

# Resolução Exame 2015

I

1. Analise as seguintes funções escritas em Python e explique o que fazem. Não precisa descrever o funcionamento interno das funções.

a)

```
def f(x, y, z):  
    if y == []:  
        return [x]  
  
    if z(x, y[0]):  
        return [x] + y  
  
    return [y[0]] + f(x, y[1:], z)
```

Todos os elementos de `y` são testados para a função `z()`. Se derem `false`, são adicionados à lista do resultado devolvido. Assim que algum elemento der `true` ou `y` estiver vazio, `x` é adicionado à lista do resultado devolvido e os restantes elementos de `y` são *appended* à lista, se existentes.

b)

```
def h(x, y, z):  
    if x == []:  
        return y[:]  
  
    if y == []:  
        return x[:]  
  
    if z(x[0]) < z(y[0]):  
        return [x[0]] + h(x[1:], y, z)  
  
    return [y[0]] + h(x, y[1:], z)
```

A função `z()` devolve um número.

1. `h([1,3,5], [-1, 2, 7], z)` devolve `[-1]` + o que deu no passo 2
2. `h([1,3,5], [2, 7], z)` devolve `[1]` + o que deu no passo 3
3. `h([3,5], [2, 7], z)` devolve `[2]` + o que deu no passo 4
4. `h([3,5], [7], z)` devolve `[3]` + o que deu no passo 5
5. `h([5], [7], z)` devolve `[5]` + o que deu no passo 6
6. `h([], [7], z)` devolve `[7]`.

portanto fica.

1. `h([1,3,5], [-1, 2, 7], z)` devolve `[-1] + [1, 2, 3, 5, 7]`
2. `h([1,3,5], [2, 7], z)` devolve `[1] + [2, 3, 5, 7]`
3. `h([3,5], [2, 7], z)` devolve `[2] + [3, 5, 7]`
4. `h([3,5], [7], z)` devolve `[3] + [5, 7]`
5. `h([5], [7], z)` devolve `[5] + [7]`
6. `h([], [7], z)` devolve `[7]`.

`h([1,3,5], [-1, 2, 7], z) = [-1, 1, 2, 3, 5, 7]`

A transformação do elemento inicial de cada lista é comparada e o resultado menor dessa transformação é adicionado à lista de resultados e esse valor descartado da próxima iteração.

**2. Para efeitos de implementação de redes de Bayes em Python, as probabilidades condicionadas podem ser representadas como tuplos `(Var, Mothers, Prob)`, em que `Var` é uma das variáveis da rede, `Mothers` é uma lista de tuplos representando uma das combinações possíveis de valores das variáveis mães de `Var`, e `Prob` é a probabilidade condicionada de `Var` dado `Mother`):** Programe uma função que, dada uma lista de tuplos representando todas as probabilidades condicionadas de uma rede, e dada ainda uma determinada variável da rede, retorna uma lista com todas as variáveis ascendentes dessa variável. Exemplo:

```
bn = [ ( "C", [ ("A",True), ("B",True) ], 0.95), ( "C", [ ('A',True), ("B",False)],
0.7), ("C", [("A", False), ("B", True)], 0.65), ("C", [("A", False), ("B", False)],
0.1), ("D", [("C", True)], 0.77), ("D", [("C", False)], 0.22), ("B", [], 0.33)]

>>> get_ancestors(bn, "D")
["A", "B", "C"]
```

```
def get_ancestors(bn, event):
    ancestors = []

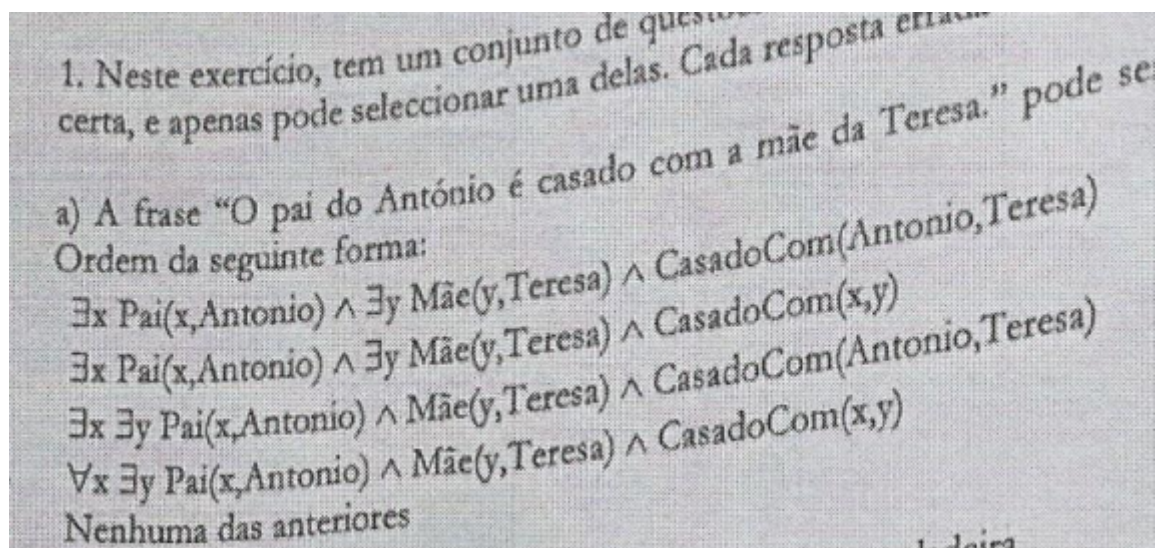
    for i in bn:
        if i[0] == event:
            for j in i[1]:
                ancestors.append(j[0])
                ancestors.extend(get_ancestors(bn, j[0]))

    return list(set(ancestors))
```

II

1.

a)



$$\exists x (P(x) \wedge Q(x)) \Rightarrow \exists x P(x) \wedge \exists x Q(x)$$

R: b)

b)



Nenhuma das anteriores

b) Relativamente às redes de Bayes, indique a afirmação verdadeira

Não permitem representar conhecimento impreciso  
 São representadas por grafos não dirigidos  
 Permitem representar as dependências entre as variáveis de um problema  
 Permitem representar relações de herança entre entidades  
 Nenhuma das anteriores

R: c)

c)

c) Relativamente às redes semânticas, indique a afirmação verdadeira

Não é possível representar hierarquias de tipos  
 Permitem representar conhecimento por omissão  
 Nenhum tipo de rede semântica permite representar a negação ou a disjunção  
 A rede semântica estudada na componente prática da disciplina é tão expressiva como a lógica de primeira ordem  
 Nenhuma das anteriores

R: b)

d)

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

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

Substituir virgulas por conjunções.

R: e)



e)

e) O algoritmo de pesquisa em grafo (*graph-search*) difere do algoritmo de pesquisa em árvore (*tree-search*) em c

A pesquisa em grafo utiliza-se transições sempre na forma de operadores STRIPS

A pesquisa em grafo trabalha com um grafo de restrições

A pesquisa em grafo não cria nós com estados repetidos no caminho de cada nó até à raiz

A pesquisa em grafo é completa e óptima

Nenhuma das anteriores

R: c) ou b) idk

2.

2. Está a ser desenvolvido um novo robô aspirador, sendo necessário implementar o respectivo algoritmo de controlo. Um pressuposto do algoritmo é que o espaço a limpar está organizado na forma de uma grelha de células quadradas. O aspirador consegue determinar se existe lixo para aspirar na célula em que ele está, bem como nas células vizinhas (*aqui, em frente, à esquerda, à direita e atrás*). As acções que o aspirador consegue executar são: aspirar o lixo na célula actual; mover-se para a célula em frente; e rodar 90° para a direita. Sempre que detecta lixo na vizinhança, o robô move-se para a célula em que está o lixo. No caso de não detectar lixo, o aspirador move-se para a célula em frente caso o tempo actual (em segundos) seja par, ou roda 90° para a direita caso contrário. Não precisa de preocupar-se com obstáculos, já que a acção de mover é automaticamente omitida caso haja um obstáculo em frente.

a) Identifique e caracterize as varias condições (predicados/proposições) que podem ser usadas para descrever as situações em que se pode encontrar o robo aspirador.

lixo\_na\_vizinhança(x), x -> direção  
lixo\_na\_atual()  
tempo\_par()

b) Especifique um conjunto de regras situação-ação que definam um comportamento adequado do robo aspirador. Pode fazê-lo na forma de uma tabela com as seguintes colunas:

- Situação - uma conjunção de condições
- Atualização - Atualização das variaveis de estado, caso existam
- Ação - ação a executar pelo agente na situação indicada

Situação	Atualização	Ação
----------	-------------	------

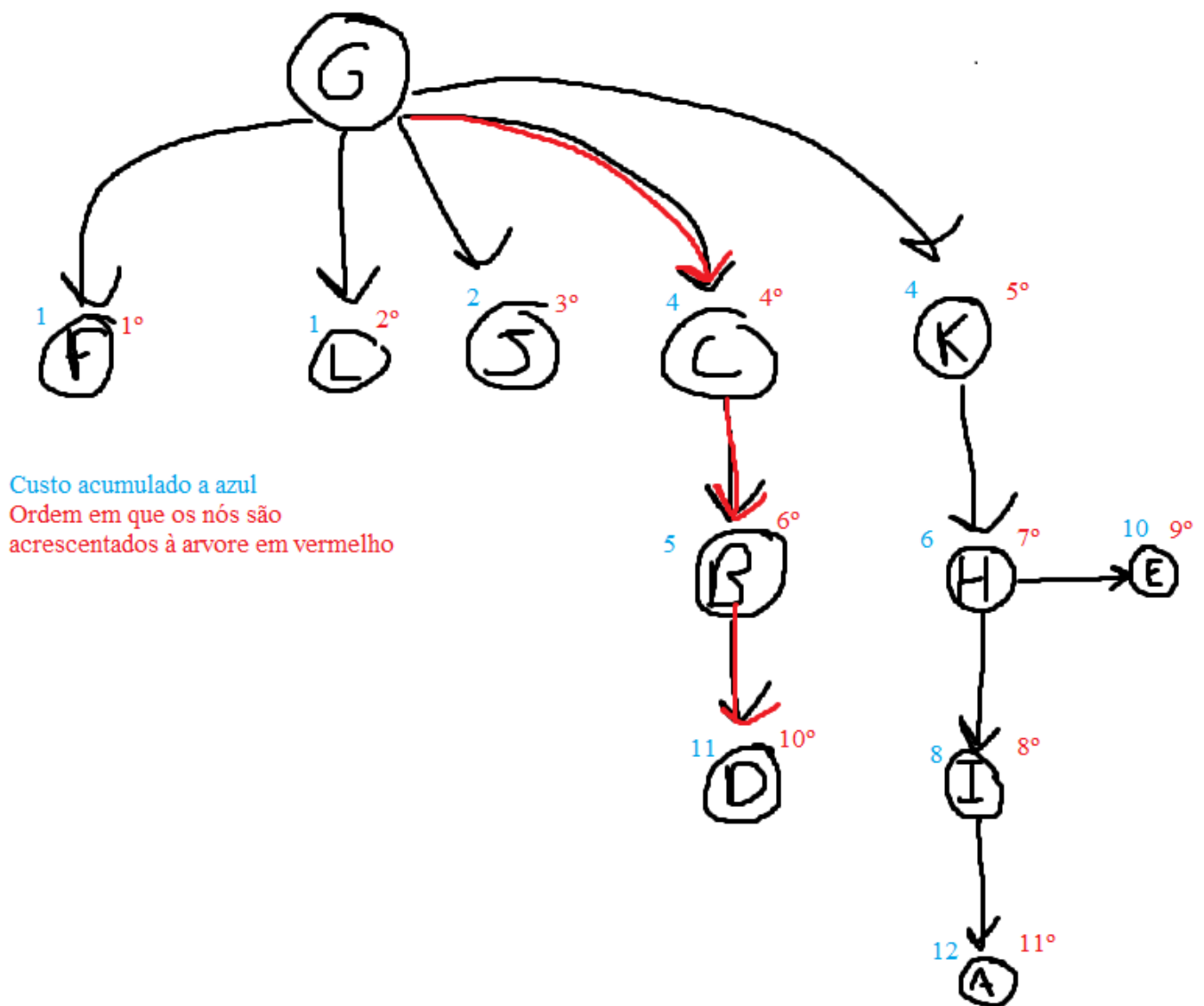
Situação	Atualização	Ação
Existe lixo na célula atual: <code>lixo_na_atual()</code>	<code>!lixo_na_atual()</code>	Aspirar
Existe lixo na proximidade, excluindo célula atual: <code>lixo_na_vizinhança(x)</code>	-	Mover na direção de x
Não há lixo na proximidade, tempo par: <code>!lixo_na_vizinhança(x) E tempo_par()</code>	-	Mover em frente
Não há lixo na proximidade, tempo ímpar: <code>!lixo_na_vizinhança(x) E !tempo_par()</code>	-	rodar 90° para a direita

3.

a)

3. O grafo a seguir apresentado representa um espaço de estados num problema de pesquisa, sendo D o estado objectivo (solução). Os custos das transições estão anotados junto às ligações do grafo.

a) Tomando o estado G como estado inicial, apresente a árvore de pesquisa gerada quando se realiza uma pesquisa de custo uniforme. Esta pesquisa é feita sem repetição de estados no caminho de qualquer nó até à raiz da árvore. 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 desempate com base na ordem alfabética dos respectivos estados.



pode tar mal

**b) Calcule o fator de ramificação médio da árvore gerada.**

**ramificação média** indica nos a dificuldade do problema

N - número de nós resultantes da expansão (nós filho)

X - número de nós expandidos (mesmo que não tenham gerado filhos)

$$RM = \frac{N - 1}{X}$$

N = 11 e X = 11

RM = (N - 1) / X = 1?

pode tar mal

### c) Identifique semelhanças e diferenças entre a pesquisa em largura e a pesquisa de custo uniforme.

- A pesquisa em largura consiste em avaliar todos os nós de um determinado nível antes de prosseguir para a avaliação dos nós do próximo nível. É completo e ótimo.
- A pesquisa de custo uniforme, apesar de ser um caso particular da A\*, é idêntica à pesquisa em largura. Invés de começar pelo primeiro nó expandido, que está na lista aguardando processamento, o nó que possui o menor custo será escolhido para ser expandido. Caso exista, solução, a primeira encontrada é ótimo.

## 4.

4. Considere o seguinte problema:  
“André, Bernardo e Cláudio dão um passeio de bicicleta. Cada um anda na bicicleta de um dos amigos e leva o chapéu de um dos outros. O que leva o chapéu de Cláudio anda na bicicleta de Bernardo. Que bicicleta e que chapéu levam cada um dos amigos?”  
Com vista à resolução do problema através de pesquisa com propagação de restrições, identifique as variáveis e respectivos valores possíveis, e represente a informação disponível através de um grafo de restrições.

Amanasdiaojsoyk+pdl'+ç'°c

x