá ser entregue em mão ao docente no laboratório no dia indicado na Tabela 1.

Note-se que o projecto só é considerado entregue aquando da entrega do relatório em papel. As submissões electrónicas do código não são suficientes para concretizar a entrega. Um grupo que faça a submissão electrónica do código e a entrega do relatório em papel, por exemplo, na 1ª data de entrega

Tabela 1: Datas de Avaliação do Projecto

Data	Documentos a Entregar
até 4 de Novembro de 2015, 4ª feira	Enunciado do projecto disponibilizado na página da disciplina
até 9 de Novembro de 2015, 4ª feira	Inscrição dos grupos no sistema Fenix
9 de Dezembro de 2015, 4ª feira 12h 15h-16h	**1ª Data de entrega do projecto: submissão electrónica final** **[Não haverá adiamentos a este prazo]** Submissão electrónica do projecto Entrega do relatório do projecto em papel, SCDEEC
10 de Dezembro de 2015, 5ª feira 12h 15h-16h	**2ª Data de entrega do projecto: submissão electrónica penalização de um (1) valor** Submissão electrónica do projecto Entrega do relatório do projecto em papel, SCDEEC
11 de Dezembro de 2015, 6ª feira 12h 15h-16h	**3ª Data de entrega do projecto: submissão electrónica penalização de dois (2) valores** Submissão electrónica do projecto. Entrega do relatório do projecto em papel, SCDEEC
	Submissões posteriores a 11 de Dezembro têm penalização de 20 valores.
14 a 18 de Dezembro de 2015	Discussão do trabalho (data combinada com cada grupo).

pode fazer submissões nas datas seguintes, mas se fizer a entrega de um novo relatório em papel, será este, e as respectivas submissões, o considerado para avaliação, com a penalização indicada.

5.1 Funcionamento

A verificação do funcionamento do código a desenvolver no âmbito do projecto será exclusivamente efectuada nas máquinas do laboratório da disciplina, embora o desenvolvimento possa ser efectuado em qualquer plataforma ou sistema que os alunos escolham. Esta regra será estritamente seguida, não se aceitando quaisquer excepções. Por esta razão, é essencial que os alunos, independentemente do local e ambiente em que desenvolvam os seus trabalhos, os verifiquem no laboratório antes de os submeterem, de forma a evitar problemas de última hora. Uma vez que os laboratórios estão abertos e disponíveis para os alunos em largos períodos fora do horário das aulas, este facto não deverá causar qualquer tipo de problemas.

5.2 Código

Não deve ser entregue código em papel. Os alunos devem entregar por via electrónica o código do programa (ficheiros `.h` e `.c`) e uma `Makefile` para gerar o executável. Todos os ficheiros (`*.c`, `*.h`, `Makefile`, etc.) devem estar localizados na directoria raiz.

O código deve ser estruturado de forma lógica em vários ficheiros (`*.c` e `*.h`). As funções devem ter um cabeçalho curto mas explicativo e o código deve estar correctamente indentado e com

comentários que facilitem a sua legibilidade.

5.3 Relatório

Os relatórios devem ser entregues na altura indicada na Tabela 1 e dirigidos ao docente do laboratório respectivo. Os grupos compostos por alunos que frequentam laboratórios distintos devem indicar um dos docentes desses laboratórios. Os alunos que não estão a frequentar o laboratório devem mencionar essa situação.

O relatório do projecto deverá contemplar os aspectos seguidamente indicados. A apresentação do mesmo deverá iniciar-se por aspectos de ordem geral, seguindo-se descrições mais detalhadas dos módulos e estruturas de dados utilizados. Sugere-se a seguinte estrutura para o relatório:

- Uma capa com os dados dos membros do grupo, incluindo nome, número e e-mail. Esta capa deverá seguir o formato indicado na página da disciplina (oportunamente será disponibilizado);
- Uma página com o índice das secções em que o relatório se divide;
- Uma descrição do problema que foi resolvido, com indicação clara das especificações do mesmo tal como foram entendidas;
- Um texto simples que indique como os alunos abordaram e resolveram o problema;
- Uma descrição completa da arquitectura do programa, incluído um fluxograma detalhado e um texto claro, mas sucinto, indicando a divisão lógica e funcional dos módulos desenvolvidos para a resolução do problema, indicando os respectivos objectivos, as funções utilizadas e as estruturas de dados de suporte;
- Uma descrição detalhada das estruturas de dados utilizadas e justificação das mesmas;
- Descrição dos algoritmos usados (por exemplo, na manipulação das estruturas de dados);
- Uma descrição dos subsistemas funcionais que existam e, para cada um
  - a descrição dos objectivos do subsistema (até 5 linhas);
  - o nome do módulo onde estão definidas as estruturas de dados (ficheiros `.h`) a utilizar no subsistema;
  - o nome do módulo `C` onde estão as funções do respectivo subsistema;
  - listagem das funções a implementar no subsistema, indicando para cada uma a respectiva assinatura e os objectivos da função (descrição sumária, sem código);
- Uma análise dos requisitos computacionais do programa desenvolvido, tanto em termos da memória que utiliza como da complexidade computacional, com particular ênfase no custo das operações de processamento sobre as estruturas de dados;
- Uma análise crítica do funcionamento do programa e a avaliação do desempenho do projecto implementado;
- Pelo menos, um pequeno exemplo completo de aplicação, com descrição da utilização das estruturas de dados em cada passo.

Oportunamente será disponibilizado na página da disciplina um relatório modelo.

5.4 Critérios de Avaliação

Os projectos submetidos electronicamente serão avaliados de acordo com a seguinte grelha.

- Testes passados na submissão electrónica – 75%
- Estruturação do código e comentários – 5%
- Gestão de memória e tipos abstractos – 5
- Relatório escrito – 15%

Na submissão electrónica cada projecto será testado com cerca de 20 problemas de diferentes graus de complexidade, onde se avaliará a capacidade em produzir soluções correctas dentro de limites de tempo e memória. Para o limite de tempo, cada um dos testes terá de ser resolvido em menos de 60 segundos. Para o limite de memória, cada um dos testes não poderá exceder 100MB como pico de memória usada. Cada teste resolvido dentro dos orçamentos temporal e de memória que produza soluções correctas recebe um ponto.

Se o corpo docente entender necessário, face à complexidade dos problemas a resolver, poderão os limites de tempo e/ou memória ser aumentados.

Caso o desempenho de alguma submissão não seja suficientemente conclusivo, poderá ser sujeita a testes adicionais fora do contexto da submissão electrónica. Alguns desses testes adicionais poderão ser os mesmos da submissão electrónica, que serão corridos sem limites de tempo nem de memória. Outros poderão ser diversos daqueles. O desempenho nesses testes adicionais poderá contribuir para subir ou baixar a pontuação obtida na submissão electrónica.

No que à avaliação do relatório diz respeito, os elementos de avaliação incluem: apreciação da abordagem geral ao problema e respectiva implementação; análise de complexidade temporal e de memória; exemplo de aplicação; clareza e suficiência do texto, na sua capacidade de descrever e justificar com precisão o que está feito; e qualidade do texto escrito e estruturação do relatório.

6 Código de Honestidade Académica

(Retirado da página da disciplina)

Nesta disciplina, espera-se de cada aluno que subscreva os mais altos padrões de honestidade académica. Isto significa que cada ideia que não seja do aluno deve ser explicitamente creditada ao respectivo autor. O não cumprimento disto constitui plágio.

O plágio inclui a utilização de ideias, código ou conjuntos de soluções de outros alunos ou indivíduos, ou quaisquer outras fontes para além dos textos de apoio à disciplina, sem dar o respectivo crédito a essas fontes. Nesta disciplina, o método de avaliação inclui vários testes, trabalhos, relatórios, etc. Os alunos são encorajados a discutir os problemas com outros alunos e devem mencionar essa discussão quando submetem os resultados. Essa menção NÃO influenciará a nota. Mas os alunos não deverão copiar código de outros alunos, ou dar o seu próprio código a outros em qualquer circunstância. De facto, não devem sequer deitar listagens fora sem primeiro as destruir, nem deixar o código desenvolvido em computadores de uso partilhado.

A desonestidade académica inclui também a cópia em testes e exames. Nesta disciplina, estes devem ser feitos sem consulta, nem de qualquer texto, nem de outros colegas. Receber ou dar ajuda durante estas provas é um acto de desonestidade académica. Devem ser evitadas situações que possam dar azo a suspeitas de desonestidade (abrir as mochilas para buscar papel, olhar para todos os lados em vez de se concentrar na folha do exame, etc.). Nesta disciplina, a desonestidade académica é considerada fraude, com todas as consequências legais que daí advêm. Qualquer fraude terá como consequência imediata a reprovação de todos os alunos envolvidos (incluindo os que possibilitaram a ocorrência). Qualquer suspeita de desonestidade académica será relatada aos órgãos superiores da escola para instauração de um processo disciplinar. Este poderá resultar em reprovação à disciplina, reprovação de ano, suspensão temporária ou definitiva do IST ou mesmo da UTL.