Projeto Nº 1: Época Normal

Inteligência Artificial - Escola Superior de Tecnologia de Setúbal
2023/2024

Prof. Joaquim Filipe
Eng. Filipe Mariano

1. Jogo do Cavalo: Descrição Geral

O Jogo do Cavalo é uma variante do problema matemático conhecido como o Passeio do Cavalo, cujo objetivo é, através dos movimentos do cavalo, visitar todas as casas de um tabuleiro similar ao de xadrez. Esta versão decorrerá num tabuleiro de `10` linhas e `10` colunas (10x10), em que cada casa possui uma pontuação. Nesta secção pretende-se dar uma perspetiva geral do que é este jogo, deixando para a secção seguinte a explicação do que se pretende que seja desenvolvido no projeto de Inteligência Artificial relativo à resolução de um problema neste contexto por procura em Espaço de Estados.

	A	B	C	D	E	F	G	H	I	J	
1	92	52	04	33	15	20	30	23	82	94	1
2	18	51	22	08	35	59	69	61	36	49	2
3	91	12	74	93	11	05	19	75	28	21	3
4	26	88	50	99	37	64	47	97	98	62	4
5	44	83	54	41	43	76	67	09	96	79	5
6	32	06	34	48	27	16	85	68	60	07	6
7	55	87	72	58	81	40	17	73	38	95	7
8	00	29	70	42	13	10	80	78	14	24	8
9	56	01	71	39	84	77	03	25	63	66	9
10	65	90	89	45	46	02	57	31	86	53	10
	A	B	C	D	E	F	G	H	I	J	

Figura 1: Exemplo de tabuleiro inicial.

1.1. Tabuleiro

O jogo possui as seguintes características:

- O tabuleiro tem as dimensões 10x10 em que os valores de cada casa são entre `00` e `99`, sem repetição.
- Cada vez que é iniciado um jogo é construído um novo tabuleiro com o valor das casas distribuído aleatoriamente.

- O objectivo do jogo é acumular mais pontos que o adversário, usando um cavalo de xadrez. Cada jogador tem um cavalo da sua cor (branco ou preto).

1.2. Desenrolar do Jogo

- O jogo começa com a colocação do cavalo branco numa casa da **1ª linha (A1-J1 do tabuleiro)**. Esta casa é escolhida pelo jogador com o cavalo branco.
- Se a casa escolhida tiver um número com dois dígitos diferentes, por exemplo 57, então, em consequência, o número simétrico 75 é apagado do tabuleiro, tornando esta casa inacessível durante o resto do jogo. Ou seja, nenhum cavalo pode terminar outra jogada nessa casa.
- Se um cavalo for colocado numa casa com um número "duplo", por exemplo 66, então qualquer outro número duplo pode ser removido e o jogador deve escolher qual em função da sua estratégia (por *default* remover a de maior valor). Depois de um jogador deixar a casa para se movimentar para outra, a casa onde estava fica também inacessível para o jogo, ficando o numero da casa apagado.
- Após a primeira jogada (colocar o cavalo branco) segue-se a jogada do adversário com colocação do cavalo preto numa casa da **10ª linha (A10-J10)** do tabuleiro, casa essa que é escolhida pelo 2º jogador. O número simétrico da casa correspondente é também apagado.
- Depois de ambos os cavalos serem colocados, todas as jogadas seguintes são efectuadas através de um movimento de cavalo (usando as regras tradicionais do Xadrez para o cavalo). **Um cavalo não pode saltar para uma casa vazia (sem número) e também não pode fazê-lo para uma casa que esteja ameaçada pelo cavalo adversário.**
- A cada jogada de um jogador repete-se a regra do simétrico ou duplo.
- Um jogador ganha pontos por cada casa visitada pelo seu cavalo (igual ao valor da casa). Os pontos são contabilizados apenas para as casas visitadas, não pelos números simétricos ou duplos removidos.

1.3. Determinar o Vencedor

O jogo termina quando não for possível movimentar qualquer um dos cavalos no tabuleiro, sendo o vencedor o jogador que ganhou **mais** pontos.

2. Objetivo do Projeto: Jogador Único

No âmbito deste projeto vamos considerar o Problema do Cavalo como uma versão simplificada do jogo mencionado anteriormente, em que o principal objetivo consiste em atingir uma dada pontuação definida para o problema, no menor número possível de jogadas, i.e. de saltos de cavalo. Para isso, será necessário, a partir da casa inicial, deslocar o cavalo branco ao longo do tabuleiro em jogadas sucessivas até não ser possível efetuar qualquer movimento ou até atingir o objetivo.

A transformação do jogo num problema, para esta fase do projeto, pressupõe as seguintes diferenças ao nível das regras:

- Existe apenas um jogador (cavalo branco);
- O jogador começa por colocar o cavalo numa casa da primeira linha do tabuleiro;
- O estado final é atingido quando o cavalo chega a uma casa que lhe permite obter uma pontuação igual ou superior ao objetivo definido;
- Se não for possível atingir o objetivo, o programa deverá informar o utilizador de que o problema não tem solução;

- Os objetivos para os problemas A-F, fornecidos em anexo e que têm de ser resolvidos no âmbito deste projeto, são: A: 70, B: 60, C:270, D:600, E: 300, F:2000;
- A inicialização do processo de resolução do problema consiste na aplicação de um operador especial, de colocação do cavalo numa casa da primeira linha que tenha uma pontuação numérica. Esse operador permite fazer a geração dos sucessores do nível 1 a partir do nó raiz do grafo que representa cada um dos problemas acima referidos. A partir daí, são aplicáveis os operadores de movimentação do cavalo.

Pretende-se que os alunos escrevam e testem um programa, em `LISP`, para apresentar a sequência de estados ou de jogadas que conduzem da posição inicial do problema até à posição final, recorrendo aos algoritmos de procura lecionados nas aulas conforme se indica detalhadamente mais adiante. A solução óptima consiste no caminho com menor número de jogadas entre o estado inicial e o estado final.

3. Formulação do Problema

3.1. Tabuleiro

O tabuleiro é representado sob a forma de uma lista composta por `10` outras listas, cada uma delas com `10` átomos. Cada uma das listas representa uma linha do tabuleiro, enquanto que cada um dos átomos possui o valor da respetiva casa. Por outras palavras, o tabuleiro é representado por uma lista de listas em LISP, sendo que cada átomo representa uma casa em que a linha corresponde ao índice da sublista e a coluna ao índice do átomo dentro da linha. **As casas já visitadas terão o valor `NIL`, enquanto que a casa onde se encontra o cavalo deverá ter o valor `T`.**

De seguida mostram-se respetivamente a representação do 1º tabuleiro e do 2º tabuleiro da figura 2.

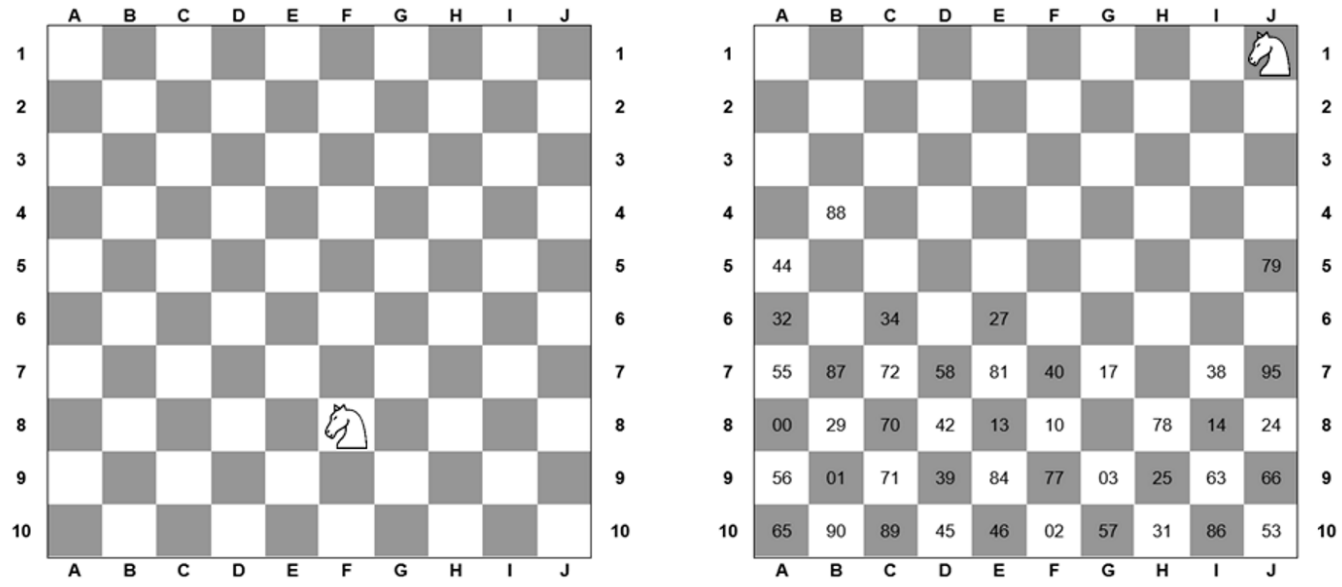


Figura 2: Exemplos de representações de tabuleiro (ambas as posições são terminais).

```
(
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
```

```
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL T NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
)
```

Figura 3: Representação do 1º tabuleiro da figura 2.

```
(
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL T)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL NIL NIL NIL NIL NIL NIL NIL NIL NIL)
 (NIL 88 NIL NIL NIL NIL NIL NIL NIL NIL)
 (44 NIL NIL NIL NIL NIL NIL NIL NIL 79)
 (32 NIL 34 NIL 27 NIL NIL NIL NIL NIL)
 (55 87 72 58 81 40 17 NIL 38 95)
 (0 29 70 42 13 10 NIL 78 14 24)
 (56 1 71 39 84 77 3 25 63 66)
 (65 90 89 45 46 2 57 31 86 53)
)
```

Figura 4: Representação do 2º tabuleiro da figura 3.

3.2. Solução Encontrada

A solução pode representar–se por uma sequência de estados, desde o estado inicial até ao estado final, ou então – por razões de legibilidade – pela lista de operações (i.e. jogadas) realizadas sobre as peças do tabuleiro. Cada jogada é identificada pela casa destino do cavalo (uma letra e um número).

3.3. Operadores

Os operadores representam os movimentos possíveis num determinado estado. Para o Problema do Cavalo o máximo de movimentos possíveis serão 8, desde que essas casas não tenham sido ainda visitadas ou removidas pela regra dos simétricos ou duplos.

Os custos de aplicação dos operadores são considerados constantes e unitários.

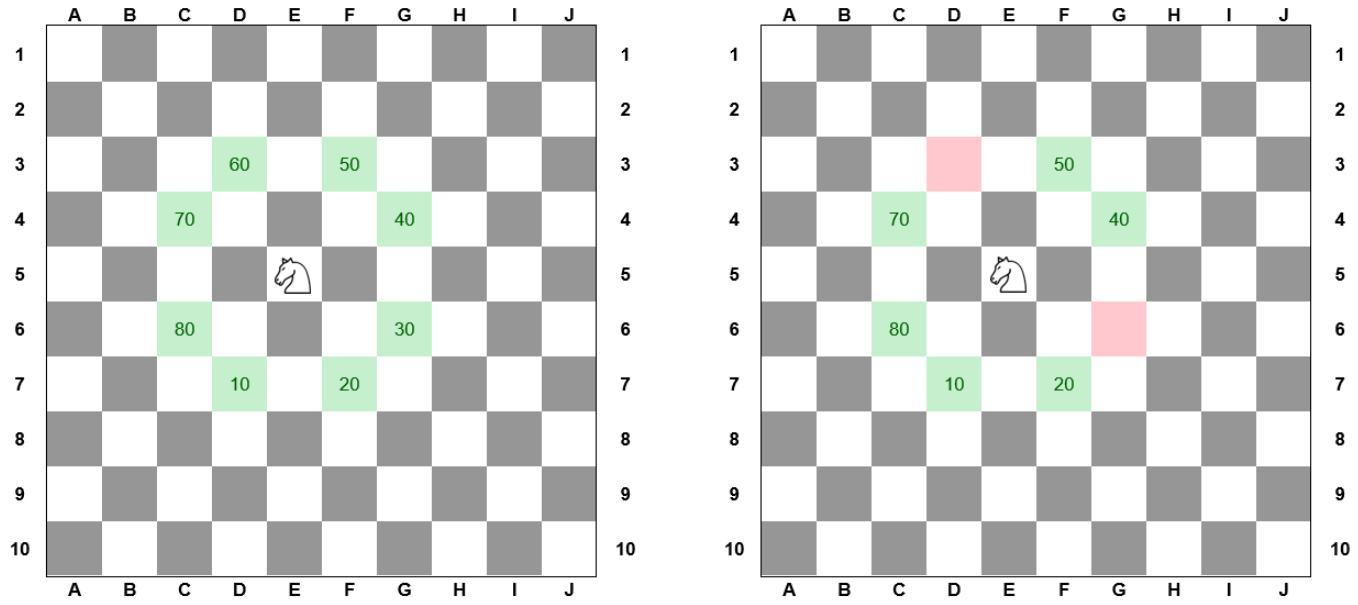


Figura 5: No 1º tabuleiro encontra-se uma situação em que os 8 movimentos possíveis poderão ser realizados porque as casas ainda não foram visitadas (têm pontos). No 2º tabuleiro apenas é possível realizar 6 movimentos porque duas das casas já foram visitadas.

3.4. Estrutura do Programa

O programa deverá estar dividido em três partes, cada uma num ficheiro diferente:

1. Uma parte com a implementação dos métodos de procura e com a implementação das métricas de análise de eficiência, i.e. a parte do programa que é independente do domínio de aplicação;
2. Outra para implementar tudo o que envolve a resolução do problema concreto, incluindo a definição dos operadores e heurísticas, específicos do domínio de aplicação;
3. E a terceira parte para fazer a interação com o utilizador e para proceder à escrita e leitura de ficheiros.

O projeto deverá obrigatoriamente apresentar um estudo comparativo do comportamento dos três métodos: procura em largura primeiro (BFS), procura em profundidade primeiro (DFS) e A*. Será ainda permitido que os alunos desenvolvam, opcionalmente, outros algoritmos conforme indicado na secção 7.

No caso dos métodos informados, o programa deverá utilizar funções heurísticas modulares, ou seja, que possam ser colocadas ou retiradas do programa de procura como módulos.

As heurísticas não devem estar embutidas de forma rígida no programa de procura. Exige–se a utilização de duas heurísticas, uma fornecida no fim do presente documento e outra desenvolvida pelos alunos.

O projeto deverá incluir a implementação de cada um dos algoritmos, de forma modular, permitindo que o utilizador escolha qualquer um deles, conjuntamente com os respetivos parâmetros (heurística, profundidade, etc.) para a resolução de um dado problema.

4. Experiências

Pretende-se que o projeto estude, para cada problema fornecido em anexo, o desempenho de cada algoritmo e, no caso do A*, de cada uma das heurísticas propostas, apresentando, em relação a cada problema indicado em anexo, a solução encontrada e dados estatísticos sobre a sua eficiência, nomeadamente o número de nós gerados, o número de nós expandidos, a penetrância, o fator de ramificação média e o tempo de execução.

Os projetos deverão apresentar os dados acima referidos num ficheiro produzido automaticamente pelo programa, sendo descontado `0,5` valor por cada problema não resolvido. No caso de ser apresentada a solução, mas não o estudo de desempenho das heurísticas o desconto é de apenas `0,2` valor por cada caso.

5. Heurísticas

Sugere-se usar como heurística de base, uma heurística que privilegia visitar as casas com o maior número de pontos. Para um determinado tabuleiro x:

h(x) = o(x)/m(x)

em que:
`m(x)` é a média por casa dos pontos que constam no tabuleiro x,
`o(x)` é o número de pontos que faltam para atingir o valor definido como objetivo.

Esta heurística pode ser melhorada para refletir de forma mais adequada o conhecimento acerca do domínio e assim contribuir para uma maior eficiência dos algoritmos de procura informados. Além da heurística acima sugerida, deve ser definida pelo menos uma segunda heurística que deverá melhorar o desempenho dos algoritmos de procura informados.

6. Grupos

Os projetos deverão ser realizados em grupos de, no máximo, duas pessoas sendo contudo sempre sujeitos a avaliação oral individual para confirmação da capacidade de compreensão dos algoritmos e de desenvolvimento de código em LISP.

O grupo poderá ser constituido por alunos que frequentam turmas ou turnos diferentes.

7. Bónus

O projeto inclui uma parte opcional que permite atribuir um bónus aos alunos que a conseguirem implementar.

Para além dos três métodos de procura anteriormente mencionados, cada grupo tem a opção de programar, aplicar e estudar uma ou mais das seguintes estratégias *Simplified Memory–Bounded A** (SMA*), *Iterative Deepening A** (IDA*) ou *Recursive Best First Search* (RBFS).

A programação e estudo dos métodos opcionais vale 1 valor cada, a somar ao valor total do projeto, sendo a nota final do projeto limitada ao máximo de 20 valores.

8. Datas

Entrega do projeto: 18 de Dezembro de 2023, até as 23:00.
Discussão do projeto: Inicio de Fevereiro de 2024, conjuntamente com a discussão do segundo projeto.

9. Documentação a entregar

A entrega do projeto e da respetiva documentação deverá ser feita através do Moodle, na zona do evento "Entrega do 1º Projeto". Todos os ficheiros a entregar deverão ser devidamente arquivados num ficheiro comprimido (`ZIP` com um tamanho máximo de 5Mb), até à data acima indicada. O nome do arquivo deve seguir a estrutura `<iniciaisAluno1>_<numeroAluno1>_<iniciaisAluno2>_<numeroAluno2>_P1`.

9.1. Código fonte

Os ficheiros de código devem ser devidamente comentados e organizados da seguinte forma:

projeto.lisp Carrega os outros ficheiros de código, escreve e lê ficheiros, e trata da interação com o utilizador.

puzzle.lisp Código relacionado com o problema.

procura.lisp Deve conter a implementação de:

1. Algoritmo de Procura de Largura Primeiro (BFS)
2. Algoritmo de Procura do Profundidade Primeiro (DFS)
3. Algoritmo de Procura do Melhor Primeiro (A*)
4. Os algoritmos SMA*, IDA* e/ou RBFS (caso optem por implementar o bónus)

9.2. Exercícios

Deverá haver um ficheiro de exercícios, com a designação `problemas.dat`, contendo todos os exemplos de tabuleiro que se quiser fornecer ao utilizador, organizados de forma sequencial, e que este deverá escolher mediante um número inserido no interface com o utilizador.

Esse número representa o número de ordem na sequência de exemplos. O ficheiro deverá ter várias listas, separadas umas das outras por um separador legal, e não uma lista de listas. Essas listas serão tantas quantos os problemas fornecidos.

Na oral, os docentes irão solicitar que se adicione mais um exemplo, numa dada posição do ficheiro, que deverá imediatamente passar a ser selecionável através do interface com o utilizador e ser resolvido normalmente.

9.3. Manuais

No âmbito da Unidade Curricular de Inteligência Artificial pretende−se que os alunos pratiquem a escrita de documentos recorrendo à linguagem de marcação ***Markdown***, que é amplamente utilizada para os ficheiros ***ReadMe*** no ***GitHub***. Na Secção 4 do guia de Laboratório nº 2, encontrará toda a informação relativa à estrutura recomendada e sugestões de ferramentas de edição para ***Markdown***.

Para além de entregar os ficheiros de código e de problemas, é necessário elaborar e entregar 2 manuais (o manual de utilizador e o manual técnico), em formato PDF juntamente com os sources em MD, incluídos no arquivo acima referido.

Manual Técnico:

O Manual Técnico deverá conter o algoritmo geral, por partes e devidamente comentado; descrição dos objetos que compõem o projeto, incluindo dados e procedimentos; identificação das limitações e opções técnicas. Deverá ser apresentada uma análise critica dos resultados das execuções do programa, onde deverá transparecer a compreensão das limitações do projeto. Deverão usar uma análise comparativa do conjunto de

execuções do programa para cada algoritmo e cada problema, permitindo verificar o desempenho de cada algoritmo e das heurísticas. Deverá, por fim, apresentar a lista dos requisitos do projeto (listados neste documento) que não foram implementados.

Manual de Utilizador:

O Manual do Utilizador deverá conter a identificação dos objetivos do programa, juntamente com descrição geral do seu funcionamento; explicação da forma como se usa o programa (acompanhada de exemplos); descrição da informação necessária e da informação produzida (ecrã/teclado e ficheiros); limitações do programa (do ponto de vista do utilizador, de natureza não técnica).

10. Avaliação

Funcionalidade	**Valores**
Representação do estado e operadores	2.5
Funções referentes ao puzzle	2.5
Procura em profundidade e largura	2.5
Procura com A* e heurística dada	2.5
Implementação de nova heurística	1.5
Resolução dos problemas a) a f)	3
Resolução do problema g) (dado na avalição oral)	0.5
Qualidade do código	2
Interação com o utilizador	1
Manuais (utilizador e técnico)	2
Total	20
Bónus	3

O valor máximo da nota são 20 valores, já com a integração do valor do bónus.

Os problemas a) a f) estão descritos na Secção Experiências, enquanto que o problema g) será facultado durante a avaliação oral. A avaliação do projeto levará em linha de conta os seguintes aspectos:

- Data de entrega final – Existe uma tolerância de 3 dias em relação ao prazo de entrega, com a penalização de 1 valor por cada dia de atraso. Findo este período a nota do projeto será 0.
- Correção processual da entrega do projeto – (Moodle; manuais no formato correto). Anomalias processuais darão origem a uma penalização que pode ir até 3 valores.
- Qualidade técnica – Objetivos atingidos; Código correto; Facilidade de leitura e manutenção do programa; Opções técnicas corretas.
- Qualidade da documentação – Estrutura e conteúdo dos manuais que acompanham o projeto.

- Avaliação oral – Eficácia e eficiência da exposição; Compreensão das limitações e possibilidades de desenvolvimento do programa. Nesta fase poderá haver lugar a uma revisão total da nota de projeto.

11. Recomendações finais

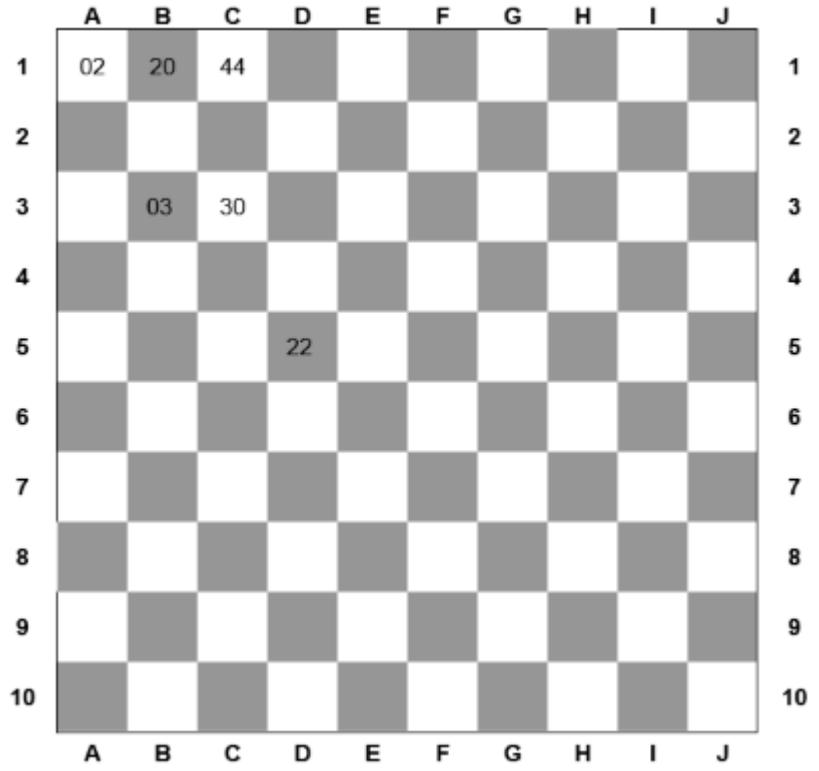
Com este projeto pretende–se motivar o paradigma de programação funcional. A utilização de variáveis globais, de instruções de atribuição do tipo set, setq, setf, de ciclos, de funções destrutivas ou de quaisquer funções com efeitos laterais é fortemente desincentivada dado que denota normalmente uma baixa qualidade técnica. A sequenciação só será permitida no contexto das funções de leitura/escrita.

As únicas excepções permitidas a estas regras poderão ser a utilização da instrução loop para implementar os ciclos principais dos algoritmos implementados (como alternativa a uma solução puramente recursiva), em conjugação com as variáveis globais Abertos e Fechados para manutenção das listas de nós.

ATENÇÃO: Suspeitas confirmadas de plágio serão penalizadas com a anulação de ambos os projetos envolvidos (fonte e destino), e os responsáveis ficam sujeitos à instauração de processo disciplinar.

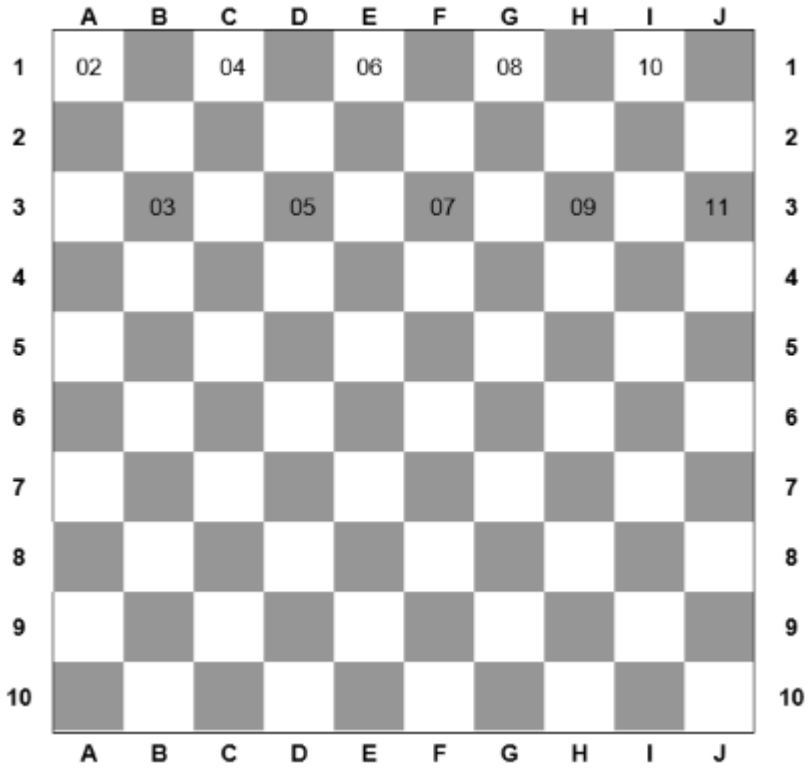
Anexos – Problemas

Problema A:



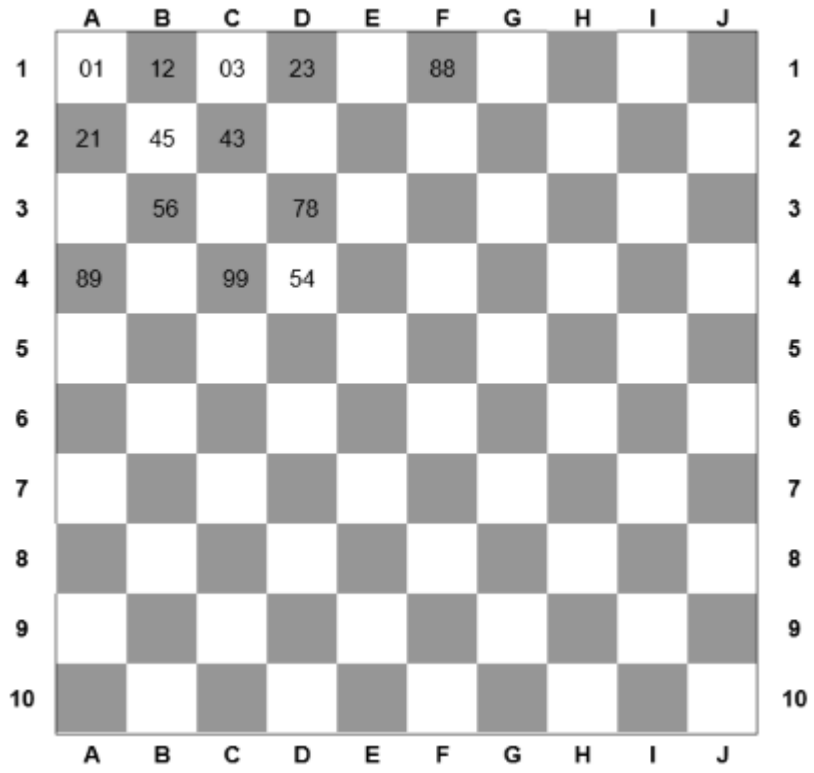
Objetivo: 70 pontos

Problema B:



Objetivo: 60 pontos

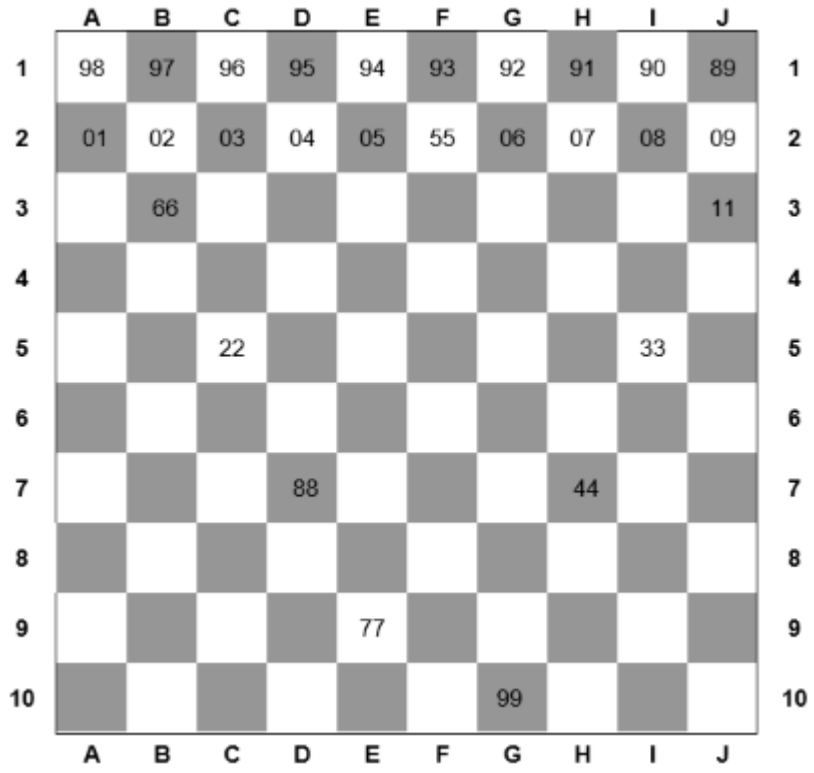
Problema C:



	A	B	C	D	E	F	G	H	I	J	
1	01	12	03	23		88					1
2	21	45	43								2
3		56		78							3
4	89		99	54							4
5											5
6											6
7											7
8											8
9											9
10											10
	A	B	C	D	E	F	G	H	I	J	

Objetivo: 270 pontos

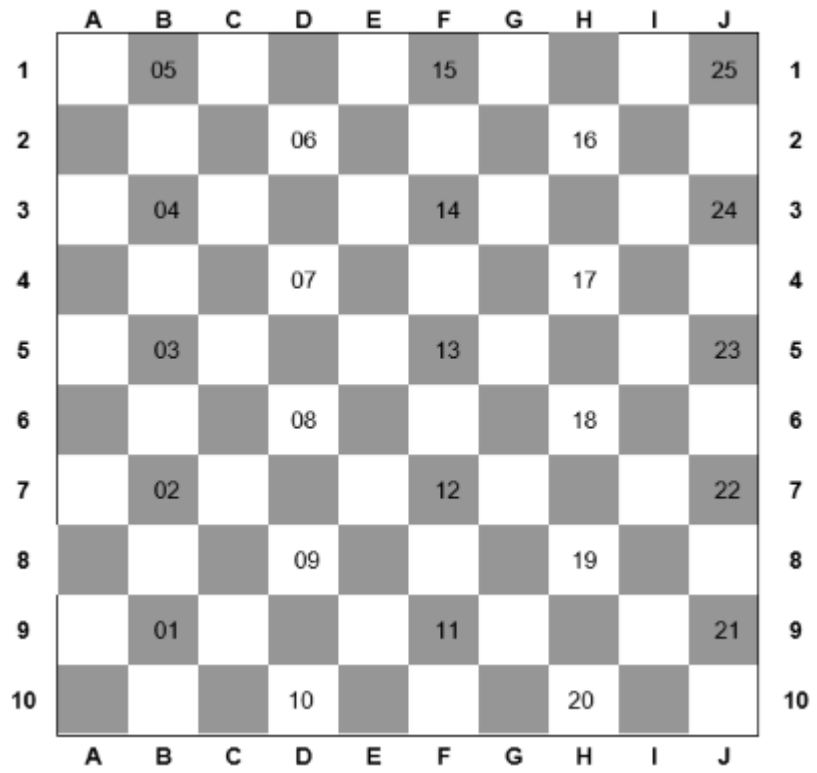
Problema D:



	A	B	C	D	E	F	G	H	I	J	
1	98	97	96	95	94	93	92	91	90	89	1
2	01	02	03	04	05	55	06	07	08	09	2
3		66								11	3
4											4
5			22						33		5
6											6
7				88				44			7
8											8
9					77						9
10							99				10
	A	B	C	D	E	F	G	H	I	J	

Objetivo: 600 pontos

Problema E:



Objetivo: 300 pontos

Problema F:

Deverá ser estudado para os vários algoritmos a partir de um tabuleiro completo gerado aleatoriamente e gravado no ficheiro de problemas.

Objetivo: 2000 pontos

Problema G:

Será disponibilizado durante a discussão do projeto.