Escola Politécnica da USP - Depto. de Enga. Mecatrônica

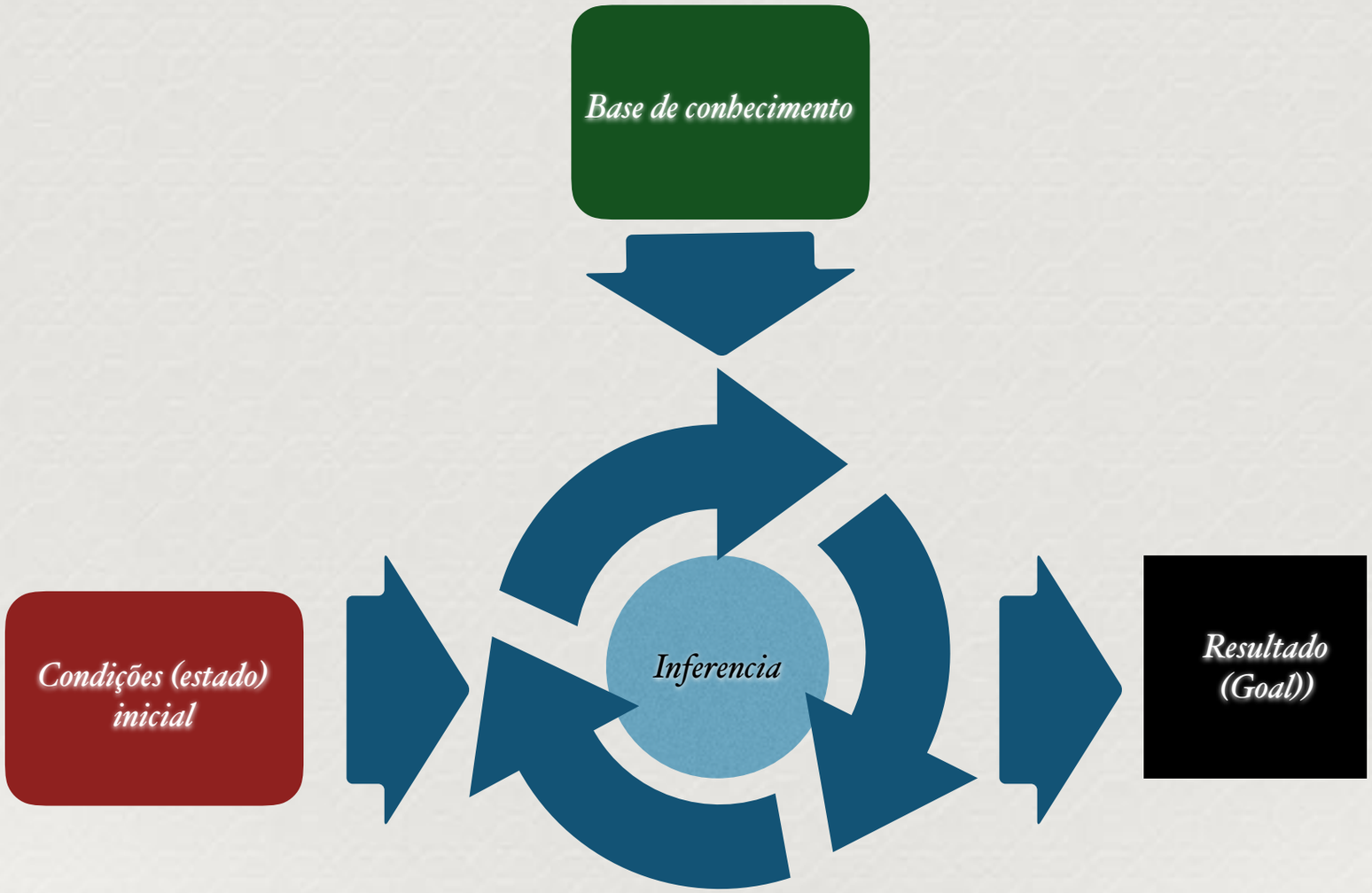
PMR-3510 Inteligência Artificial

Aula 5- Resolução de problemas por máquinas

Prof. José Reinaldo Silva

reinaldo@usp.br

Em uma primeira abordagem, gostaríamos de ter "agentes inteligentes" capazes de "resolver problemas". O que significa isso?



7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Como "resolver problemas automaticamente"

Podemos utilizar duas grande abordagens para resolver problemas automaticamente:

Problema

1. achar uma abordagem geral que leva do estado inicial ao estado final;
2. testar esta abordagem em alguns casos (sem levar em conta o tempo para chegar à solução);
3. checar se a abordagem é completa, isto é, resolve todos os casos ou há casos especiais onde o problema "não converge";
4. preparar a implementação do revolvedor (estrutura de dados e base de conhecimento, regras de dedução);

Solução

Achar um método geral de resolução de problemas

Estrutura de um "resolvedor automático de problemas"

Para dotar uma máquina da capacidade de resolver problemas (ou uma classe de problemas) é preciso ter uma estrutura com os seguintes atributos:

1. uma descrição clara do "estado inicial" ou seja das condições iniciais do problema a ser resolvido;
2. uma descrição clara do objetivo ou "estado final", de modo que seja possível saber quando (e se) o problema foi resolvido;
3. em cada estágio do processo de solução saber quais os próximos estados que podem ser atingidos;
4. poder escolher um (ou o melhor) caminho entre os estados acima;
5. saber que operadores (ou passos) aplicar para fazer a "transição" para um próximo estado;
6. discernir se estamos convergindo para a solução.

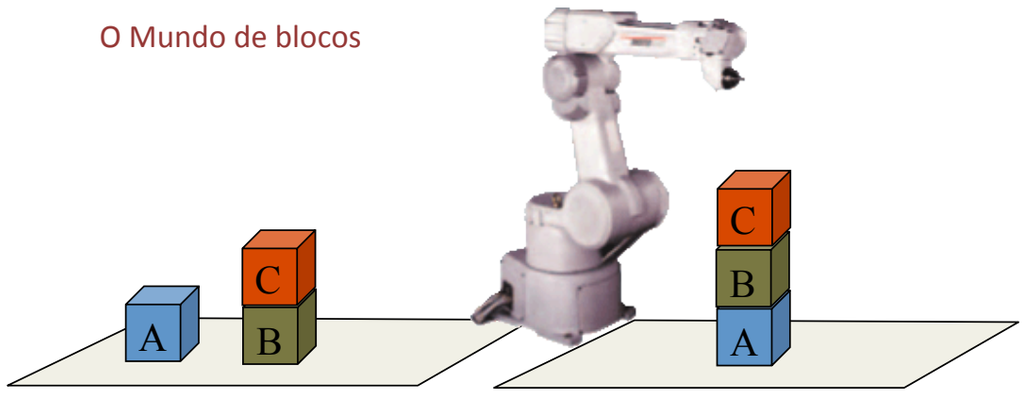
Vamos agora pensar na resolução de problemas por uma máquina!

Usando ainda o paradigma estado/transição, devemos, além de especificar claramente os estados inicial e final, saber agora quais são os demais estados e como se relacionam, isto é, o espaço de estados. O problema pode ser então reduzido a uma busca por um caminho neste espaço que leve do estado inicial ao estado final.

IA Planning: o STRIPS

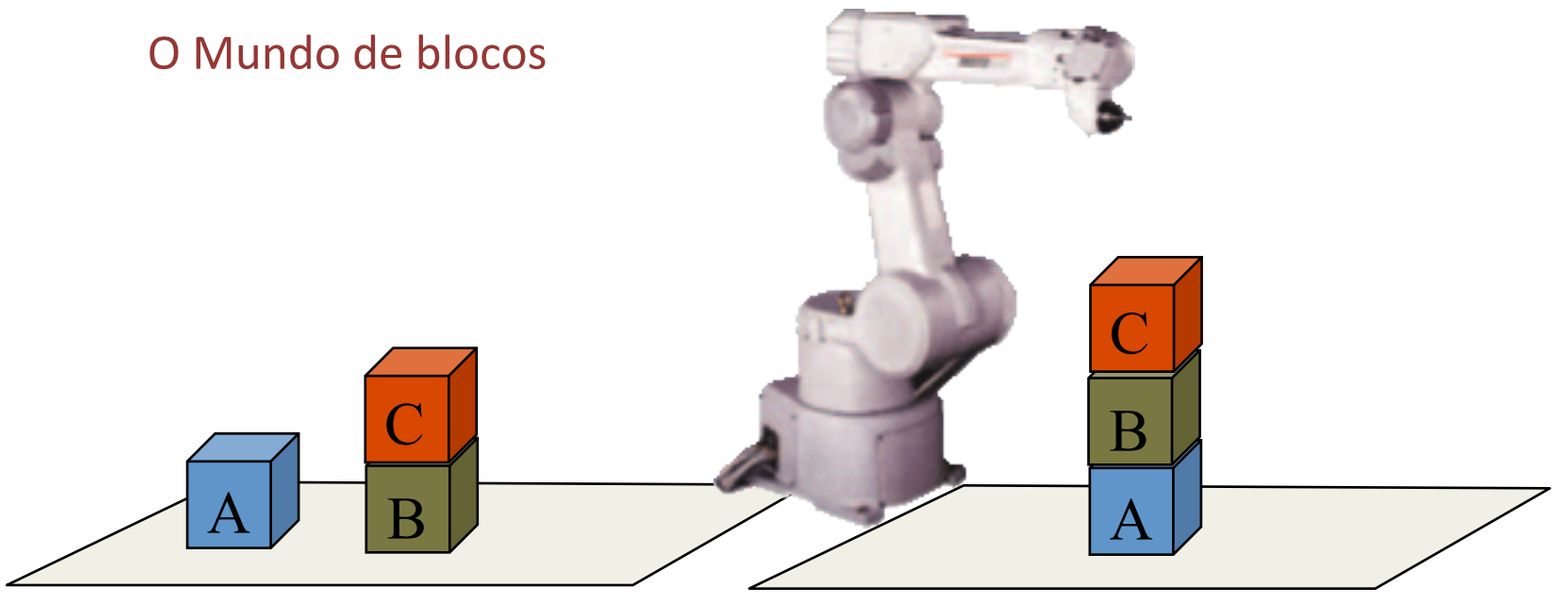
O sistema STRIPS é a estratégia de resolução de problemas mais usada em planning. Note-se que é uma estratégia baseada no método estado-transição e por isso é passível de ser analisada em Redes de Petri. O problema modelo mais conhecido resolvido com o sistema STRIPS é o problema do mundo de blocos.

O Mundo de blocos

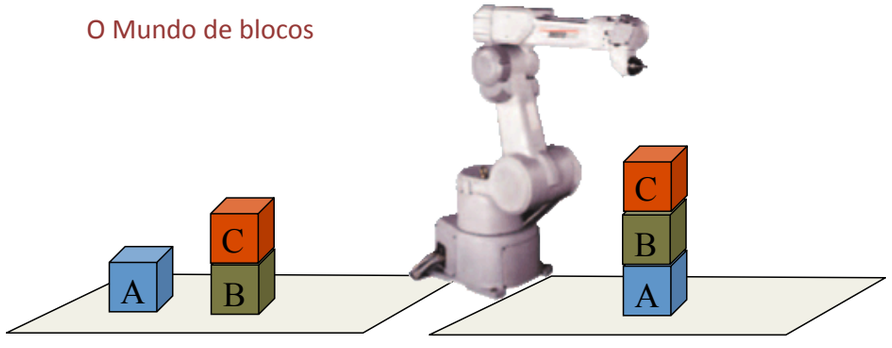


O exemplo mais simples e intuitivo do sistema STRIPS é o chamado "mundo de blocos" que consiste em mudar blocos de configuração usando robôs.

O Mundo de blocos

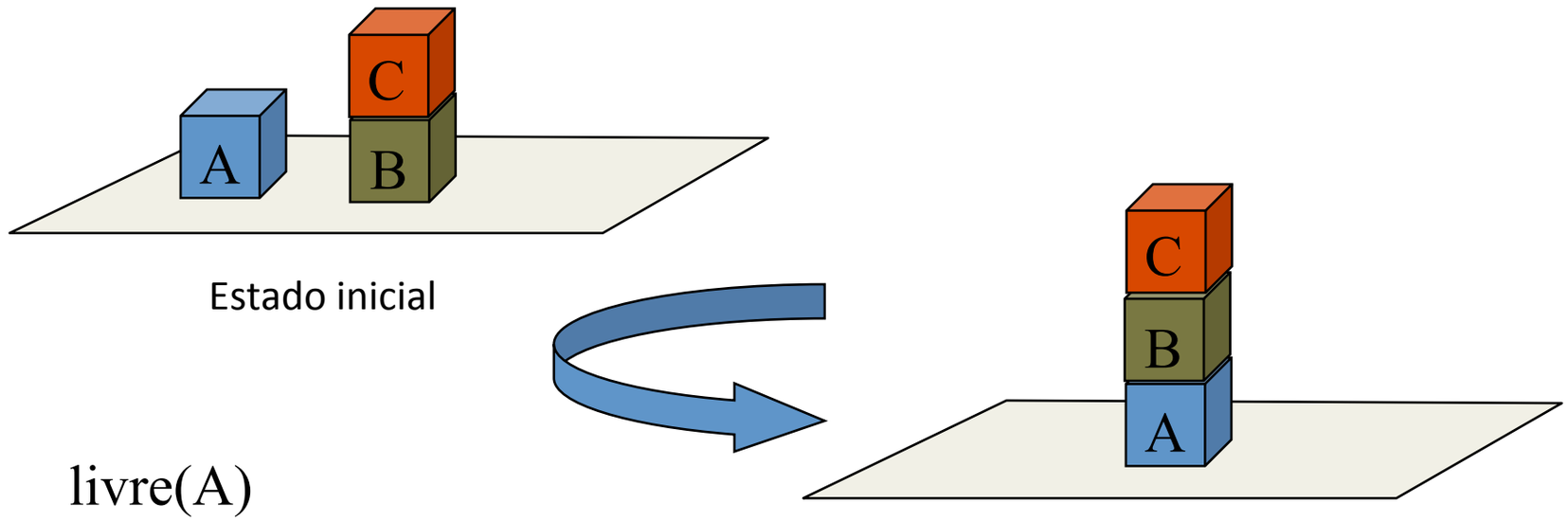


O Mundo de blocos



O exercício consiste em analisar a dinâmica deste sistema. Vamos portanto separar a "planta" do "controle", onde a planta são os possíveis estados do "mundo de blocos" e que movimento é preciso fazer para passar de um estado para o outro. Existe um agente dinâmico capaz de operar a planta, isto é, capaz de, dentro de algumas condições realizar uma transição. Este agente é o robô. No momento vamos apenas modelar a "planta".

Um plano é a solução de um problema composto de um estado inicial, um estado final, e uma sequência de ações (ou um passo) que transforma o estado inicial no estado final, ou, em outras palavras que os coloca na mesma localidade

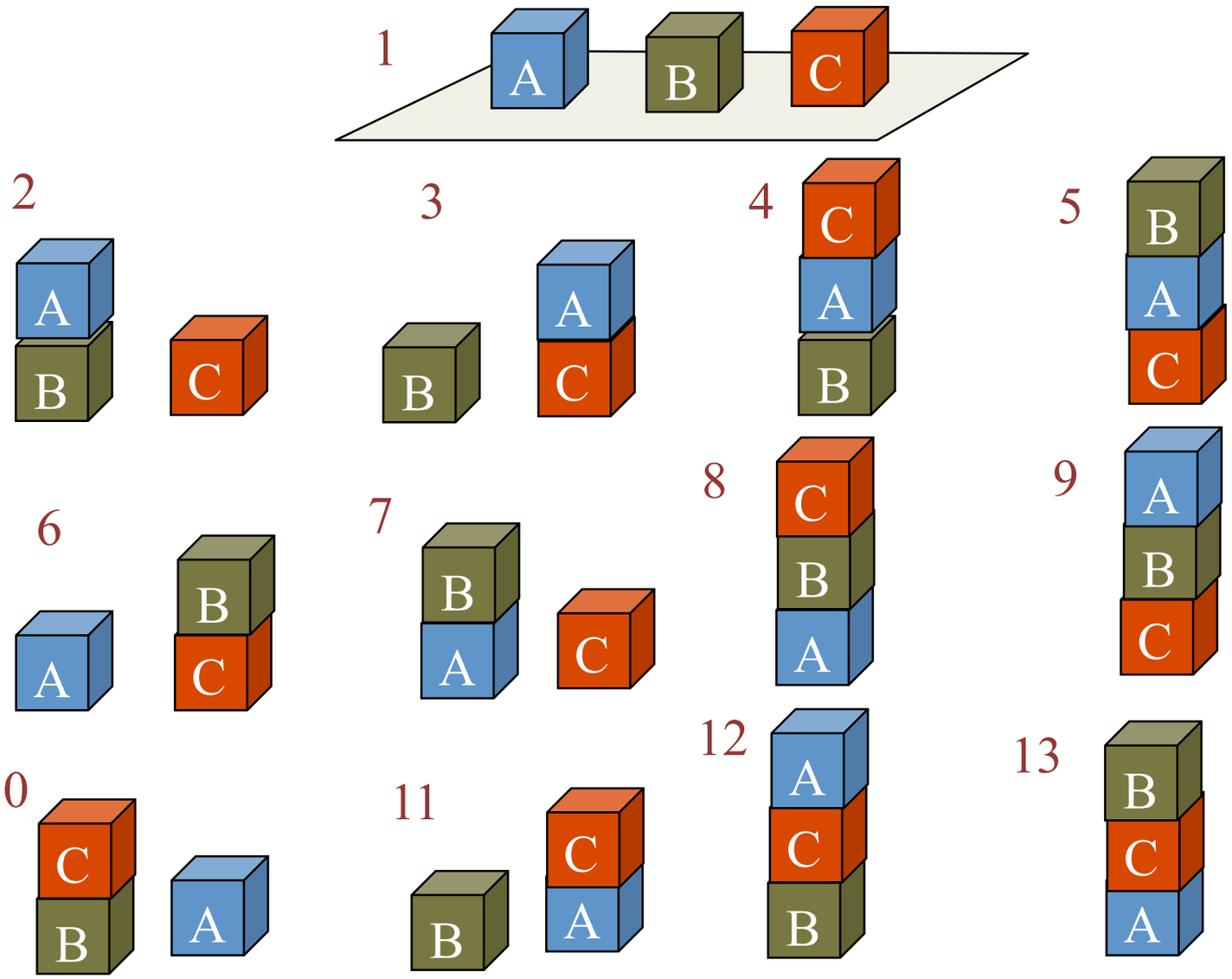


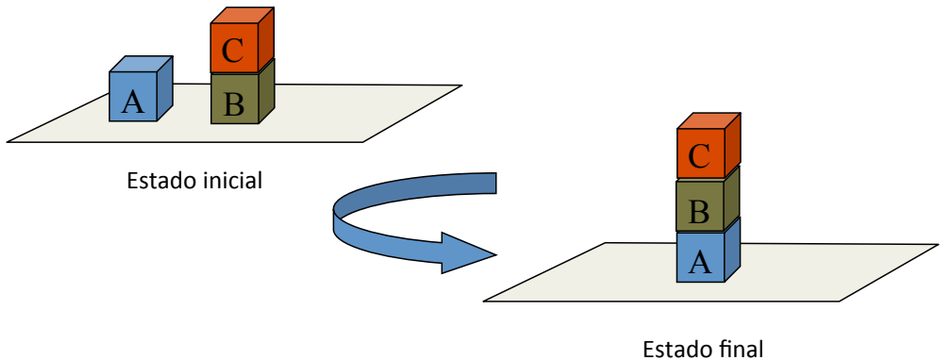
Estado inicial

livre(A)
sobre(A,mesa)
sobre(C, B)
sobre (B, mesa)
livre(C)

Estado final

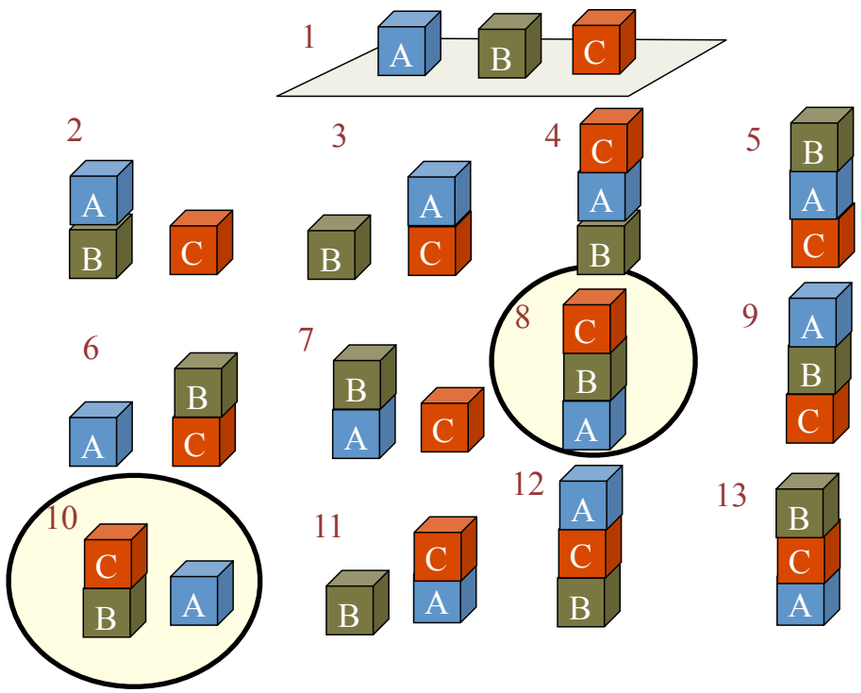
livre(C)
sobre(C,B)
sobre(B,A)
sobre(A,mesa)

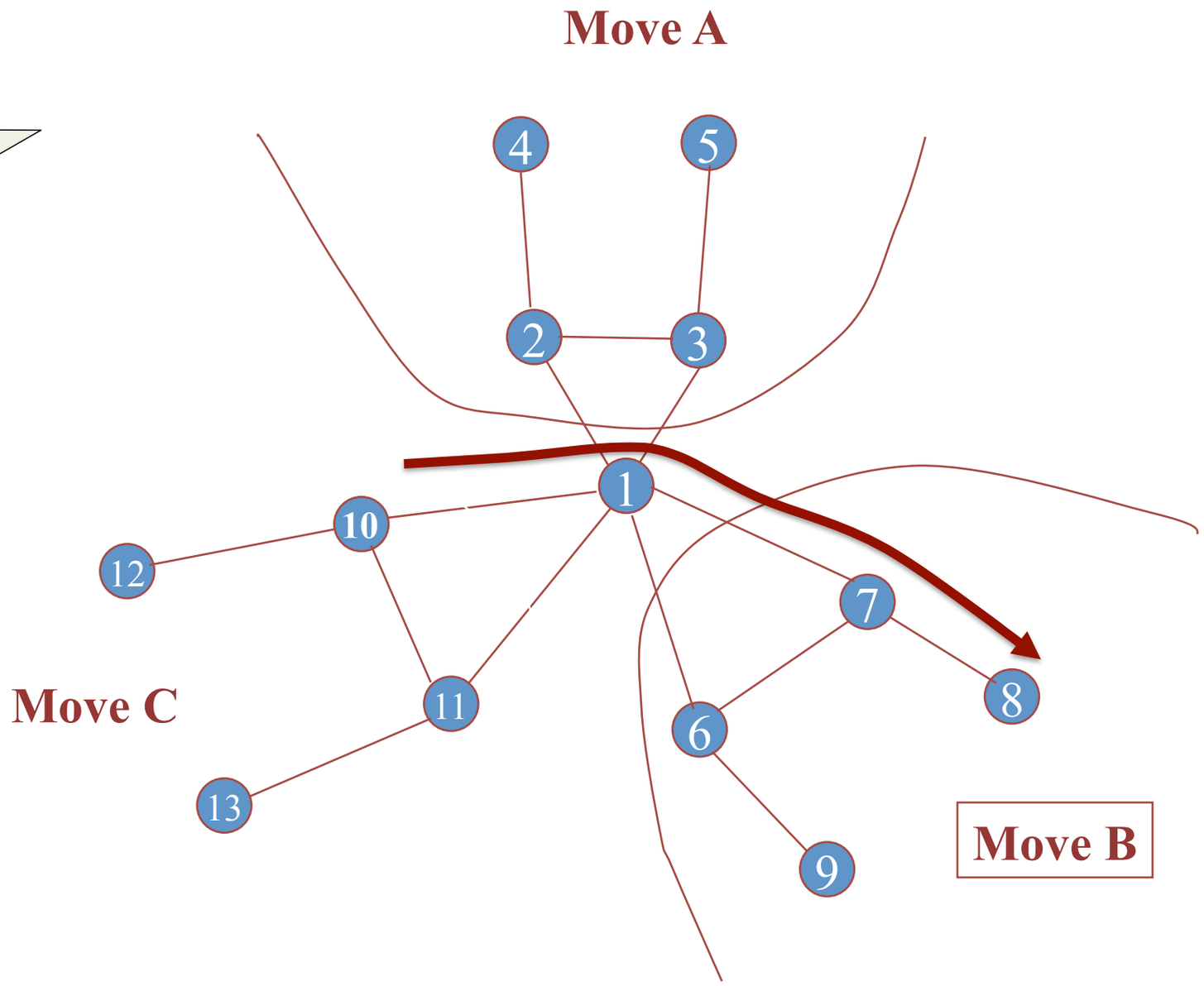

1
A
B
C
2
A
B
C
3
B
A
C
4
C
A
B
5
B
A
C
6
A
B
C
7
B
A
C
8
C
B
A
9
A
B
C
10
C
B
A
11
B
C
A
12
A
C
B
13
B
C
A

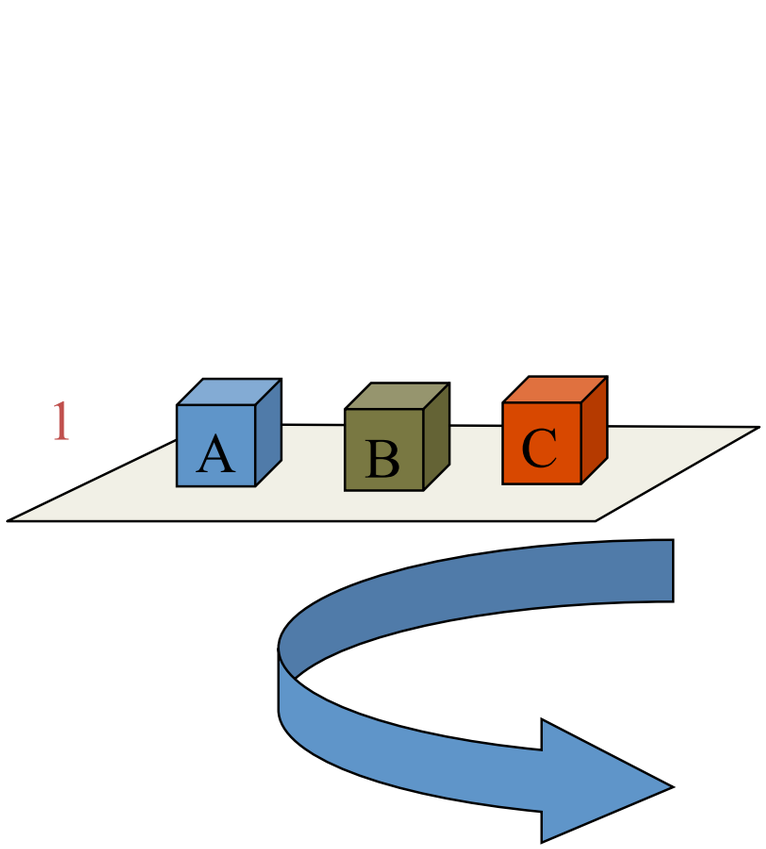


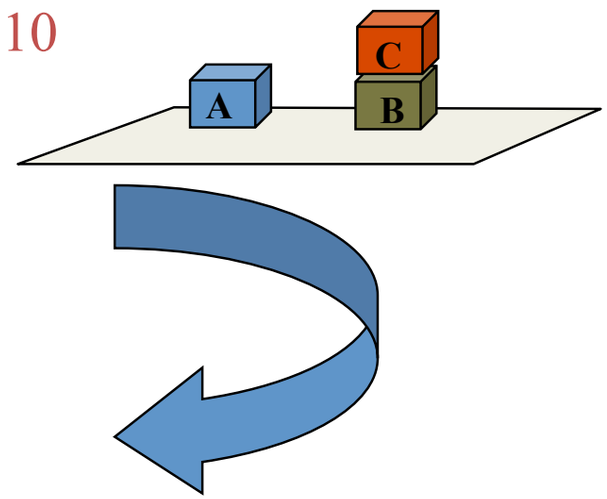
Estado inicial

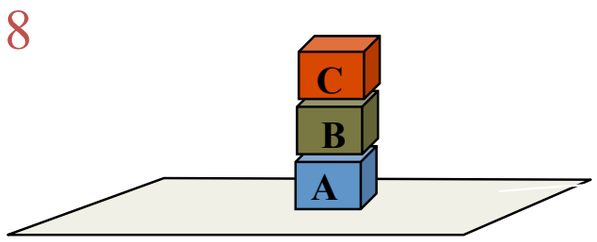
Estado final

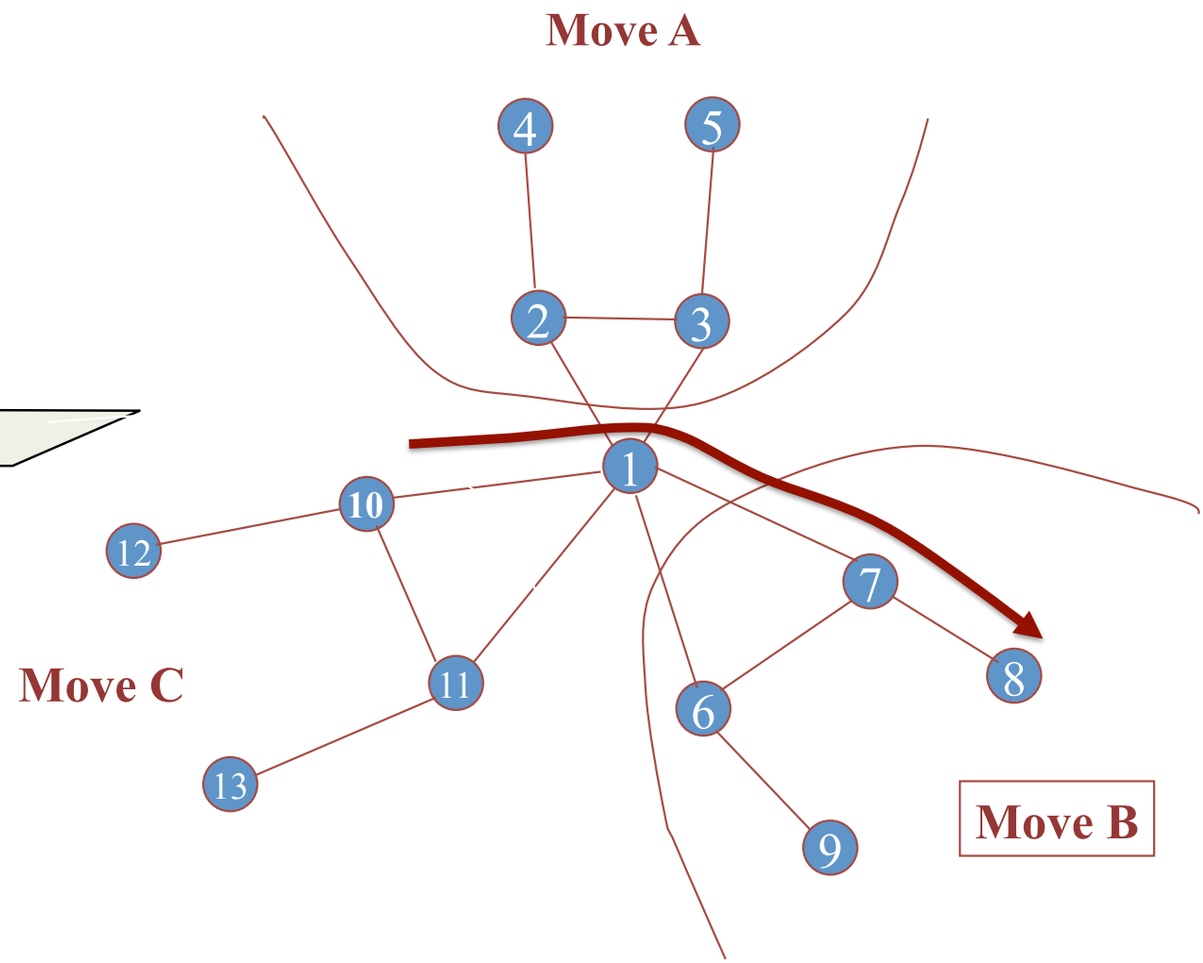





1
A
B
C


10
A
C
B


8
C
B
A


Move A
4
5
2
3
1
10
12
7
8
Move C
11
6
13
9
Move B

Será este um caminho possível para qualquer problema?

Na prática são poucos os problemas onde se pode listar todo o espaço de estados. Exatamente por isso o "mundo de blocos" é um problema-modelo muito importante e um excelente caso de estudo. Portanto embora parece atraente este não pode ser o método adotado para resolver problemas automaticamente por máquinas.

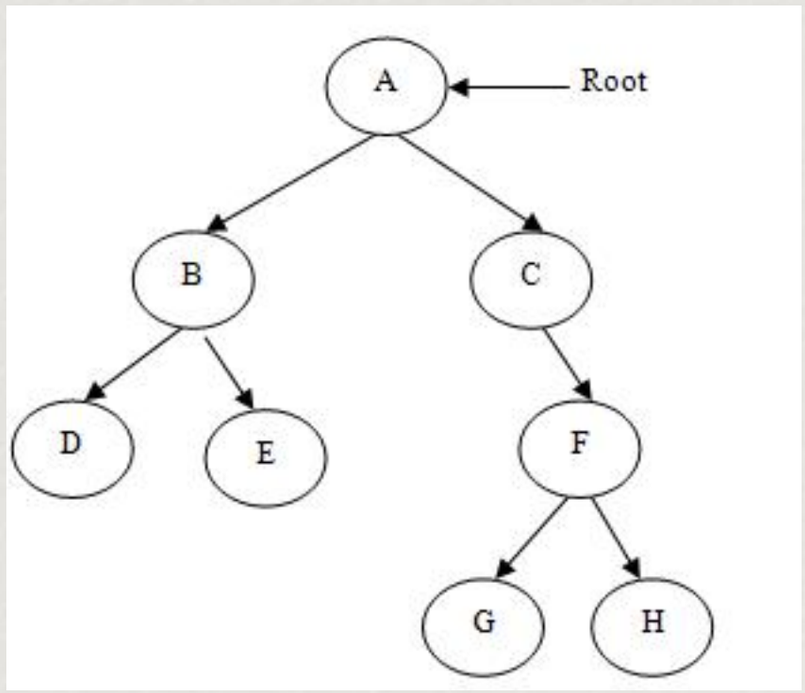
Uma outra forma seria gerar o espaço de estados enquanto se busca a solução

1. uma descrição clara do "estado inicial" ou seja das condições iniciais do problema a ser resolvido;
2. uma descrição clara do objetivo ou "estado final", de modo que seja possível saber quando (e se) o problema foi resolvido;
3. em cada estágio do processo de solução saber quais os próximos estados que podem ser atingidos;
4. poder escolher um (ou o melhor) caminho entre os estados acima;
5. saber que operadores (ou passos) aplicar para fazer a "transição" para um próximo estado;
6. discernir se estamos convergindo para a solução.

→ *estrutura*

Usando árvores como base para a solução

Uma opção muito interessante é modelar o espaço de estados na forma de uma árvore



Algoritmos

Existem prós e contras em cada estratégia descrita e nos algoritmos gerados para implementa-las:

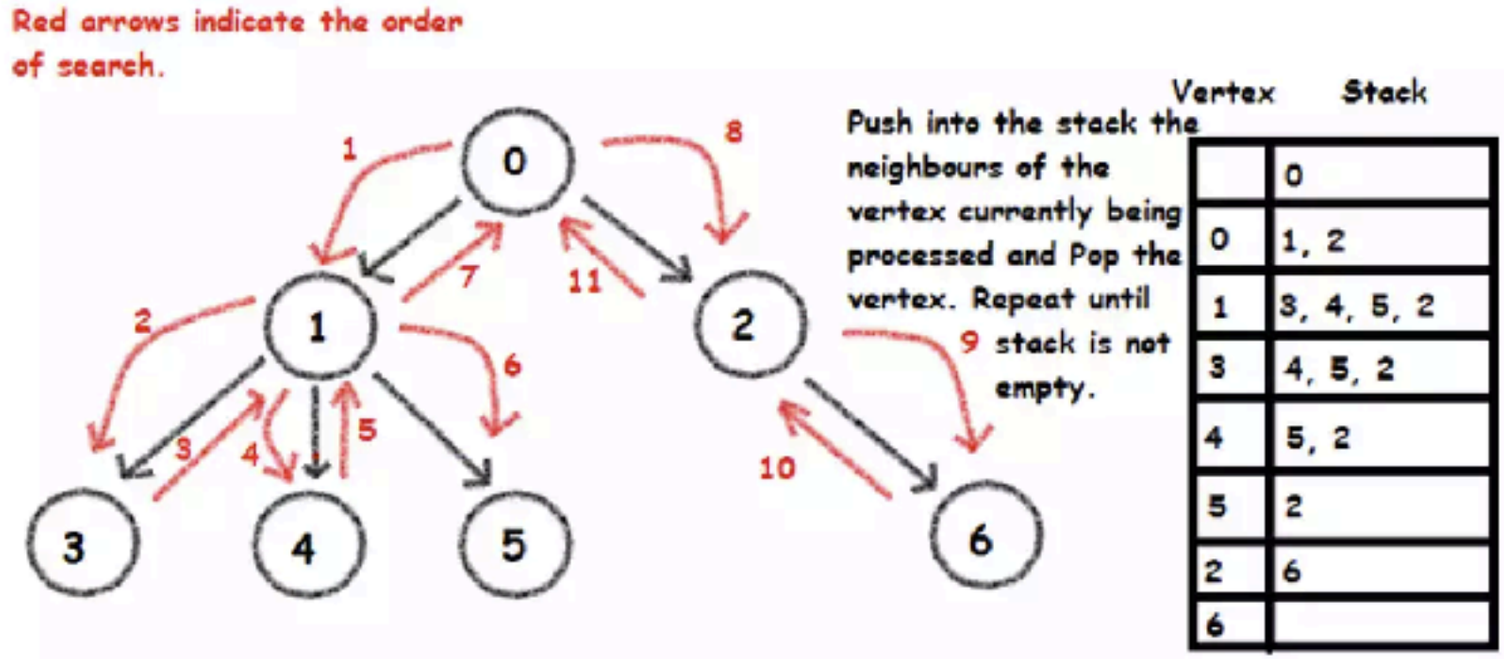
- Gerar todo o espaço de estados pode ser proibitivo dependendo do problema, mas, se não for este o caso poderemos usar este espaço para todas as instâncias do problema.
- Gerar o espaço de estados enquanto se faz a busca pode ser interessante, mas temos que levar em conta que teremos que repetir a mesma geração para cada instância do problema.

Algoritmos de busca

Busca não informada - quando todos os nós gerados são igualmente promissores, ou não se tem informação sobre o seu potencial: busca em profundidade, busca em largura, busca de custo uniforme

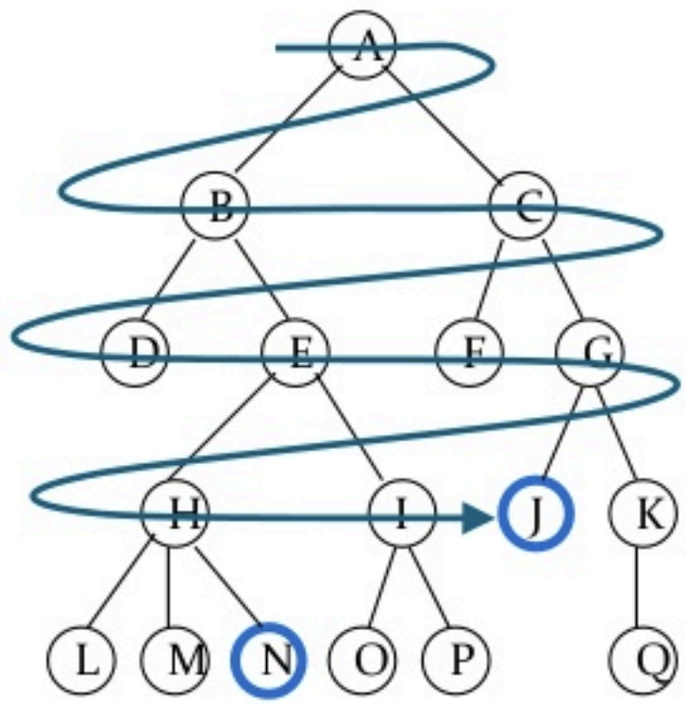
Busca informada - quando conhecimento heurístico pode ser levantado que distingue entre os nós gerados em um mesmo nível da árvore.

Busca em profundidade



Busca em largura

Breadth-first searching[1]

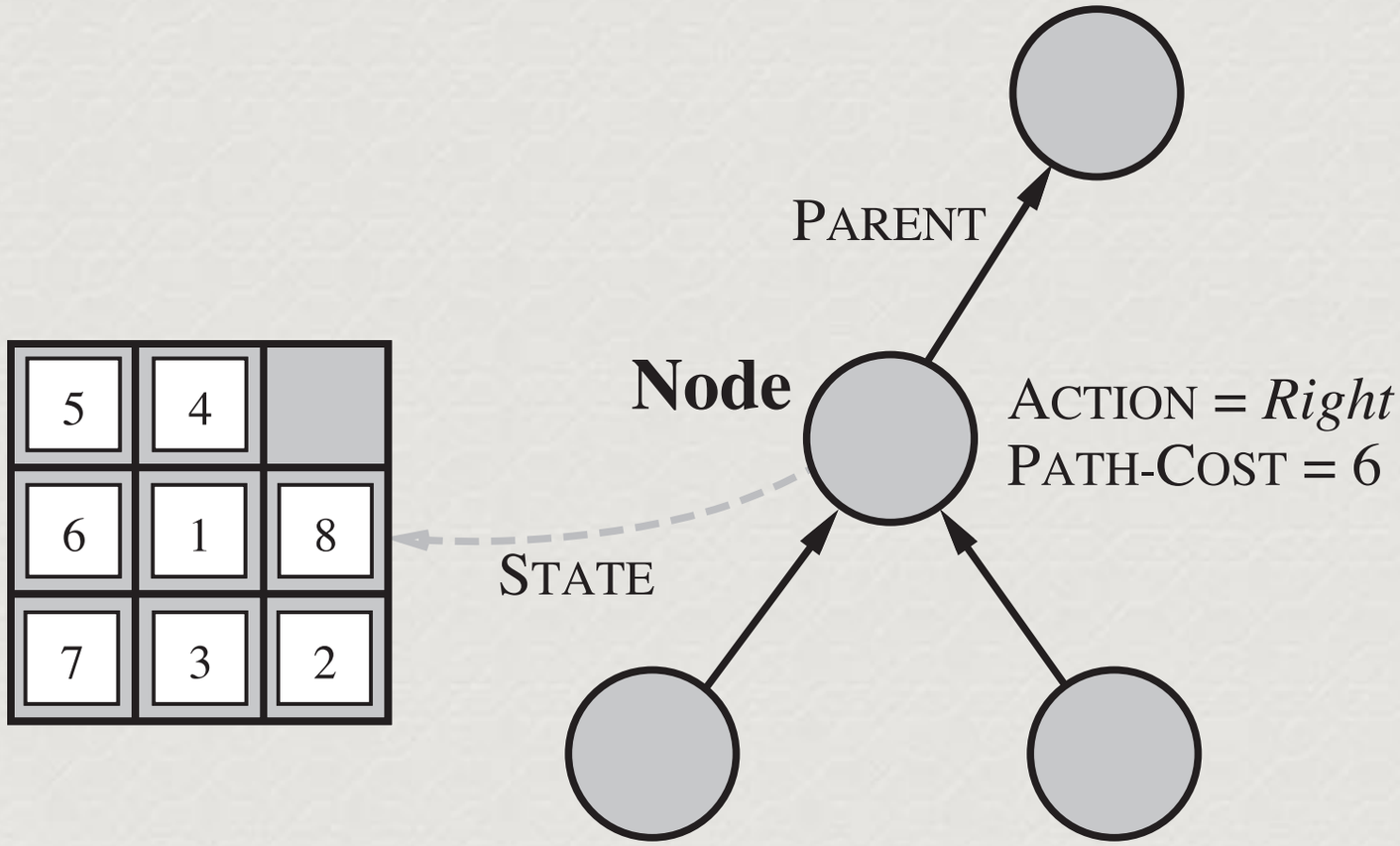


- A breadth-first search (BFS) explores nodes nearest the root before exploring nodes further away
- For example, after searching A, then B, then C, the search proceeds with D, E, F, G
- Node are explored in the order A B C D E F G H I J K L M N O P Q
- J will be found before N

Usando os métodos clássicos de busca não-informada

Comparando os dois métodos podemos concluir que cada um deles pode ser melhor aplicado em situações onde:

i) a solução está mais próxima da raiz, e a busca em largura será mais eficiente;
ii) a solução está nos nós de maior profundidade ou nas folhas, a neste caso a busca em profundidade será mais apropriada;


PARENT
Node
ACTION = Right
PATH-COST = 6
STATE
5
4
6
1
8
7
3
2

Na verdade, gerar todo o espaço de estados neste problema é proibitivo, portanto devemos pensar primeiro na geração enquanto se constrói o espaço de estados e em busca informada, que será vista na próxima aula, juntamente com a busca com custo uniforme.

Vamos começar o exercício buscando um algoritmo (antes de começar a programação) que primeiro construa "on the fly" o espaço de busca, depois acrescentar heurísticas a este processo.

Até a próxima aula!