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` sao 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 transformaçao 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 proxima iteraçao.

2. Para efeitos de implementaçao de redes de Bayes em Python, as probabilidades condicionadas podem ser representadas como tuplos `(Var, Mothers, Prob)`, em que Var é uma das variaveis da rede, `Mothers` é uma lista de tuplos representando uma das combinaçoes possiveis de valores das variaveis maes de `Var`, e `Prob` é a probabilidade condicionada de `Var` dado `Mother`): Programe uma funçao que, dada uma lista de tuplos representando todas as probabilidades condicionadas de uma rede, e dada ainda uma determinada variavel da rede, retorna uma lista com todas as variaveis ascendentes dessa variavel. Exemplo:

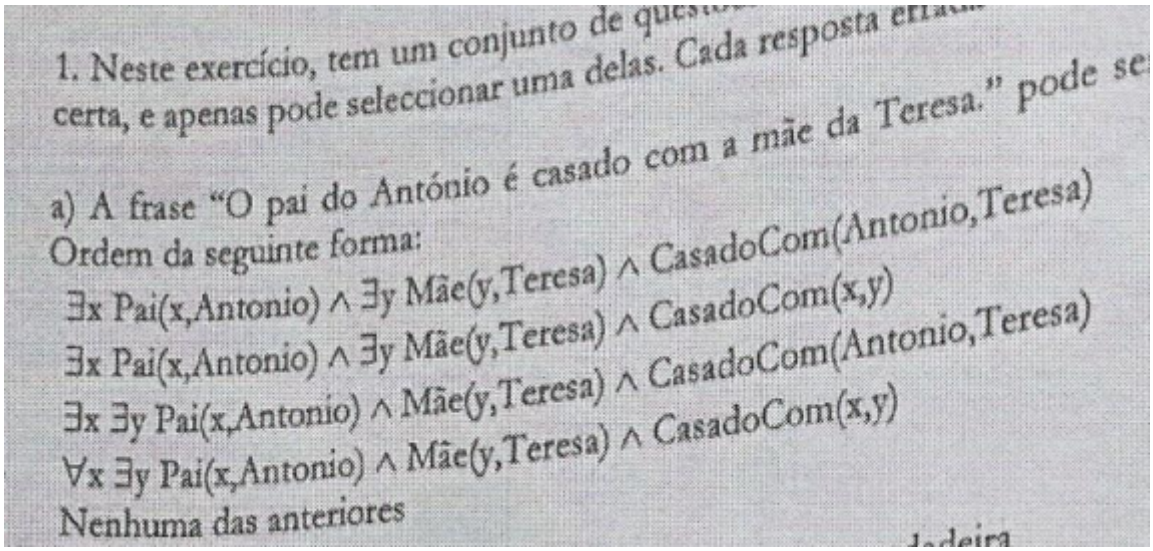
```
bn = [ ( "C", [ ("A",True), ("B",True) ], 0.95), ( "C", [('A',True), ("B",False)], 0.7), ("C", [("A", False), ("B", True)], 0.65), ("C", [("A", False), ("B", False)], 0.1), ("D", [("C", True)], 0.77), ("D", [("C", False)], 0.22), ("B", [], 0.33)]
>>> get_ancestors(bn, "D")
["A", "B", "C"]
```

```python
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.

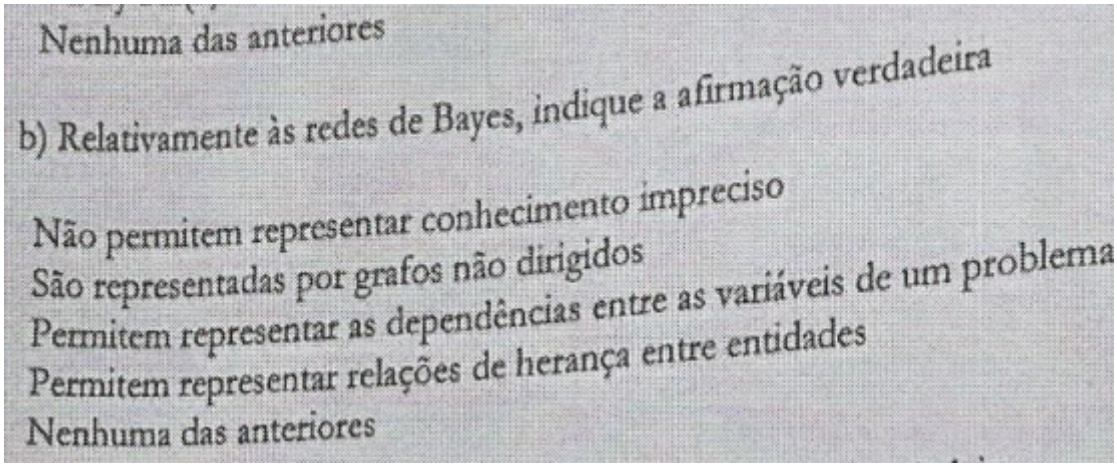
a)



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

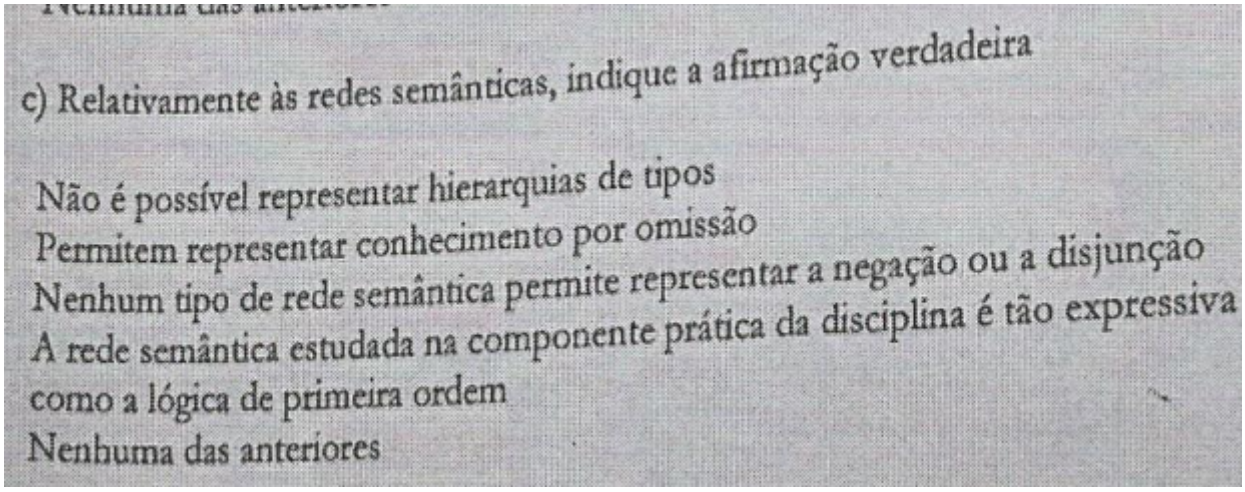
R: b)

b)



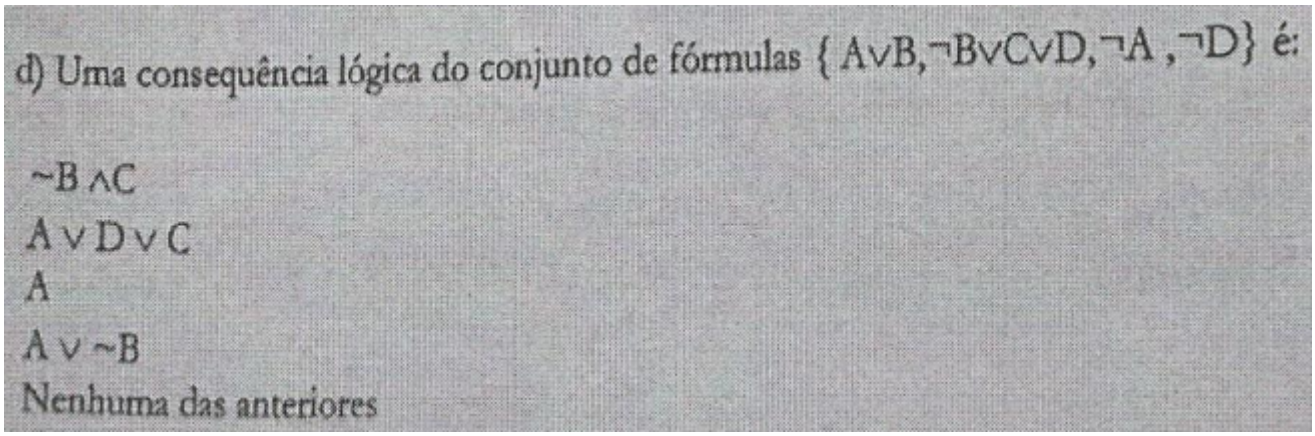
R: c)

c)



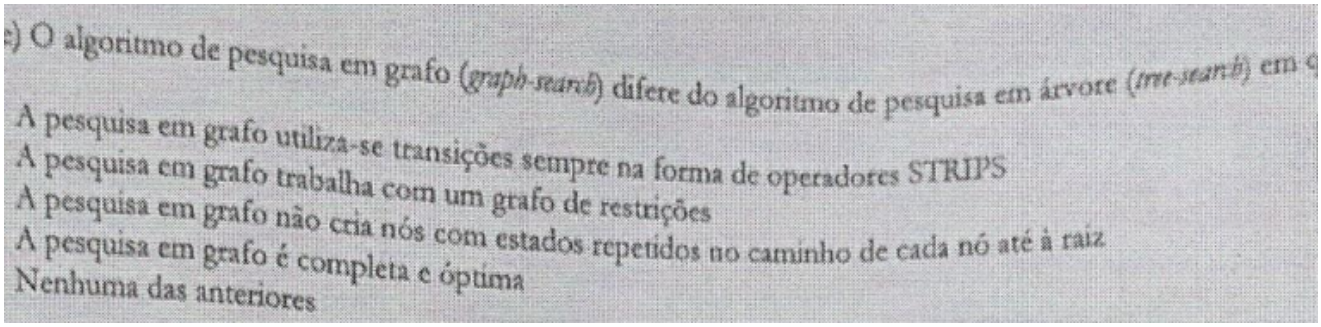
R: b)

d)



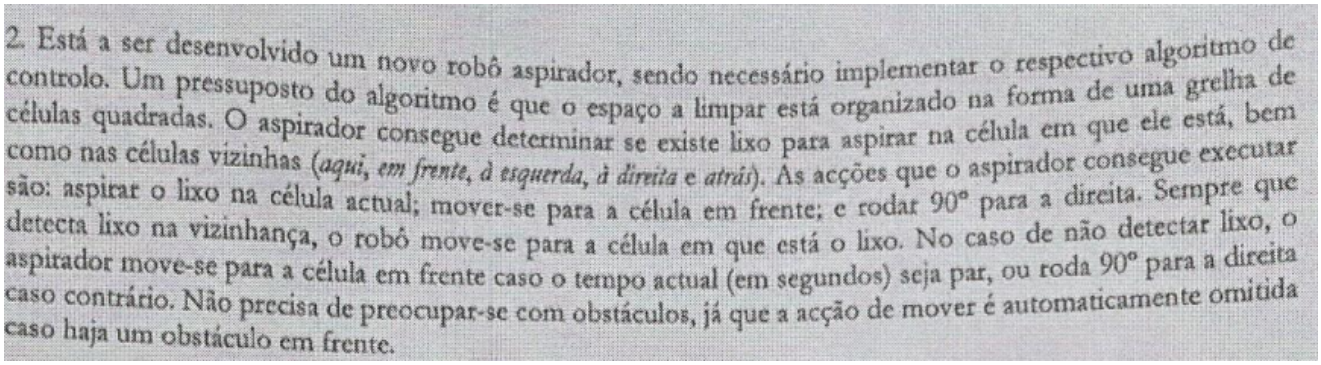
Substituir virgulas por conjunções.
R: e)

e)



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 90º 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 90º 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 condiçoes (predicados/proposiçoes) que podem ser usadas para descrever as situaçoes 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:

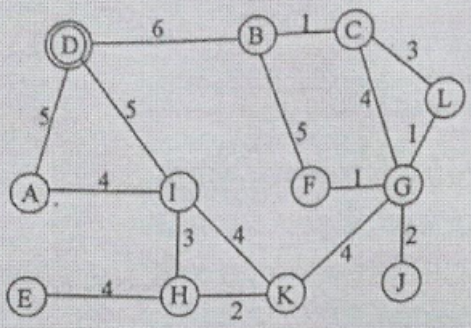
- Situação - uma conjunção de condições
- Atualização - Atualização das variaveis de estado, caso existam
- Ação - ação a executar pelo agente na situação indicada

Situação	Atualização	Ação
Existe lixo na celula atual: `lixo_na_atual()`	! `lixo_na_atual()`	Aspirar
Existe lixo na proximidade, excluindo celula atual: `lixo_na_vizinhança(x)`	-	Mover na direção de x
Não há lixo na proximidade, tempo par: `!lixo_na_vizinhança(x) E tempo_par()`	-	Mover em frente
Não há lixo na proximidade, tempo impar: `!lixo_na_vizinhança(x) E !tempo_par()`	-	rodar 90º para a direita

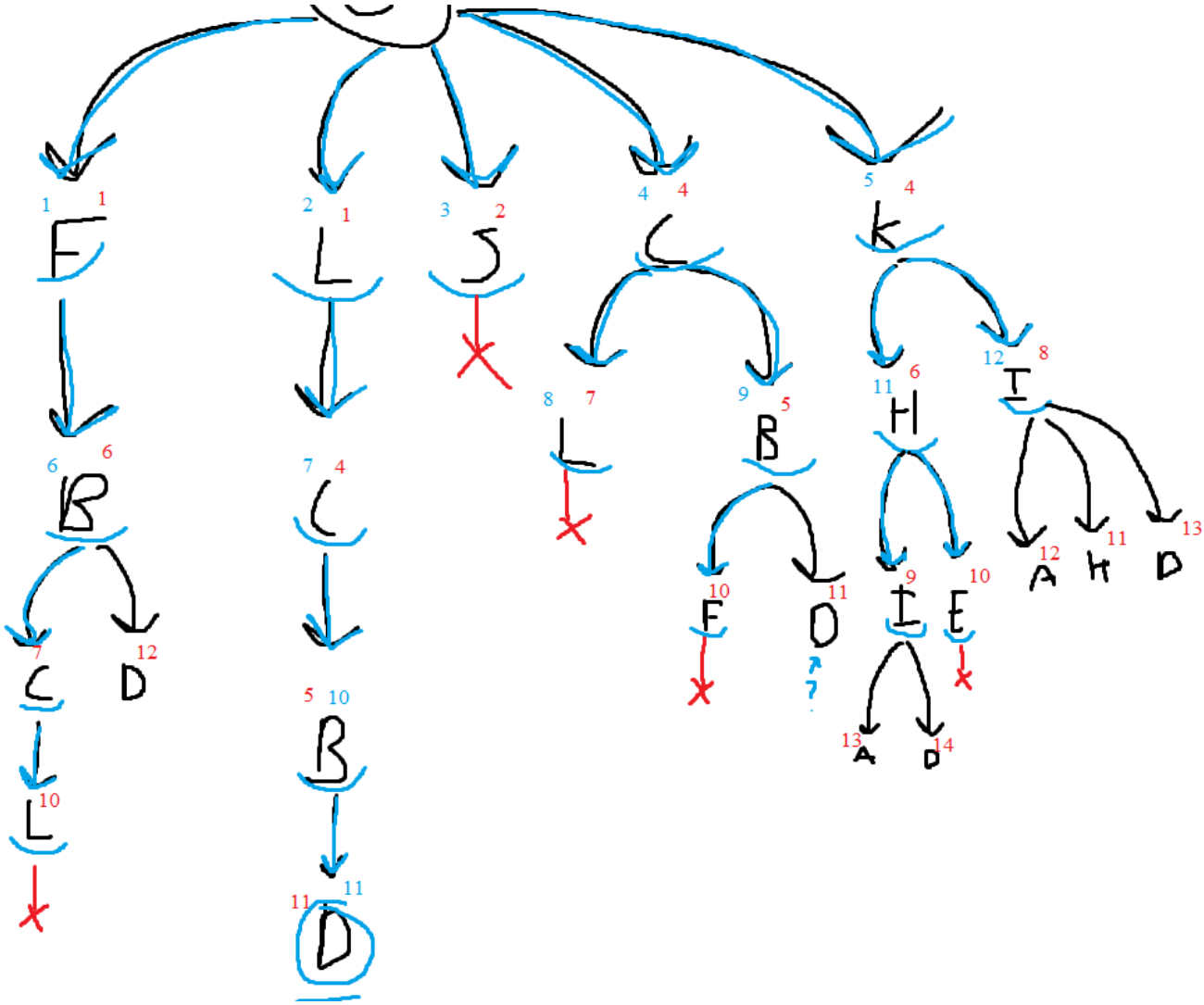
3.

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.



A ordem de inserçao na arvore pode está incompleta.

b) Calcule o fator de ramificação médio da arvore gerada.

ramificação média indica nos a dificuldade do problema
N - número de nós da árvore de pesquisa no momento em que se encontra a solução
X – Número de nós expandidos (não terminais)

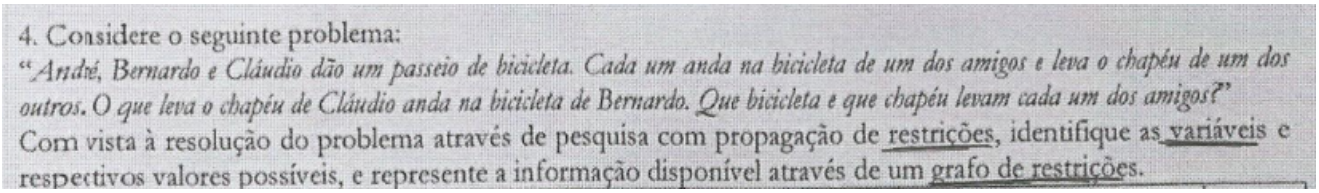
N = 26
X = 13

$$RM = \frac{N-1}{X} = \frac{26-1}{13} = 1.92$$

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 identica 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.

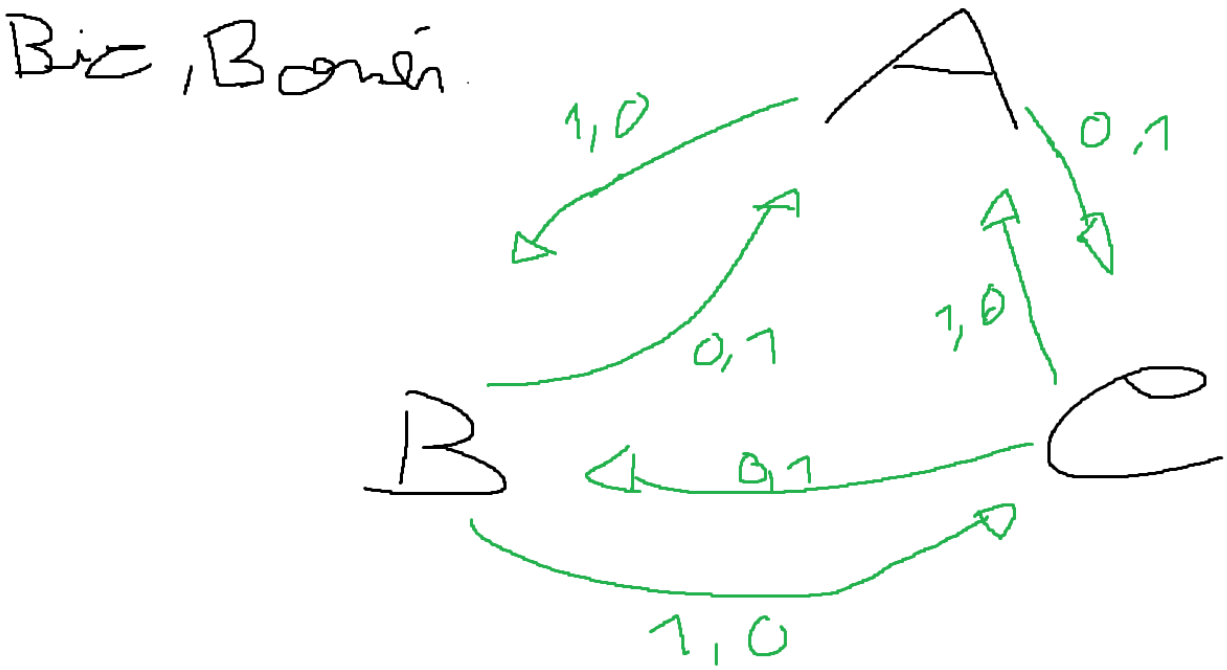

4. Considere o seguinte problema:
"*André, Bernardo e Cláudio dão um passeio de bicicleta. Cada um anda na bicicleta de um dos amigos e leva o chapéu de um dos outros. O que leva o chapéu de Cláudio anda na bicicleta de Bernardo. Que bicicleta e que chapéu levam cada um dos amigos?*"
Com vista à resolução 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.

Restrições:

- André = {Chapeu André, Bicicleta André}
- Bernardo = {Chapeu Bernardo, Bicicleta Bernardo}
- Claudio = {Chapeu Claudio, Bicicleta Claudio}

Variaveis:

- Chapeu utilizado pelo André = {}
- Bicicleta utilizado pelo André = {}
- Chapeu utilizado pelo Bernardo = {}
- Bicicleta utilizado pelo Bernardo = {}
- Chapeu utilizado pelo Claudio = {}
- Bicicleta utilizado pelo Claudio = {}



nsei fds