Inteligência Artificial

EC016

Estratégias de Busca

Prof. Felipe Andery Reis
fandery@inatel.br
Material adaptado do prof. Edielson

Fev/2018

Introdução

Objetivos

- Promover a compreensão dos algoritmos de busca informada e não informada e desenvolver aplicações utilizando os métodos aprendidos

Agenda

Parte 1

- Apresentação do ambiente de desenvolvimento e a linguagem de programação utilizada
- Introdução às estratégias de busca
- Estratégias de busca cega:
  - Busca em largura
  - Busca em profundidade
- Exercícios práticos envolvendo estratégias de busca cega

Agenda

Parte 2

- Estratégias de busca heurística:
  - Busca gulosa
  - Busca A*
- Exercícios práticos envolvendo estratégias de busca heurística

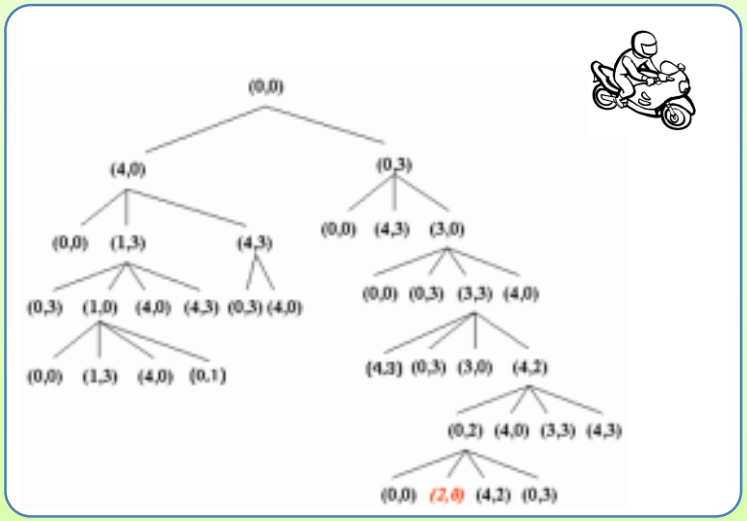
Agentes Inteligentes

Introdução

➢ Paradigma do raciocínio da IA

Metáfora linguística

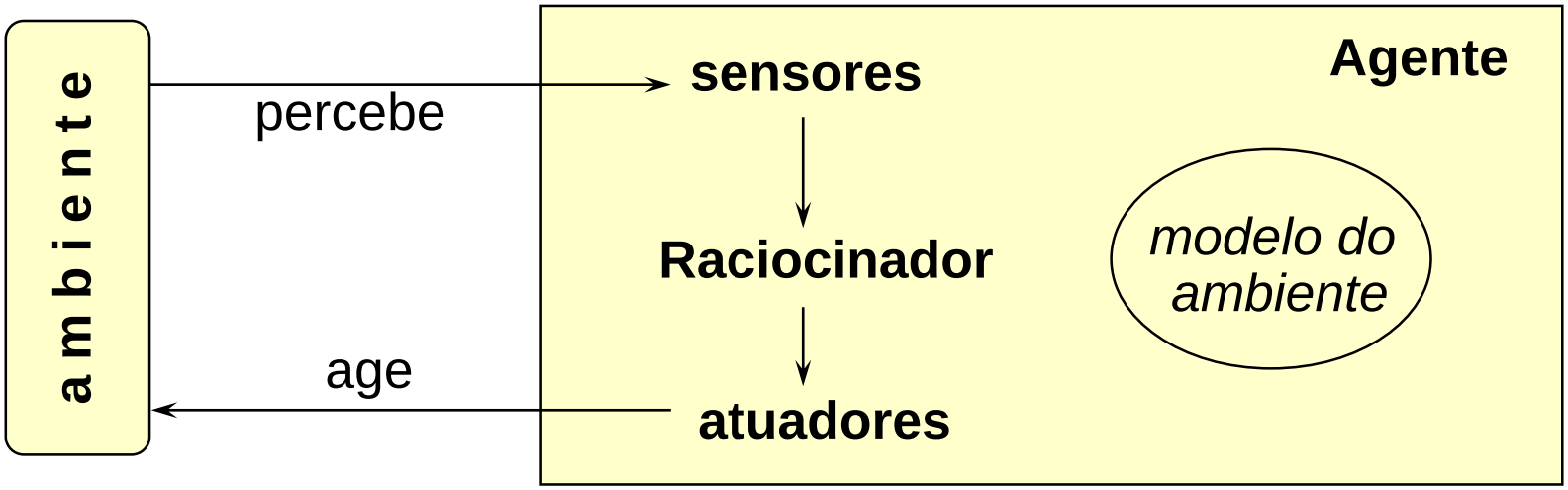
ex. sistemas especialistas, agentes,...



Simbólica

O que é um agente?

- Agente é qualquer entidade que:
  - **percebe** seu ambiente através de **sensores** (ex. câmeras, microfone, teclado, mensagens de outros agentes,...);
  - **age** sobre ele através de **atuadores** (ex. vídeo, autofalante, impressora, braços, ftp, mensagens para outros agentes,...).

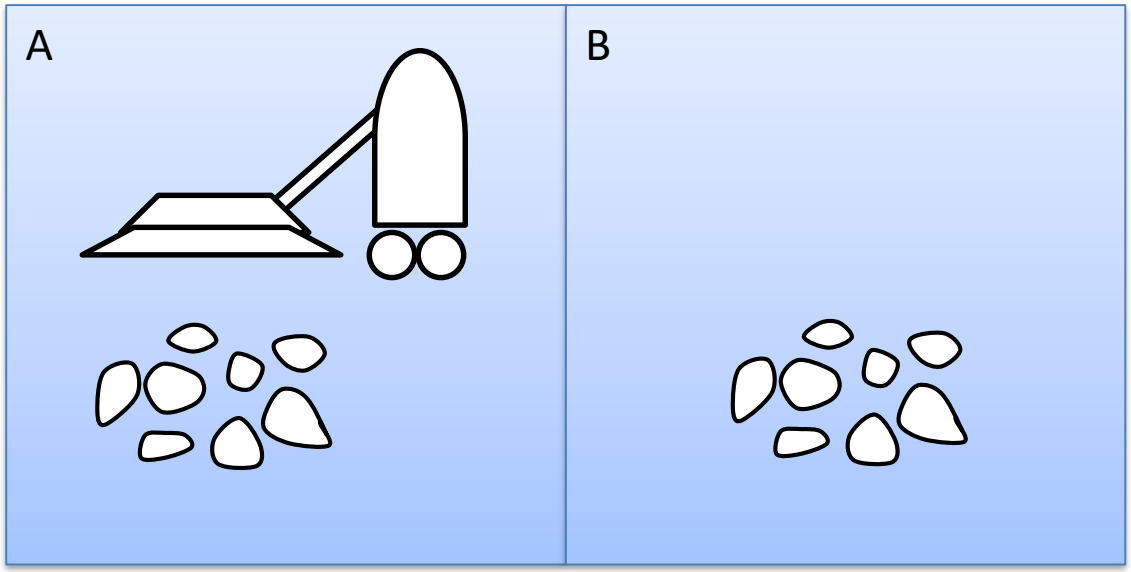


Exemplos de agentes

- Um agente humano
  - **Sensores**: olhos, ouvidos, ...
  - **Atuadores**: mãos, pernas, boca, ...
- Um agente robótico
  - **Sensores**: câmeras, detectores da faixa de infravermelho, ...
  - **Atuadores**: motores, ...
- Um agente de software
  - **Sensores**: teclas digitadas, conteúdo de arquivos, pacotes de redes, ...
  - **Atuadores**: exibição de algo na tela, gravação de arquivos, envio de pacotes de rede, ...

O mundo do aspirador de pó

- Este é o mundo do aspirador de pó com apenas 2 locais.



- Neste caso, o agente (aspirador de pó) pode mover-se para a esquerda, mover-se para a direita, aspirar ou não fazer nada.

O mundo do aspirador de pó

- A **sequencia de percepções** do agente é a história completa de tudo que o agente já percebeu.
- O comportamento do agente é descrito pela **função do agente**, que mapeia qualquer sequência de percepções específicas para uma ação.

Sequencia de percepções	Ação
[A, Limpo]	Direita
[A, Sujo]	Aspirar
[B, Limpo]	Esquerda
[B, Sujo]	Aspirar
[A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Sujo]	Aspirar
...	...
[A, Limpo], [A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Limpo], [A, Sujo]	Aspirar
...	...

Conjunto infinito de funções do agente

A natureza dos ambientes

- **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors)

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Motorista de taxi	Viagem segura, rápida, dentro da lei, confortável, maximizar lucro	Estradas, outros tipos de tráfego, pedestres, clientes	Direção, Acelerador, freio, sinal, buzina, visor	Câmeras, sonar, velocímetro, GPS, hodômetro, acelerômetro, sensores do motor, teclado
Sistema de diagnóstico médico	Paciente saudável, minimizar custos , processos judiciais	Paciente, hospital, equipe	Exibir perguntas, testes, diagnósticos, tratamento, indicações	Entrada pelo teclado para sintomas, descobertas, respostas do paciente
Análise de imagens de satélite	Definição correta da categoria da imagem	Links de transmissão de satélite em órbita	Exibir a categorização da cena	Arrays de pixels em cores
Robô de seleção de peças	Porcentagem de peças em bandejas corretas	Correia transportadora com peças; bandejas	Braço e mão articulados	Câmera, sensores angulares articulados
Controlador de refinaria	Maximizar pureza, rendimento, segurança	Refinaria, operadores	Válvulas, bombas, aquecedores, mostradores	Sensores de temperatura, pressão, produtos químicos
Instrutor de inglês interativo	Maximizar nota de aluno em teste	Conjunto de alunos, testes de agência	Exibir exercícios, sugestões, correções	Entrada pelo teclado

Teoria de problemas

Teoria de problemas

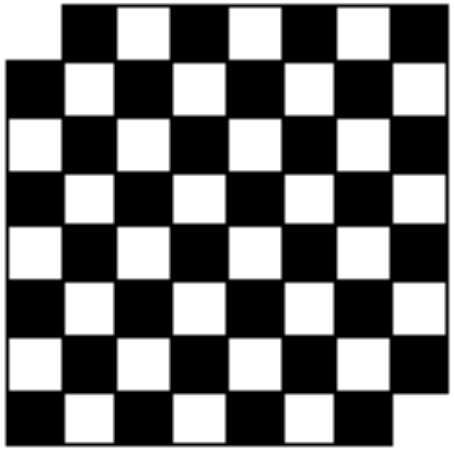
- A IA se ocupa da resolução de problemas, para tal é necessário o conhecimento sobre o problema e sobre as técnicas para manipular este conhecimento e obter a solução. Mas, o que é um PROBLEMA?

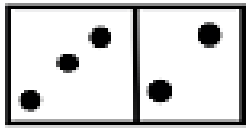
- Resolver um problema é diferente de ter um método para resolvê-lo.

- Antes de tentar buscar a solução de um problema, deve-se responder as seguintes perguntas:
  - Quais são os dados?
  - Quais são as soluções possíveis?
  - O que caracteriza uma solução satisfatória?

Teoria de problemas

- O tabuleiro de xadrez mutilado
  - Nesse problema, temos um tabuleiro de jogo de xadrez onde dois quadrados em cantos opostos foram cortados, de modo que restam apenas 62 quadrados





  - Agora, pegamos 31 dominós feitos de modo que cada dominó cubra exatamente dois quadrados.
  - É possível dispor os 31 dominós de modo que eles cubram todos os 62 quadrados do tabuleiro?

Teoria de problemas

- **Definição**: Um problema é um objeto matemático P={D,R,q}

D conjunto não vazio de dados

R conjunto não vazio com os possíveis resultados

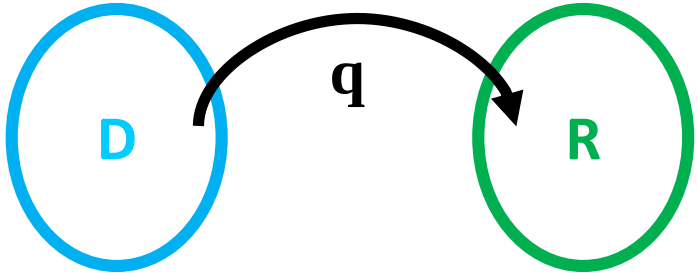
$$q \subset D \times R$$

é a condição que caracteriza uma solução satisfatória, associando cada elemento do conjunto de dados a elemento do conjunto de resultados.

Teoria de problemas

- **Exemplo**: Um problema de diagnóstico médico
  - O conjunto de dados disponíveis $d \in D$ (observação dos sintomas, exames, etc);
  - R é o conjunto de doenças possíveis;
  - Solução satisfatória: encontrar o par (d, r) onde $r \in R$ é o diagnóstico correto.

- A definição de um problema permite testar se um certo elemento é ou não solução, mas não guia na busca deste elemento.



Teoria de problemas

- Estratégias Básicas para Resolver Problemas

As estratégias constituem os modos básicos de raciocínio para resolver problemas.

1. **Pela definição do problema**: o qual se apresenta como uma função, estes modos de raciocínio devem se adaptar ao modo que a função foi definida.

2. **Por enumeração exaustiva**: o conhecimento necessário para resolver o problema está na enumeração (busca exaustiva ou força bruta)

Teoria de problemas

3. **Declarativamente**: leva frequentemente a problemas de busca.

   "Utilizar um método de busca em que, por passos sucessivos se aproxima da solução, usando algumas vezes técnicas sem grande justificativa teórica". ESTA É A ABORDAGEM DA IA SIMBÓLICA!

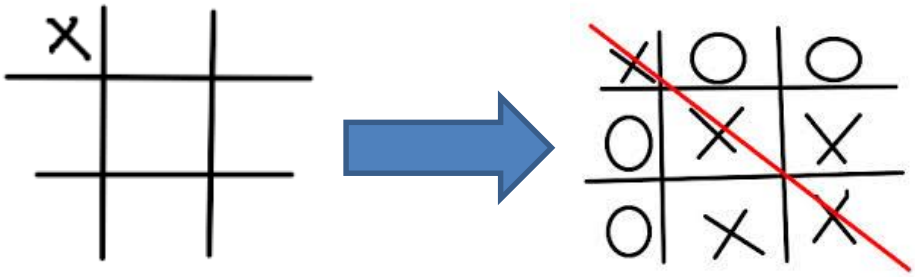
4. **Por exemplos**: Se o problema foi definido por exemplos, deverá se usar um método para aproximar a função. ESTA É A ABORDAGEM DA IA CONEXIONISTA!

➢ ALGUNS PROBLEMAS CLÁSSICOS:

Missionários e canibais; Torres de Hanói; Baldes de Água; Jogo do Oito; Reconhecimento de Caracteres, Previsão, etc.

Teoria de problemas

- Jogo da velha



- Missionários e Canibais

- As Torres de Hanói

Resolução de problemas

Resolução de problemas

Dedica-se ao estudo e elaboração de algoritmos, capazes de resolver, por exemplo, problemas considerados intratáveis do ponto de vista da computação convencional.

Resolução de problemas

- Primeiros problemas por computador: prova automática de teoremas e jogos.
- Capacidade de cálculo e memória dos computadores: insuficientes perante o enorme número de caminhos de solução.
- **Exemplo**: jogo de xadrez
- **Um dos objetivos de IA**: resolver problemas que o homem não sabe resolver facilmente ou num tempo razoável, desde que sejam completamente formalizados.

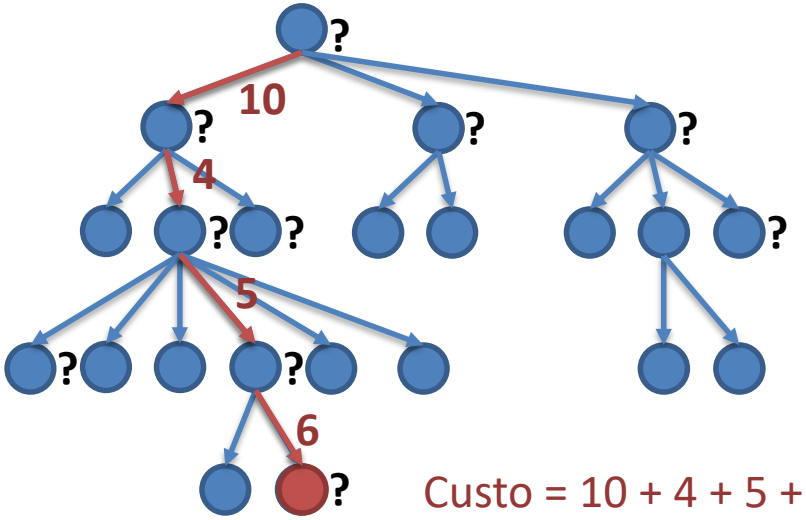
Resolução de problemas

- O quebra-cabeças 3x3
- O Caixeiro Viajante
- Cálculo Integral Formal
- Empilhamento de blocos: a partir de uma configuração de blocos iniciais, qual a sequência de movimentos para se chegar a uma configuração final?
- As Torres de Hanói

Formulação de Problemas – Espaço de estados

- O espaço de estados é a árvore de todos os estados que podemos produzir a partir do estado inicial.

- Elementos:
  - Estado Inicial
  - Função Sucessor
  - Teste de Objetivo
  - Custo de Caminho

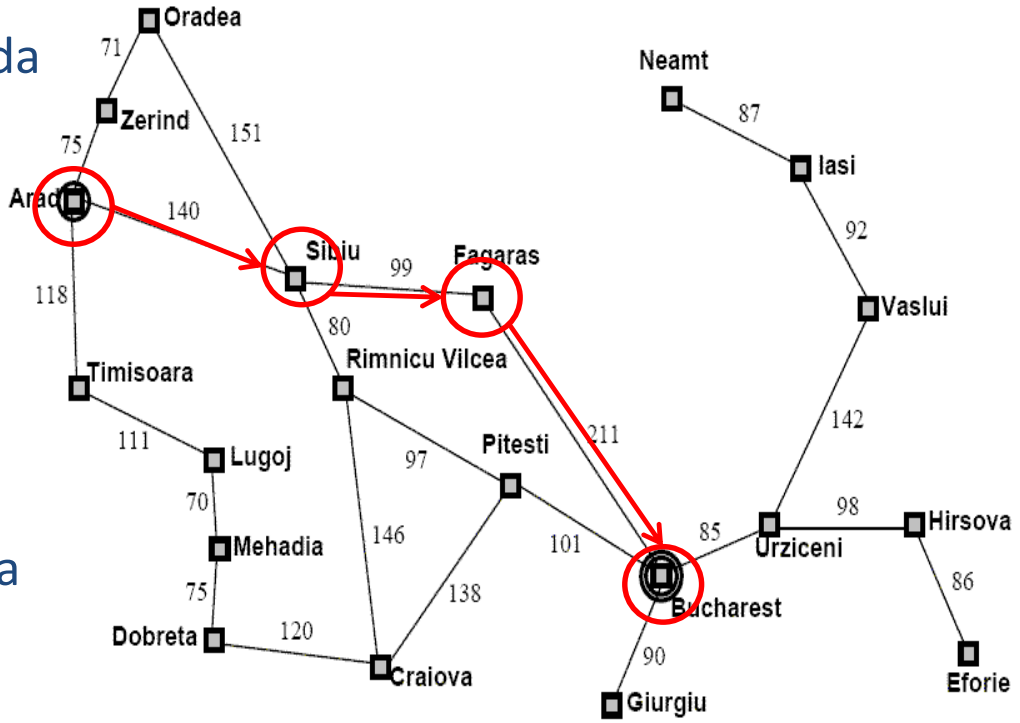


Custo = 10 + 4 + 5 + 6 = 25

Resolução de problemas

➢ Formulação de Problemas – Espaço de estados

1. **Espaço de estados:** Mapa da Romênia
2. **Estado inicial:** Arad
3. **Estado Objetivo:** Bucharest
4. **Ações Possíveis:**
   Ir de uma cidade para outra
5. **Custo:**
   Distância entre as cidades



Custo: 140 + 99 + 211 = 450Km

Resolução de problemas

➢ Problemas e soluções bem definidos

➢ Quatro componentes para definir um problema:

1. O **estado inicial** em que o agente começa.

2. Uma descrição das ações possíveis que estão disponíveis para o agente.
   - ➢ Formulação mais comum: uso de uma **função sucessor**.
   - ➢ Estado inicial e função sucessor: definem o espaço de estados do problema.
   - ➢ **Caminho** no espaço de estados – sequência de estados conectados por uma sequência de ações.

3. O **teste de objetivo** – determina se um dado estado é um estado objetivo.

4. Função de **custo de caminho** – atribui um custo numérico a cada caminho.

Resolução de problemas

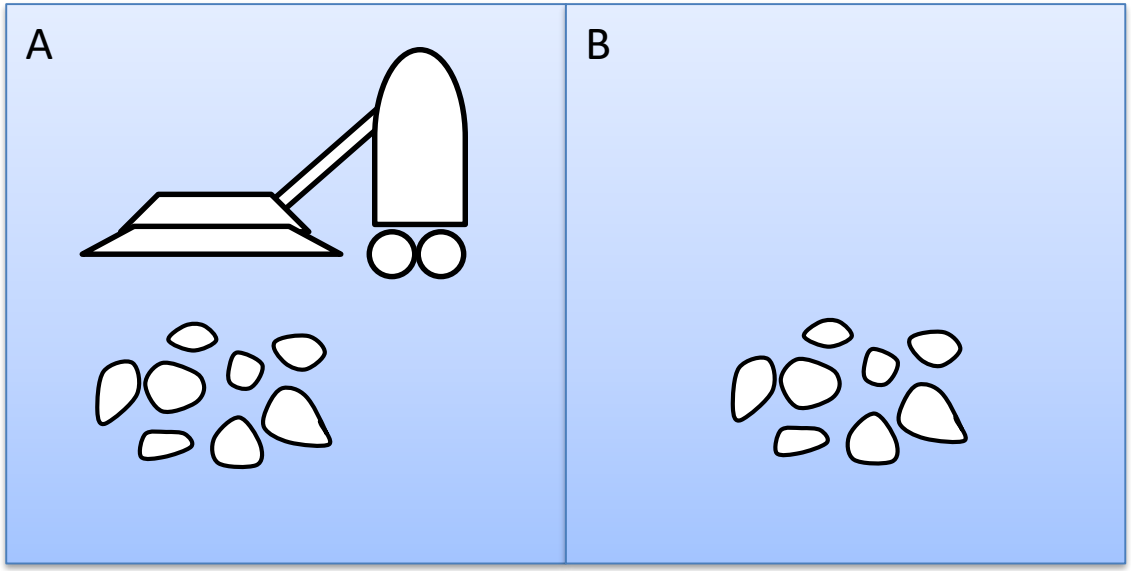
Exemplos de Problemas

- **Exemplos:**
  - Mundo do Aspirador de Pó
  - Quebra-cabeça de 8 peças
  - Quebra-cabeça das 8 rainhas

- **Exemplos: Problemas do mundo real**
  - Problema de Exames médicos
  - Problema de Diagnósticos médicos
  - Problema de Roteamento
  - Problema de Layout de placas
  - Problema de Navegação de Robôs
  - Problema de Pesquisas na Internet

Miniproblemas

- **Exemplo 1**: Mundo do aspirador de pó com apenas 2 locais.



Miniproblemas

- Problema do Mundo do Aspirador de Pó - Formulação

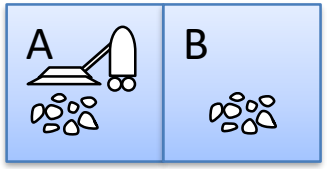
- **Estados:**
  - O agente está em uma entre duas posições, cada uma das quais pode conter sujeira ou não.
  - Há $2 \times 2^2 = 8$ estados do mundo possíveis (vistos a seguir).

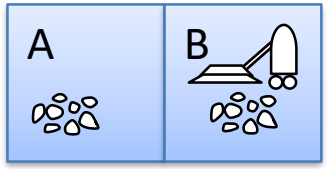
- **Estado inicial:**
  - Qualquer estado pode ser designado como estado inicial.

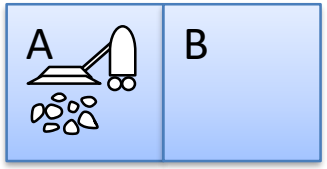
- **Função Sucessor:**
  - Gera os estados válidos que resultam da tentativa de executar as três ações (*Esquerda*, *Direita* e *Aspirar*).

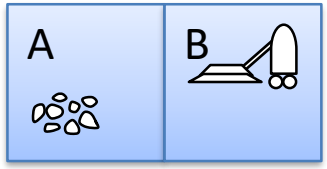
Miniproblemas

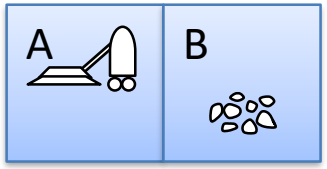
- **Estados:**

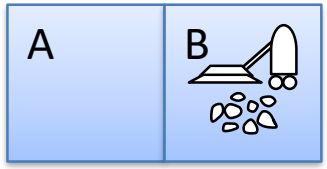
1 

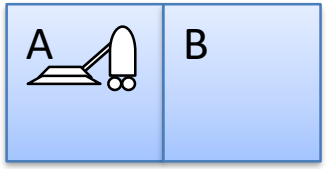
2 

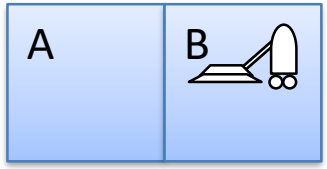
3 

4 

5 

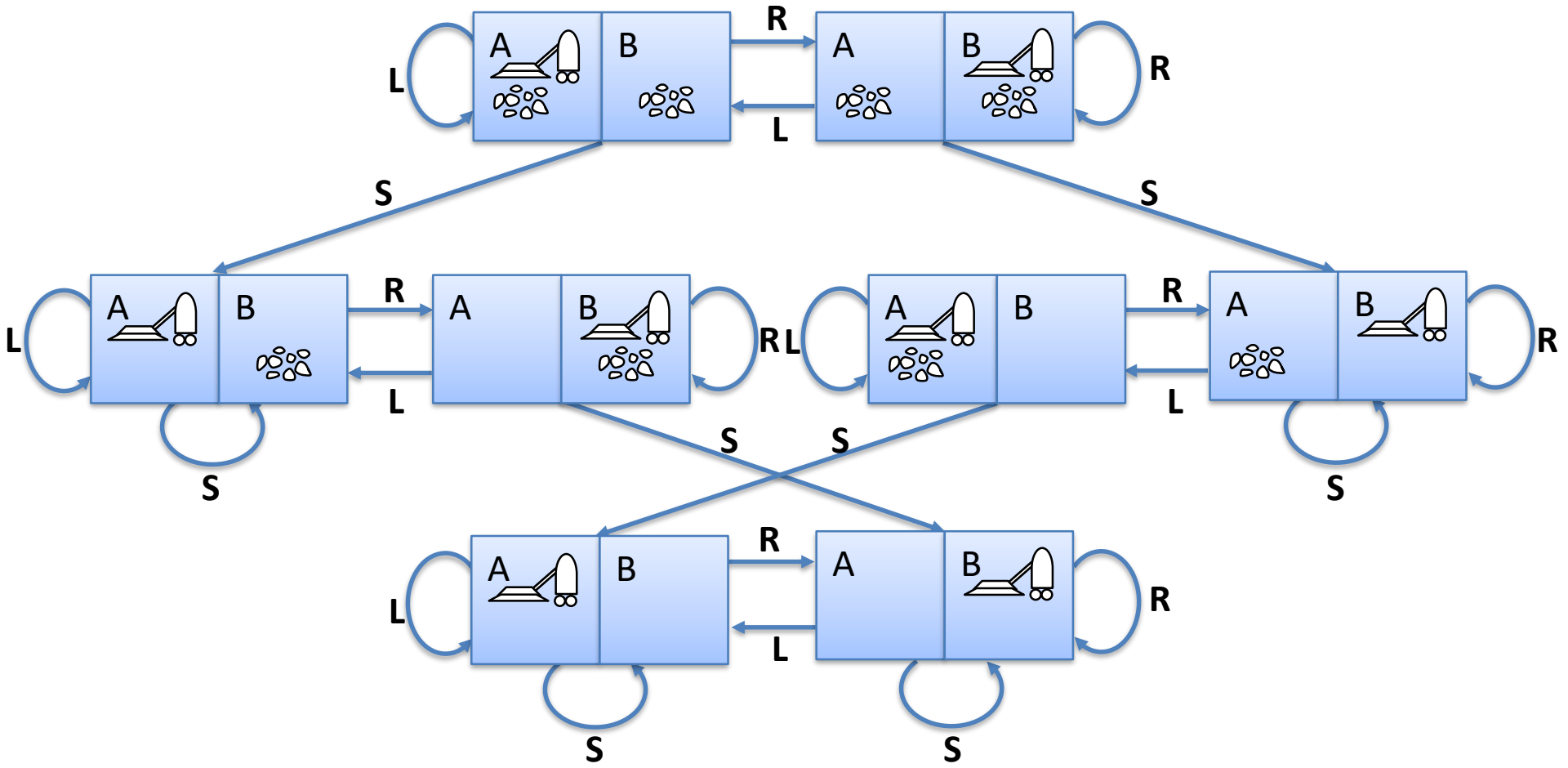
6 

7 

8 

Miniproblemas

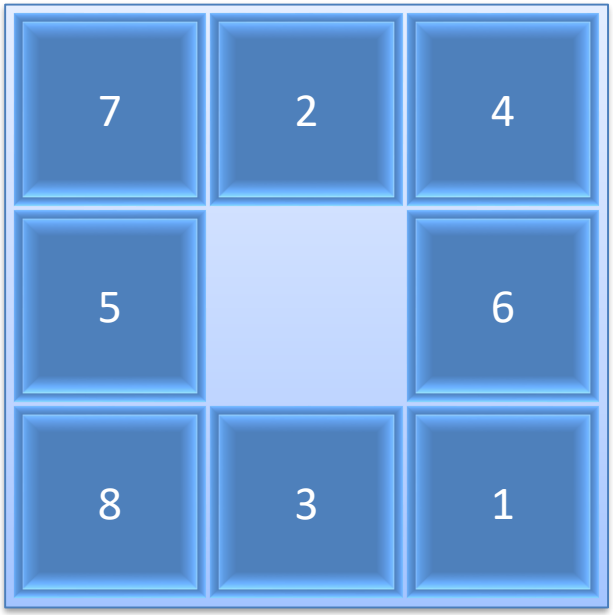
- **Espaço de estados:** (Notações L=Esquerda, R= Direita, S=Aspirar)



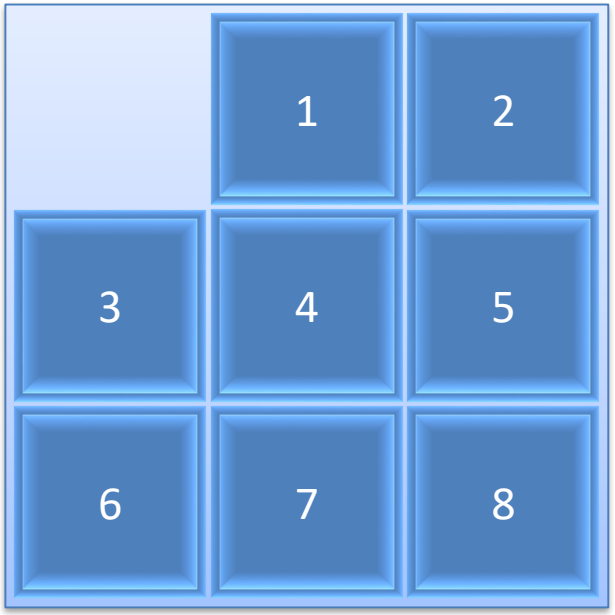
Miniproblemas

- **Exemplo 2**: Uma instância típica do quebra-cabeça de 8 peças

Estado inicial



Estado objetivo

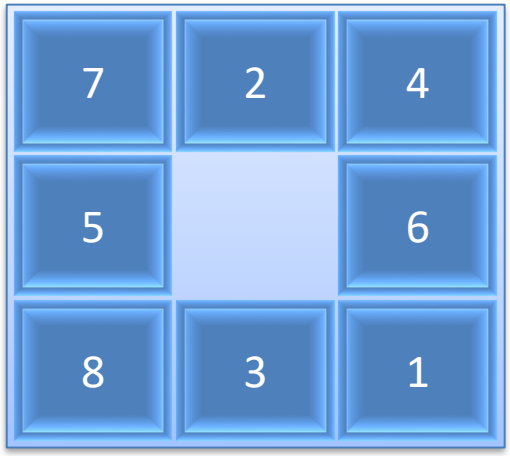


Miniproblemas

- Problema do Quebra-cabeça de 8 Peças - Formulação

- **Estados:**
  - A posição de cada uma das oito peças e do espaço vazio em um dos nove quadrados.

- **Estado inicial:**
  - Qualquer estado pode ser designado como estado inicial.



Miniproblemas

- Problema do Quebra-cabeça de 8 Peças - Formulação

- **Função Sucessor:**
  - Gera os estados válidos que resultam da tentativa de executar as quatro ações (o espaço vazio se desloca para a *Esquerda*, *Direita*, *Acima* ou *Abaixo*).

- **Estado objetivo:**
  - Qualquer estado pode ser designado como estado final.

Miniproblemas

- Problema do Quebra-cabeça de 8 Peças - Formulação

- **Abstração incluídas**
  - As ações são reduzidas a seus estados iniciais e finais, ignorando-se as posições intermediárias por onde o bloco está deslizando.
  - Foram abstraídas ações como sacudir o tabuleiro quando as peças ficam presas ou extrair as peças com uma faca e colocá-las de volta no tabuleiro.

Miniproblemas

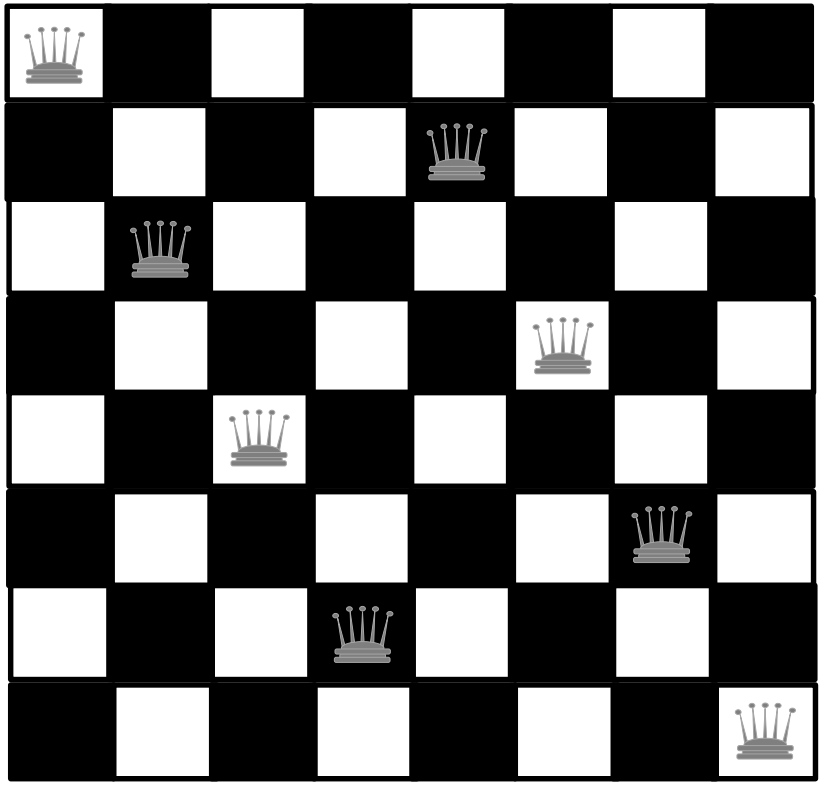
- Problema do Quebra-cabeça de 8 Peças - Formulação

- Pertence à família de **quebra-cabeças de blocos deslizantes** – usados com frequência como problemas de teste para novos algoritmos de busca em IA.

- **Número de estados acessíveis**
  - Quebra-cabeça de 8 peças: 9!/2 = 181.440
  - Quebra-cabeça de 15 peças (tabuleiro 4 x 4) possui aproximadamente 1,3 trilhão (instâncias aleatórias podem ser resolvidas de forma ótima em alguns ms pelos melhores algoritmos de busca).
  - Quebra-cabeça de 24 peças (tabuleiro 5 x 5) possui cerca de 10^{25} estados. Não conhecemos algoritmo que resolva de forma ótima em tempo viável

Miniproblemas

- **Exemplo 3**: Problema das 8 rainhas: O objetivo é posicionar as 8 rainhas no tabuleiro de tal forma que nenhuma rainha possa atacar qualquer uma das outras .



Miniproblemas

- Problema das 8 rainhas - Formulação

- **Estados:**
  - Qualquer disposição de 0 a 8 rainhas no tabuleiro.

- **Estado inicial:**
  - Nenhuma rainha no tabuleiro.

- **Função Sucessor:**
  - Colocar uma rainha em qualquer quadrado vazio.

Miniproblemas

- Problema das 8 rainhas - Formulação

- **Teste de objetivo:**
  - As 8 rainhas estão posicionadas no tabuleiro e nenhuma delas pode ser atacada por outra.

- As combinações possíveis formam um conjunto de *64x63x … x57*, que nos remete a aproximadamente $3x10^{14}$ sequências possíveis.

Miniproblemas

- Problema das 8 rainhas - Simplificação

- Considerando que uma nova rainha só pode ocupar um quadrado que não esteja sobre ataque. Neste caso reduzimos as possibilidades para apenas 2057, que é facilmente resolvido.

- **Estado:**
  - Disposições de n rainhas, uma por coluna nas n colunas mais à esquerda, sem que nenhuma rainha ataque a outra.

- **Função sucessor:**
  - Adicione uma rainha a qualquer quadrado na coluna vazia mais à esquerda de tal modo que ela não seja atacada por qualquer outra rainha.

Problemas no mundo real

- São aqueles cujas soluções de fato preocupam as pessoas.
- Eles tendem a não representar uma única descrição consensual, mas tentaremos dar uma ideia geral de suas formulações.
- Sua descrição é normalmente complexa.

Problemas no mundo real

- **Exemplo**: Problema de roteamento.

- É um problema clássico com várias aplicações reais, tais como rotas de pacotes na internet, rotas de aviões em aeroportos e passagem de trilhas em placas de circuitos.

- Vamos analisar uma versão simplificada deste problema.

Problemas no mundo real

- **Estados**: Cada um é representado por uma posição (por exemplo das aeronaves) e pela hora atual.
- **Estado Inicial**: Situação das aeronaves na hora 00:00am.
- **Função sucessor**: Definida conforme movimento previsto para as aeronaves ao longo do dia.
- **Teste de objetivo**: Os voos devem chegar e partir no horário correto.
- **Custo de caminho**: Consumo de recursos (combustível, tempo de pessoal, etc.).

Problemas no mundo real

- **Exemplo 2:** Diagnóstico médico.

- Em um sistema de diagnóstico médico, podemos partir de um conjunto de perguntas que serão respondidas por exames clínicos e laboratoriais, por exemplo:
  - O Cliente tem febre?
  - Está com algum sintoma na pele?
  - Reclama de alguma dor?
  - Está com catarro no pulmão?
  - Quais os valores máximo e mínimos da pressão arterial?
  - Qual o valor do colesterol HDL?

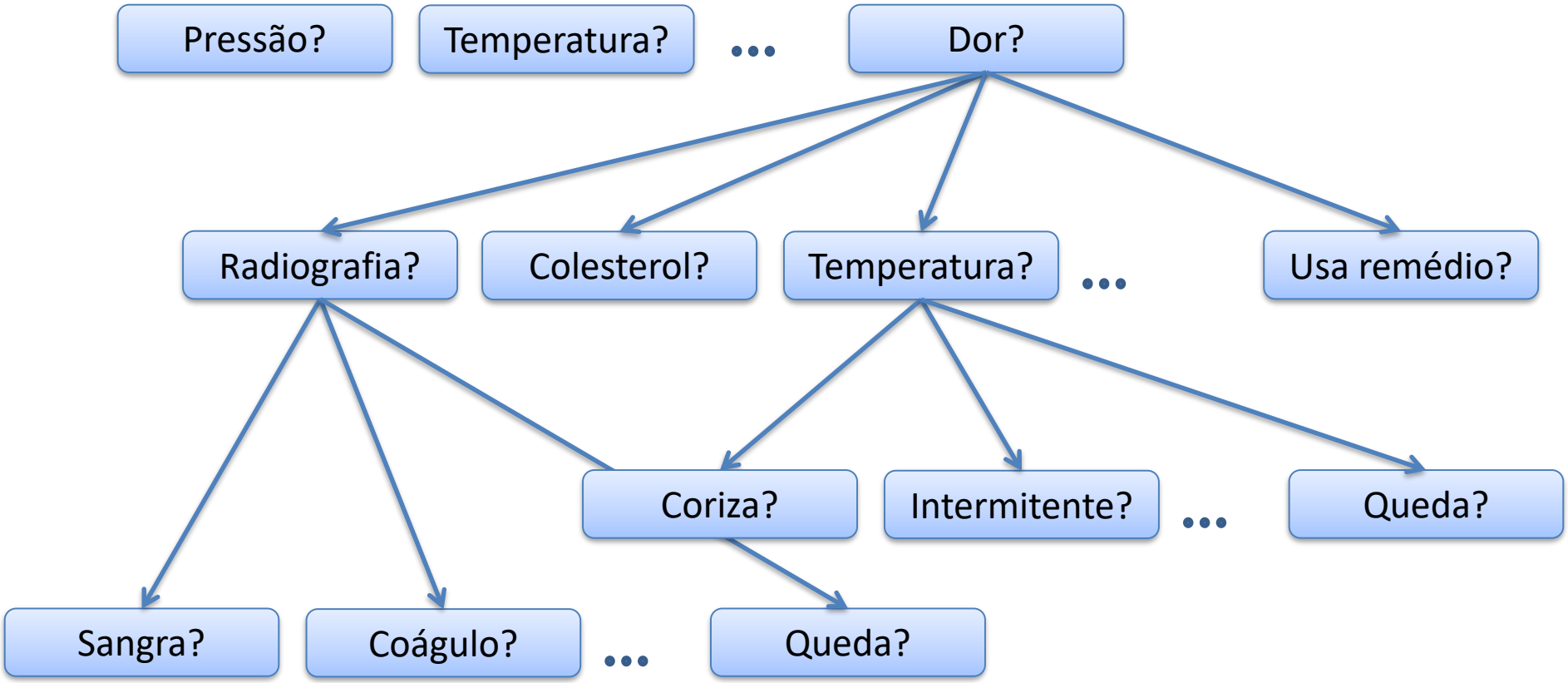
- Também podemos observar que determinadas perguntas possuem relação entre si, formando uma arvore.

Problemas no mundo real

- **Estados**: Cada pergunta que é feita ocupa um estado.
- **Estado Inicial**: Desconhecimento total da condição do paciente.
- **Função sucessor**: Dependendo da pergunta pode ser sim ou não (ex: o paciente tem febre), também pode ser limites de um exame (ex: qual o valor da preção arterial máxima).
- **Teste de objetivo**: Encontrar o diagnostico correto para a doença do paciente.
- **Custo de caminho**: Novamente depende da pergunta, pode ser minutos do médico ou valores cobrados pelo laboratório.

Problemas no mundo real

- Montagem da árvore



Estratégias de Busca

Porque busca?

"Um **agente** com várias opções imediatas de valor desconhecido pode decidir o que fazer examinando, em primeiro lugar, diferentes possíveis sequencias de ações que levam a estados de valores conhecidos e, em seguida, escolher a melhor sequência de ações “

- A busca simula ações no mundo com certo nível de abstração para entender como proceder ou resolver um problema.

Agentes de Resolução de Problemas

- Um tipo de agente baseado em objetivo

Estratégias de busca

- **Busca:**

- Um agente com várias opções imediatas pode decidir o que fazer comparando diferentes sequencias de ações possíveis.

- Esse processo de procurar pela melhor sequencia é chamado de busca.

Formular objetivo → buscar → executar

Estratégias de busca

- **Abordagens de busca básicas num espaço de estados:**

- **Busca Cega** (Sem informação/Não informada)
  - Não tem informação sobre qual sucessor é mais promissor para atingir a meta.

- **Busca Heurística** (Busca Com Informação/Informada)
  - Possui informação (estimativa) de qual sucessor é mais promissor para atingir a meta.
  - É uma busca cega com algum guia ou orientação.

- Todas as estratégias de busca se distinguem pela ordem em que os nós são expandidos.

Busca Cega

Estratégias de busca cega

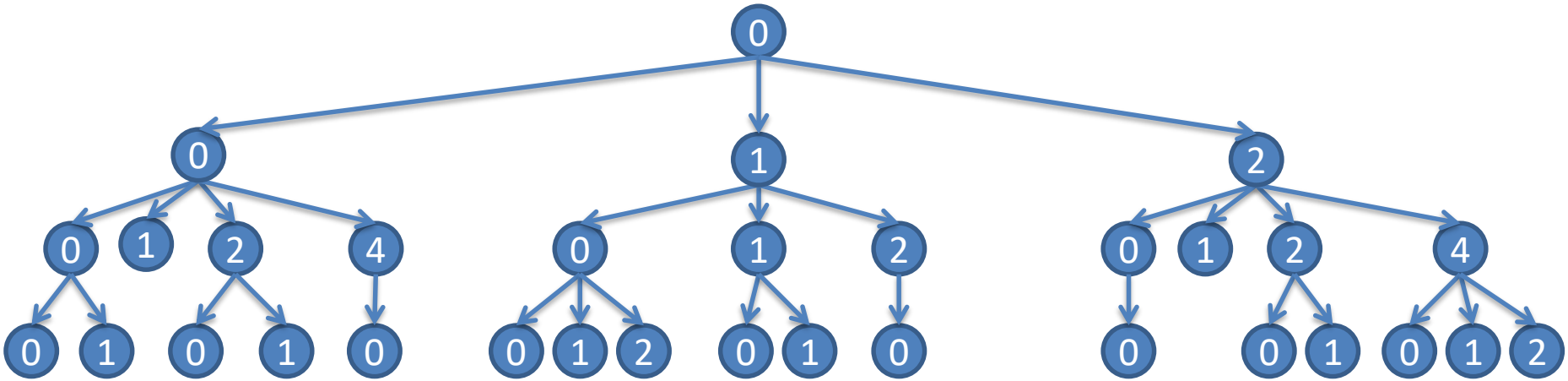
- **Busca Cega** (Blind Search ou Uninformed Search)

- Uma estratégia de busca é dita cega se ela não leva em conta informações específicas sobre o problema a ser resolvido.

- Tipos de Busca Cega
  - Busca em largura
  - Busca pelo custo uniforme
  - Busca em profundidade
  - Busca em profundidade limitada
  - Busca por aprofundamento iterativo
  - Busca bidirecional

Estratégias de busca cega

- **Busca em Largura (BrFS – *Breadth-first search*)**
  - Consiste em construir uma árvore de estados a partir do estado inicial, aplicando a cada momento, todas as regras possíveis aos estados do nível mais baixo, gerando todos os estados sucessores de cada um destes estados. Assim, cada nível da árvore é completamente construído antes de adicionar qualquer nó do próximo nível à árvore.



Estratégias de busca cega

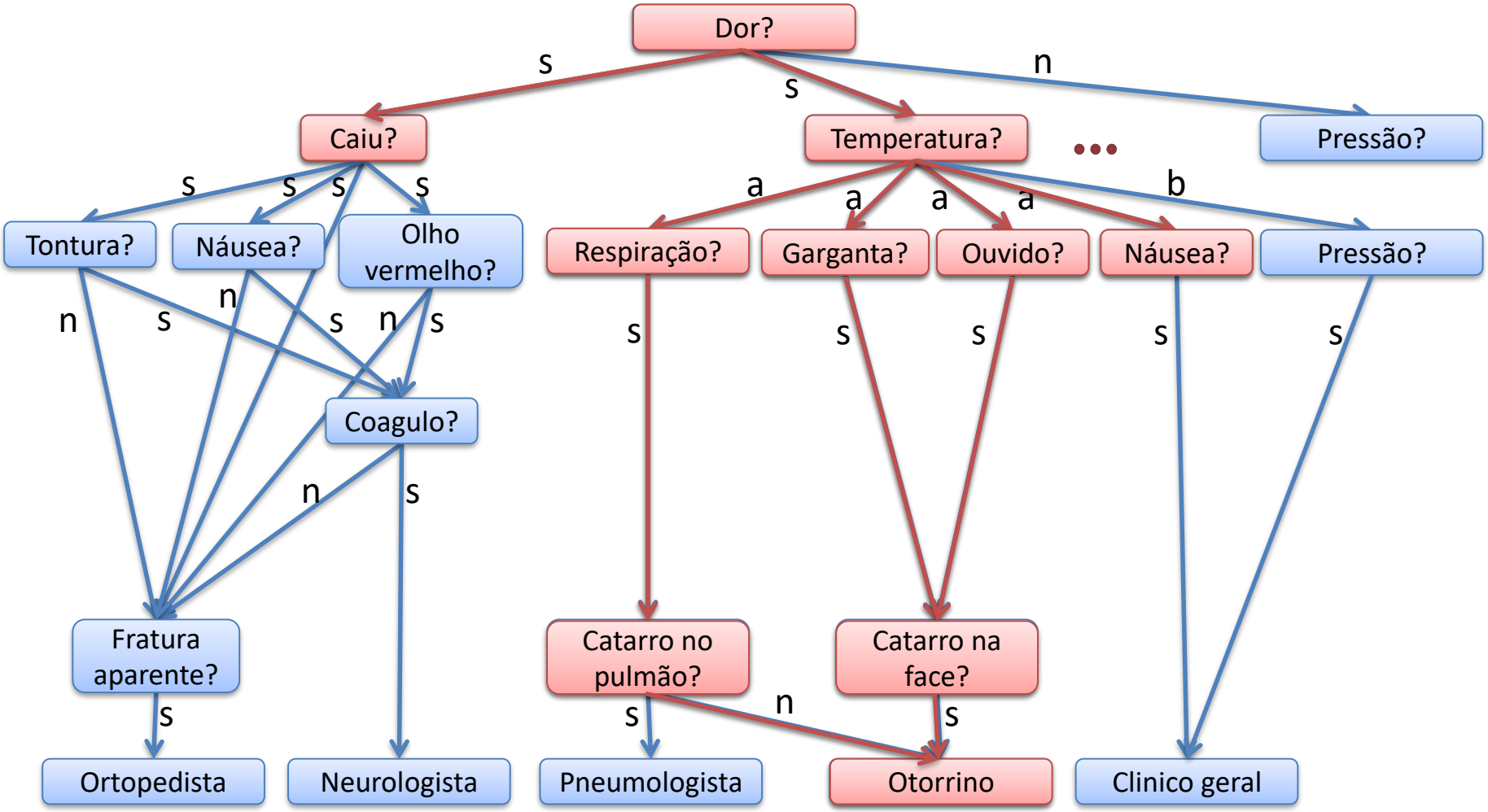
- **Busca em Largura**

Ordem de expansão dos nós:

1. Nó raiz
2. Todos os nós de profundidade 1
3. Todos os nós de profundidade 2, etc …

Estratégias de busca cega

- **Busca em Largura:** Exemplo do pré-atendimento no pronto socorro.



Estratégias de busca cega

- **Busca em Largura**

- Características: Completa e Ótima
  - Se existe solução, esta será encontrada;
  - A solução encontrada primeiro será a de menor profundidade.

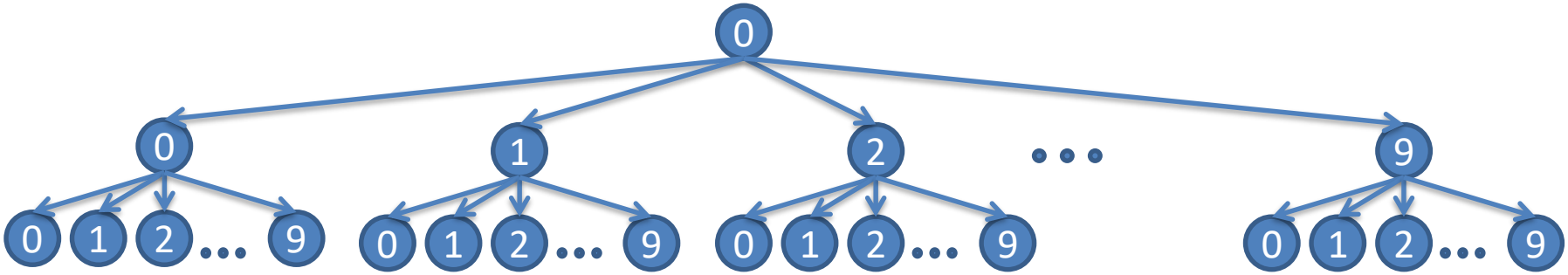
- Vantagens:
  - Completo
  - Ótimo, sob certas condições (por exemplo, é ótimo se os operadores sempre têm o mesmo custo).

- Desvantagens:
  - Requer muita memória e tempo (complexidade exponencial). Todos os caminhos selecionados possuem a mesma importância.

Estratégias de busca

- **Busca em Largura**

- Suponha uma árvore onde cada nó pode ser derivado em outros 10 nós.



- O número de nós na primeira linha é 10. Na segunda linha, são 10 nós filhos para cada um dos pais, assim teremos 10^2=100 nós. Na terceira teremos 10 nós filhos para cada um dos 100 nós pais, ou seja, 10^3=1000, e assim sucessivamente.
- Assim, o número total de nós é a soma do número total de nós de cada camada, ou seja, se tivermos 3 camadas, o número total de nós é de 10+100+1000=1110.

Estratégias de busca cega

- **Busca em Largura**

- Assim,
  - Para um fator de ramificação de 10;
  - Tempo de processamento de um milhão de nós/segundo;
  - Consumo de memória de 1000 bytes/nó.

Profundidade	Nós	Tempo	Memória
2	110	0,11 milissegundos	107 KB
4	11.110	11 milissegundos	10,6 MB
6	10^6	1,1 segundo	1 GB
8	10^8	2 minutos	103 GB
10	10^{10}	3 horas	10 TB
12	10^{12}	13 dias	1 PB
14	10^{14}	3.5 anos	99 PB
16	10^{16}	350 anos	10 EB

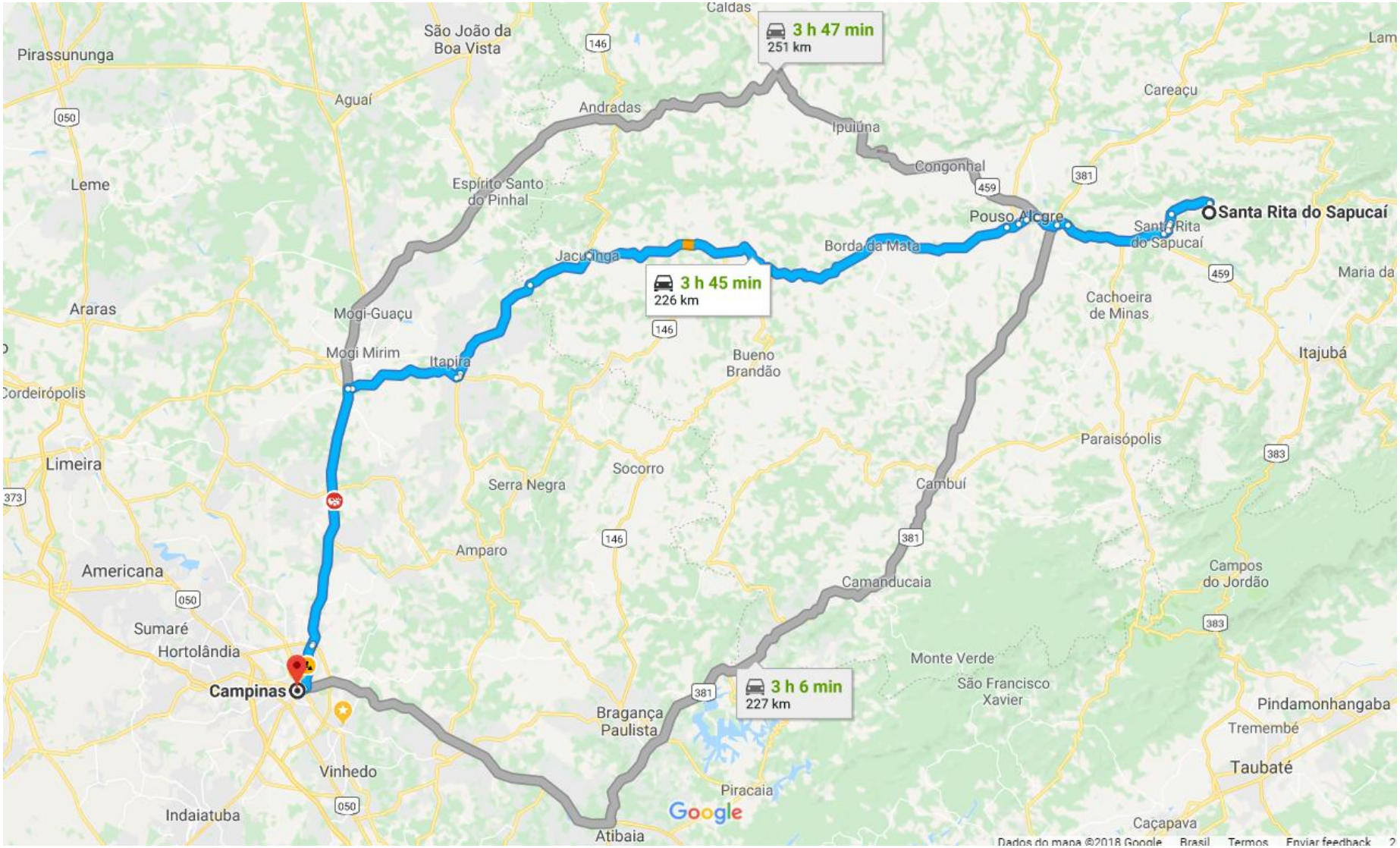
Estratégias de busca cega

- **Busca em Largura:** Exemplo de algoritmo.

```
função BFS (problema) retorna solução/falha
    nó ← um nó com ESTADO = problema.ESTADO_INICIAL, CUSTO_DE_CAMINHO=0;
    se problema.TESTE_DE_OBJETIVO(nó.OBJETIVO) então retorne SOLUÇÃO(nó);
    senão
        borda ← uma fila FIFO com nó como elemento único
        explorado ← conjunto vazio
        repita
            se VAZIO?(borda) então retorne falha;
            nó ← POP(borda) /*escolhe o nó mais raso na borda*/
            adicione nó.ESTADO para explorado
            para cada ação em problema.AÇÕES(nó.ESTADO) faça
                filho ← NÓ_FILHO(problema, nó, ação)
                se (filho.ESTADO) não está em explorado ou borda então
                    se problema.TESTE_DE_OBJETIVO(filho.ESTADO)
                        então retorne SOLUÇÃO(filho);
                        senão borda ← INSIRA(filho , borda)
```

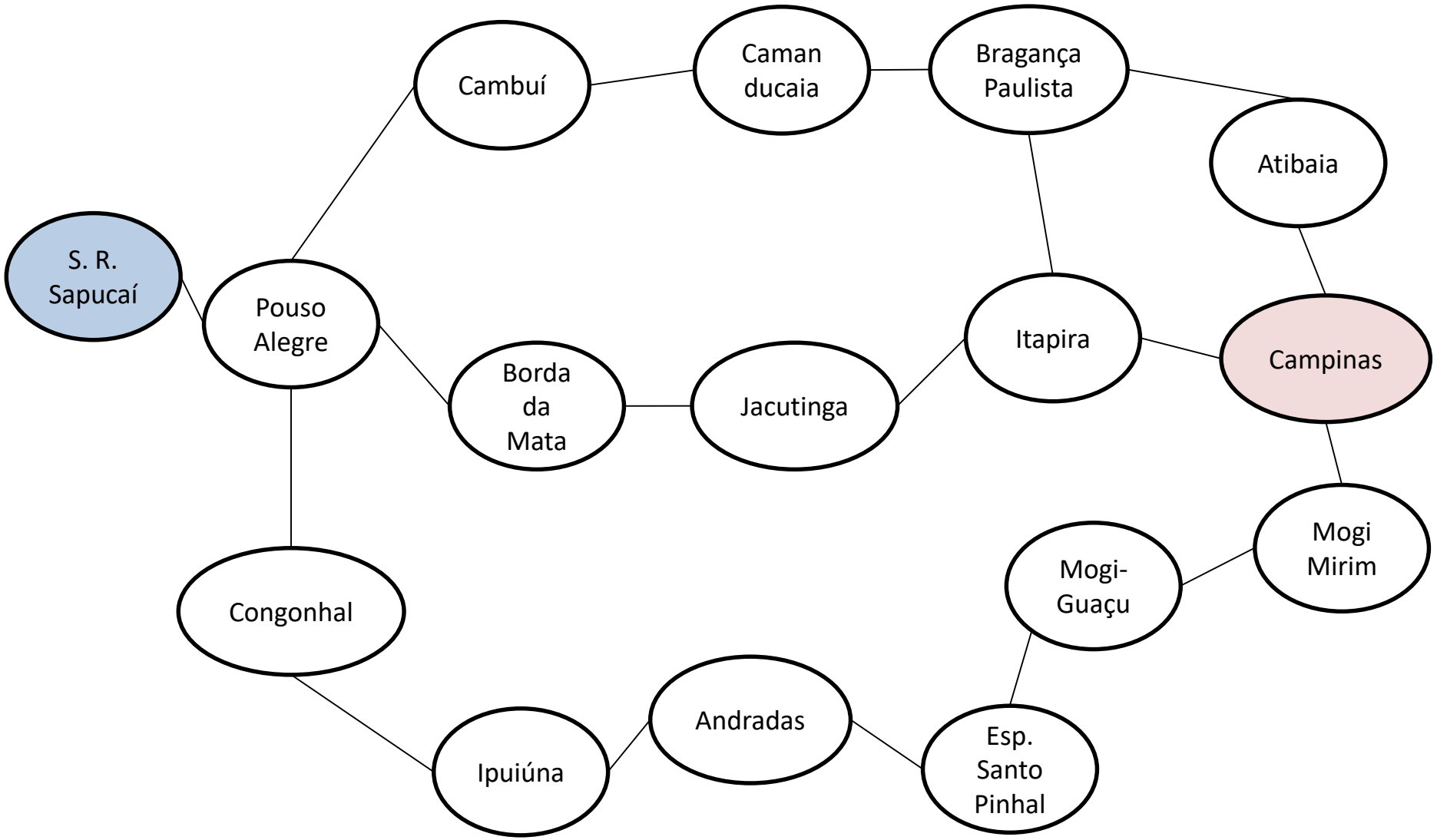
Exemplo – busca em largura

- **Busca em Largura:** Mapa das cidades (**origem**: S.R.Sapucaí, **destino**: Campinas)



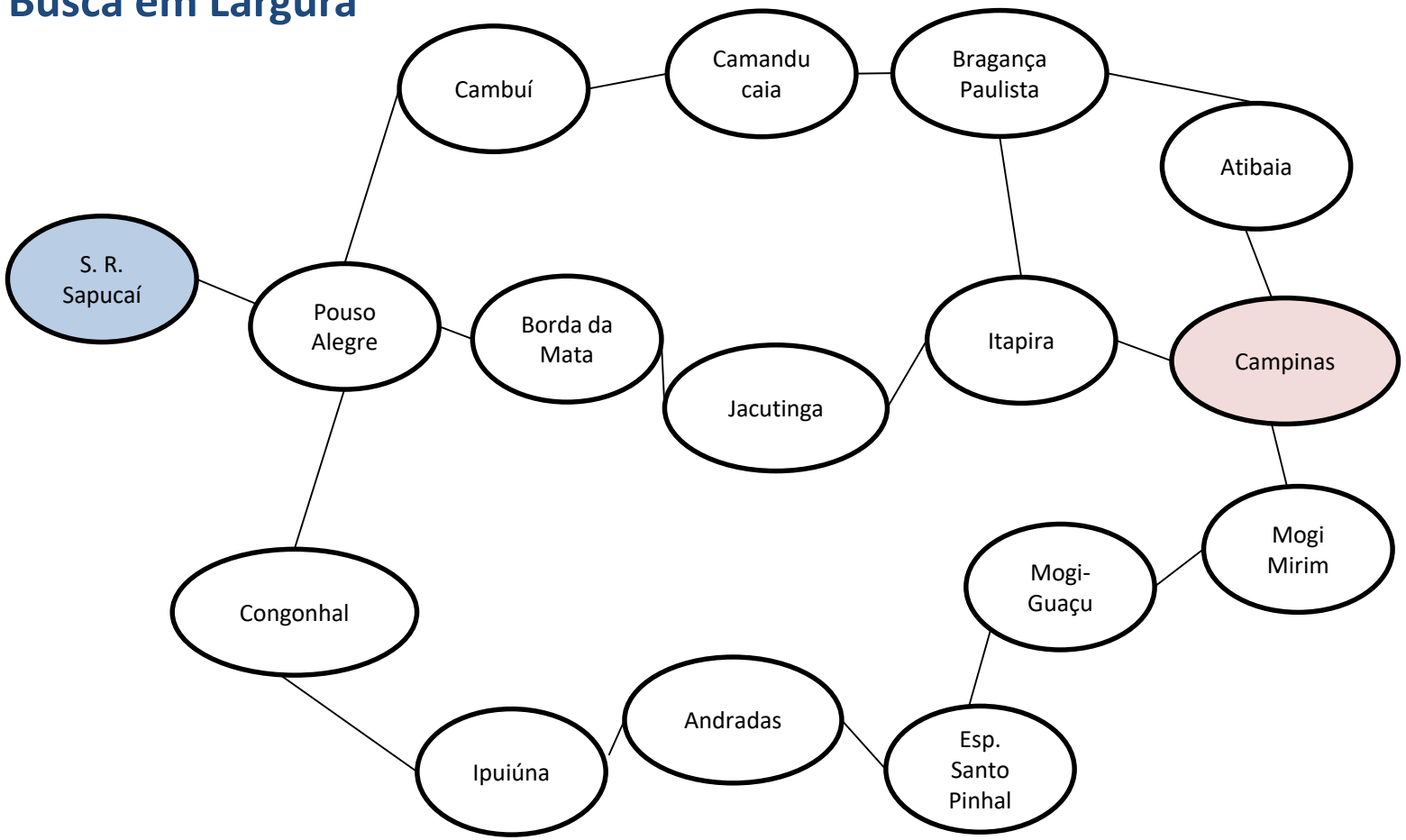
Exemplo – busca em largura

- **Busca em Largura**



Exemplo – busca em largura

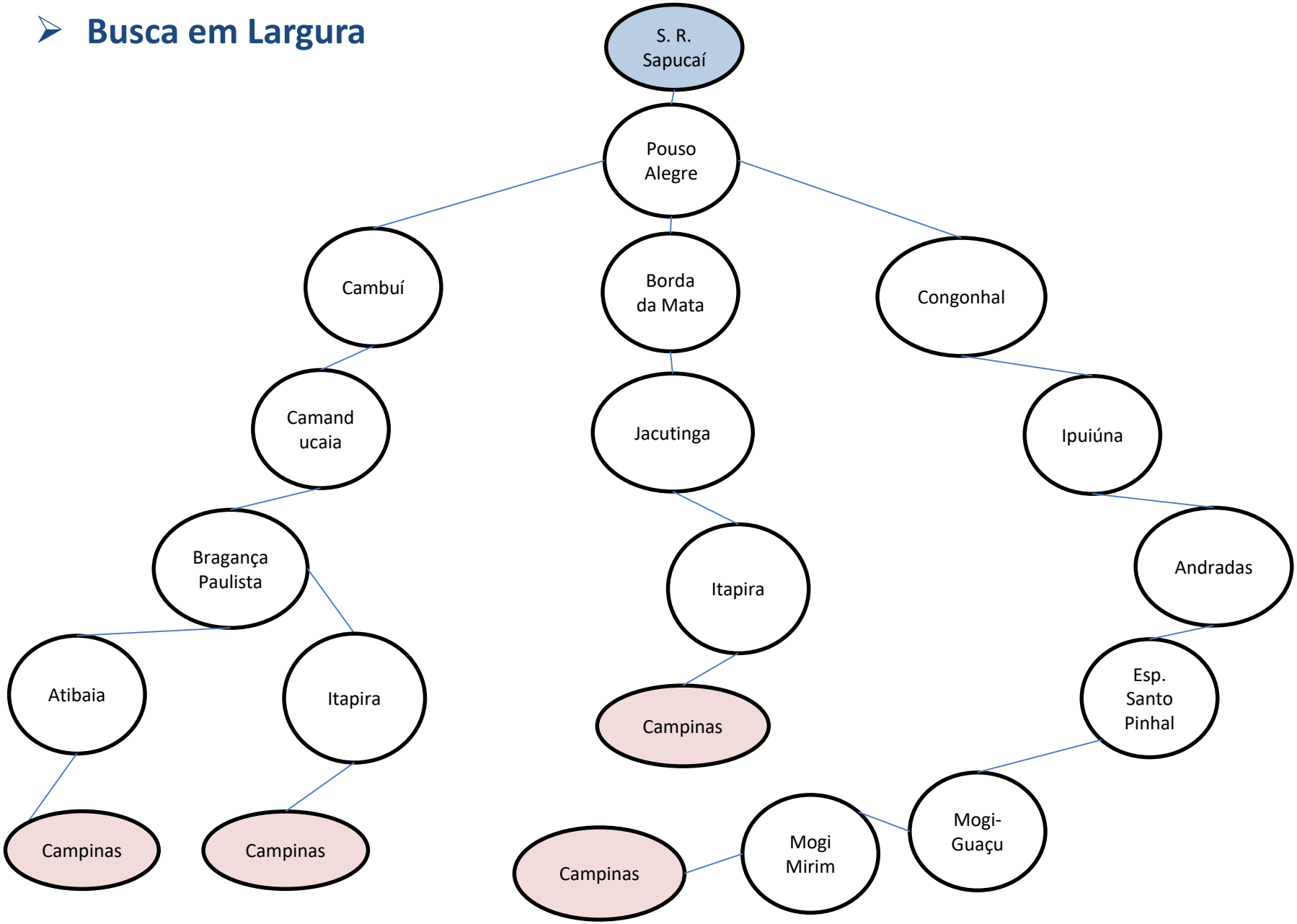
- **Busca em Largura**



S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanduc aia	Jacutinga	Ipuiúna	Bragança Paulista	Itapira	Andradas	Atibaia	Campinas	Esp. Santo Pinhal	

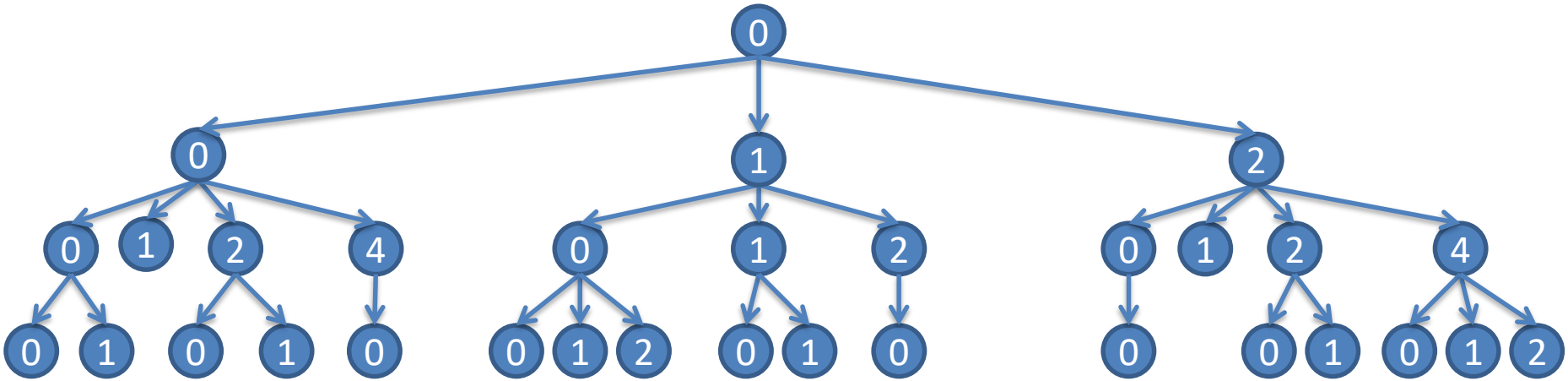
Exemplo – busca em largura

- **Busca em Largura**



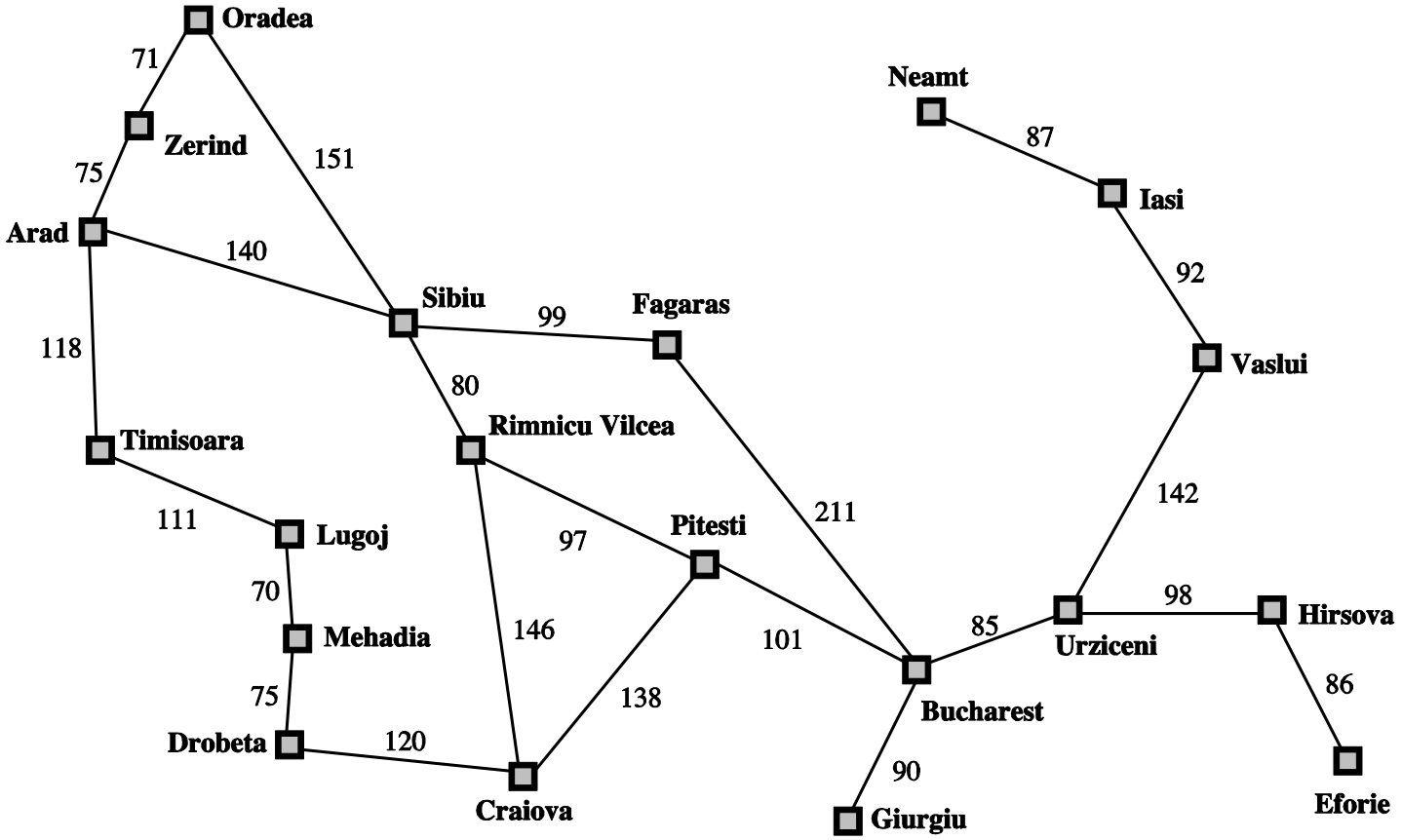
Estratégias de busca cega

- **Busca em Profundidade (DFS – *Depth-first search*):**
  - Procurar explorar completamente cada ramo da árvore antes de tentar o ramo vizinho.
  - O que acontece quando nenhuma regra pode ser aplicada, ou a árvore atinge uma profundidade muito grande sem que tenha encontrado uma solução?
    - Neste caso ocorre o **BACKTRACKING**, ou seja, o algoritmo "volta atrás" e tenta outro caminho.



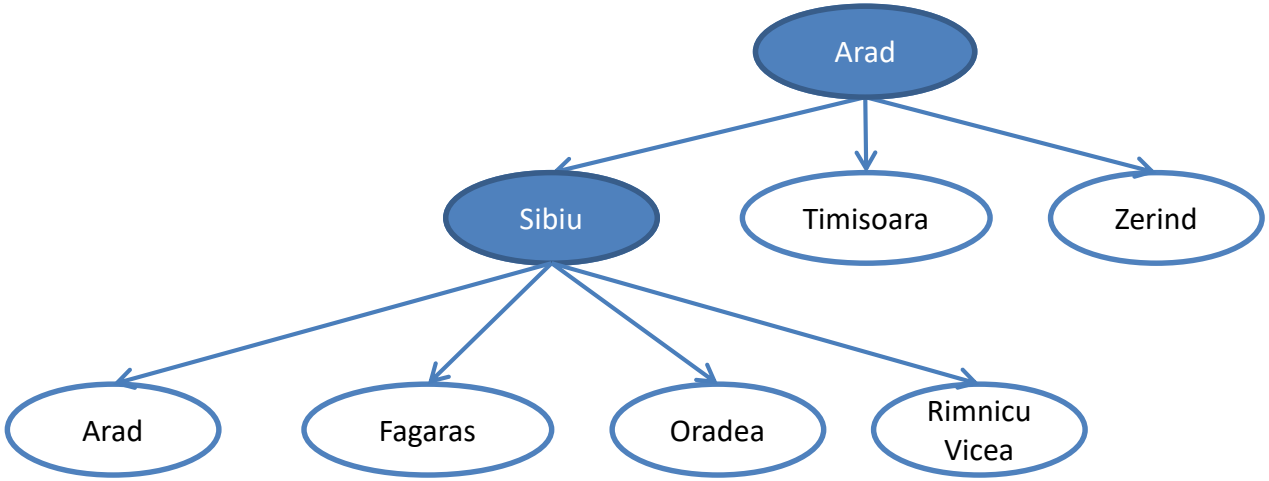
Estratégias de busca cega

- **Busca em Profundidade:** Exemplo do mapa rodoviário de parte da Romênia.



Estratégias de busca cega

- **Busca em Profundidade:** Exemplo do mapa rodoviário de parte da Romênia.



Estratégias de busca cega

- **Busca em Profundidade:**

- Características: Não é Completa e Não é Ótima

  - Realizando uma busca em árvore onde um nó já visitado pode ser visitado novamente, poderia gerar um laço infinito, como no caso Arad-Sibiu-Arad;

  - Se admitir estados repetidos ou um nível máximo de profundidade, pode nunca encontrar a solução.

  - O algoritmo não encontra necessariamente a solução ótima, mas pode ser MAIS EFICIENTE se o problema possui um grande número de soluções ou se a maioria dos caminhos pode levar a uma solução.

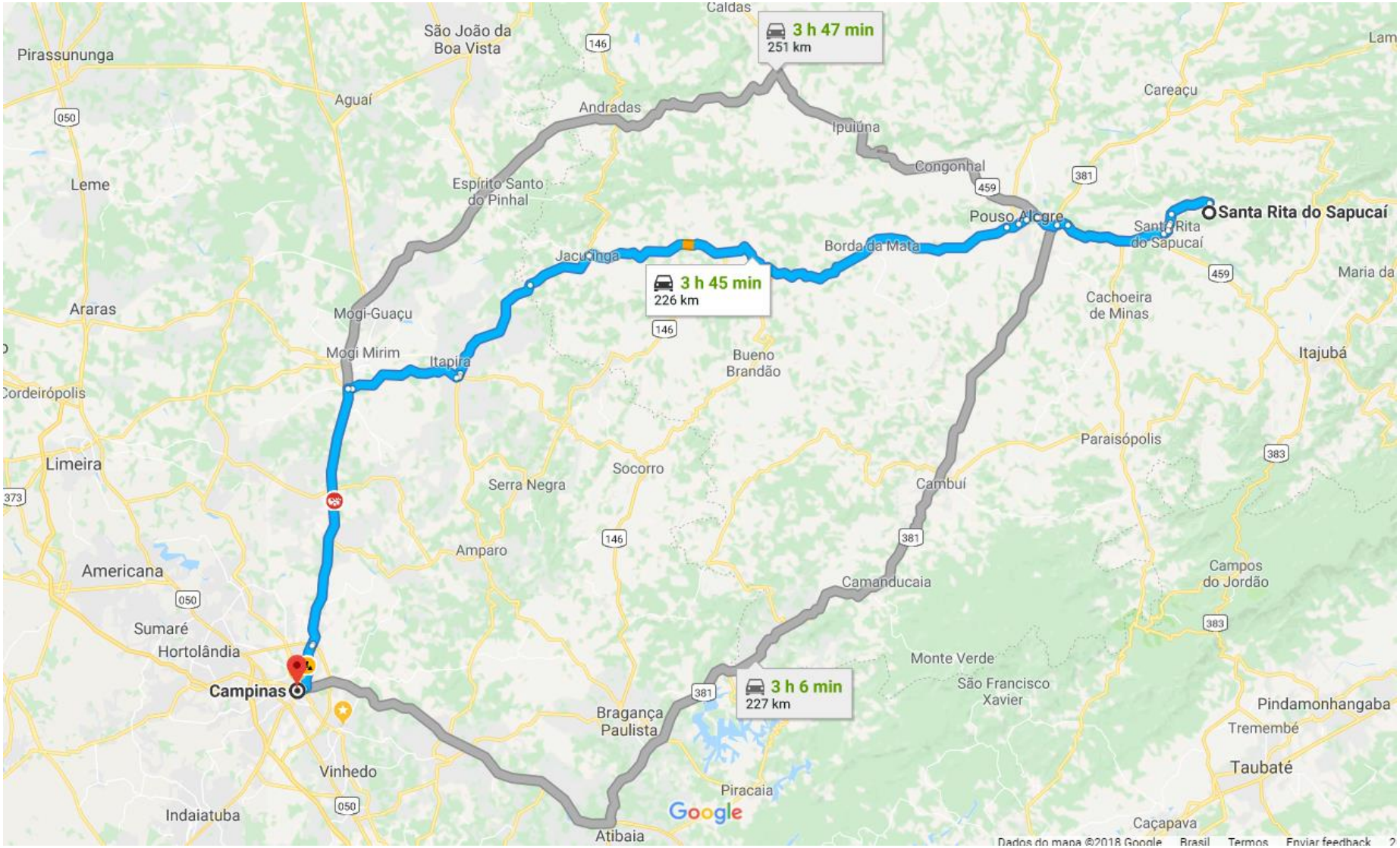
Estratégias de busca cega

- **Busca em Profundidade:** Exemplo de algoritmo.

```
função BPL-RECURSIVA(raiz nó; problema, limite inteiro) retorna solução/falha
declara
    resultado    nó;
    conta        inteiro;
inicio
    se ESTADO[raiz]==problema então retorna SOLUÇÃO[raiz];
    senão se PROFUNDIDADE[raiz]==limite então retorna falha;
    senão
        resultado ← falha;
        para conta ← 1 até MAX-FILHO[raiz] faça
            resultado ← BPL-RECURSIVA(FILHO[raiz][conta], problema, limite);
            se resultado!=falha retorna resultado;
        fim para.
    fim senão.
    retorna falha;
fim.
```

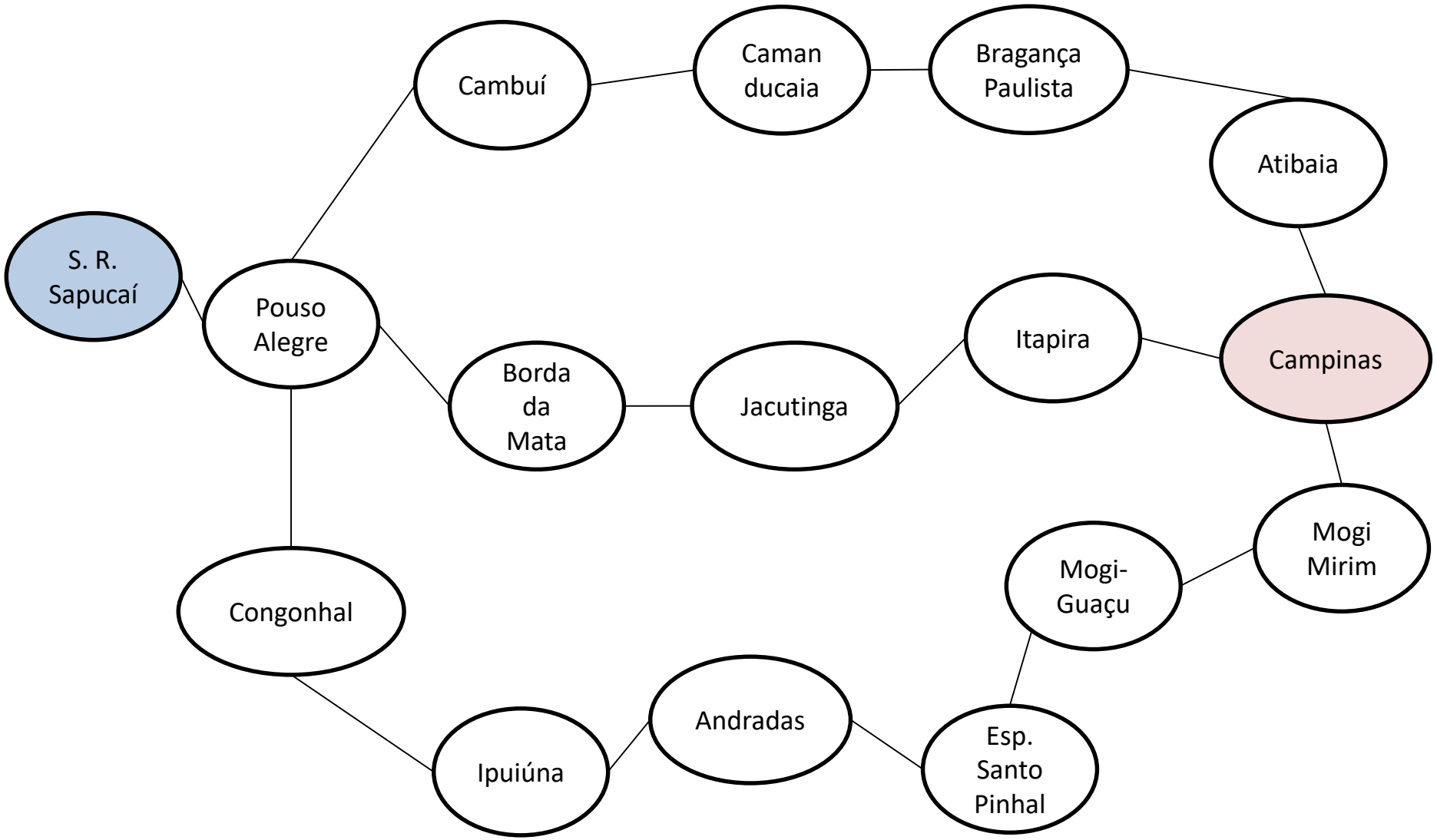
Exemplo – busca em profundidade

- **Busca em Largura:** Mapa das cidades (**origem**: S.R.Sapucaí, **destino**: Campinas)



Exemplo – busca em profundidade

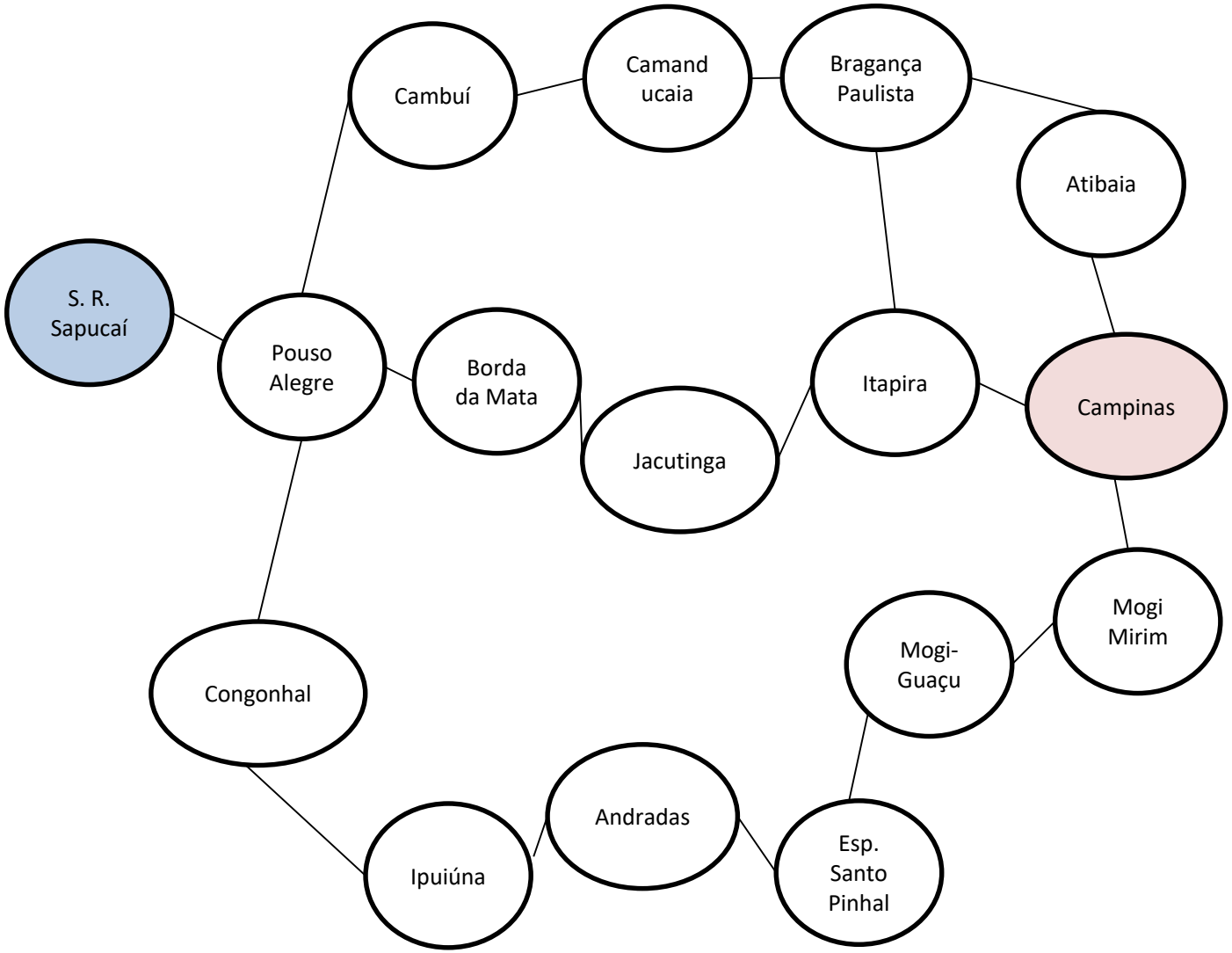
- **Busca em Profundidade**



Exemplo – busca em profundidade

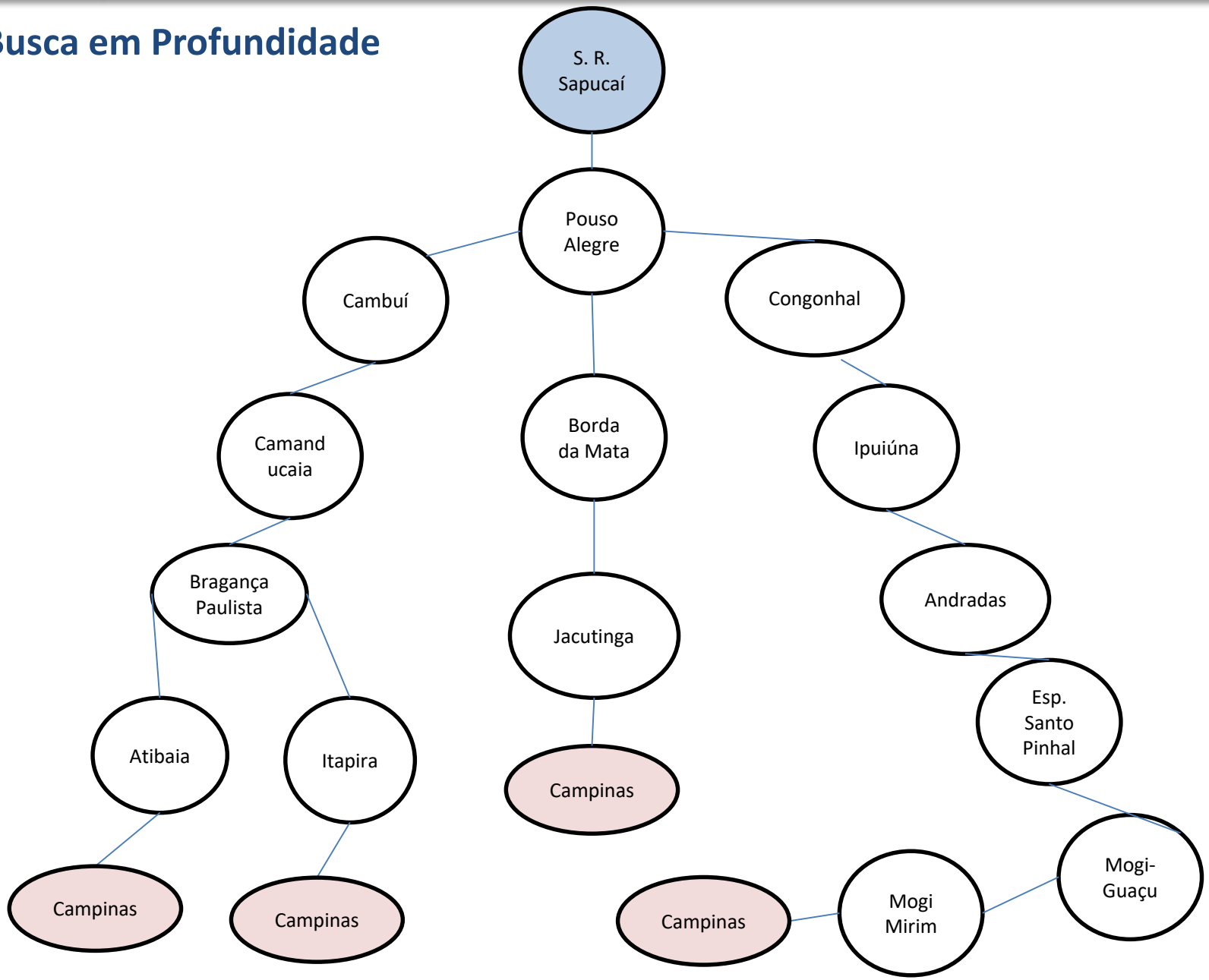
- **Busca em Profundidade**

Campinas
Atibaia
Bragança Paulista
Camanducaia
Cambuí
Pouso Alegre
S. R. Sapucaí



Exemplo – busca em profundidade

- **Busca em Profundidade**



Estratégias de busca cega

- **Observações:**

- Busca em profundidade utiliza menos memória porque armazena apenas um único caminho do nó raiz até o nó folha;

- Quanto ao tempo, a busca em profundidade é geralmente mais rápida.

- As buscas em largura e profundidade não fazem uso de nenhum conhecimento para encontrar sua solução, fazendo uma busca exaustiva dentro do seu espaço. Para contornar este problema, pode-se usar os **métodos heurísticos**.

Referências

[RUSSELL, S.; NORVIG, P.; **Inteligência Artificial**, 2ºed. – Cap. 1 e 2]

[RUSSELL, S.; NORVIG, P.; **Inteligência Artificial**, 2ºed. – Cap. 3]