Lista - Busca não informada

Questão 01.

Objetivo: a jarra de capacidade de 4 litros conter exatamente 2 litros.

Estados: (J_1, J_2) -> par contendo a quantidade de J_1 e de J_2.
(0, 0) -> estado inicial. Ambas vazias
(2, –) -> estado final. Uma jarra com 2 litros e a outra com qualquer quantidade (0, 1, 2 ou 3)

Regras: * encher uma jarra completamente
* esvaziar uma jarra completamente
* passar a água de uma jarra para a outra
} ações

Custo: cada ação tem custo 1, soma das ações

Questão 02.

a) Objetivo: Fazer com que todos os elementos cruzem o rio, sem que nenhum seja comido.

Estados: (Você, Barco, Couve, Cabra, Lobo).
↳ contém a posição de cada um desses elementos. Os valores que as variáveis podem receber são ou "direita" ou "esquerda", que representam cada árvore.

(esquerda, esquerda, esquerda, esquerda, esquerda)
↳ estado inicial, onde todos estão na margem esquerda.

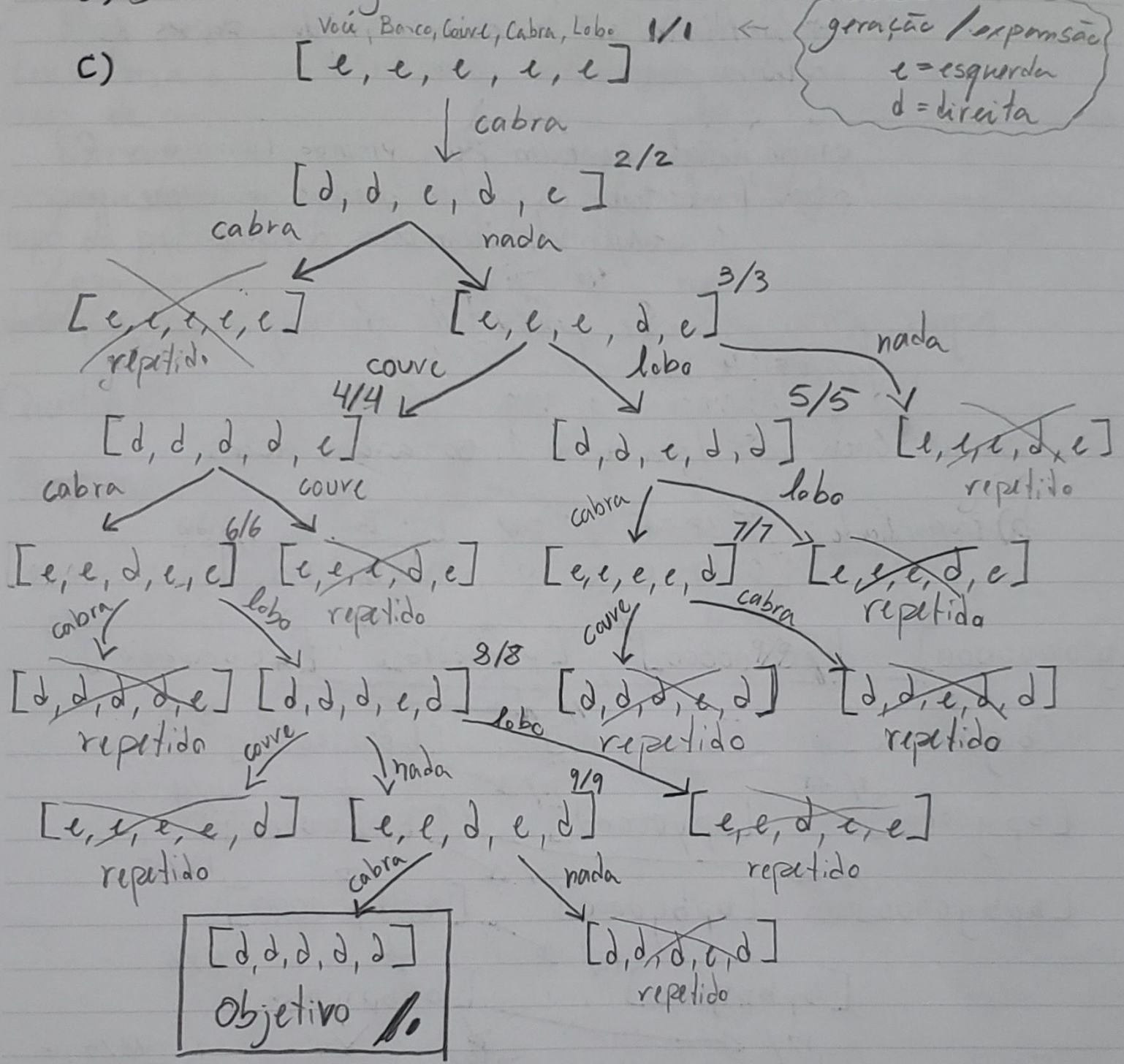
(direita, direita, direita, direita, direita)
↳ estado final, onde todos cruzaram a margem

Regras: * levar elemento no barco (podendo não levar nada) desde que Você esteja na mesma margem que a Cabra, ou que a Cabra esteja sozinha.

Custo: cada "levar elemento no barco" custa 1. Somatório das vezes que essa ação for executada.

b) Busca em largura

c)



d) Solução: levar cabra → trazer nada → levar couve → trazer cabra → levar lobo → trazer nada → levar cabra //

Questão 03:

a) Objetivo: Colorir países adjacentes de cores diferentes e usar até 4 cores.

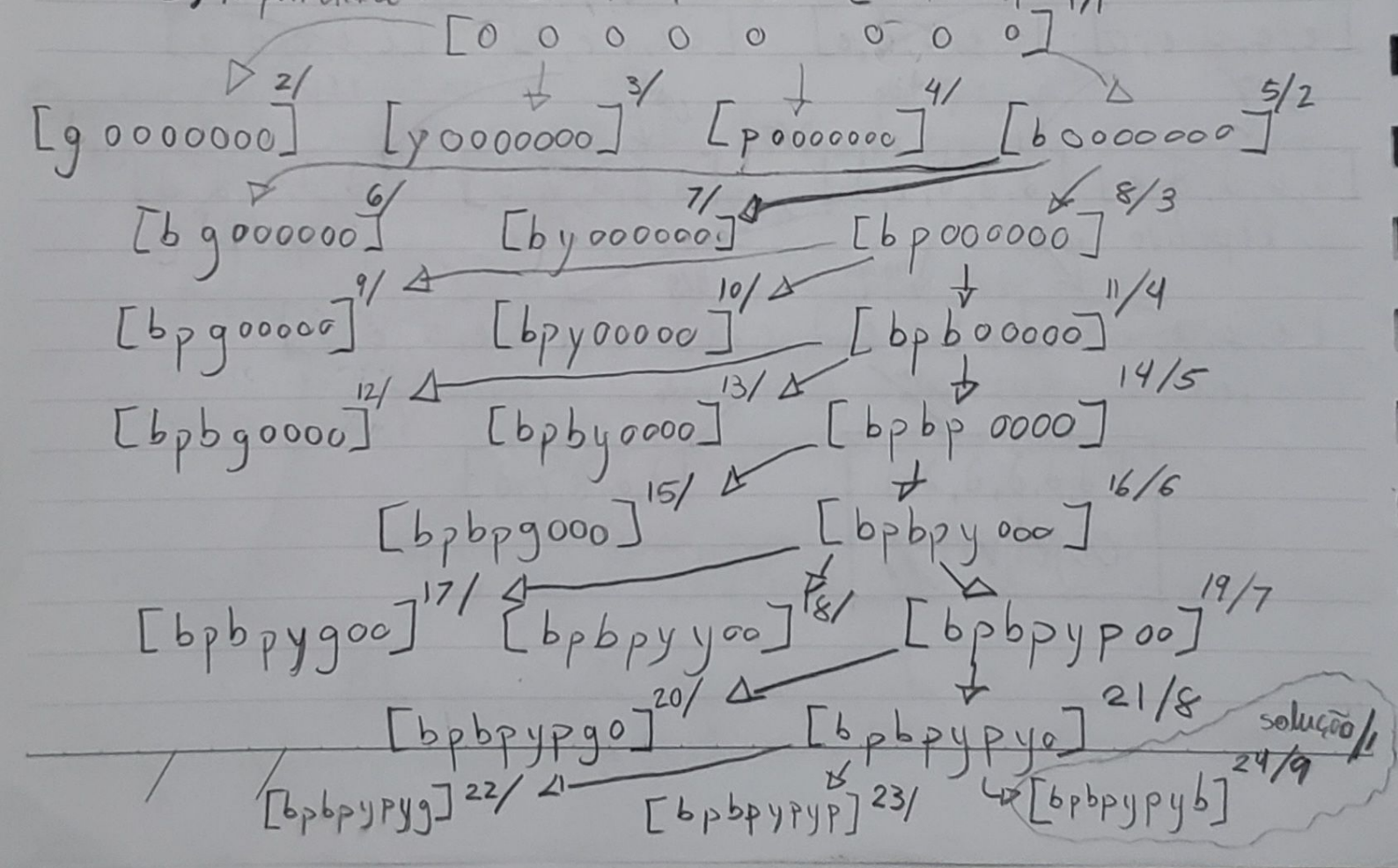
Estados: (País1, País2, País3, ...)
↳ contém a cor de cada um dos países do problema, sendo elas no máximo 4

estado inicial: nenhum país pintado (denotado por 0)
estado final: todos os países pintados, desde que países adjacentes tenham cores diferentes

Regras: pintar país. Se país tiver um adjacente já pintado pintar de nova cor.

Custo: cada ação tem custo 1. soma das ações

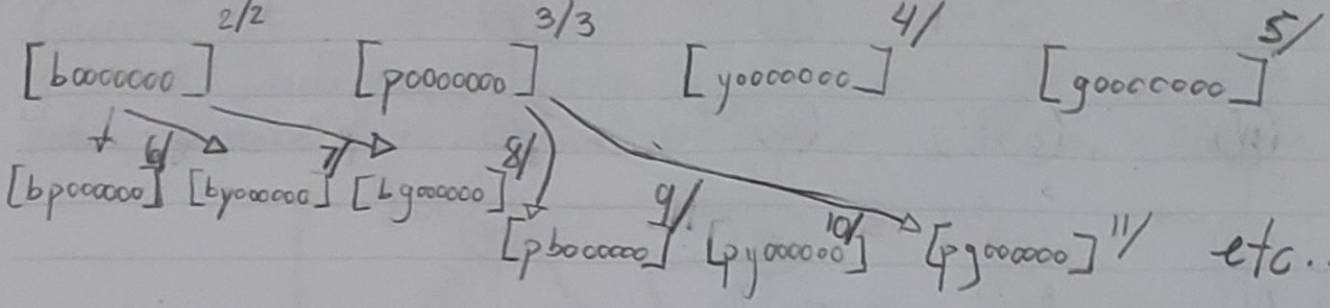
b) Profundidade: P SP F i SW L B H



largura:

P	SP	F	i	SW	L	B	H
0	0	0	0	0	0	0	0

1/1



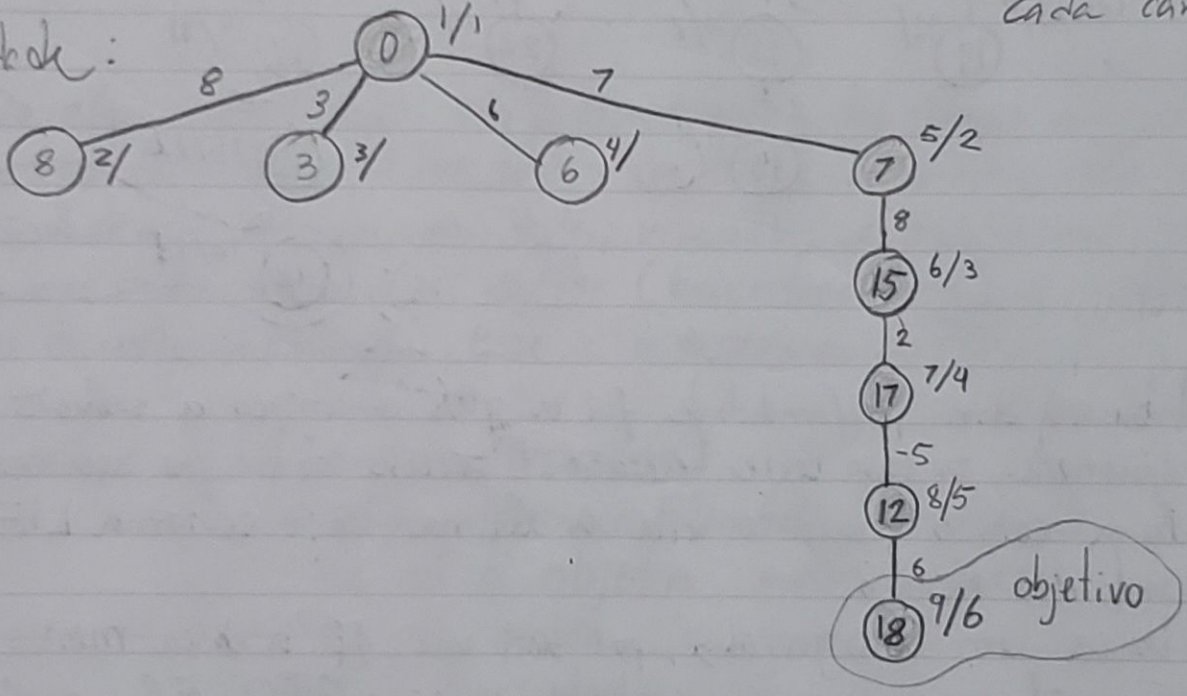
Nesse problema em específico, a busca em largura é muito trabalhosa e muito grande. Ela explora estados de todas as cores de um país antes, e aí sim passa para o próximo país.

Em contraposto, a busca em profundidade decide uma cor para todos os países, mas só explora o último, por estar no topo da pilha. Dessa forma, muito menos estados são gerados.

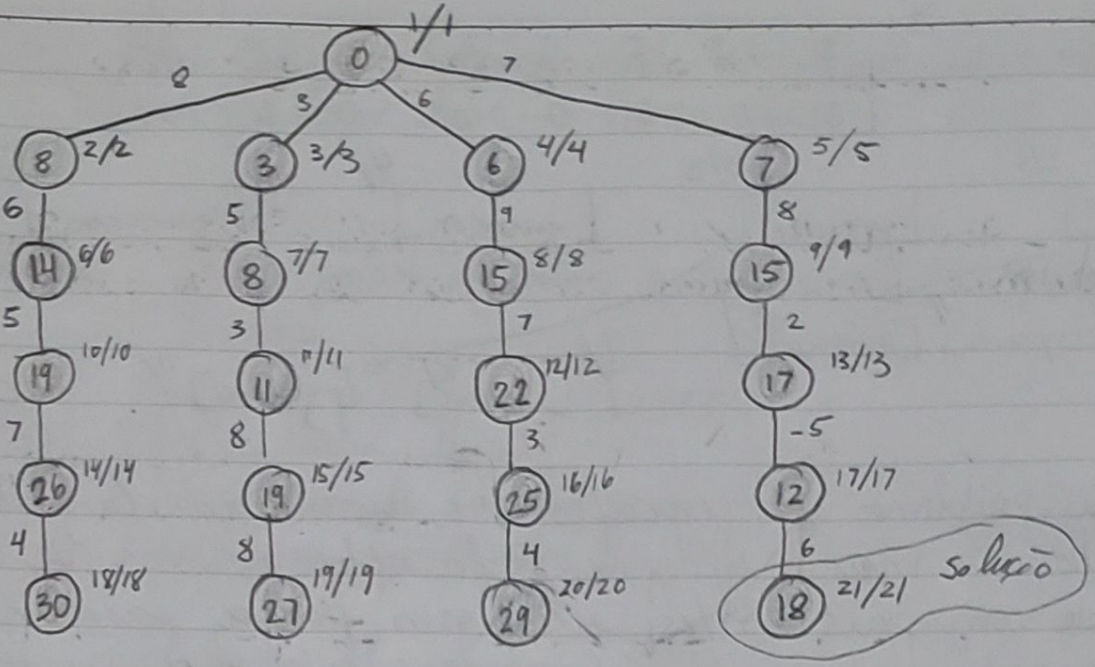
Logo, a busca em profundidade é mais eficiente nesse problema específico.

Questão 04: Cada nó representará a soma atual, logo, o custo de cada caminho.

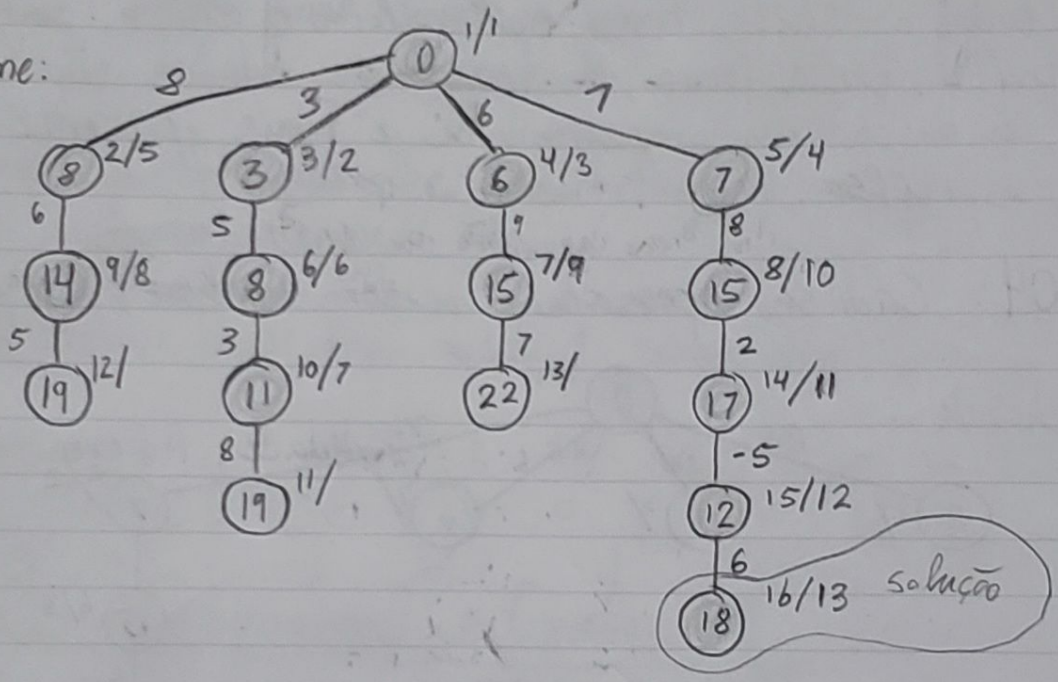
Profundidade:



Largura:



Custo uniforme:



A busca em profundidade foi a que encontrou a solução de maneira mais eficiente, no caso deste exercício. É preciso notar que isso ocorreu porque a coluna com o resultado era a última, e o sistema LiFO propiciou esse encontro rápido.

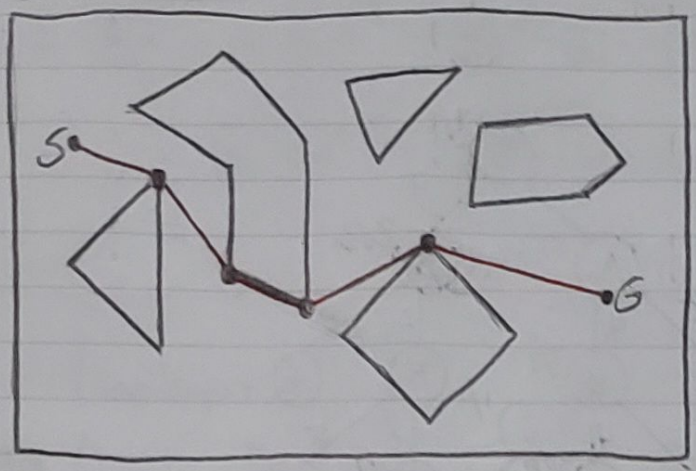
A busca em largura, por sua vez, foi a busca menos eficiente, também influenciada por seu sistema, no caso FiFO. Ela explora todos os nós de cada nível, e, além disso, a solução estava no último caminho.

Já a busca em custo uniforme encontrou a solução medianamente eficientemente. Alguns nós "desnecessários" foram explorados, mas em casos gerais isso garante um melhor balanceamento da exploração. Também é importante ressaltar que, neste caso, haver um custo negativo

não foi um problema, mas nem sempre é o caso.

Questão 05:

a) Dado que o menor caminho entre dois pontos é uma reta, o robô deve seguir em direção a G da seguinte maneira:



Então, sendo S o ponto inicial e G o ponto final, o conjunto de pontos mínimo é o da figura à esquerda e contém 6 pontos.

b) O sucessor de um ponto é o ponto mais próximo dele, desde que esteja caminhando na direção do ponto objetivo.
Os pontos sucessores de S estão marcados na letra a).

Questão 06:

a) Do algoritmo de busca em profundidade. Ao chegar em uma célula, em vez de ele explorar todas as suas vizinhas, ele explora uma vizinha, e depois a vizinha desta vizinha, e assim vai, até chegar em um ponto morto e aí voltar (backtracking) e explorar outra vizinha de alguma célula. Esse é o sistema LiFO.

b) função: dfs (fronteira, marcados) ← remove da fronteira

```
nó ← primeiroElemento (fronteira)
se nó é objetivo, então return
se não, então
    marcados ← nó
    vizinhos ← achaVizinhos (nó)
    novaFronteira ← [vizinhos, fronteira]
    dfs (novaFronteira, marcados)
```

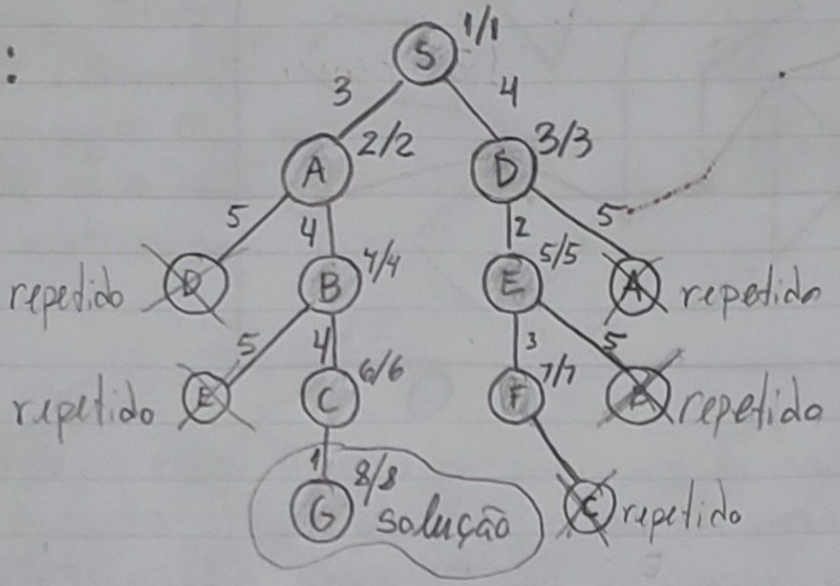
Questão 07. Máximo = $b^0 + b^1 + b^2 + \dots + b^d$.

no caso de expandir todos os nós do nível d antes de procurar checar a solução.

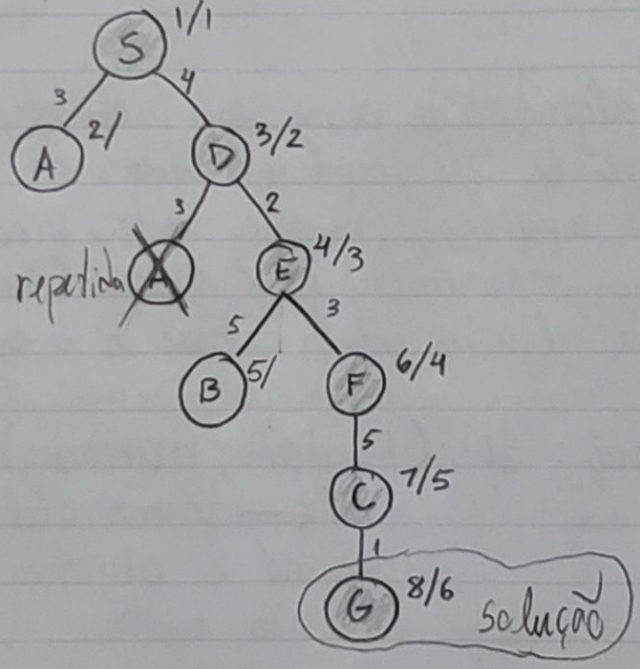
Se o nó solução não for o último de seu nível, e a checagem for feita ao gerar um novo nó, encontramos essa solução sem gerar todos os nós.

Questão 08.

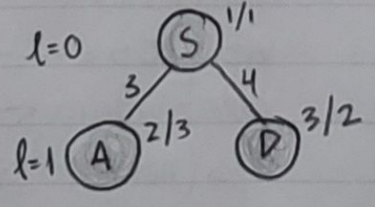
• Largura:



• Profundidade:



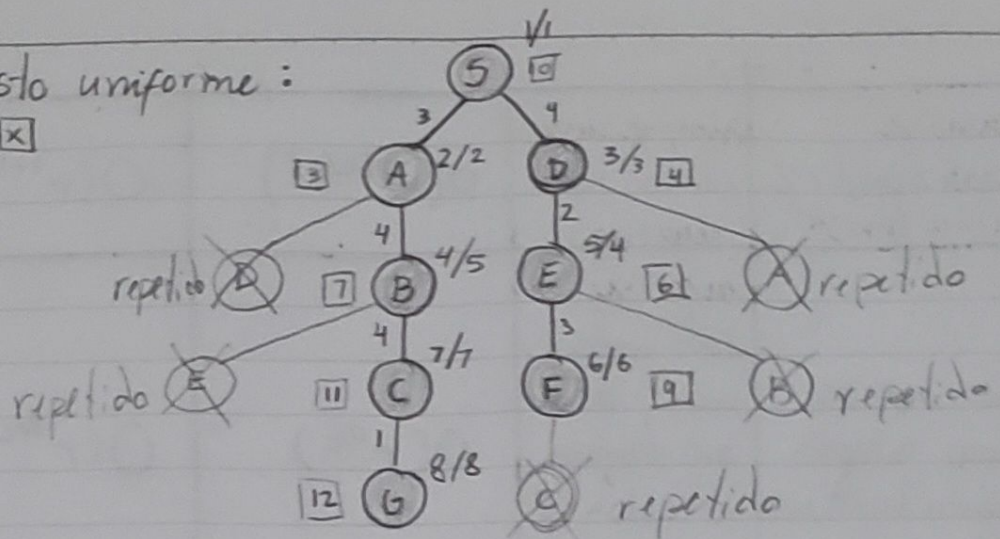
• Profundidade limitada: l=1



não encontra solução com limite 1.

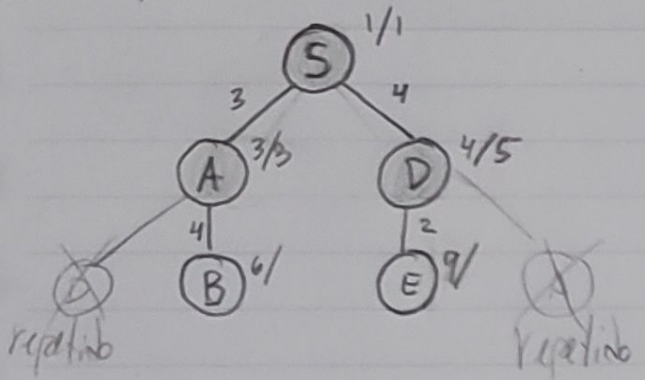
• Custo uniforme:

custo = [x]

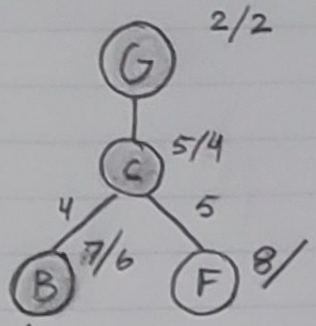


• Bidirecional:

para frente:



para trás:



encontrou o mesmo nó na fronteira da outra busca.

Questão 09:

r → fator de ramificação
m → profundidade máxima
c → custo da solução ótima
d → profundidade da solução mais rasa
l → limite de profundidade
ε → custo mínimo das ações

	completa	ótima	comp. tempo	comp. espaço
largura	sim	sim, se o custo dos passos for igual	$O(r^d)$	$O(r^d)$
profundidade	não	não	$O(r^m)$	$O(r \cdot m)$
prof. limitada	não	não	$O(r^l)$	$O(r \cdot l)$
prof. iterativa	sim	sim, se o custo dos passos for igual	$O(r^d)$	$O(r \cdot d)$

...

custo uniforme	sim, se custo $\geq$ positivo ε (evita loops)	sim, se custos forem não negativos	$O(r^{1+[C/\varepsilon]})$	$O(r^{1+[C/\varepsilon]})$
bidirecional	sim, se as duas direções usam busca em largura	sim, se as duas direções usam busca em largura e custos são iguais para os passos	$O(r^{d/2})$	$O(r^{d/2})$