UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

DANIEL LA RUBIA ROLIM - DRE: 115033904

LISTA DE EXERCÍCIOS

BUSCAS NÃO INFORMADAS

Rio de Janeiro

2021

1. No problema das jarras, temos duas jarras, uma com capacidade de 4 litros e outra com capacidade de 3 litros de água. As jarras não possuem nenhuma marcação e precisamos colocar exatamente 2 litros de água na jarra com capacidade de 4 litros. Podemos encher as jarras com água de uma torneira, assim como podemos jogar fora a água que estiver nas jarras e passar água de uma jarra para outra. Faça a formulação deste problema indicando como você o representaria, qual seria o espaço de estados, qual o estado inicial, quais os estados finais, quais as regras usadas e o custo até encontrar uma solução.

Fazendo uso das informações do enunciado, podemos pensar no problema da seguinte forma:

- Duas jarras: **J1** e **J2,** podendo ser representado por uma tupla de duas posições.
- Capacidades: **4L** e **3L**, respectivamente. **J1** $\in$ [0,4] e **J2** $\in$ [0,3]
- Estado inicial: Jarras vazias **(0,0)**
- Estado final: objetivo(**2**, N), podendo **N** ser qualquer valor, contanto que a primeira jarra possua apenas dois litros.
- Regras que realizam as ações:
    - **encher_J1** e **encherJ2**
    - **passar_12** e **passar21** (representando a transferência da quantidade de água da jarra 1 para a jarra 2 e vice-versa)
    - **esvaziar1** e **esvaziar2**
- Custo: Variável de acordo com o algoritmo escolhido para resolver o problema. Montada a árvore de configurações, o algoritmo percorre os estados somando 1 de custo, até encontrar uma configuração que corresponda ao *objetivo*.

2. Você está na margem de um rio com um barco, um maço de couve, uma cabra e um lobo. Sua tarefa é levar todas essas coisas para a margem oposta do rio. Apenas você sabe navegar com o barco, e há espaço no barco para apenas você e mais um item (independente do tamanho do item). Você não pode deixar a cabra com o lobo ou a couve na mesma margem do rio sem estar monitorando, ou algo será comido.

a) Formule este problema para ser resolvido utilizando um algoritmo de busca.

- Três elementos: **Cabra**, **Couve** e **Lobo,** podendo ser representado por uma tupla de 3 posições.
- Três estados para um elemento:
    - 0 → Não atravessou
    - 1 → Atravessando
    - 2 → Atravessou
- Estado inicial: **(0, 0, 0)** → Nenhum dos itens atravessou
- Estado final: **(2, 2, 2)** → Todos os itens atravessaram
- Regras para as alterar os estados:
    - **move02** (passando o valor da posição na tupla de 0 → 1 → 2)
    - **move20** (fazendo o caminho caminho inverso, 2 → 1 → 0)

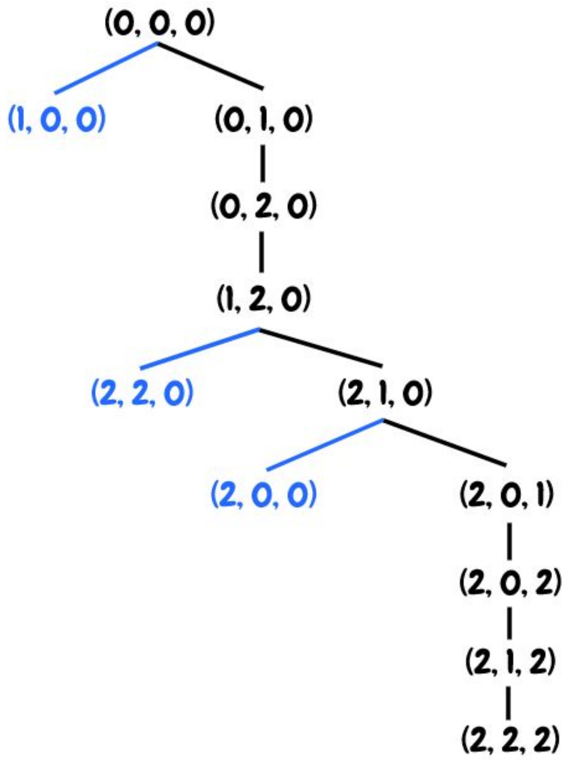
- Testes:
  - Verificar a limitação imposta onde a **Cabra** não pode ficar **sozinha** tanto com a **couve**, quanto com o **lobo**
  - Verificar que qualquer um dos itens só pode atingir o estado **1** isolado (visto que o *ser humano* só pode carregar 1 item por vez no barco)

b) Escolha uma estratégia de busca que consiga encontrar uma solução para o problema.

Para resolver o problema, será utilizado o algoritmo de busca em profundidade.

c) Desenhe a árvore de busca que será gerada pela estratégia que você escolheu, indicando a ordem de geração e expansão dos nós.

(Couve, Cabra, Lobo) = (0, 0, 0)



d) Diga qual foi a solução encontrada.

Acompanhando pela árvore, podemos ver que devemos atravessar a cabra primeiro, chegando ao estado (0,**2**,0). Na volta, transportamos a couve no estado **(1,2,0)** e deixamos a couve do outro lado do rio. Entretanto, como pode ser observado, no ato de levar a couve, a cabra é inserida de volta no barco, indo ao estado **(2,1,0)**. Então ela é deixada na margem inicial e o lobo transportado, ficando o lobo com a couve no outro lado da margem **(2,0,2)**. Por mim, a cabra é transportada, atingindo o estado **(2,2,2)**.

3. O problema de coloração de mapas consiste em colorir um mapa usando no máximo 4 cores distintas, de forma que regiões adjacentes tenham cores diferentes.

a) Apresente uma formulação para este problema definindo o objetivo, estado inicial, operadores e a função de custo.

Não consegui resolver este problema quando foi passado como tarefa em Prolog, portanto vou descrever aqui a maneira como pensei em resolver, embora não tenha tido capacidade de implementar na época.

Dada uma lista de países (no caso, podendo utilizar a lista de países da América do Sul como exemplo), definir os predicados usando uma lista que relaciona as adjacências, de forma que:

- `fronteiras(brasil, [guianaFrancesa, suriname, guiana, venezuela, colombia, peru, bolivia, paraguai, argentina, uruguai]).`
- `fronteiras(guianaFrancesa, [brasil, suriname]).`
- …

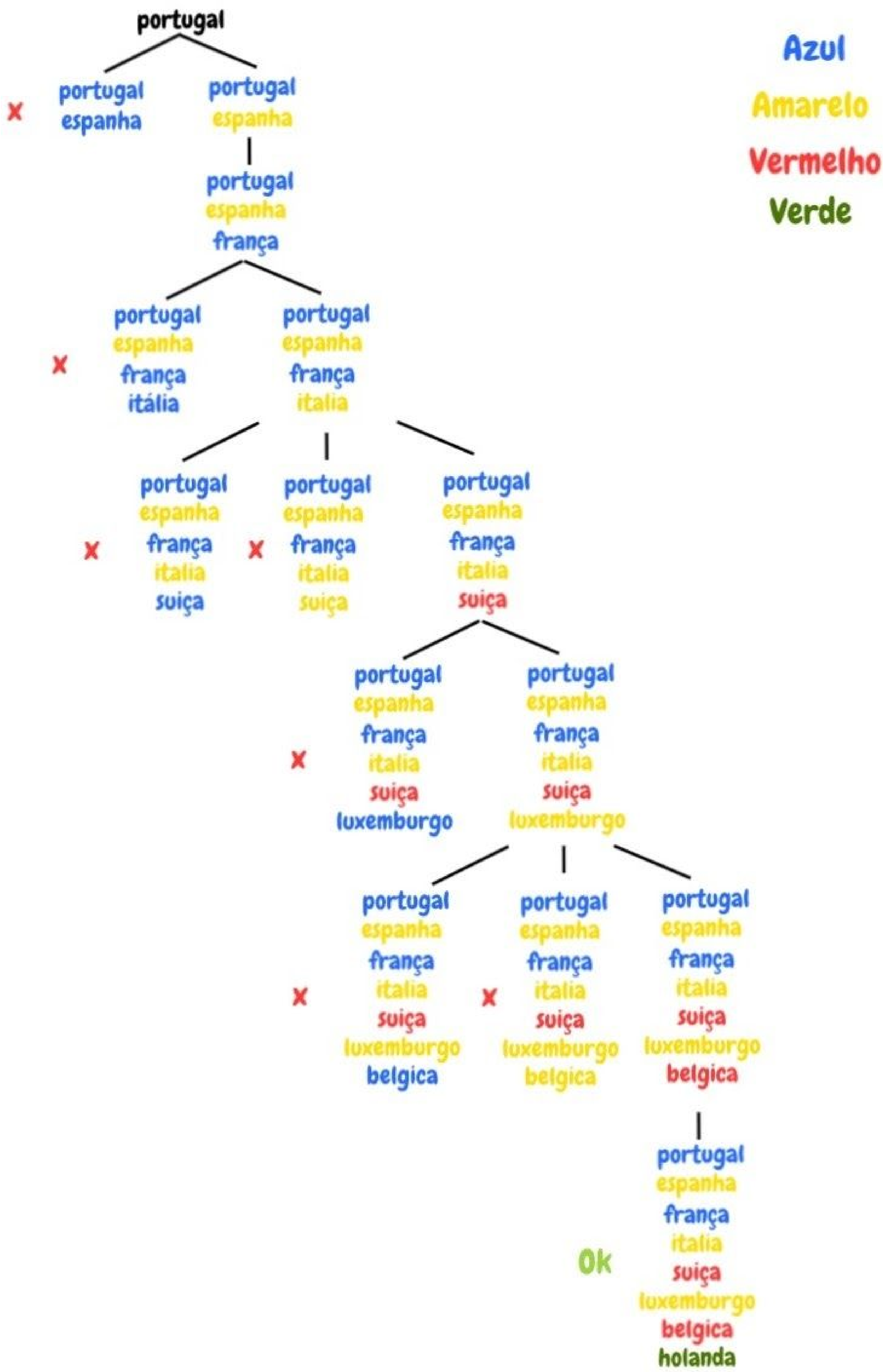
O objetivo deste problema consiste em que os países vizinhos não possuam a mesma cor. Poderia ser definido, como estado inicial, que nenhum dos países está pintado (cor preta). Para fazer o preenchimento, pensei em um algoritmo que faria *backtrack*. Isso seria necessário porque poderia chegar um determinado momento onde nosso programa **não teria opções de cores para serem utilizadas**, visto que todos os adjacentes do país **x** que seria pintado no momento já possuem as quatro cores disponíveis. Dessa forma, seria necessário retornar ao nó anterior e tentar uma nova combinação. Sendo possível, o algoritmo prosseguiria a partir dali. Não sendo, retornaria mais um nó e tentaria novamente (e assim por diante).

b) Considere o mapa abaixo e construa a árvore de busca para o problema acima, indicando em cada nó a ordem de geração e expansão, para os algoritmos de busca em largura e em profundidade.

A árvore da busca em largura fica enorme e achei que não seria viável colocar aqui na resposta. Quanto ao início da coloração, decidi iniciar por Portugal por ser o país mais à esquerda.

Já referente à performance, o melhor algoritmo é o de busca em profundidade, uma vez que menos nós precisam ser visitados até chegar em uma solução viável.

A árvore de busca em profundidade está na página a seguir.

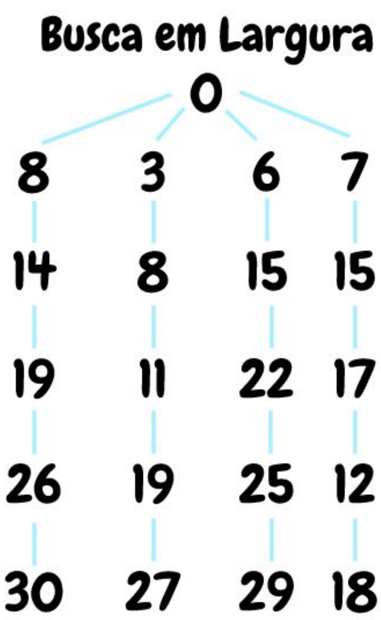

portugal
portugal
espanha
x
portugal
espanha
portugal
espanha
frança
portugal
espanha
frança
itália
x
portugal
espanha
frança
italia
portugal
espanha
frança
italia
suiça
x
portugal
espanha
frança
italia
suiça
x
portugal
espanha
frança
italia
suiça
portugal
espanha
frança
italia
suiça
luxemburgo
x
portugal
espanha
frança
italia
suiça
luxemburgo
portugal
espanha
frança
italia
suiça
luxemburgo
belgica
x
portugal
espanha
frança
italia
suiça
luxemburgo
belgica
x
portugal
espanha
frança
italia
suiça
luxemburgo
belgica
portugal
espanha
frança
italia
suiça
luxemburgo
belgica
holanda
Ok
Azul
Amarelo
Vermelho
Verde

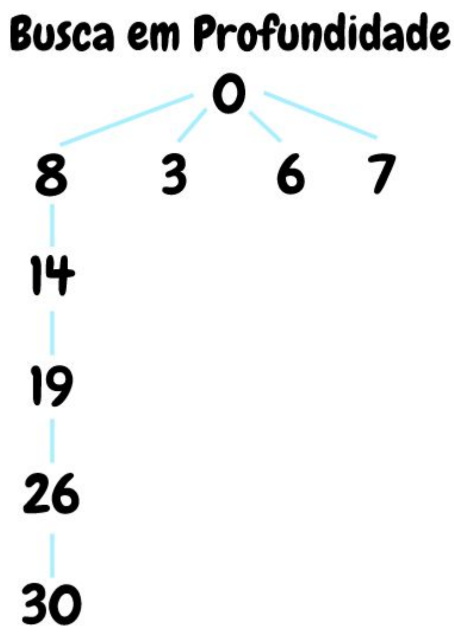
4. Calcule o valor da coluna de soma mínima da matriz abaixo, utilizando os métodos de busca em profundidade, largura e custo-uniforme. Compare as respostas e discuta os resultados.

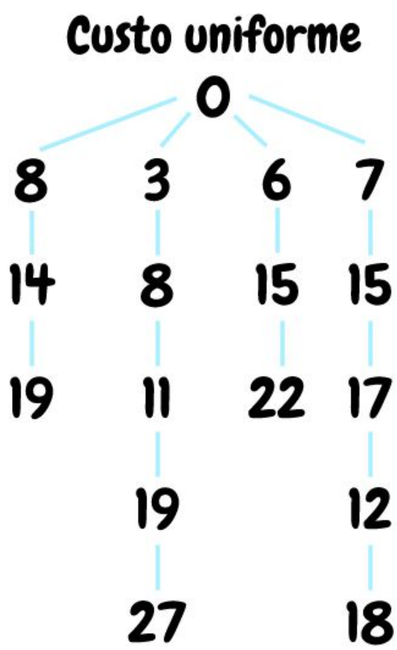
$$\begin{pmatrix} 8 & 3 & 6 & 7 \\ 6 & 5 & 9 & 8 \\ 5 & 3 & 7 & 2 \\ 7 & 8 & 3 & -5 \\ 4 & 8 & 4 & 6 \end{pmatrix}$$

Conforme podemos observar, a solução que fornece a menor soma é a de **custo uniforme**.

Busca em profundidade e busca em largura → 30
Busca uniforme → **18**







5. Um robô deve encontrar o caminho mais curto entre um ponto de partida **S** e um objetivo **G**, no espaço bidimensional povoado por polígonos, conforme a figura abaixo. Considere que o robô possui tamanho infinitesimal e que o caminho pode ser adjacente aos obstáculos, mas não pode cortar nenhum deles.

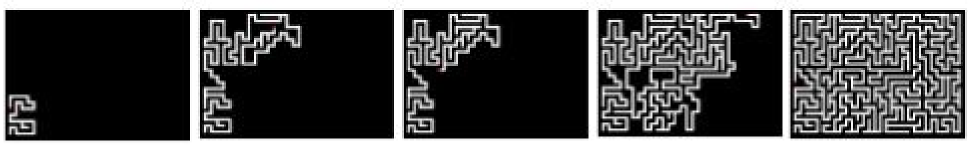
a) Qual o conjunto mínimo de pontos que deve ser considerado na busca do caminho de menor tamanho entre um ponto de partida e um objetivo quaisquer? Justifique.

O menor caminho possível seria formado por uma reta que ligaria os pontos **S** e **G**. Como existem polígonos no caminho, é necessário considerar como conjunto mínimo de pontos o conjunto formado pelos vértices dos polígonos. Os vértice não-côncavo do polígono não-convexo presente na figura deve ser desconsiderado, uma vez que é possível traçar uma reta entre os vértices convexos deste polígono e, utilizar um vértice côncavo aumentaria necessariamente o tamanho do caminho.

b) Dado um dos pontos dentro deste conjunto mínimo, descreva um método para gerar os sucessores deste ponto no grafo de busca correspondente. Quais são os pontos sucessores para o ponto S na figura acima?

Os pontos sucessores para o ponto **S** são todos aqueles em que é possível traçar uma reta que o ligue ao ponto **S**. Neste caso, pela imagem, temos três vértices.

6. O seguinte algoritmo é um dos mais simples utilizado para gerar labirintos computacionalmente: Considere a área do labirinto como uma matriz de células onde cada célula é inicialmente preenchida por quatro paredes (ou seja, todas as células estão bloqueadas). Iniciando em uma célula qualquer na borda da área, o algoritmo sorteia randomicamente uma célula vizinha que ainda não foi visitada, remove a "parede" entre estas duas células, e adiciona a nova célula a uma pilha (o que significaria abrir um caminho entre elas). O algoritmo repete este processo com a célula escolhida no último passo. Uma célula que não tenha vizinhos ainda não visitados é considerada um ponto morto. Ao atingir um ponto morto, o algoritmo faz um backtracking no caminho entre células vizinhas na ordem inversa à ordem em que foram visitadas, até atingir uma célula com um vizinho ainda não visitado, e recomeça então a geração do caminho visitando essa célula que ainda não tinha sido visitada (criando uma nova junção). Este processo continua até que todas as células tenham sido visitadas, forçando o algoritmo a fazer *backtracking* até a célula inicial, o que garante que o espaço do labirinto será completamente visitado.



a) O algoritmo descrito é uma versão ligeiramente randomizada de que algoritmo de busca conhecido? Justifique.

O algoritmo é uma versão da busca em profundidade. É possível afirmar isto porque uma vez que o algoritmo inicia em um nó, ele sempre está buscando realizar a expansão dos nós filhos do **nó atual**, explorando o máximo possível o ramo antes de fazer o *backtracking*.

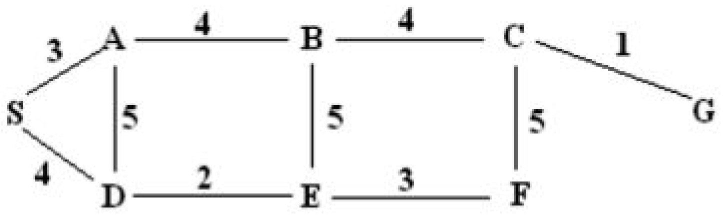
b) Escreva este algoritmo em pseudocódigo.

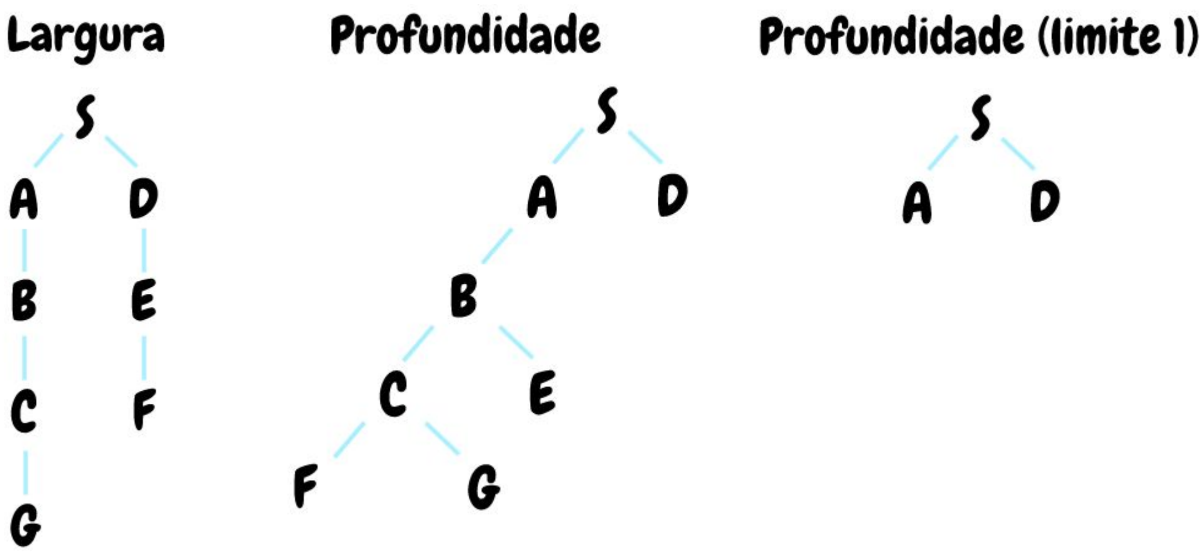
~~Completar depois.~~

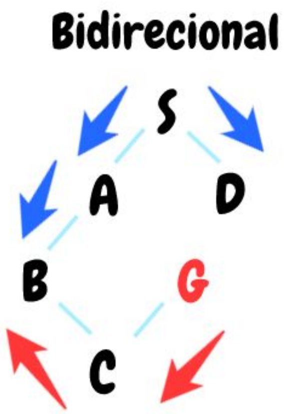
7. Considere um problema com as seguintes características: na sua árvore de busca, cada estado gera **b** novos estados. Supondo que estamos fazendo uma busca em largura e que a solução deste problema possui comprimento **d**, qual o número máximo de nós que serão gerados antes que uma solução seja encontrada? Seja **n** o valor máximo que você encontrou. É possível obter a solução sem que todos os **n** vértices sejam gerados? Por quê?

O número máximo de nós gerados até encontrar uma solução é de $\mathbf{b}^{d}$. Por conta de estar sendo utilizado busca em largura para encontrar as soluções, é necessário que todos os nós tenham sido gerados, mesmo que não sejam visitados.

8. Considere a figura abaixo, onde **S** é o estado inicial e **G** é o estado final de um problema qualquer. Construa a árvore de busca do seguintes métodos: *largura, profundidade, profundidade com limite igual a 1, custo uniforme e bidirecional*.







9. Para cada método de busca não-informada, indique se ele é completo, ótimo e as complexidades de tempo e espaço.

	Completo	Tempo	Espaço	Ótimo
Largura	Sim[a]	$O(b^d)$	$O(b^d)$	Sim[c]
Profundidade	Não	$O(b^m)$	$O(b^m)$	Não
Custo Uniforme	Sim[a,b]	$O(b^{1+[c*/E]})$	$O(b^{1+[c*/E]})$	Sim
Profundidade Lim.	Não	$O(bL)$	$O(bL)$	Não
Profundidade Iter.	Sim[a]	$O(bd)$	$O(bd)$	Sim[c]
Bidirecional	Sim[a,d]	$O(b^{d/2})$	$O(b^{d/2})$	Sim[c,d]

Tal que:

- **b** → Fator de ramificação
- **d** → Profundidade da solução mais rasa
- **m** → Profundidade máxima da árvore de busca
- **L** → Limite de profundidade
- [a] → Completo caso **b** seja infinito
- [b] → Completo se o custo de cada passo for menor do que **e** para **e** > **0**.
- [c] → Ótimo caso o custo de cada passo seja idêntico
- [d] → Caso ambas as direções usem busca em largura