Resolução de Problemas de Busca

Ao final desta aula a gente deve...

- Compreender o que é um problema de busca em IA
- Ser capaz de formulá-lo
- Conhecer algumas aplicações
- Entender como buscar a solução do problema
  - Busca Cega
    - Em profundidade
    - Em largura

Um problema de busca em IA pode ser definido em termos de... Algumas definições básicas (1/2)

- Um espaço de estados possíveis, incluindo:
  - um estado inicial
    - Em (Recife)
    - Estar (pobre)
  - um ou mais estados finais => objetivo
    - Em (João Pessoa)
    - Estar (rico)
  - Espaço de Estados:
    - conjunto de todos os estados alcançáveis a partir do estado inicial por qualquer seqüência de ações.
    - pode ser representado como uma árvore onde os estados são nós e as operações são arcos.
  - Ex., dirigir de Recife a João Pessoa
    - Espaço de estados: todas as cidades da região

Um problema de busca em IA pode ser definido em termos de... Algumas definições básicas(2/2)

- Um conjunto de ações (ou operadores) que permitem passar de um estado a outro
  - Ex., dirigir de Recife a João Pessoa
    - Ações: dirigir de uma cidade a outra na região
    - E.g. Dirigir (Recife, Abreu e Lima)
  - Ficar rico
    - Jogar(megasena).

Um problema de busca em IA pode ser definido em termos de...

- Um estado inicial
- Um conjunto de ações (ou operadores)
  - que permitem passar de um estado a outro
  - os estados podem não “estar lá” concretamente.
    - No caso do problema de dirigir... as cidades estão lá
    - No caso de ficar rico... não necessariamente.
- Um teste de término
  - Verifica se um dado estado é o objetivo
  - Objetivo => um ou mais estados finais
    - propriedade abstrata (em intenção)
      - ex., condição de xeque-mate no Xadrez
    - conjunto de estados finais do problema (em extensão)
      - ex., estar em João Pessoa

Um problema de busca em IA pode ser definido em termos de...

- Custo de caminho
  - Função que associa um custo a cada caminho possível
  - Um caminho é uma sequência de estados conectados por ações possíveis.
  - Cada ação tem um custo associado
    - O custo de dirigir de Recife a Abreu e Lima, por exemplo, poderia ser a distância entre as duas cidades.

Algumas definições

- Solução
  - caminho (sequência de ações) que leva do estado inicial a um estado final (objetivo).
  - Cuidado! A solução **não** é o estado final!

Solucionando o problema: formulação, busca e execução

- Formulação do problema e do objetivo (manual)
  - quais são os estados e as ações a considerar?
  - qual é (e como representar) o objetivo?
    - Em extensão ou intensão?
- Busca (processo automático)
  - processo que gera/analisa seqüências de ações para alcançar um objetivo
  - solução = caminho entre estado inicial e estado final.
- Execução (manual ou automática)

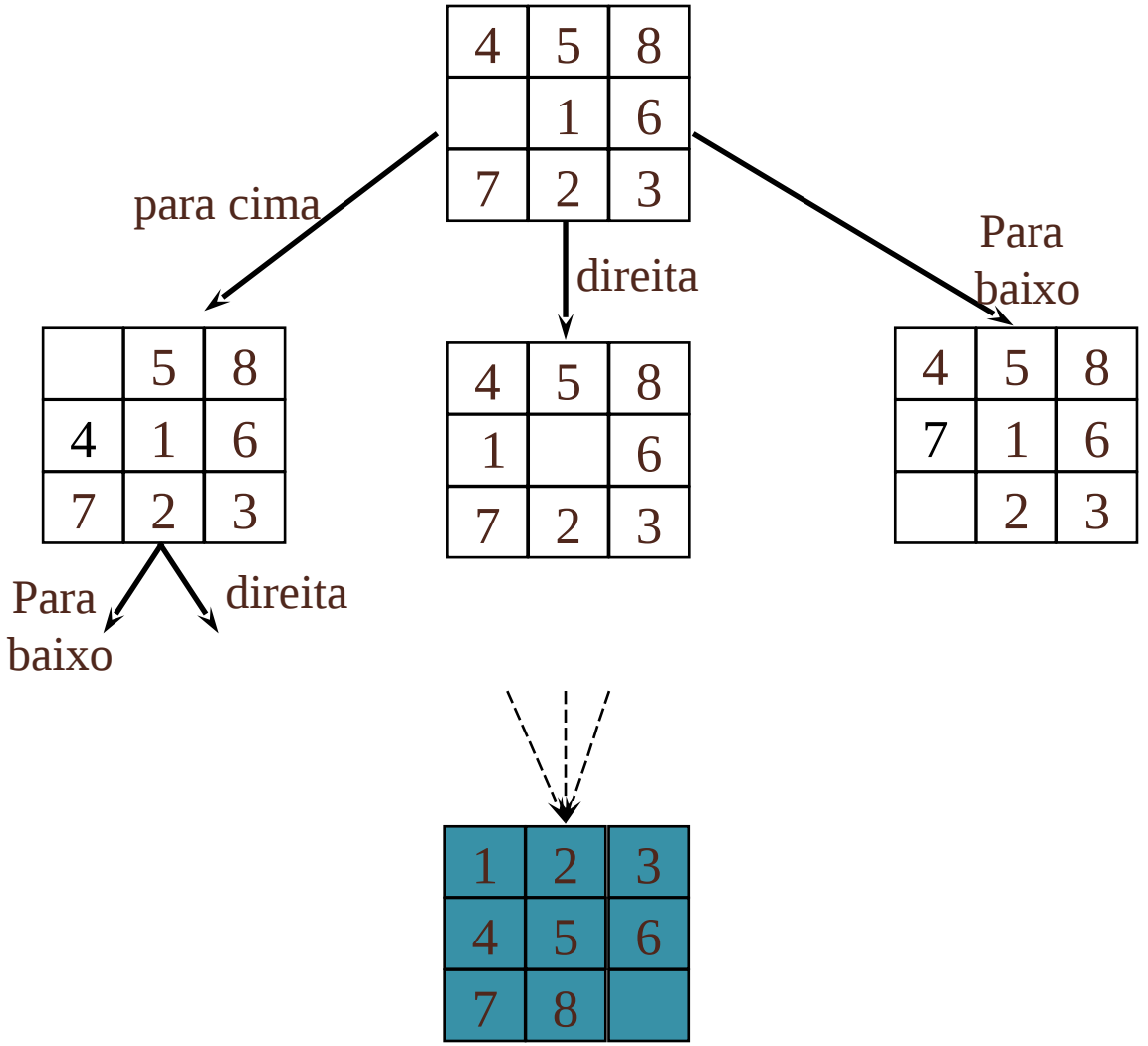
Custo da Busca

- Custo total da busca =
  - custo de busca (tempo e memória, i.e. custo computacional) -> busca da solução
  - + custo do caminho -> execução da solução
- Espaço de estados grande
  - compromisso (conflito) entre determinar
    - a melhor solução em termos de custo do caminho (é uma boa solução?) e
    - a melhor solução em termos de custo computacional (é computacionalmente barata?)

Exemplos de Formulação de problema
Jogo de 8 números

- Espaço de estados = todas as possíveis configurações do tabuleiro
- Estado inicial = qualquer um dos estados possíveis
- Teste de término = tabuleiro ordenado, com branco na posição [3,3]
- Ações/operadores = mover peças numéricas para espaços livres (em branco) (esquerda, direita, para cima e para baixo)
- Custo do caminho = número de passos da solução
- Custo de busca = depende do computador e da estratégia de busca utilizada
  - Próximas aulas

Árvore de busca para o Jogo dos 8 números



Exemplos de formulação de problema

- Dirigir de Recife (PE) a Juazeiro do Norte (CE)
  - Espaço de estados = todas as cidades do mapa alcançáveis a partir do estado inicial
  - Estado inicial = estar em Recife
  - Teste de término (já atingimos o objetivo?) = estar em Juazeiro do Norte
  - Ações/operadores = dirigir de uma cidade para outra (se houver estrada entre elas!)
  - Função Custo do caminho = número de cidades visitadas, distância percorrida, tempo de viagem, grau de divertimento, etc

Custo do caminho diferente => Solução diferente

- Função de *custo de caminho*

  (1) distância entre as cidades

  (2) tempo de viagem, etc.

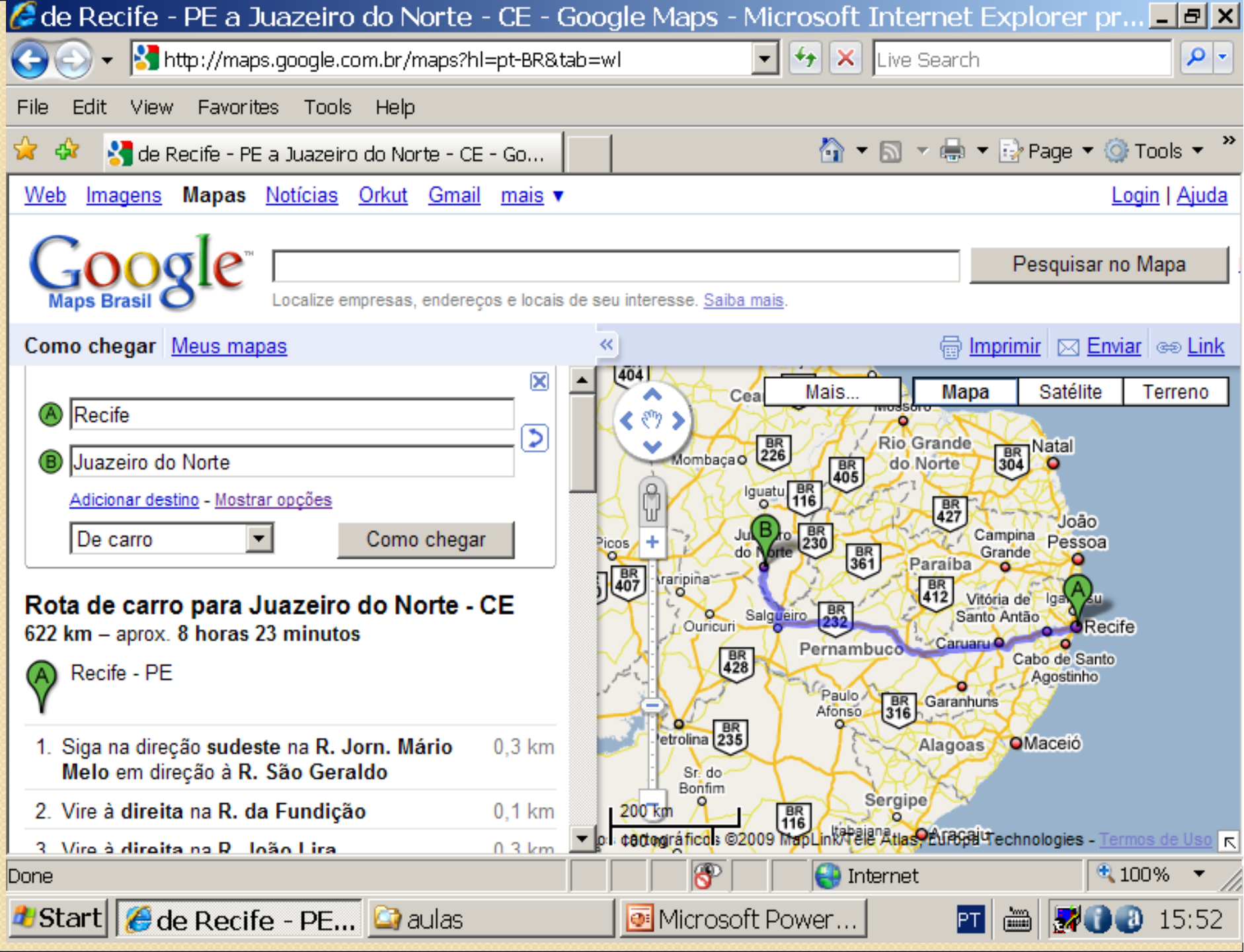
- Solução mais barata:

  (1) Camaragibe, Carpina, Patos, Milagres,...

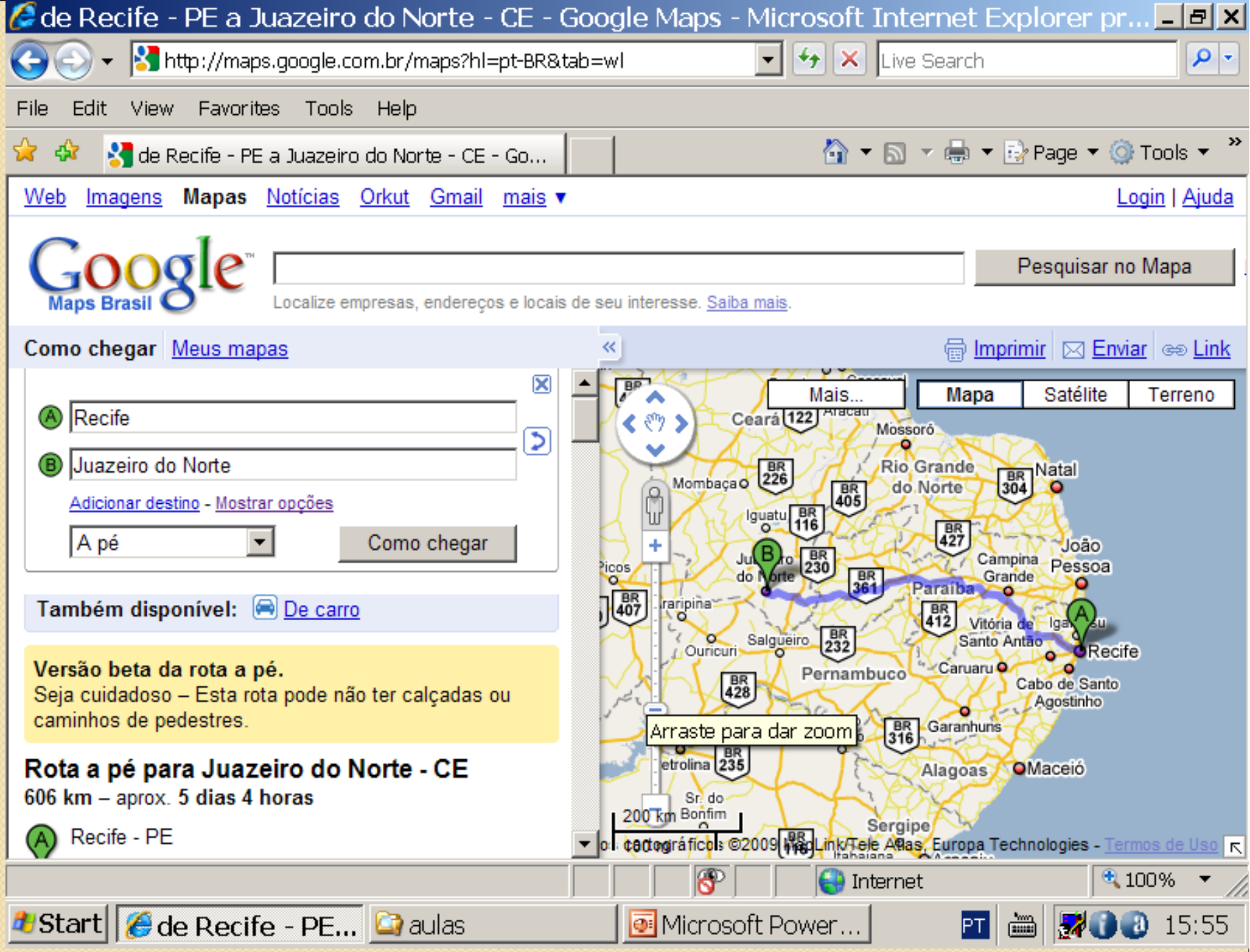
  (2) Moreno, Vitória de S. Antão, Caruaru, Salgueiro,...

  apesar de mais longa, pega estradas melhores e evita as cidades.

Recife - Juazeiro do Norte

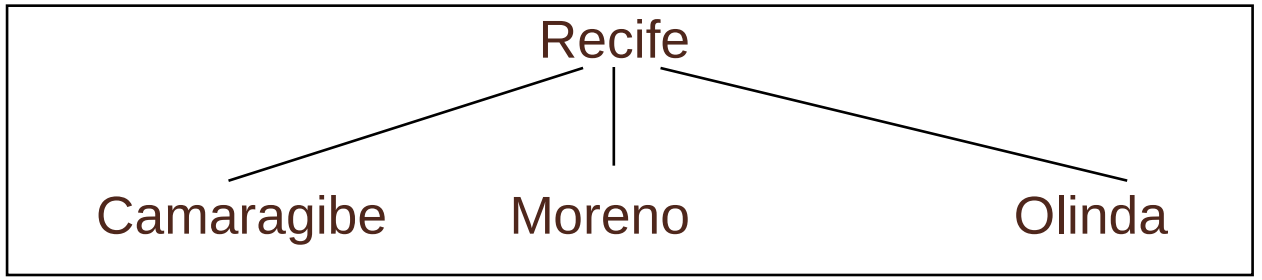


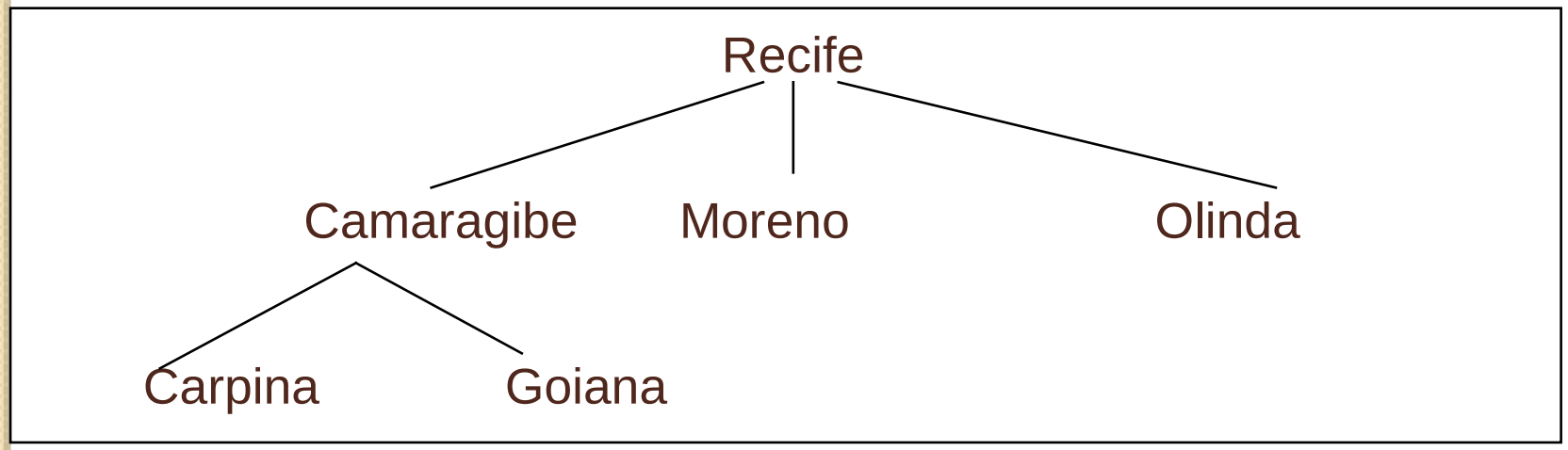
Recife – Juazeiro do Norte



Exemplo: viajar de Recife a Juazeiro

Estado inicial =>



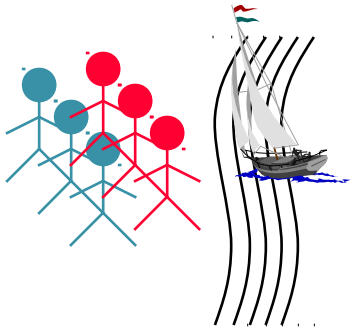


Aplicações de Busca: “Toy Problems”

- Jogo das 8 rainhas
- Jogo dos n números (*n-puzzle*)
- Criptoaritmética
- Palavras cruzadas

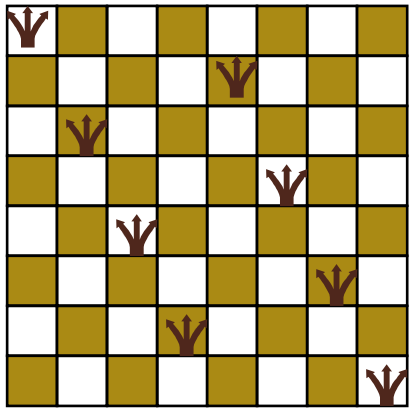
```
  send
+ more
---------
 money
```

- Canibais e missionários



Importância da formulação
Ex.: Jogo das 8 Rainhas

- Objetivo: dispor 8 rainhas no tabuleiro sem possibilitar "ataques"
  - i.e., não pode haver mais de uma rainha em uma mesma linha, coluna ou diagonal



- Existem diferentes estados e operadores possíveis
  - essa escolha pode ter consequências boas ou nefastas na complexidade da busca ou no tamanho do espaço de estados

Formulações para 8 Rainhas

- Formulação A
  - estados: qualquer disposição com n ($n \leq 8$) rainhas
  - operadores: adicionar uma rainha a qualquer quadrado
  - $64x63x...x57 = 3x10^{14}$ possibilidades: vai até o fim para testar se dá certo
- Formulação B
  - estados: disposição com n ($n \leq 8$) rainhas sem ataque mútuo (teste gradual)
  - operadores: adicionar uma rainha na coluna vazia mais à direita em que não possa ser atacada
  - melhor (2057 possibilidades), mas pode não haver ação possível
- Formulação C
  - estados: disposição com 8 rainhas, uma em cada coluna
  - operadores: mover uma rainha atacada para outra casa na mesma coluna

Aplicações de Busca: Problemas Reais

- Cálculo de rotas
  - rotas em redes de computadores
  - sistemas de planejamento de viagens
  - planejamento de rotas de aviões
  - caixeiro viajante
- Alocação (Scheduling)
  - Salas de aula
  - Máquinas industriais (job shop)
- Projeto de VLSI
  - Cell layout
  - Channel routing

Aplicações de Busca: Problemas Reais

- Navegação de robôs:
  - generalização do problema da navegação
  - robôs movem-se em espaços contínuos, com um conjunto (infinito) de possíveis ações e estados
  - controlar os movimentos do robô no chão, e de seus braços e pernas requer espaço multi-dimensional
- Montagem de objetos complexos por robôs:
  - ordenar a montagem das diversas partes do objeto
- etc...

Problemas de Busca

Formulação, Busca e Execução

Algoritmo de Busca

Solucionando o problema: formulação, busca e execução

- Formulação do problema e do objetivo (manual)
  - quais são os estados e as ações a considerar?
  - qual é (e como representar) o objetivo?
- Busca (processo automático)
  - processo que gera/analisa seqüências de ações para alcançar um objetivo
  - solução = caminho entre estado inicial e estado final.
- Execução (manual ou automática)

Busca em Espaço de Estados

- Depois de formular adequadamente o problema, a solução deve ser “buscada” automaticamente
  - Solução: caminho (sequência de ações) que leva do estado inicial a um estado final (objetivo).
- Deve-se usar um método de busca para determinar a (melhor) solução para o problema
- Uma vez a busca terminada com sucesso, é só executar a solução
  - De forma manual ou automática (ex., um robô)

Busca em Espaço de Estados
Algoritmo de Geração e Teste

- **Fronteira** do espaço de estados
  - Lista contendo os nós (estados) a serem expandidos
  - Inicialmente, a fronteira contém apenas o estado inicial do problema
- **Algoritmo:**

1. Selecionar o primeiro nó (estado) da fronteira do espaço de estados;

   - se a fronteira está vazia, o algoritmo termina com falha.

2. Testar se o nó selecionado é um estado final (objetivo):

   - se “sim”, então retornar nó - a busca termina com sucesso.

3. Gerar um novo conjunto de estados aplicando ações ao estado selecionado;

4. Inserir os nós gerados na fronteira, de acordo com a estratégia de busca usada, e voltar para o passo (1).

Busca em Espaço de Estados
Implementação do Algoritmo

- Os nós da fronteira devem guardar mais informação do que apenas o estado:
  - Na verdade nós são uma estrutura de dados com 5 componentes:
    1. o estado (configuração) correspondente ao nó atual
    2. o seu nó pai – ou o caminho inteiro para não precisar de operações extras
    3. a ação aplicada ao pai para gerar o nó – verifica de onde veio para evitar loops
    4. o custo do nó desde a raiz ($g(n)$)
    5. a profundidade do nó – se guardar o caminho não precisa!

Busca em Espaço de Estados
Implementação do Algoritmo

Função-Insere: controla a ordem de inserção de nós na fronteira do espaço de estados.

função **Busca-Genérica (*problema formulado*, Função-Insere)** retorna **uma solução** ou **falha**

fronteira ← Estado-Inicial (*problema)*

loop do

se *fronteira* está vazia então retorna falha

nó ← Remove-Primeiro (*fronteira*)

se Teste-Término (*problema, nó)* tiver sucesso

então retorna *nó*

fronteira ← Função-Insere (*fronteira*, Ações (*nó)*)

end

Métodos de Busca

- Busca exaustiva (cega)
  - Não sabe qual o melhor nó da fronteira a ser expandido
    - i.e., menor custo de caminho desse nó até um nó final (objetivo).
  - Estratégias de Busca (ordem de expansão dos nós):
    - caminhamento em largura
    - caminhamento em profundidade

- Busca heurística (informada)
  - Estima qual o melhor nó da fronteira a ser expandido com base em funções heurísticas => conhecimento
  - Estratégia de busca: *best-first search* (melhor escolha)

Critérios de Avaliação das Estratégias de Busca

- Completude:
  - a estratégia sempre encontra uma solução quando existe alguma?
- Qualidade (“otimalidade” - *optimality*):
  - a estratégia encontra a melhor solução quando existem diferentes soluções?
  - i.e., solução de menor custo de caminho
- Custo do tempo:
  - quanto tempo gasta para encontrar a 1ª solução?
- Custo de memória:
  - quanta memória é necessária para realizar a busca?

Estratégias de Busca Exaustiva (Cega)

- Encontram soluções para problemas pela geração *sistemática* de novos estados, que são comparados ao objetivo;
- São *ineficientes* na maioria dos casos:
  - utilizam apenas o *custo de caminho* do nó atual ao nó inicial (função *g*) para decidir qual o próximo nó da fronteira a ser expandido.
  - essa medida nem sempre conduz a busca na direção do objetivo.
- Como encontrar um barco perdido?
  - não podemos procurar no oceano inteiro...
  - observamos as correntes marítimas, o vento, etc...

Busca Cega (Exaustiva)

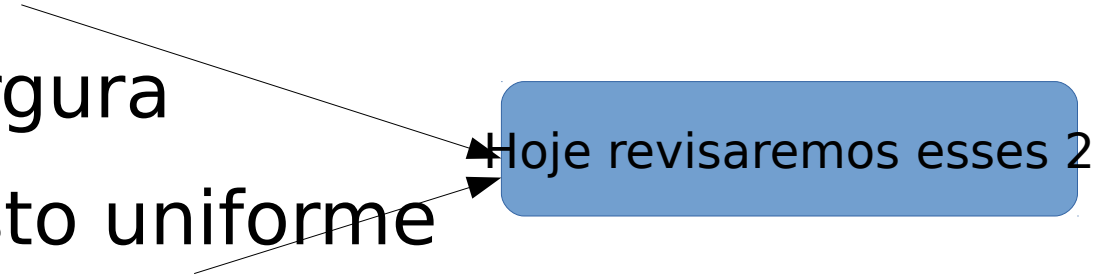
- Estratégias para determinar a ordem de expansão dos nós

1. Busca em largura
2. Busca de custo uniforme
3. Busca em profundidade
4. Busca com aprofundamento iterativo

Busca Cega (Exaustiva)

- Estratégias para determinar a ordem de expansão dos nós

1. Busca em largura
2. Busca de custo uniforme
3. Busca em profundidade
4. Busca com aprofundamento iterativo



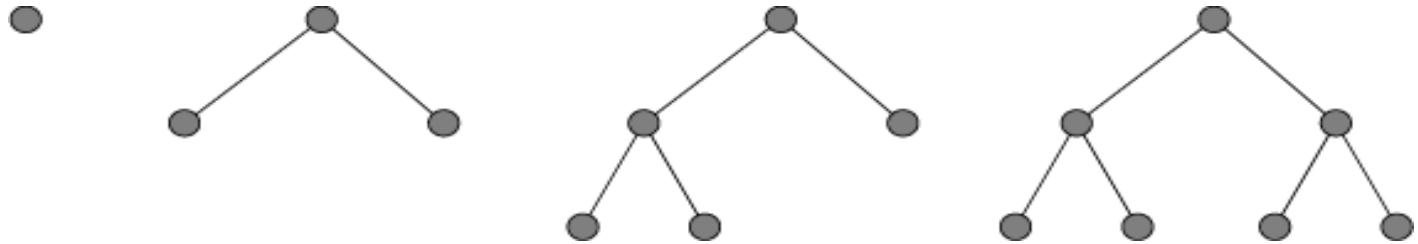
Busca em Largura

- Ordem de expansão dos nós:
  1. Nó raiz
  2. Todos os nós de profundidade 1
  3. Todos os nós de profundidade 2, etc...

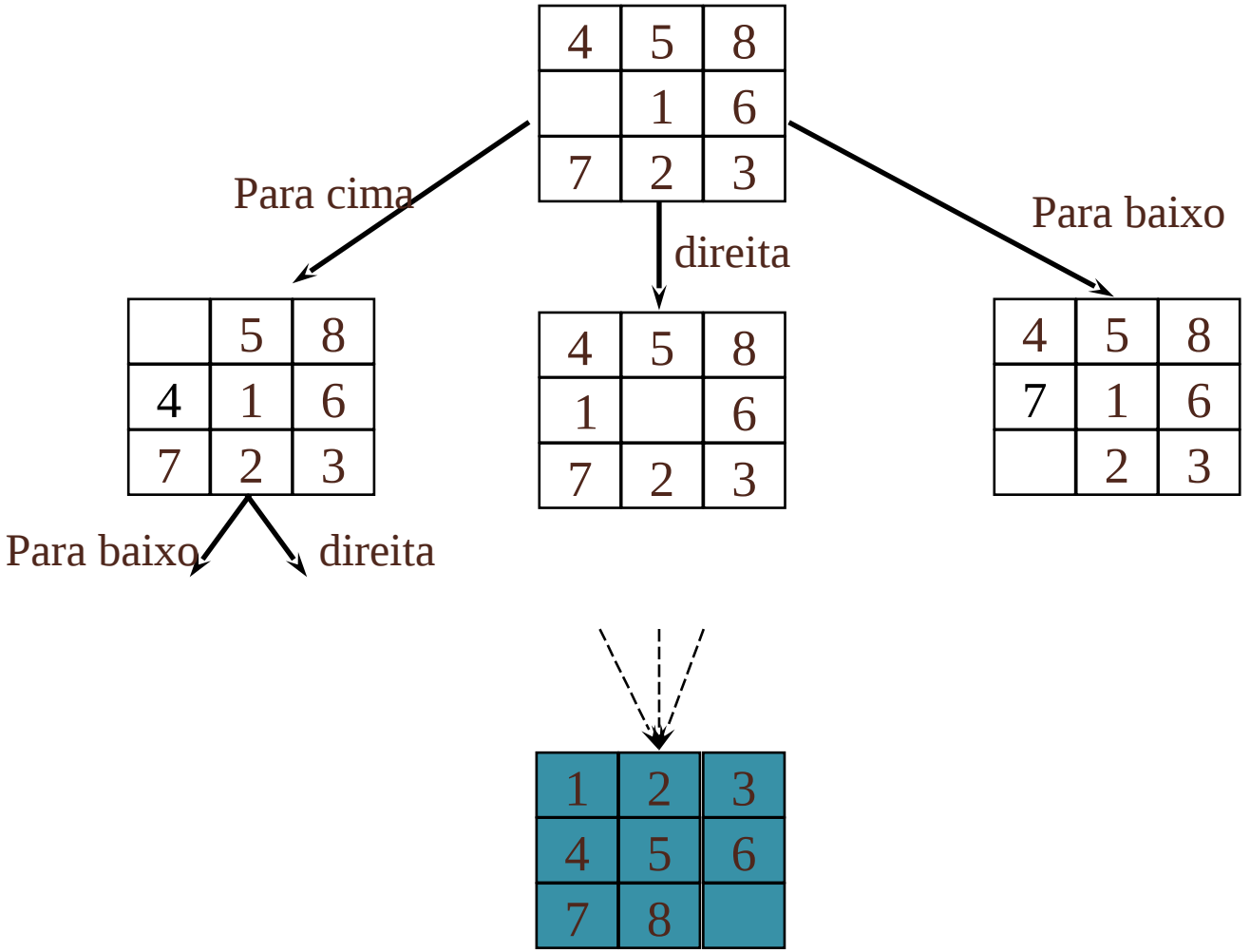
- Algoritmo:

função **Busca-em-Largura** (***problema***)
retorna **uma solução ou falha**

Busca-Genérica (*problema*, Insere-no-Fim)



Exemplo: Jogo dos 8 números



Busca em Largura
Qualidade

- Esta estratégia é *completa*
- É *ótima* ?
  - Sempre encontra a solução mais “ rasa”

    → que nem sempre é a solução de menor **custo de caminho,** caso os operadores tenham valores diferentes.
- É *ótima* se
  - ∀n,n’ profundidade(n’) ≥ profundidade(n) ⇒ custo de caminho(n’) ≥ custo de caminho (n).
    - A função **custo de caminho** é não-decrescente com a profundidade do nó.
    - Essa função acumula o custo do caminho da origem ao nó atual.
  - Geralmente, isto só ocorre quando todos os operadores têm o mesmo custo (=1)

Busca em Largura
Custo

- Fator de expansão da árvore de busca:
  - número de nós gerados a partir de cada nó (b)
- Custo de tempo:
  - se o fator de expansão do problema = b, e a primeira solução para o problema está no nível d,
  - então o número máximo de nós gerados até se encontrar a solução = $1 + b + b^2 + b^3 + \ldots + b^d$
  - **custo exponencial** = $O(b^d)$.
- Custo de memória:
  - a *fronteira* do espaço de estados deve permanecer na memória
  - é um problema mais crucial do que o tempo de execução da busca

Busca em Largura

- Esta estratégia só dá bons resultados quando a *profundidade* da árvore de busca é *pequena*.
- Exemplo:
  - fator de expansão b = 10
  - 1.000 nós gerados por segundo
  - cada nó ocupa 100 bytes

Profundidade	Nós	Tempo	Memória
0	1	1 milissegundo	100 bytes
2	111	0.1 segundo	11 quilobytes
4	11111	11 segundos	1 megabytes
6	10^6	18 minutos	111 megabytes
8	10^8	31 horas	11 gigabytes
10	10^{10}	128 dias	1 terabyte
12	10^{12}	35 anos	111 terabytes
14	10^{14}	3500 anos	11111 terabytes

Busca em Profundidade

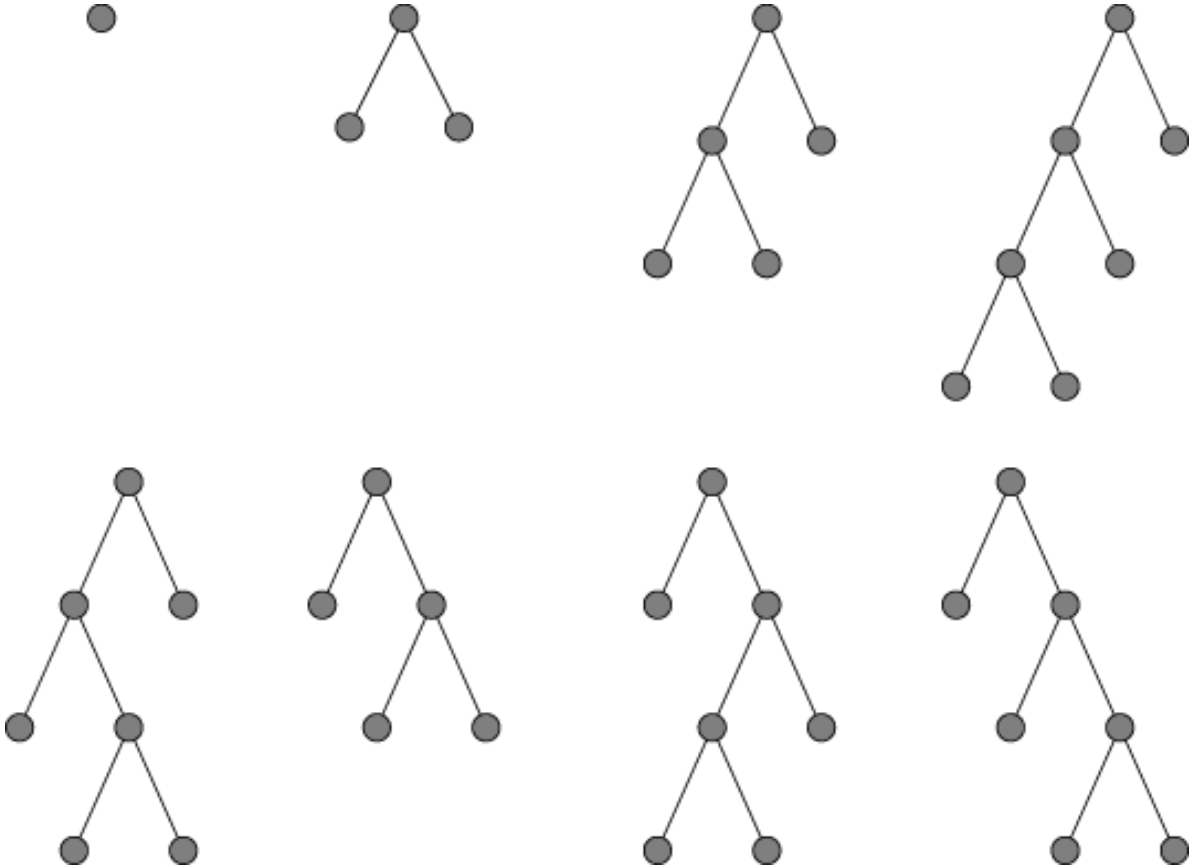
- Ordem de expansão dos nós:
  - sempre expande o nó no *nível mais profundo* da árvore:

    1. nó raiz
    2. primeiro nó de profundidade 1
    3. primeiro nó de profundidade 2, etc....
  - Quando um nó final não é solução, o algoritmo volta para expandir os nós que ainda estão na fronteira do espaço de estados
- Algoritmo:

função Busca-em-Profundidade (*problema*)
retorna **uma solução ou falha**

  Busca-Genérica (*problema*, Insere-no-Começo)

Busca em Profundidade



Busca em Profundidade

- Esta estratégia não é *completa* nem é *ótima*.
- Custo de memória:
  - mantém na memória o caminho sendo expandido no momento, e os nós irmãos dos nós no caminho (para possibilitar o *backtracking*)
    - necessita armazenar apenas $b.m$ nós para um espaço de estados com fator de expansão b e profundidade m, onde m pode ser maior que d (profundidade da 1a. solução)
- Custo de tempo: $O(b^m)$, no pior caso.
- Observações:
  - Para problemas com várias soluções, esta estratégia pode ser bem mais rápida do que busca em largura.
  - Esta estratégia deve ser evitada quando as árvores geradas são muito *profundas* ou geram *caminhos infinitos*.

Como Evitar Geração de Estados Repetidos?

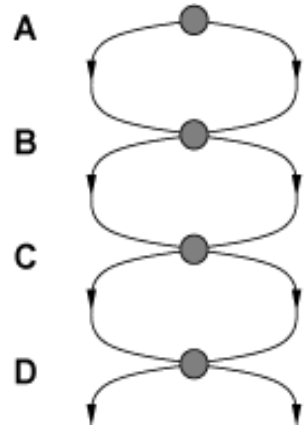
Evitar Geração de Estados Repetidos

- Problema geral em Busca
  - expandir estados presentes em caminhos já explorados
- É inevitável quando existe operadores reversíveis
  - ex. encontrar rotas, canibais e missionários, 8-números, etc.
  - a árvore de busca é potencialmente infinita
- Ideia
  - **podar** (*prune*) estados repetidos, para gerar apenas a parte da árvore que corresponde ao grafo do espaço de estados (que é finito!)
  - mesmo quando esta árvore é finita, evitar estados repetidos pode reduzir exponencialmente o custo da busca

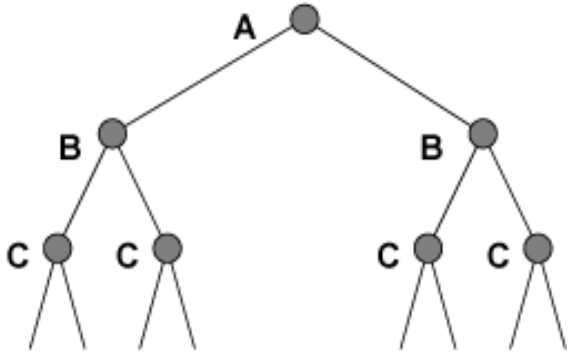
Evitar Geração de Estados Repetidos

- Exemplo:
  - (m + 1) estados no espaço => 2^m caminhos na árvore

Espaço de estados



Árvore de busca



- Questões
  - Como evitar expandir estados presentes em caminhos já explorados?
  - Em ordem crescente de eficácia e custo computacional?

Evitando operadores reversíveis

- se os operadores são reversíveis:
  - conjunto de predecessores do nó = conjunto de sucessores do nó
  - porém, esses operadores podem gerar árvores *infinitas*!

Como Evitar Estados Repetidos ? Algumas Dicas

1. Não retornar ao estado “ pai”
   - função que rejeita geração de sucessor igual ao pai
2. Não criar caminhos com ciclos
   - não gerar sucessores para qualquer estado que já apareceu no caminho sendo expandido
3. Não gerar qualquer estado que já tenha sido criado antes (em qualquer ramo)
   - requer que todos os estados gerados permaneçam na memória
   - custo de memória: $O(b^d)$
   - pode ser implementado mais eficientemente com *hash tables*

Conflito (*trade-off*)

- Problema:
  - Custo de armazenamento e verificação x Custo extra de busca
- Solução
  - depende do problema
  - quanto mais “ loops” , mais vantagem em evitá-los!

A seguir...

- Busca cega, variações dos métodos em profundidade e em largura
- Busca heurística