% INTELIGÊNCIA ARTIFICIAL Exame, xx/xx/xxxx (Tempo: 3h)

Identificação do aluno:

Nome: _____ #: _____

I

1. Analise as seguintes funções escritas em Python e explique o que fazem:

a)
```
def f(x):
    if x==[]:
        return 0
    if x[0]>0:
        return x[0] + f(x[1:])
    return f(x[1:])
```

Esta função recebe uma lista 'x' como entrada e retorna a soma de todos os números positivos na lista. A função usa recursão para percorrer os elementos da lista.
· O caso base é quando a lista está vazia, caso em que a função retorna 0.
· Se o primeiro elemento da lista (x[0]) for maior que 0, ele é adicionado à soma e a função é chamada novamente com o restante da lista (x[1:]).
· Se o primeiro elemento da lista não for maior que 0, a função é chamada novamente com o resto da lista.

b)
```
def g(x):
    if x==[]:
        return [[]]
    y = g(x[1:])
    return y + [ [x[0]]+z for z in y ]
```

É uma implementação de um algoritmo de geração de combinações. Ela funciona da seguinte maneira:

· Se a lista x for vazia, a função retorna uma lista com uma lista vazia (representando a combinação vazia).
· Caso contrário, a função chama ela mesma recursivamente para gerar as combinações da lista x[1:] (isto é, da lista x sem o primeiro elemento). Essas combinações são armazenadas na variável y.
· Em seguida, a função adiciona o primeiro elemento da lista x a cada uma das combinações geradas para a lista x[1:], e retorna essas combinações modificadas juntamente com as combinações geradas para a lista x[1:].

2. No contexto da geração de todas as interpretações de uma fórmula em lógica proposicional, é necessário gerar todas as combinações de valores possíveis das diversas variáveis proposicionais contidas na fórmula. Assim, programe uma função que, dada uma lista de variáveis proposicionais, gere todas as combinações de valores possíveis.
Exemplo:
```
>>> interpretacoes(["a","b"])
[ [ ("a",True), ("b",True) ], [ ("a",True), ("b",False) ], [ ("a",False), ("b",True) ], [ ("a",False), ("b",False) ] ]
```

```
def interpretacoes(variaveis):
  # Criar uma lista vazia de interpretações
  interpretacoes = []

  # Gerar todas as combinações possíveis de valores para as variáveis
  for valor_a in [True, False]:
    for valor_b in [True, False]:
      # Adicionar uma nova interpretação à lista
      interpretacoes.append([(variaveis[0], valor_a), (variaveis[1], valor_b)])

  # Retornar a lista de interpretações
  return interpretacoes
```

II

1. Neste exercício, tem um conjunto de questões de escolha. Em cada alínea, apenas uma das opções dadas está certa, e apenas pode seleccionar uma delas. Cada resposta errada desconta 20% da cotação da alínea.

a) A frase "Todos os livros de Banda Desenhada têm capa dura" pode ser representada em Lógica de 1ª ordem da seguinte forma:

$\forall x\ (Livro(x) \wedge BandaDesenhada(x)) \Rightarrow \neg\ CapaDura(x)$	
$\forall x\ BandaDesenhada(x) \Rightarrow \neg\ Capa(x,Dura)$	
$\forall x\ Livro(x) \vee (\ BandaDesenhada(x)) \Rightarrow \neg\ Capa(x,Dura)$	
$\forall x\ Livro(x) \wedge (\ BandaDesenhada(x)) \Rightarrow \neg\ Capa(x,Dura)$	
Nenhuma das anteriores	X

b) A frase "A melhor nota a Português foi a da Ana" pode ser representada em Lógica de 1ª ordem da seguinte forma:

$\forall x\ Nota(Ana,Português) > Nota(x,Português) \wedge Aluno(x)$	
$\forall x,y,z\ Nota(Ana,Português,y) > Nota(x,Português,z) \wedge y>z$	
$\forall x\ Nota(Ana,Português) \geq Nota(x,Português) \vee Aluno(x)$	
$\forall x\ Aluno(x) \Rightarrow (\ Nota(Ana,Português) \geq Nota(x,Português)\ )$	X
Nenhuma das anteriores	

c) Pesquisa por melhorias sucessivas é:

Uma técnica de pesquisa para resolução problemas de atribuição	
Uma técnica para combinação de heurísticas	
Uma técnica de pesquisa para optimização de soluções	X
Um caso particular de recozimento simulado em que a evolução da temperatura faz lembrar uma paisagem de montanhas	
Nenhuma das anteriores	

d) Uma consequência lógica do conjunto de fórmulas { $A \vee B, \neg B \vee C \vee D, \neg A, \neg D$} é:

$B \wedge A$	
C	X
A	
$A \vee D$	
Nenhuma das anteriores	

e) Os operadores STRIPS são:

Um formato de representação de acções para sistemas reactivos com estado interno	
Um formato de representação de transições de estados para pesquisa por melhorias sucessivas	
Mecanismos de modificação da solução em pesquisa por recozimento simulado	
Mecanismos para geração de planos no mundo dos blocos	
Nenhuma das anteriores	X

2. Identifique semelhanças e diferenças entre a pesquisa em árvore em profundidade e a pesquisa por montanhismo.

A pesquisa em árvore em profundidade (Depth-first search, DFS) é um algoritmo de busca que explora um grafo ou árvore de maneira recursiva, seguindo ao máximo possível a ramificação de um nó até chegar a um nó folha, antes de retornar para o nó anterior e explorar outra ramificação. A pesquisa por montanhismo (Hill climbing) é um algoritmo de otimização que tenta encontrar a solução ótima de um problema através de uma sequência de melhorias sucessivas na solução atual.

As principais semelhanças entre a pesquisa em árvore em profundidade e a pesquisa por montanhismo são:

· Ambas são técnicas de pesquisa iterativas que visam encontrar a melhor solução possível para um problema.
· Ambas são usadas para resolver problemas de busca, otimização e planejamento.
· Ambas podem ser aplicadas a uma ampla variedade de problemas, incluindo problemas de Inteligência Artificial, ciência da computação e outras áreas.

As principais diferenças entre a pesquisa em árvore em profundidade e a pesquisa por montanhismo são:

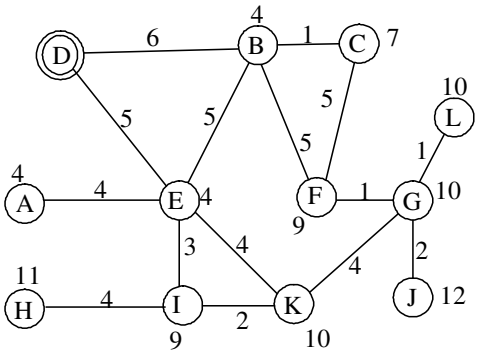
· A pesquisa em árvore em profundidade explora um grafo ou árvore de maneira recursiva, enquanto a pesquisa por montanhismo tenta encontrar a solução ótima através de uma sequência de melhorias sucessivas.
· A pesquisa em árvore em profundidade é usada principalmente para encontrar uma solução exata para um problema, enquanto a pesquisa por montanhismo é usada principalmente para encontrar uma solução aproximada.
· A pesquisa em árvore em profundidade pode ser aplicada a grafos ou árvores, enquanto a pesquisa por montanhismo é usada principalmente para problemas de otimização.

3. As casas têm divisões de diferentes tipos, por exemplo, salas de estar, salas de jantar, quartos de dormir, cozinhas e quartos de banho. As divisões da casa têm peças de mobiliário, como por exemplo, mesas, cadeiras, camas, cómodas e estantes. A casa da Gabriela é em Aveiro. Essa casa tem um quarto com uma cama, em que a Gabriela dorme, e uma cómoda. Represente este conhecimento através de uma rede semântica.

[Casas] <-- [têm divisões de diferentes tipos] --> [Divisões da casa]
<-- [têm peças de mobiliário] --> [Peças de mobiliário]
<-- [é em] --> [Aveiro]
[Gabriela] <-- [é dona da casa] --> [Casas]
[Quarto] <-- [é uma divisão da casa] --> [Divisões da casa]
<-- [tem] --> [Cama]
<-- [tem] --> [Cómoda]
[Cama] <-- [é uma peça de mobiliário] --> [Peças de mobiliário]
<-- [é usada para dormir] --> [Dormir]
[Cómoda] <-- [é uma peça de mobiliário] --> [Peças de mobiliário]
<-- [é usada para armazenar objetos] --> [Armazenar objetos]

Nesta representação, as casas são representadas como um conceito geral, que pode ter diferentes divisões e peças de mobiliário. A casa da Gabriela é uma casa específica, localizada em Aveiro, e tem um quarto com uma cama e uma cómoda. A cama e a cómoda são peças de mobiliário que são usadas, respectivamente, para dormir e armazenar objetos.

4. O grafo a seguir apresentado representa um espaço de estados num problema de pesquisa, sendo **D** o estado objectivo (solução). As estimativas do custo de chegar à solução a partir de cada estado estão anotadas junto aos mesmos. Os custos das transições estão anotados junto às ligações do grafo.



a) Verifique se as estimativas de custo anotadas junto a cada nó constituem uma heurística admissível para a pesquisa A*. Se não for esse o caso, introduza (na própria figura) alterações que a tornem admissível. Justifique.

Para uma heurística ser admissível em um problema de pesquisa, ela deve satisfazer a propriedade de consistência, ou seja, para qualquer nó N e qualquer nó vizinho V de N, a diferença entre a estimativa de custo para chegar à solução a partir de V e a estimativa de custo para chegar à solução a partir de N deve ser menor ou igual ao custo da transição entre N e V.

No caso do grafo apresentado, podemos verificar que as estimativas de custo para os nós B, E e I são admissíveis, pois atendem à propriedade de consistência para todas as suas transições. No entanto, as estimativas de custo para os nós C, F e G não são admissíveis, pois não atendem à propriedade de consistência para todas as suas transições.

Continuação na última página

b) Tomando o estado **G** como estado inicial, apresente a árvore de pesquisa gerada quando se realiza uma pesquisa A* com repetição de estados. Numere os nós pela ordem em que são acrescentados à árvore e anote também o valor da função de avaliação em cada nó. Em caso de empate nos valores da função de avaliação em dois ou mais nós, utilize a ordem alfabética dos respectivos estados.

A árvore de pesquisa gerada pela pesquisa A* com repetição de estados, a partir do estado inicial G, seria a seguinte:

1. G (f = 9 + 1 = 10)
2. F (f = 8 + 1 = 9)
3. L (f = 10 + 1 = 11)
4. K (f = 10 + 4 = 14)
5. I (f = 9 + 2 = 11)
6. E (f = 4 + 3 = 7)
7. B (f = 4 + 5 = 9)
8. D (f = 6 + 6 = 12)

Observe que, na hora de expandir o nó G, escolhemos expandir primeiro o nó F, pois ele tem o menor valor da função de avaliação (f) entre os nós vizinhos de G. Na hora de expandir o nó F, escolhemos expandir primeiro o nó L, pois ele também tem o menor valor da função de avaliação entre os nós vizinhos de F. E assim por diante, até chegarmos ao nó D, que é o nó objetivo (solução).

Observe também que, na hora de expandir o nó K, tanto o nó I quanto o nó E têm o mesmo valor da função de avaliação (f = 11). Nesse caso, utilizamos a ordem alfabética para escolher qual dos dois nós expandir primeiro

5. Considere um veículo autónomo que se movimenta num ambiente estruturado em nós e ligações, ou seja, estruturado como um grafo. As ligações correspondem a ruas. Os nós representam confluências de uma ou mais ruas. Além disso, em cada nó pode haver 0 ou mais parques de estacionamento. As ruas começam e terminam em nós adjacentes no grafo. O agente é capaz de realizar as seguintes acções: atravessar (passar para outra rua do nó), estacionar num dos parques do mesmo nó, percorrer (seguir até ao nó no outro extremo da rua actual), sair do estacionamento para uma dada rua que começa ou termina no mesmo nó.

a) Identifique e caracterize um conjunto de predicados em lógica de primeira ordem que possam ser usados para especificar condições sobre estados de planeamento neste domínio. Identifique os valores possíveis dos argumentos desses predicados. (Nota: Para responder a esta pergunta, é aconselhável ver também a alínea b), onde estes predicados também são usados.)

Um conjunto de predicados em lógica de primeira ordem que pode ser usado para especificar condições sobre os estados de planeamento neste domínio inclui:

· Ponto(x): este predicado é verdadeiro se o nó x é um ponto de confluência de ruas.
· Rua(x): este predicado é verdadeiro se x é uma rua.
· NóInicial(x): este predicado é verdadeiro se o veículo está no nó x no início da viagem.

· NóAtual(x): este predicado é verdadeiro se o veículo está no nó x no momento atual.
· NóFinal(x): este predicado é verdadeiro se o veículo deve chegar ao nó x no final da viagem.
· RuaAtual(x): este predicado é verdadeiro se o veículo está na rua x no momento atual.
· RuaPara(x, y): este predicado é verdadeiro se a rua x leva ao nó y.
· Estacionamento(x): este predicado é verdadeiro se o nó x tem um parque de estacionamento.
· Estacionado(x): este predicado é verdadeiro se o veículo está estacionado no parque x.

Os valores possíveis dos argumentos destes predicados são:

· x: um nó ou uma rua
· y: um nó

b) Usando os predicados que propôs, defina um conjunto de operadores STRIPS para representar as acções que podem ser realizadas neste domínio.

Usando os predicados sugeridos na alínea a), um conjunto de operadores STRIPS para representar as ações que podem ser realizadas neste domínio pode ser definido da seguinte maneira:

Atravessar(x, y):

· Pré-condições: Ponto(x), Rua(y)
· Efeitos: NóAtual(y), RuaAtual(y)

Estacionar(x):
· Pré-condições: Estacionamento(x), NóAtual(x)
· Efeitos: Estacionado(x)

Percorrer(x, y):
· Pré-condições: Rua(x), NóAtual(x), RuaPara(x, y)
· Efeitos: NóAtual(y)

SairEstacionamento(x, y):
· Pré-condições: Estacionado(x), NóAtual(x), Rua(y), NóAtual(y)
· Efeitos: Estacionado(False), RuaAtual(y)

Estes operadores permitem ao agente atravessar para outra rua do nó atual, estacionar no parque de estacionamento do nó atual, percorrer a rua atual até ao nó no outro extremo e sair do estacionamento para uma dada rua que começa ou termina no mesmo nó.

CONTINUAÇÃO

4. a)
Para tornar a heurística admissível, podemos fazer as seguintes alterações:

No nó C, podemos alterar a estimativa de custo de 7 para 6, pois assim atenderíamos à propriedade de consistência para as transições C->B e C->F.

No nó F, podemos alterar a estimativa de custo de 9 para 8, pois assim atenderíamos à propriedade de consistência para as transições F->B e F->C.

No nó G, podemos alterar a estimativa de custo de 10 para 9, pois assim atenderíamos à propriedade de consistência para as transições G->F e G->L.

Após essas alterações, a heurística passa a ser admissível, pois atende à propriedade de consistência para todas as transições do grafo.

Alterado:

Estimativas do custo de chegar à solução:
A:4 B:4 C:6 E:4 F:8 G:9 H:11 I:9 J:12 K:10 L:10

Custos das transições:
A->E:4 B->D:6 B->E:5 B->F:5 B->C:1 C->B:1 C->F:5 D->B:6 D->E:5 E->D:5 E->B:5 E->A:4
E->I:3 E->K:4 F->B:5 F->C:5 F->G:1 G->F:1 G->L:1 G->K:4 G->J:2 H->I:4 I->H:4 I->E:3 I->K:2

Identificação do aluno:

Nome: _____ #: _____