Inteligência artificial

Aula 2- Busca

Inteligência artificial

Prof. RELTON ALVES DA SILVA

Aula 2 - Busca

Sumario

Introdução
Gerar e Testar
Busca em Profundidade
Busca em Largura
Busca em Espaco de Estados

Introdução

- Os algoritmos a serem abordados enquadram-se na categoria sem informação. Trata-se dos métodos mais simples, pois não dispõem de dados adicionais além de sua própria definição formal. Em outras palavras, esses algoritmos são aplicados em ambientes caracterizados pelas seguintes dimensões:

- Determinístico;
- Completamente observáveis;
- Estáticos;
- Completamente conhecidos;

Introdução

A abordagem de resolução de problemas é aplicada a uma série de ambientes de tarefas que Inicialmente vamos dividir em duas classes:

- Miniproblemas
- Problemas do mundo real

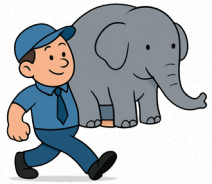
Miniproblemas

Se destina a ilustrar problemas ou exercitar diversos métodos de resolução de problemas. Ele pode ter uma descrição concisa e exata.

Isso significa que ele pode ser usado com facilidade por diferentes buscadores, com a finalidade de comparar o desempenho de algoritmos.

Problemas do mundo real

São aqueles cujas soluções de fato preocupam as pessoas. Eles tendem a não representar uma única descrição consensual, mas tentaremos dar uma idéia geral de suas formulações.

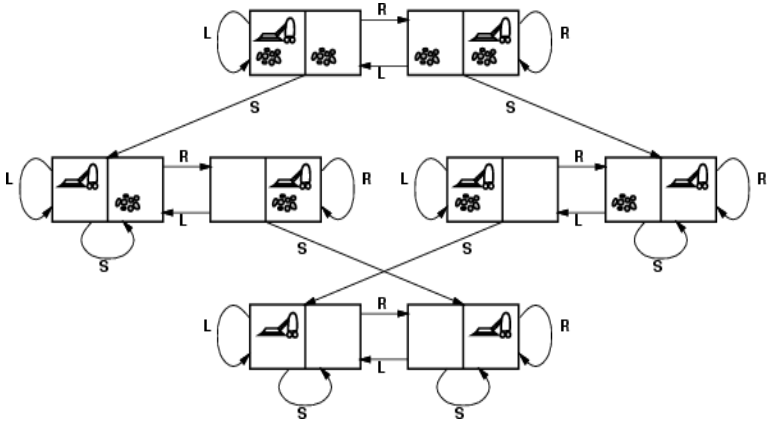


Exemplos de Miniproblemas

Mundo do aspirador de pó

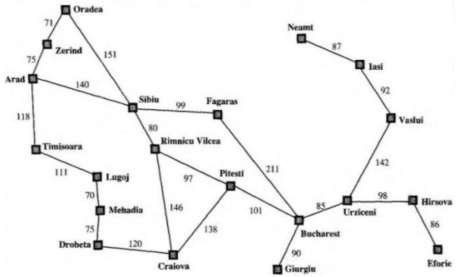
O agente aspirador de pó percebe em que quadrado está e se existe sujeira no quadrado.

Ele pode mover-se para direita ou para esquerda, aspirar à sujeira ou não fazer nada.



Problemas do mundo real

- Problema de roteamento
  - Encontrar a melhor rota de um ponto a outro (aplicações: redes de computadores, planejamento militar, planejamento de viagens aéreas)
- Problemas de tour
  - Visitar cada ponto pelo menos uma vez
- Caixeiro viajante
  - Visitar cada cidade exatamente uma vez
  - Encontrar o caminho mais curto
- Layout de VLSI
  - Posicionamento de componentes e conexões em um chip



Agentes de resolução de problemas

- Agentes reativos não funcionam em ambientes para os quais o número de regras condição-ação é grande demais para armazenar.
- Nesse caso podemos construir um tipo de agente baseado em objetivo chamado de agente de resolução de problemas.

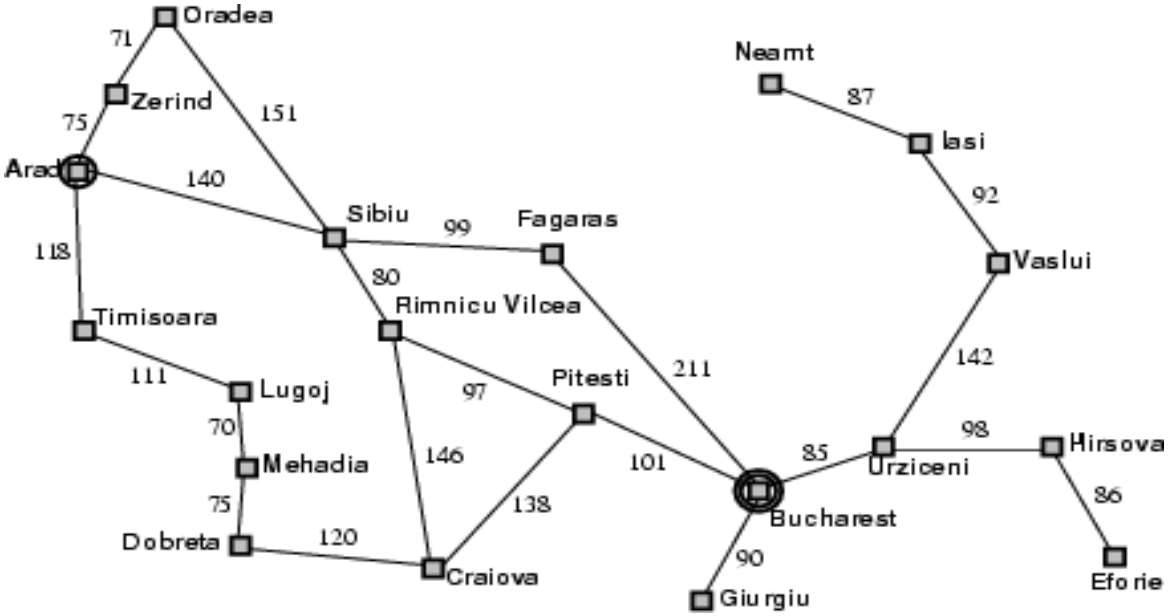
Agente

- Um agente com várias opções imediatas pode decidir o que fazer comparando diferentes seqüências de ações possíveis.
- Esse processo de procurar pela melhor seqüência é chamado de **busca**.
- Formular objetivo → buscar → executar

Exemplo: Romênia

- De férias na Romênia; atualmente em Arad.
- Vôo sai amanhã de Bucareste.
- Formular objetivo:
  - Estar em Bucareste
- Formular problema:
  - estados: cidades
  - ações: dirigir entre as cidades
- Encontrar solução:
  - sequência de cidades, ex., Arad, Sibiu, Fagaras, Bucareste.

Exemplo: Romênia



Formulação de problemas

- Um **problema** é definido por quatro itens:

1. **Estado inicial** ex., “em Arad"
2. **Ações** ou **função sucessor** *S(x)* = conjunto de pares estado-ação
   - ex., *S(Arad)* = {*<Arad → Zerind, Zerind>, …* }
3. **Teste de objetivo**, pode ser
   - explícito, ex., *x* = “em Bucharest"
   - implícito, ex., *Cheque-mate(x)*
4. **Custo de caminho** (aditivo)
   - ex., Soma das distâncias, Número de ações executadas, etc.

- Uma **solução** é uma seqüência de ações que levam do estado inicial para
- o estado objetivo.
- Uma **solução ótima** é uma solução com o menor custo de caminho.

Exemplo Prático

SCAN ME

https://colab.research.google.com/drive/1MZ9lLEzHoQjlpnNSI7EPFIbQmMTjP3oo?usp=sharing#scrollTo=GysK3eFiu5DV

Introdução

- Soluções de problemas de busca são parte dos problemas estudados pelos algoritmos de Inteligência Artificial;
- Um problema consiste de:
  - Objetivo (Estado Objetivo);
  - Estados (iniciando no estado Inicial);
  - Busca (é o método utilizado para examinar o espaço do problema para encontrar o objetivo)
  - Espaço de busca;

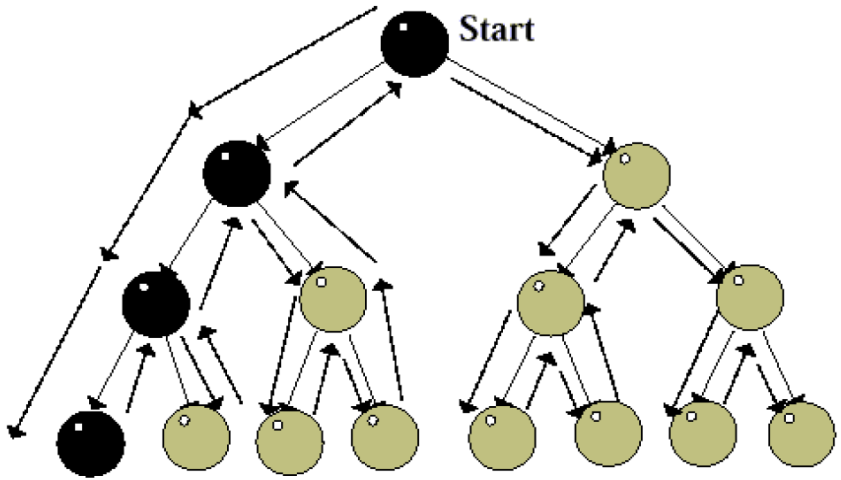
Gerar e Testar

- Solução mais simples;
- Selecionar um estado e verificar se é o objetivo, caso contrário iria para um próximo nó. Esta é uma forma de **força bruta**;
- Força bruta (ou busca exaustiva) não pressupõe ou tem informações extras. Simplesmente irá percorrer todos os estados;
- Para ter sucesso um algoritmo Gerar e Testar, deve:
  - Ser completo (gerar todas as soluções possíveis);
  - Não ser redundante;
  - Deve ser bem informado (Não deve examinar estados não adequados com o espaço de busca);
- Também chamado de busca **cega**, é muito utilizado quando não se tem informações extras. Exemplo: Busca em **Largura** e **Profundidade**.

Busca em *Profundidade*

- Tem este nome pois segue um caminho até a sua maior profundidade, daí começa um novo caminho;
- A busca em profundidade utiliza um método chamado **retrocesso cronológico** para voltar até a árvore de busca, uma vez que um caminho sem saída foi detectado;
- Um exemplo da aplicação da busca em profundidade é para indexação de páginas na Internet;
- Tem a desvantagem quando os caminhos são muito longos ou mesmo infinitos;
- Tem a vantagem quando os caminhos tem mesmo comprimento ou todos os caminhos levam ao estado objetivo;

Busca em *Profundidade*



Algoritmos de Busca

Um **algoritmos de busca** em vetor é um algoritmo para procurar a presença ou não de determinado valor em uma sequência de dados em memória (em um *vetor*). Deve-se destacar que a realização de busca é provavelmente o algoritmo mais empregado na prática, por exemplo, sempre que acessa um sistema com usuário e senha será necessário *buscar pelo seu usuário* e depois vefificar se a senha está correta.
Podemos separar a *classe dos algoritmos busca* em duas grandes categoria, aqueles cujos elementos **estão ordenados** e aqueles **não ordenados**.

Algoritmos de Busca

Na segunda categoria o tempo de busca é diretamente proporcional ao número de elementos, se existem N elementos, no pior caso (e.g. quando o elemento procurado não existe no vetor) serão realizadas N comparações (ou 2*N se algoritmo implementado não usar sentinela - vide seção 1.1). Por outro lado, se os dados estiverem ordenados, é possível implementar um algoritmo tremendamente mais rápido, o algoritmo de busca binária, que no pior caso realiza apenas log2(N) comparações.

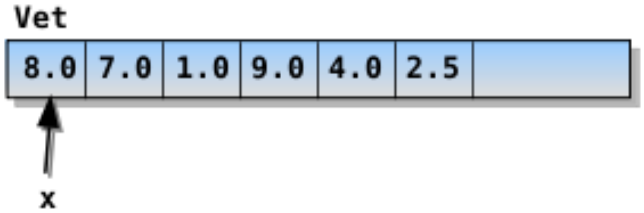
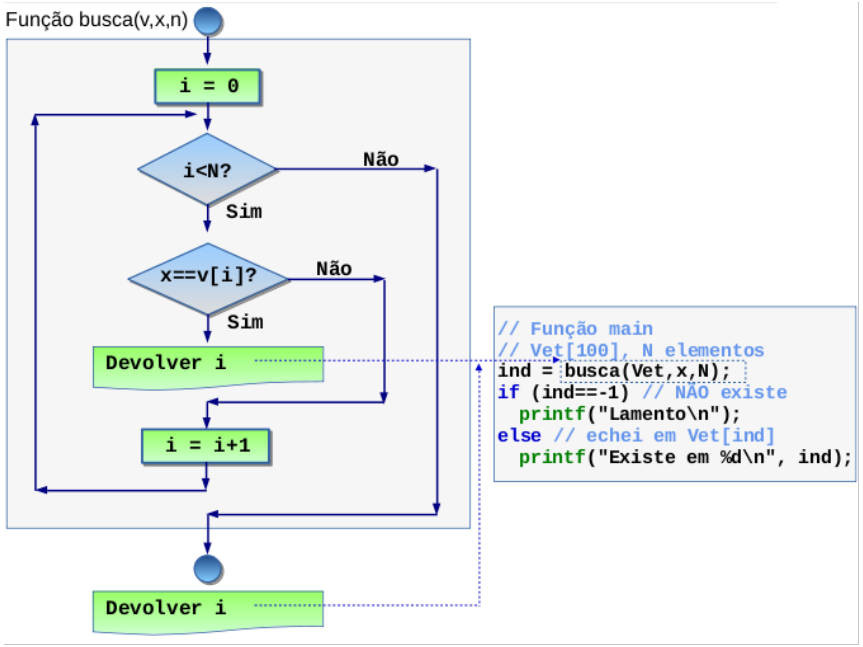


Fig. 1. Ilustração da busca por x em um vetor não ordenado Vet em memória.

Algoritmos de Busca

Um primeiro algoritmo para buscar um dado elemento em um vetor não ordenado pode ser obtido usando um enumerador (contador) iniciado com o primeiro elemento do vetor, se for igual ao buscado devolve esse índice, senão examine o próximo a assim por diante, como ilustrado na figura..



Hora de Programar

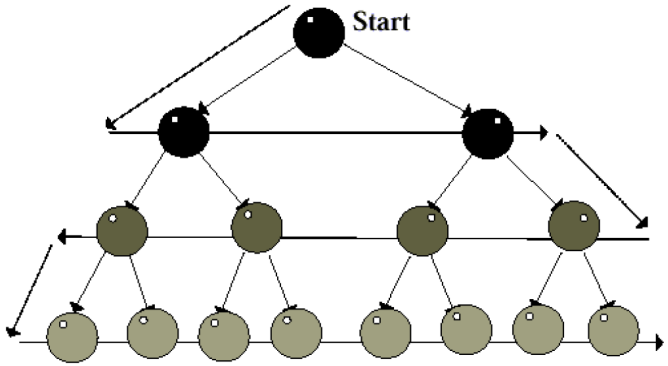
```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

int main()
{
    cout << "Algoritimo de Busca - Aula Relton "<< endl;

int matriz[3][3] = {{0, 0, 0}, {0, 1, 0}, {0, 0, 0}};
   int linha, coluna;
   bool encontrado = false;

for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 3; j++) {
        if(matriz[i][j] == 1) {
            linha = i;
            coluna = j;
            encontrado = true;
            break;
        }
    }
    if(encontrado) {
        break;
    }
}

if(encontrado) {
    cout << "O valor 1 foi encontrado na posicao: (" << linha << ", " << coluna << ")" << endl;
} else {
    cout << "O valor 1 nao foi encontrado na matriz." << endl;
}

    return 0;

}
```

SCAN ME

https://www.mycompiler.io/pt/new/cpp

Busca em Largura

- Este algoritmo começa avaliando os nós por nível e não por caminho;
- Este algoritmo tem a vantagem quando os caminhos possíveis são muito profundos;
- Este algoritmo tem vantagens quando o estado objetivo se localiza em estados em uma parte mais rasa da árvore;
- A busca em largura não é adequada quando deve-se investigar árvores onde todos os caminhos levam ao estado objetivo;
- Não é adequada para soluções como jogos, pois estes tem árvores muito profundas;



Comparando Algoritmos

- b = número de caminhos alternativos/fator de bifurcação/ramificação (*branching factor*)
- d = profundidade da solução
- h = profundidade máxima da árvore de busca
- l = limite de profundidade

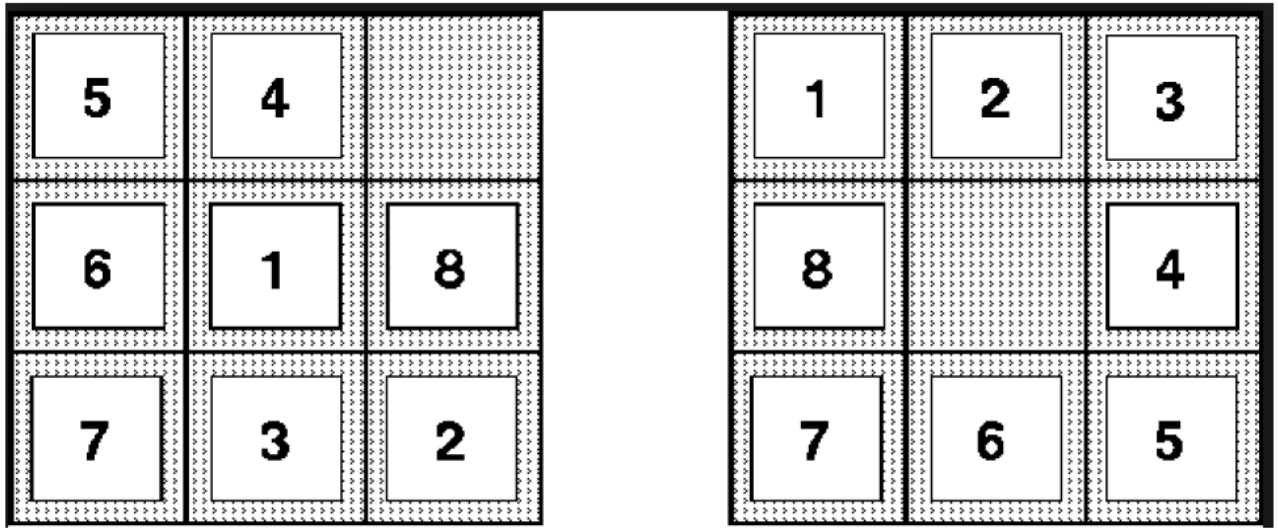
	Tempo	Espaço	Admissível? (solução mais curta)	Completa? (encontra uma solução quando ela existe)
Profundidade	$O(b^h)$	$O(bh)$	Não	Sim (espaços finitos) Não (espaços infinitos)
Profundidade limitada	$O(b^l)$	$O(bl)$	Não	Sim se $l \geq d$
Largura	$O(b^d)$	$O(b^d)$	Sim	Sim
Bidirecional	$O(b^{d/2})$	$O(b^{d/2})$	Sim	Sim

Busca em Estado de Espaço

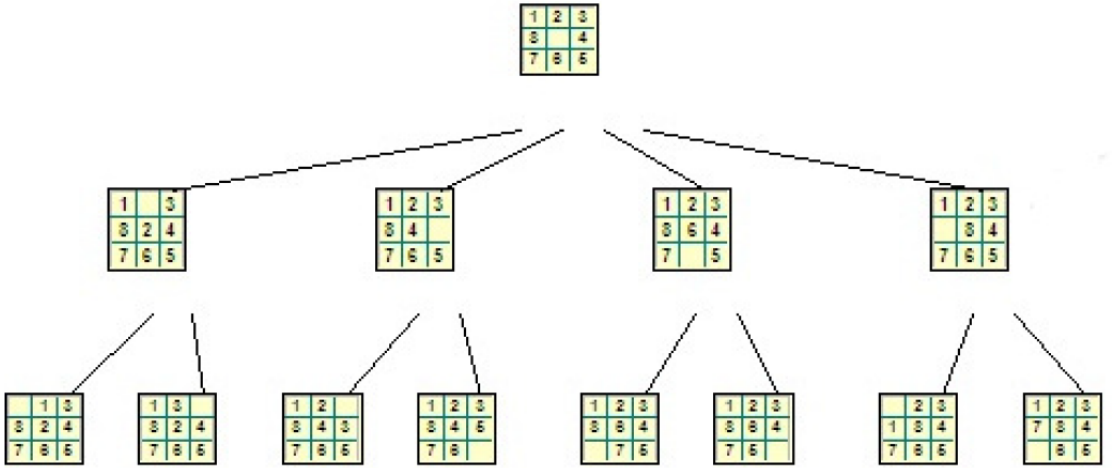
- Um grafo pode ser usado para representar um espaço de estados, onde:
  - Os nós correspondem a situações de um problema;
  - As arestas correspondem aos movimentos permitidos;
  - Um dado problema é solucionado encontrando-se um caminho no grafo;
- Se não houver custos, há interesse em soluções de caminho mínimo;

Busca em Estado de Espaço

Quebra cabeça 8 (Estado Inicial e Final)



Busca em Estado de Espaço



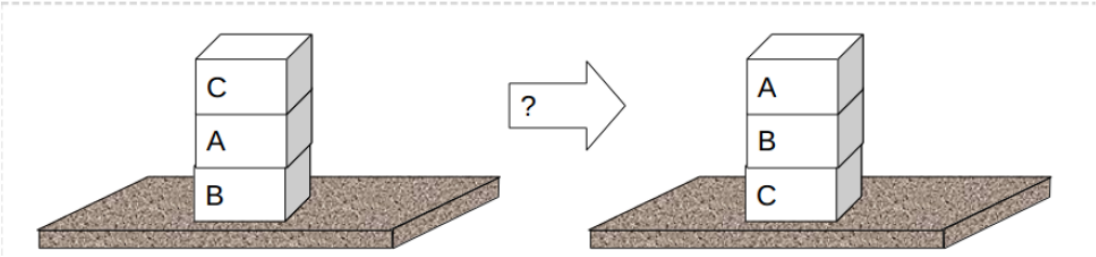
Exemplo Prático

SCAN ME

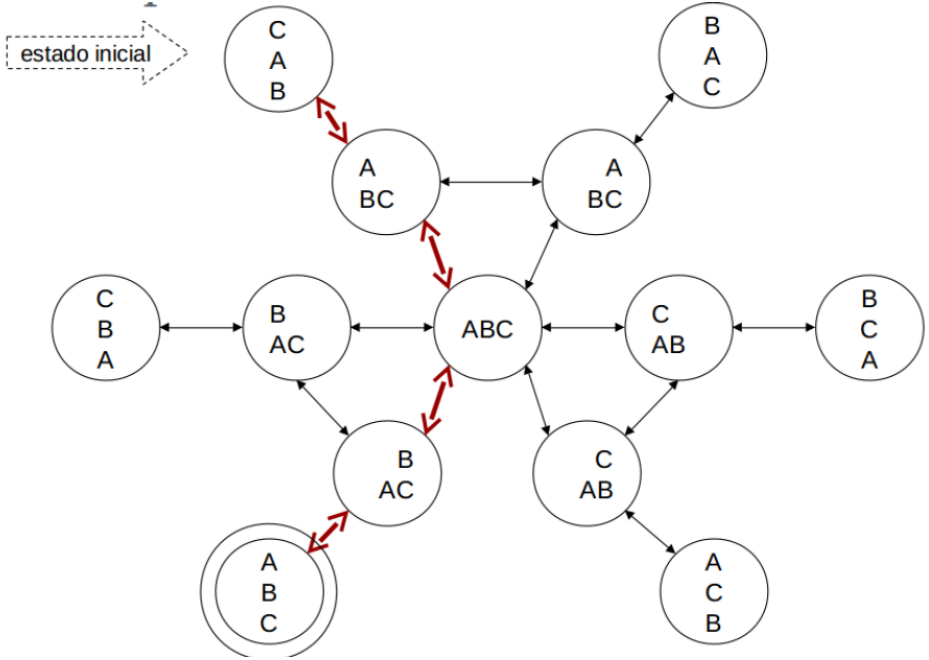
https://colab.research.google.com/drive/1z8BFS-lzqJjPT7KsMjij9oi8c6Lh4cBJ?usp=sharing

Busca em Estado de Espaço

- Considere o problema de encontrar um plano (estratégia) para rearranjar uma pilha de blocos como na figura;
- Somente é permitido um movimento por vez
- Um bloco somente pode ser movido se não há nada em seu topo
- Um bloco pode ser colocado na mesa ou acima de outro bloco



Busca em Estado de Espaço



Aspirador de Pó

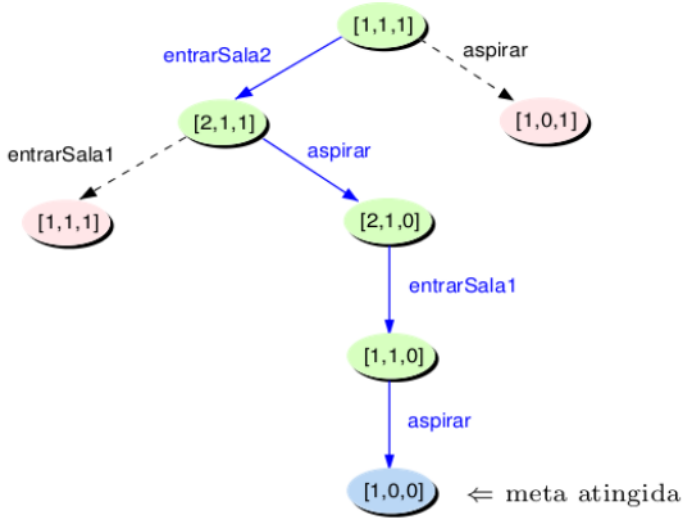
Um espaço de estados e definido por um conjunto S de estados e por um conjunto A de ações que mapeiam um estado em outro. Estados S ao representados por estruturas, onde cada componente denota um atributo do estado representado. Por exemplo, no Mundo do Aspirador, cada estado pode ser representado por uma estrutura da forma $[X, Y, Z]$, onde $X \in \{1, 2\}$ indica a posição do aspirador, $Y \in \{0, 1\}$ indica se a primeira sala está suja e $Z \in \{0, 1\}$ indica se a segunda sala está suja. Dessa forma, o estado em que o aspirador encontra-se na segunda sala e apenas essa sala está suja e representada por $[2, 0, 1]$.

Aspirador de Pó

*** Representação dos estados

- O conjunto de estados para o Mundo do Aspirador é $S =$ $\{[1,0,0],[1,0,1],[1,1,0],[1,1,1],[2,0,0],[2,0,1],[2,1,0],[2,1,1]\}$.
- O problema de busca conta com os seguintes componentes:
  - um espaço de estados (denotado pelos conjuntos S e A);
  - um estado inicial (denotado por um estado particular $s_0 \in S$);
  - um conjunto de estados meta (denotado por um conjunto $G \subseteq S$);

Aspirador de Pó



Aspirador de Pó

***** Representação das ações**

- As ações são representadas por operadores da forma:
  $oper\ (, s, s') = \beta$
  - α: É a ação que transforma um estado;
  - s: estado inicial;
  - s': estado final;
  - β: condição
- Então, temos:
  $A = \{oper(entrarSala1, [2, Y, Z], [1, Y, Z]),$
  $oper(entrarSala2, [1, Y, Z], [2, Y, Z]),$
  $oper(aspirar, [1, 1, Z], [1, 0, Z]),$
  $oper(aspirar, [2, Y, 1], [2, Y, 0])\}$

Aspirador de Pó

***** O problema de busca**

- espaço de estados: conjuntos S e A;
- estado inicial: $[1, 1, 1]$ e estados meta: $G = \{[1, 0, 0], [2, 0, 0]\}$.
- A solução para um problema de busca consiste numa sequência de ações que rotulam o caminho que leva do estado inicial a um dos estados meta no espaço de estados do problema.
- O rastreamento da chamada $Busca(A, s_0, G)$ pode produzir, por exemplo, a árvore de busca apresentada no próximo slide:

OBRIGADO