UNIDADE II: RESOLUÇÃO DE PROBLEMAS ATRAVÉS DE BUSCA

- Sumário:
  - Introdução
  - Agentes de resolução de problemas
  - Exemplos de problemas
  - Busca de soluções

Objectivos

- Adquirir uma noção acerca da resolução de problemas através de busca
- Descrever as características básicas de um agente de resolução de problemas
- Descrever alguns exemplos de problemas que podem ser resolvidos mediante busca
- Adquirir uma noção sobre o paradigma da busca num espaço de estados

Problema(1/3)

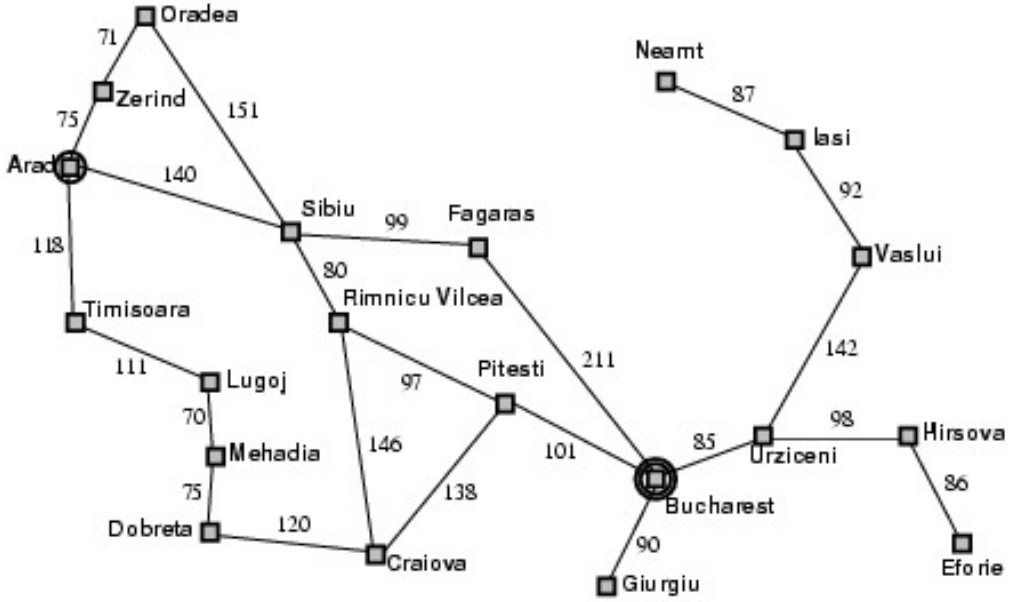
- Um agente se encontra na cidade de Arad (Roménia) em gozo de férias. O mesmo possui um bilhete de regresso não reembolsável, com partida de Bucareste no dia seguinte
- Se considera as acções do agente ao nível da condução de uma cidade para outra, avaliando as diversas alternativas

Problema(2/3)

- Objectivo -> chegar à Bucareste…
- A ***formulação de um objectivo***, com base na ***situação actual*** e na ***medida de desempenho*** do agente é o primeiro passo na ***resolução de problemas***

Problema(3/3)

- Como proceder?
- O agente pode utilizar a informação proporcionada por um mapa para considerar as diferentes etapas de uma hipotética viagem até Bucareste



Agentes e resolução de problemas(1/3)

- Nestes casos se utiliza um tipo de *agente baseado em objectivos* designado ***agente de resolução de problemas***
- Diante de várias alternativas imediatas de valor desconhecido, estes decidem o que fazer analisando possíveis sequências de acções que o conduzem a estados de valor conhecido e seleccionam a melhor sequência

Agentes e resolução de problemas(2/3)

- Determinação da sequência designada ***busca***
- Desenho do agente:
  - Formular problema -> buscar -> executar

Agentes e resolução de problemas(3/3)

- **Função** AgenteResoluçãoProblemaSimples(*percepção*)
  retorna uma acção
  - **Static:**
    - *seq*, sequência de acções, inicialmente vazia
    - *estado*, uma descrição do actual estado do mundo
    - *objectivo*, um objectivo, inicialmente nulo
    - *problema*, formulação de um problema
  - *estado* <- ActualizaEstado(*estado*, *percepção*)
  - **Se** *seq* está vazia **então do**
    - *objectivo* <- FormulaObjectivo(*estado*)
    - *problema* <- FormulaProblema(*estado*, *objectivo*)
    - *seq* <- Busca(*problema*)
  - *acção* <- Primeira(*seq*)
  - *seq* <- Resto(*seq*)
  - **Retorna** *acção*

Acerca do ambiente

- Estático
  - A formulação e resolução do problema é efectuada sem atender às mudanças que ocorrem no ambiente
- Observável
  - Se assume que o estado inicial do ambiente é previamente conhecido
- Discreto
  - Pela enumeração de diferentes sequências alternativas
- Determinístico
  - O próximo estado do agente deve ser determinado pelo estado *actual* + *acção*

Formulação do problema

- Consiste em decidir que acções e estados considerar dado um objectivo
- Um problema é definido formalmente por 4 componentes:
  - O ***estado inicial***
  - As possíveis ***acções*** do agente
    - Geralmente utiliza uma função sucessor
    - Estado inicial e função sucessor definem o ***espaço de estados***
  - ***Teste de satisfação do objectivo***
    - Utilizado para determinar se um estado constitui ou não o objectivo
  - O ***custo do caminho***
    - Função que atribui um custo numérico a cada trajectória

Formulação do problema: exemplo

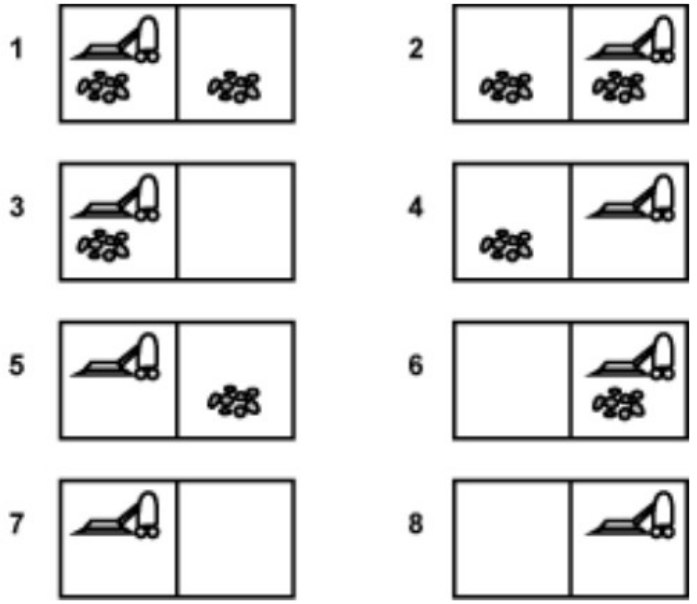
- De férias na Romênia, objectivo -> Bucareste
- Problema:
  - estados: cidades da Roménia
  - Estado inicial: Arad
  - Função sucessor: cidades alcançáveis dirigindo a partir da actual
  - Teste obj: estar em Bucareste
  - Custo: distância entre cidades
- Encontrar solução:
  - sequência de cidades, ex., Arad, Sibiu, Fagaras, Bucareste

Exemplos de problemas

- Resolução de problemas tem sido aplicada a uma grande variedade de ambientes de tarefa
- Os problemas geralmente se subdividem em
  - Miniproblemas
    - Utilizados para ilustrar ou exercitar métodos de resolução
  - Problemas reais
    - Problemas que se pretende realmente solucionar

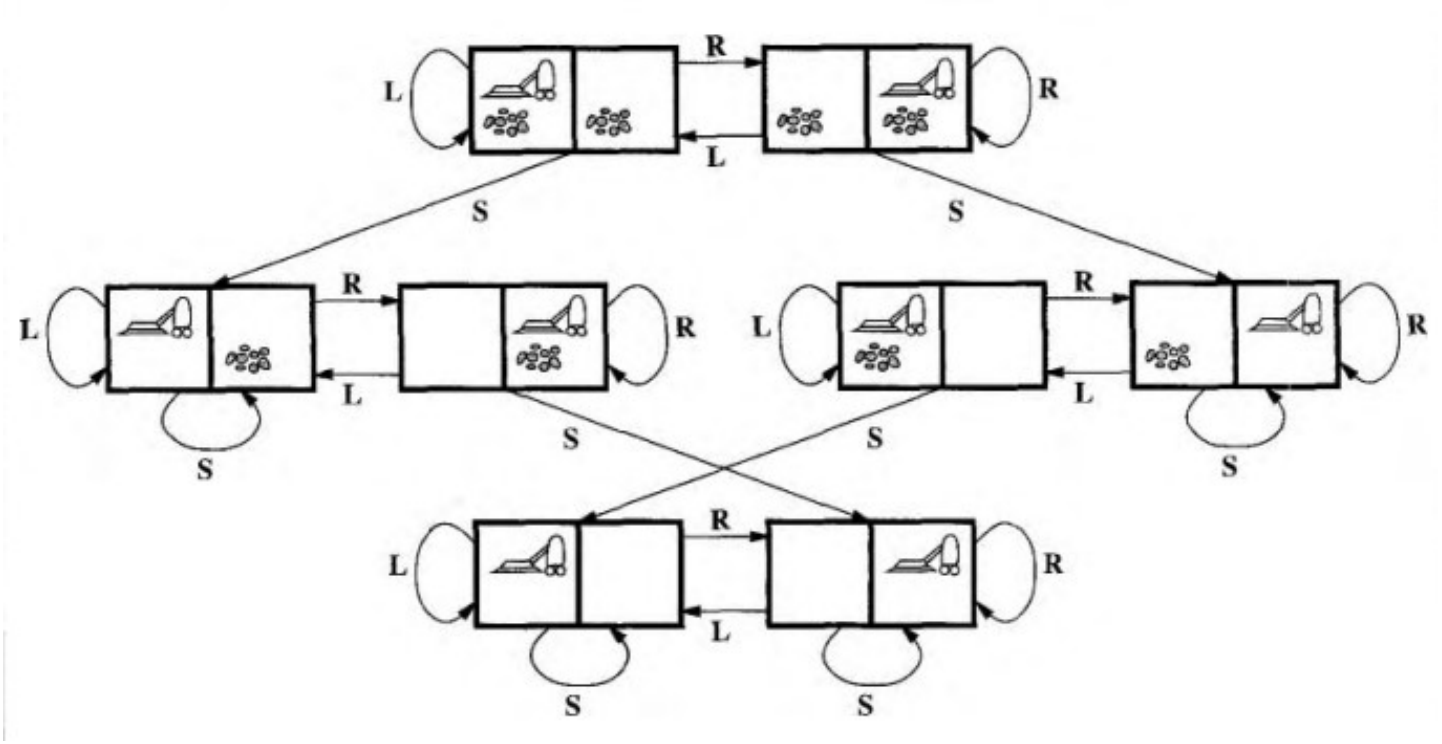
Exemplos: miniproblemas(1/4)

- Mundo do aspirador…
  - Estados: o agente se encontra num de dois quartos que podem estar limpos ou sujos. 8 possíveis estados
  - Estado inicial: qualquer
  - Função sucessor: estados legais derivados de 3 acções (esquerda, direita e limpar)
  - Teste de Obj: se todos os quartos estão limpos
  - Custo: cada passo custa 1



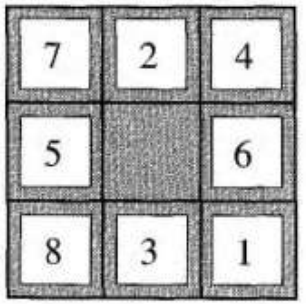
Exemplos: miniproblemas(2/4)

- Mundo do aspirador…

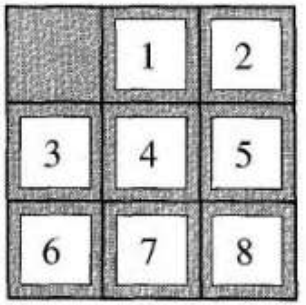


Exemplos: miniproblemas(3/4)

- Puzzle de 8 peças
  - Estados: especifica o posicionamento das 8 peças e do espaço em branco
  - Estado inicial: qualquer
  - Função sucessor: estados legais derivados de quatro acções (espaço em branco move para Esquerda, direita, cima e baixo)
  - Teste obj: verifica se estado coincide com configuração pretendida
  - Custo: cada passo custa 1



Start State



Goal State

Exemplos: miniproblemas(4/4)

- Outros:
  - 8 rainhas
  - Torre de Hanói
  - Vasilhames
  - Missionários e canibais…

Exemplos: problemas reais(1/2)

- Problemas de roteamento
  - Planeamento de rotas de aviões
  - Sistemas de planeamento de viagens
  - Planeamento de operações militares
  - Caixeiro viajante
  - Rotas em redes de computadores
- Circuitos electrónicos
  - Posicionamento de componentes
  - Rotas de circuitos

Exemplos: problemas reais(2/2)

- Navegação de robots
  - Navegação e busca de rotas em ambientes reais
  - Montagem de objetos por robôs
- …

Busca de soluções

- Realizada através de uma ***busca no espaço de estados***
- Se utiliza uma ***árvore de busca*** que é determinada pelo ***estado inicial*** e a ***função sucessor***
  - Expande-se o estado actual, gerando um novo conjunto de sucessores
  - Próximo estado a ser expandido definido por uma estratégia de busca
  - Prossegue-se até chegar ao estado final (objectivo) ou falhar a busca

Busca de soluções: descrição informal

Função BuscaEmArvore(*Problema*, *Estratégia*) **retorna** solução ou falha

Inicio

Inicializa a arvore usando o estado inicial do *Problema*

loop do

se não existem candidatos para serem expandidos **então**
retorna falha

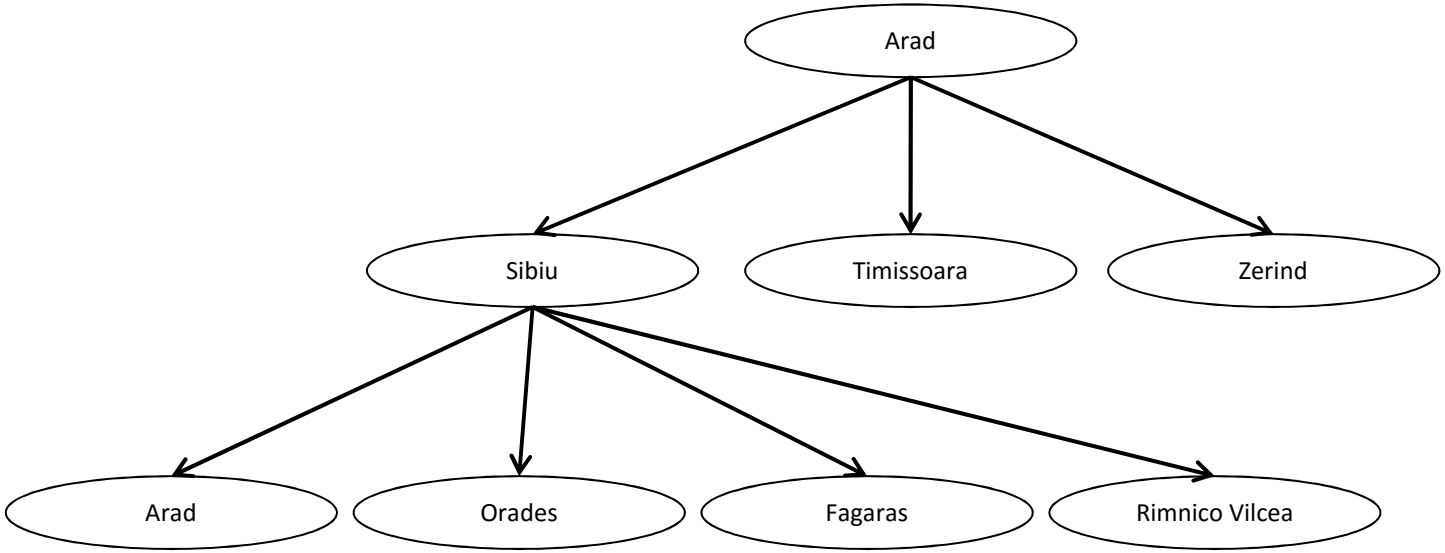
Escolhe um nó folha para ser expandido de acordo com a *Estratégia*

se Se o nó possuir o estado final **então**
retorna solução correspondente

se não
expande o nó e adiciona os nós resultantes à arvore de busca

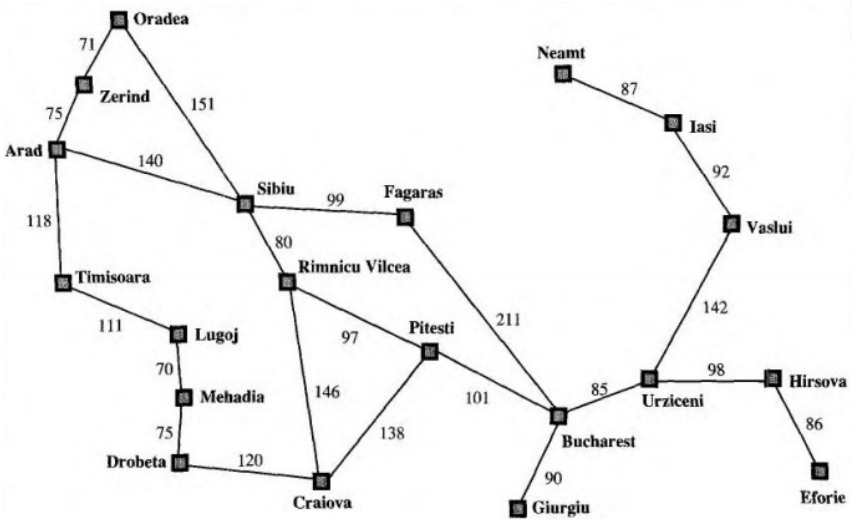
Fim

Exemplo de árvore de busca



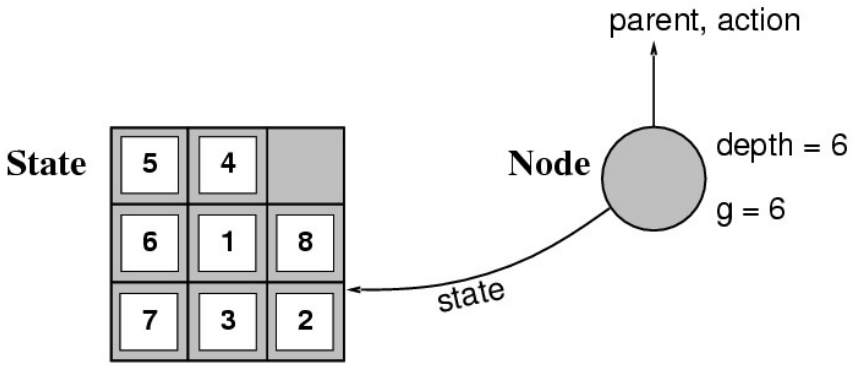
Notar que…

- O *espaço de estados* é **diferente** da *árvore de busca*
  - Existem 20 estados no espaço de estados
  - Existe um número infinito de trajectórias no espaço de estados →
    - Árvore de busca pode ter infinitos nós



Estados vs. nós

- Um ***estado*** é (uma representação de) uma configuração física
- Um ***nó*** é uma estrutura de dados que é parte da árvore de busca e inclui ***estado***, ***nó pai***, ***acção***, ***custo do caminho*** *g(x)*, ***profundidade***



- A coleção de ***nós*** que foram gerados, mas ainda não foram expandidos é chamada de ***fronteira*** (ou ***fringe***)
  - Geralmente implementados como uma fila.
  - A maneira como os nós entram na fila determina a estratégia de busca.

Busca em árvore: algoritmo

```
Função BuscaEmArvore(Problema, fronteira) retorna solução ou
   falha
Inicio
  fronteira ← InsereNaFila(FazNó(Problema[EstadoInicial]),
   fronteira)
  loop do
    se FilaVazia(fronteira) então
      retorna falha
   nó ← RemovePrimeiro(fronteira)
   se nó[Estado] for igual a Problema[EstadoFinal] então
     retorna Solução(nó)
   fronteira ← InsereNaFila(ExpandeFronteira(nó, Problema),
  fronteira)
Fim
```

Desempenho da resolução de problemas

- Estratégias são avaliadas de acordo com os seguintes critérios:
  - ***completitude***: o algoritmo sempre encontra a solução se ela existe?
  - ***optimização***: a estratégia encontra a solução óptima?
  - ***complexidade de tempo***: quanto tarda a encontrar a solução? (medido através do número de nós gerado)
  - ***complexidade de espaço***: quanta memória se necessita? (medido através do número máximo de nós na memória)
- Complexidade de tempo e espaço são medidas em termos de:
  - *b:* máximo factor de ramificação da árvore (número máximo de sucessores de qualquer nó)
  - *d:* profundidade do nó objectivo menos profundo
  - *m*: o comprimento máximo de qualquer caminho no espaço de estados (pode ser ∞)

Bibliografia

- Russell & Norvig, cap. 3, ep. 3.1, 3.2 e 3.3
- Costa & Simões, pg. 71 – 78
- Palma Méndez & Marín Morales, pg. 309 – 315