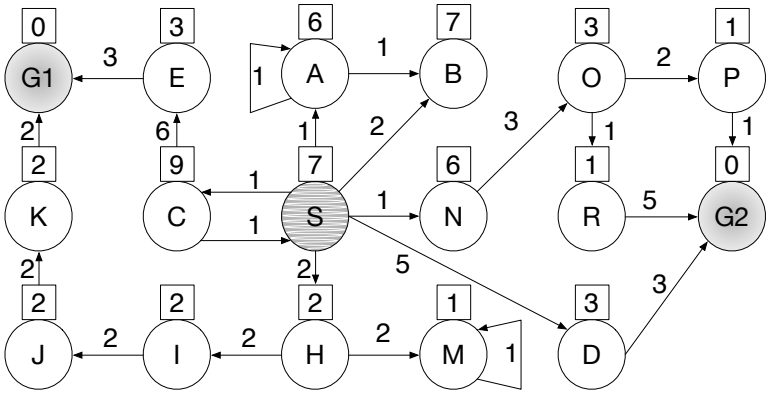
**Inteligência Artificial – 2022/23**

**26/Outubro/2022–14:30h – Duração: 2:00h**

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

# 1º Teste (Modelo A)
# – Sem consulta –

**I) [5val]** Considere o seguinte grafo de estados de um problema de procura. Os valores apresentados nos arcos correspondem ao custo do operador (ação) respetivo, enquanto os valores nos retângulos correspondem ao valor de uma heurística (estimativa do custo de chegar desse estado ao objetivo). Não se representam os nomes dos operadores, correspondendo cada arco a um operador distinto. Em todos os algoritmos cegos e em caso de empate nos algoritmos informados, quando um nó é expandido, assuma que os seus sucessores são sempre colocados na fronteira por ordem lexicográfica do nome do nó, de forma que o nó mais próximo do início do alfabeto seja selecionado antes dos seus irmãos. Pretende-se encontrar um caminho desde o estado S até G1 ou G2.

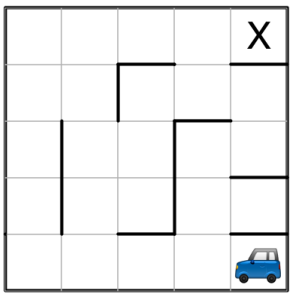



a) Caracterize a heurística quanto à admissibilidade e consistência, justificando a sua resposta.

b) Indique o caminho encontrado por cada um dos seguintes algoritmos: procura em profundidade primeiro (em árvores), procura em profundidade primeiro (em grafos), procura em largura primeiro (otimizada), procura de custo uniforme (em grafos), procura sôfrega (em árvores), procura sôfrega (em grafos) e A* (em grafos, na versão mais eficiente do algoritmo que garante a obtenção da solução ótima dadas as características da heurística).

c) O algoritmo de procura por aprofundamento progressivo é por vezes usado como alternativa ao algoritmo de procura em largura primeiro. Apresente uma vantagem e uma desvantagem do algoritmo de procura por aprofundamento progressivo sobre o algoritmo de procura em largura primeiro. Seja conciso(a) e preciso(a).

**II) [2val]** Considere um agente (carro) que pretende atingir a casa de saída (X) de um labirinto como o ilustrado na imagem ao lado. Em cada momento, o agente está virado para uma direção $d \in \{N, S, E, W\}$. Com uma única ação, o agente pode mover-se em frente a uma velocidade ajustável, ou mudar de direção. As ações de mudança de direção são *direita* e *esquerda*, que alteram a direção do agente em 90º. Mudanças de direção só são permitidas quando a velocidade é zero, deixando o carro com uma velocidade de zero, virado na nova direção. As ações de movimentação são *acelerar* e *travar*. Acelerar aumenta a velocidade em 1 e travar reduz a velocidade em 1; em ambos os casos, o agente avança o número de casas igual á sua nova velocidade. Qualquer ação que leve o agente a embater numa parede é ilegal. Qualquer ação que reduza a velocidade abaixo de 0 ou acima de $v_{max}$ é também ilegal. O objetivo do agente é encontrar uma sequência de ações que o deixe parado na casa de saída, usando o menor número de ações. Como exemplo, se o carro estivesse inicialmente parado, poderia *acelerar*, avançando uma casa na direção *W*, após o qual poderia *acelerar* novamente avançando duas casas, e depois *travar*, avançando uma casa, e *travar* novamente de modo a parar. Em seguida, poderia rodar à *direita*, repetindo a sequência de ações anteriores mais duas vezes até chegar ao objetivo.




a) Se a grelha for M por N, qual o tamanho do espaço de estados? Justifique a sua resposta. (pode assumir que todas as configurações são atingíveis a partir do estado inicial.

b) Qual o fator de ramificação máximo? Justifique a sua resposta. (pode assumir que as acções ilegais não são devolvidas pela função sucessor)

c) A distância de Manhattan entre o agente e a casa de saída é uma heurística admissível? Justifique a sua resposta.

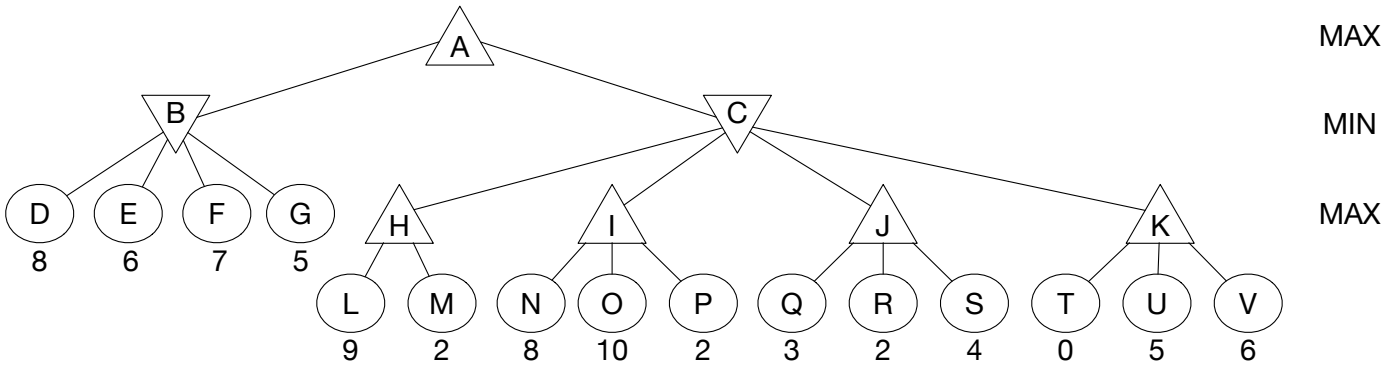
**III) [2val]** Considere a seguinte implementação do algoritmo A*, que pode estar errada!!

```
function A*(problem) returns a solution, or failure
      node ← a node with STATE=problem.INITIAL-STATE, PATH-COST = 0
      frontier ← a priority queue with node as the only element
      explored ← a singleton set with node.STATE
      loop do
            if EMPTY?(frontier) then return failure
            node ← POP(frontier)
            if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
            for each action in problem.ACTIONS(node.STATE) do
                  child ← CHILD-NODE(problem, node, action)
                  if child.STATE is not in explored then do
                        add child.STATE to explored
                        frontier ← INSERT(child, frontier)
```

Assuma que a heurística é consistente e que a fronteira está ordenada da forma habitual no A* i.e. por f = g + h crescente. Para cada uma das seguintes alíneas, indique se ela é verdadeira ou falsa, justificando de forma concisa as suas respostas.

a) A chamada a `problem.ACTIONS(node.STATE)` pode ocorrer várias vezes sobre o mesmo estado.
b) O algoritmo não é completo.
c) O algoritmo pode devolver uma solução sub-ótima.
d) A implementação está incorreta, mas nenhum dos problemas acima ocorrerá.
e) A implementação está correta.

**IV) [4val]** Considere a árvore de jogo de dois jogadores (MAX e MIN), onde os valores nas folhas são estimativas do ganho para MAX a partir desse estado e os filhos de um nó são visitados da esquerda para a direita.

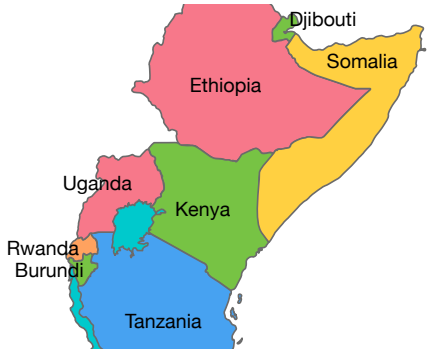


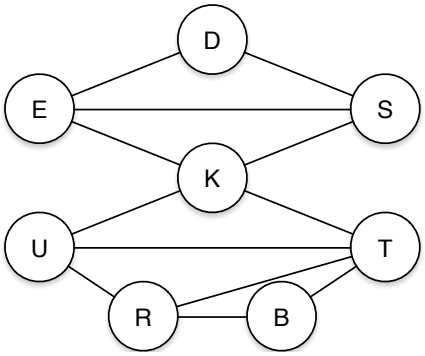

a) Calcule o valor MINIMAX de cada nó não terminal e indique qual o movimento que MAX deverá escolher.
b) Assumindo que a árvore é percorrida da esquerda para a direita, indique quais os nós que nunca chegariam a ser visitados pelo algoritmo de procura α−β.
c) Considere agora que os nós H, I, J e K da árvore da figura são nós estocásticos (CHANCE) onde cada sucessor é equiprovável. Calcule o valor EXPECTEDMINIMAX dos nós A, B e C.

**V) [3val]** Verifique, usando o algoritmo DPLL, se $(c \wedge d)$ é uma consequência lógica do seguinte conjunto de frases em lógica proposicional (sempre que tiver de escolher, deve escolher as variáveis proposicionais alfabeticamente, e atribuir VERDADEIRO antes de FALSO):

$$c \Rightarrow d$$
$$b \Leftrightarrow c$$
$$a \Rightarrow b$$
$$(a \wedge c) \Rightarrow d$$

**VI) [4val]** Considere o seguinte grafo de restrições, representando parte do mapa da África que pretendemos colorir com três cores (A)marela, (C)astanha e (M)agenta, de modo a que países adjacentes, ligados por um arco, não tenham a mesma cor.




B = Burundi
D = Djibouti
E = Ethiopia
K = Kenya
R = Rwanda
S = Somalia
T = Tanzania
U = Uganda

a) Assumindo que à variável K já foi atribuída a cor Castanha, aplique o algoritmo de Verificação para a Frente (Forward Checking) e elimine (na folha de resposta) as cores que seriam descartadas para as restantes variáveis.

b) Assumindo que à variável K já foi atribuída a cor Castanha, à variável U a cor Amarela e que não foi efetuada Verificação para a Frente (Forward Checking), aplique o algoritmo de Consistência de Arcos (AC-3) e elimine (na folha de resposta) as cores que seriam descartadas para as restantes variáveis.

c) Considere os seguintes domínios possíveis, obtidos após seleção da variável T e atribuição da cor Castanha, seguida da propagação de restrições. Das restantes variáveis, quais poderiam ser selecionadas pela heurística da variável mais constrangida para serem consideradas a seguir?

| B | D | E | K | R | S | T | U |
|---|---|---|---|---|---|---|---|
| A M | A C M | A C M | A M | A M | A C M | C | A M |

d) Considere os seguintes domínios possíveis, obtidos após seleção da variável T e atribuição da cor Castanha, seguida da propagação de restrições (o mesmo da alínea anterior). Ignore a heurística da variável mais constrangida. Das restantes variáveis, quais poderiam ser selecionadas pela heurística da variável mais constrangedora para serem consideradas a seguir?

| B | D | E | K | R | S | T | U |
|---|---|---|---|---|---|---|---|
| A M | A C M | A C M | A M | A M | A C M | C | A M |

e) Ignore as alíneas anteriores. Encontre uma solução para o problema aplicando o algoritmo de procura com retrocesso (*backtrack*) em que é usada a heurística da variável mais constrangida e desempate pela heurística da variável mais constrangedora (e em caso de persistência de empate, por ordem alfabética crescente), e após cada atribuição de um valor a uma variável é executado o algoritmo de consistência de arcos (AC3) para a redução dos domínios das variáveis. Os valores (cores) devem ser atribuídos por ordem alfabética crescente. Qual a solução encontrada? Quantas vezes teve de retroceder (*backtrack*)?

**VII) [Bónus: até 2val]** Para cada alínea, indique se ela é verdadeira ou falsa. Cada resposta correta vale 0,4val, cada resposta errada vale -0,4val. A pergunta tem uma cotação mínima de 0 valores.

Seja $h_1(s)$ uma heurística admissível e $h_2(s)=2h_1(s)$.

a) A solução encontrada pelo algoritmo A* (versão em árvore) com $h_2(s)$ é garantidamente ótima.

b) A solução encontrada pelo algoritmo A* (versão em árvore) com $h_2(s)$ tem garantidamente um custo menor ou igual ao dobro do custo da solução ótima.

Considere uma árvore de jogo de dois jogadores (MAX e MIN) cujo valor de MINIMAX do jogo (da raiz) é $v_M$. Considere uma segunda árvore de jogo em tudo idêntica à anterior, mas onde os nós do jogador MIN foram substituídos por nós CHANCE, com uma distribuição de probabilidade arbitrária mas conhecida. O valor EXPECTEDMINIMAX deste novo jogo é $v_E$.

c) O valor $v_M$ é garantidamente menor ou igual do que o valor $v_E$.

d) Usar a estratégia ótima do jogo original (MINIMAX) no jogo modificado (com os nós CHANCE) garante ao jogador MAX um ganho de pelo menos $v_M$.

e) Usar a estratégia ótima do jogo original (MINIMAX) no jogo modificado (com os nós CHANCE) garante ao jogador MAX um ganho de pelo menos $v_E$.

**FIM!**

**Inteligência Artificial – Folha de Respostas**

**26/Out/2022 – 14:30** **DI/FCT/UNL, 1º Teste, 2022/23, Duração: 2h** **Modelo: A e B**

| **Nome:** | **Número:** |
|---|---|

**I.a)** Admissível: **NÃO** Consistente: **NÃO**

Justificação: Não é admissível pois existe um estado (C) para o qual o valor da heurística (h(C)=9) é superior ao custo do caminho mais curto para chegar desse estado ao objectivo h*(C)=c(C→S→N→O→P→G2)=8..

Não é consistente pois qualquer heurística consistente também é admissível.

Outra forma de demostrar a inconsistência seria mostrar que existem estados n1, n2, ligados por um arco com custo c(n1,n2) tal que h(n1)>c(n2,n1)+h(n2), como por exemplo, com n1=S e n2=H, onde temos que h(S)=7, c(S,H)=2 e h(H)=2, logo, h(S)>c(S,H)+h(H).

| **I.b) Algoritmo** | **Solução** |
|---|---|
| Profundidade primeiro (árvores) | **Não termina (S→A→A→A ....)** |
| Profundidade primeiro (grafos) | **S→C→E→G1** |
| Largura primeiro (optimizada) | **S→D→ G2** |
| Custo uniforme (grafos) | **S→N→O→P→G2** |
| Sôfrega (árvores) | **Não termina (S→H→M→M ...)** |
| Sôfrega (grafos) | **S→H→I→J→K→G1** |
| A* | Nenhuma versão do algoritmo A* garante a solução ótima para heurísticas não admissíveis. Excecionalmente, foi aceite a resposta **S→N→O→P→G2** |

**I.c)** Vantagem: Usa menos memória. O algoritmo de procura por aprofundamento progressivo tem uma complexidade espacial de O(bd) enquanto o algoritmo de procura em largura primeiro tem uma complexidade espacial de $O(b^d)$.

Desvantagem: Expande mais nós. Ainda que ambos tenham uma complexidade temporal de $O(b^d)$, o algoritmo de procura por aprofundamento progressivo tem de gerar as árvores com profundidade d-1 antes de gerar a árvore com profundidade d, enquanto o algoritmo de procura em largura primeiro apenas gera a árvore de profundidade d.

**II.a)** Tamanho: $\mathbf{4} * \boldsymbol{M} * \boldsymbol{N} * (\boldsymbol{v_{max}} + \mathbf{1})$

Justificação: A representação do estado é (*direção*, *x*, *y*, *velocidade*), e a velocidade pode tomar qualquer valor entre 0 e $v_{max}$.

**II.b)** Fator de Ramificação Máximo: **3**

Justificação: O fator de ramificação máximo acontece quando o agente está estacionário, podendo executar uma de três acções: *acelerar*, *esquerda* e *direita*.

**II.c)** Resposta: **Não**

Justificação: A distância de Manhattan não é uma heurística admissível pois o agente pode mover-se com uma velocidade média superior a 1 (acelerando até $v_{max}$ e travando até 0 quando se aproxima do objetivo), podendo assim atingir o objetivo em menos passos do que o número de casas entre a sua localização e o objetivo. Por exemplo, se o agente estiver a 6 casas do objetivo em linha reta e sem obstáculos, e a sua velocidade for 4, ele chega ao objetivo com velocidade 0 em 4 passos, travando em cada um, pelo que o número de paços seria inferior à distância de Manhattan, que seria igual a 6.

| **III** [Verdadeira /Falsa] | a) **FALSA** | b) **FALSA** | c) **VERDADEIRA** | d) **FALSA** | e) **FALSA** |
|---|---|---|---|---|---|

**Justificação**: O erro está na inserção do estado de um nó nos explorados no momento da inserção desse nó na fronteira, e não quando o nó é retirado da fronteira. Como consequência desse erro, quando o primeiro caminho até um estado é encontrado, esse estado é colocado na lista dos explorados, impedindo que outros nós até esse estado, possivelmente correspondendo a melhores caminhos, venham a ser colocados na fronteira para posteriormente poderem ser explorados. Isto pode resultar numa solução sub-óptima. No A*, só temos a garantia que um estado foi atingido de forma otimizada quando um nó para esse estado é retirado da fronteira.

a) Antes de inserirmos um nó na fronteira, estamos a inserir o estado correspondente nos explorados. Como verificamos se o estado de um nó já faz parte dos explorados antes de acrescentar esse nó à fronteira, nunca vamos inserir na fronteira um nó com um estado igual ao estado de um nó que já lá tenha sido inserido. Assim, nunca inserimos na fronteira mais do que um nó com o mesmo estado, pelo que haverá no máximo uma chamada a `problem.ACTIONS(node.STATE)` por estado.

b) O A* (versão em árvores) é um algoritmo completo. A diferença é que o algoritmo apresentado "corta" partes da árvore de procura quando já passou por um nó para o mesmo estado. Ou seja, comparado com a versão correta do A* em árvores, o algoritmo apresentado apenas descarta sub-árvores para estados que já estão a ser considerados noutra parte da árvore, pelo que o algoritmo é completo.

c) Ver explicação inicial.

d) Dado que c) é verdadeira.

e) Dado que c) é verdadeira.

**IV.a)**

A: **5** B: **5** C: **4** H: **9** I: **10** J: **4** K: **6** Movim. de MAX: **B**

**IV.b)** **P K T U V**

**IV.c)** A: **5** B: **5** C: **3**

**V)** É consequência lógica (SIM/NÃO)? **NÃO**

Verificação: Conversão para a forma clausal do conjunto de frases e da negação de $(c \wedge d)$:

$$(\neg C \vee D) \wedge (\neg B \vee C) \wedge (\neg C \vee B) \wedge (\neg A \vee B) \wedge (\neg A \vee \neg C \vee D) \wedge (\neg C \vee \neg D)$$

Aplicação do DPLL:

$$\begin{matrix}(\neg C \vee D)\\(\neg B \vee C)\\(\neg C \vee B)\\(\neg A \vee B)\\(\neg A \vee \neg C \vee D)\\(\neg C \vee \neg D)\end{matrix} \overset{Pure\ (\neg A)}{\Longrightarrow} \begin{matrix}(\neg C \vee D)\\(\neg B \vee C)\\(\neg C \vee B)\\(\neg C \vee \neg D)\end{matrix} \overset{Split\ (B)}{\Longrightarrow} \begin{matrix}(\neg C \vee D)\\(C)\\(\neg C \vee \neg D)\end{matrix} \overset{Unit\ (C)}{\Longrightarrow} \begin{matrix}(D)\\(\neg D)\end{matrix} \overset{Unit\ (D)}{\Longrightarrow} \bot$$

$$\overset{BackTrack\ (\neg B)}{\Longrightarrow} \begin{matrix}(\neg C \vee D)\\(\neg C)\\(\neg C \vee \neg D)\end{matrix} \overset{Pure\ (\neg C)}{\Longrightarrow} \top$$

Como demonstrámos que o conjunto de frases dado em conjunto com a negação de $(c \wedge d)$ é satisfazível, concluímos que a frase $(c \wedge d)$ não é uma consequência lógica do conjunto de frases dado.

**VI**

**a)**

| **B** | **D** | **E** | **K** | **R** | **S** | **T** | **U** |
|---|---|---|---|---|---|---|---|
| A C M | A C M | A ■ M | C | A C M | A ■ M | A ■ M | A ■ M |

**b)**

| **B** | **D** | **E** | **K** | **R** | **S** | **T** | **U** |
|---|---|---|---|---|---|---|---|
| A ■ ■ | A C M | A ■ M | C | ■ C ■ | A ■ M | ■ ■ M | A |

**c)**

Variáveis: **B, K, R, U**

**d)**

Variáveis: **K, E, S**

**e)**

B: **M** D: **A** E: **C** K: **A**

R: **A** S: **M** T: **C** U: **M**

Número de retrocessos: **0 (zero)**

| **VII)** | a) **FALSA** | b) **VERDADEIRA** | c) **VERDADEIRA** | d) **VERDADEIRA** | e) **FALSA** |
|---|---|---|---|---|---|