

Teste Modelo

14 January 2020 14:11

① → a) Soma todos os números positivos de uma lista.

b) $[1, 2] \quad [2] \quad []$

$$\begin{aligned} & y = g([2]) \quad y = g([]) \quad \text{return } [[y]] \\ & y = [[1]] \\ & \text{return } \underbrace{[[1]] + [2]}_{[[1], [2]]} \\ & \text{return } [[1], [2], [1], [2]] \end{aligned}$$

Devolve todas as combinações possíveis com zero ou mais elementos da lista
(a ordem não importa)

c) Verifica se todos os elementos da lista se verificam a condição avaliada pelo função y

② → a) def interpretações (lst):

```
if lst == []:
    return []
y = interpretações (lst[1:])
return [[(lst[0], a)] + z for z in y for a in [True, False]]
```

b) def provar (f, p):

```
if [p] in f:
    return True
# n → negar
n = [x for x in f if len(x) > 1]
## negsp → negar que provam p
negsp = [x for x in n if x[0] == p]
if negsp == []:
    return False
return any([all([provar(f, y) for y in x[1:]]) for x in negsp])
```

c) def bfs_update_open_lst (open, new):

```
return open + new
# assumindo que o nó expandido já foi retirado de 'open'
```

Parte 2

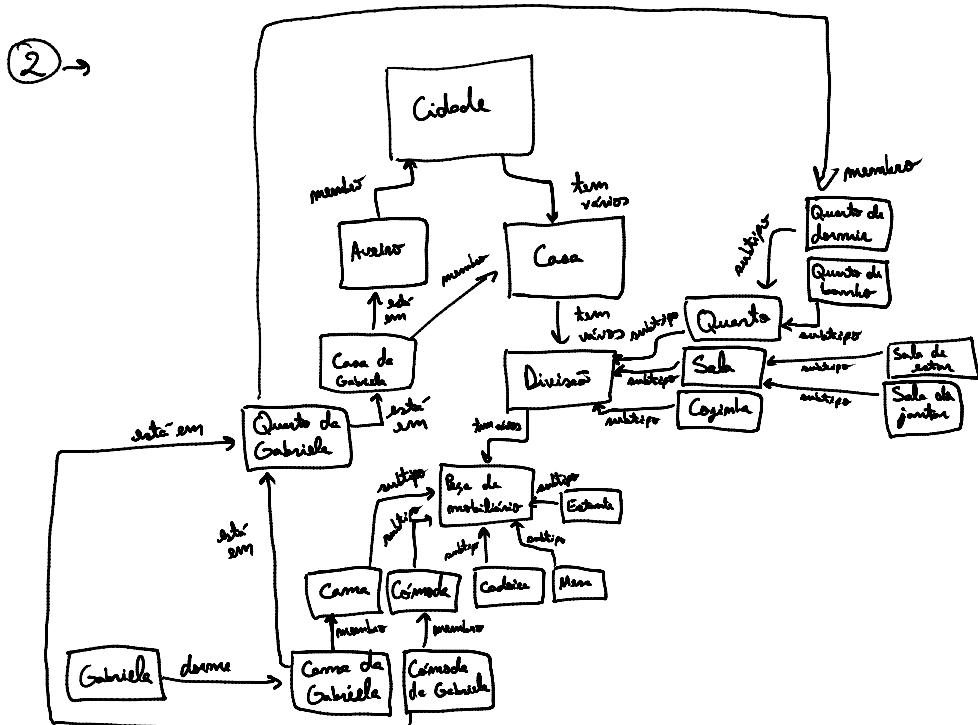
① → Após ter expandido um nó a pesquisa por montanhismo, tal como a pesquisa em profundidade, escolhe para a próxima expansão um dos nós resultantes da expansão anterior ao invés de um nó da lista de nós abertos.

1) Passo 1: No topo da árvore, o nó raiz é a Cidade. A pesquisa em profundidade escolhe para a próxima expansão um dos nós resultantes da expansão anterior ao nível de um nó da lista de nós abertos.

No entanto, enquanto que a pesquisa em profundidade escolhe o primeiro desses nós, a pesquisa por montanhismo seleciona o nó com maior valor de uma dada função heurística.

Para além disso, enquanto que a pesquisa em profundidade seleciona um nó da lista de nós abertos se o último nó expandido não tiver sido a solução nem tiver tido nós filhos; a pesquisa por montanhismo termina, já que o 'backtracking' não é permitido.

O mesmo acontece se nenhum dos nós filho tiver maior ou igual valor de heurística do que o nó não-expandido.



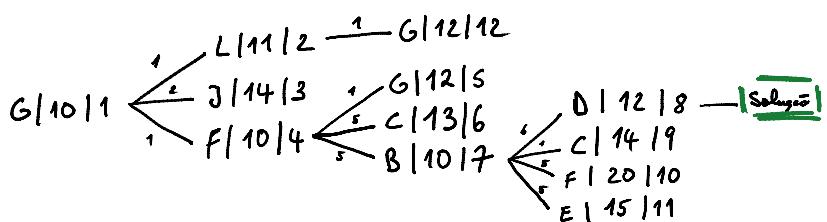
3) a) Para uma heurística ser considerada admissível esta nunca pode sobreestimar o custo real.

Os seguintes nós têm valor de heurística superior ao custo real para o destino:

I, K. Logo a sua heurística não é admissível.

Para corrigir esta situação o custo E-I aumenta para 4 e o custo E-K para 5.

b) Legenda: N° | tempo de abertura | Índice da ordem
em que foi acidentado



c) $\frac{B^{d+1} - 1}{-} = N$

$$c) \frac{B^{d+1} - 1}{B - 1} = N$$

$$d = 3$$

$$N = 12$$

$$B = ? \leftarrow \text{fator de normalização efetivo}$$

$$\text{Se } B = 2, N = 15 \times$$

$$\text{Se } B = 1, N = \frac{12}{2} \times$$

$$\text{Se } B = 1,8, N = 11,87 \checkmark$$

$$\underline{\underline{B \approx 1,8}}$$

- ④ → a) estacionado(y) → Veículo está estacionado num parque do nó y
 ma_rua(x, y) → Veículo está na rua x do nó y
 term_estacionamento(y) → Nó y possui 1 ou mais estacionamentos livres
 term_rua(x, y) → Nó y possui o fim da rua x

b) atravessar(x, y, z)

Pré-condições: ma_rua(x, z), term_rua(y, z),

Efeitos negativos: ma_rua(x, z)

Efeitos positivos: ma_rua(y, z)

estacionar(x, y)

Pré-condições: term_estacionamento(y), ma_rua(x, y)

Efeitos negativos: ma_rua(x, y)

Efeitos positivos: estacionado(y)

percorrer(x, y, z)

Pré-condições: ma_rua(x, y), term_rua(x, z)

Efeitos negativos: ma_rua(x, y)

Efeitos positivos: ma_rua(x, z)

sair(x, y)

Pré-condições: estacionado(y), term_rua(x, y)

Efeitos negativos: estacionado(y)

Efeitos positivos: ma_rua(x, y)