

# Inteligência Artificial – ACH2016

## Aula 02 – Busca como Solução de Problemas

Norton Trevisan Roman  
(norton@usp.br)

21 de fevereiro de 2019

# Solução de Problemas

Exige, em geral, algoritmo que:

# Solução de Problemas

Exige, em geral, algoritmo que:

- Considere futuras ações juntamente com quão desejáveis são seus resultados

# Solução de Problemas

Exige, em geral, algoritmo que:

- Considere futuras ações juntamente com quão desejáveis são seus resultados
- Decida o que fazer com base na sequência de ações que levam a estados desejáveis

# Solução de Problemas

## Exige, em geral, algoritmo que:

- Considere futuras ações juntamente com quão desejáveis são seus resultados
- Decida o que fazer com base na sequência de ações que levam a estados desejáveis
- É uma abordagem comum em I.A. buscar por soluções em um **espaço de estados** e explorar alternativas até o objetivo

# Busca como Solução de Problemas

Reduzir o problema a uma busca em um grafo

# Busca como Solução de Problemas

## Reduzir o problema a uma busca em um grafo

- Cada nó no espaço de busca é um estado do mundo

# Busca como Solução de Problemas

## Reduzir o problema a uma busca em um grafo

- Cada nó no espaço de busca é um estado do mundo
- As arestas são ações que levam de um estado a outro



# Busca como Solução de Problemas

## Reduzir o problema a uma busca em um grafo

- Cada nó no espaço de busca é um estado do mundo
- As arestas são ações que levam de um estado a outro
- A solução é um caminho (sequência de ações) que levam de um estado inicial a um estado final (objetivo final)

# Busca como Solução de Problemas

## Reduzir o problema a uma busca em um grafo

- Cada nó no espaço de busca é um estado do mundo
- As arestas são ações que levam de um estado a outro
- A solução é um caminho (sequência de ações) que levam de um estado inicial a um estado final (objetivo final)
  - Achar a solução torna-se então um problema de busca no grafo

# Busca como Solução de Problemas

## Exemplo

# Busca como Solução de Problemas

## Exemplo

- **Mundo**

# Busca como Solução de Problemas

## Exemplo

- **Mundo**
  - Conjunto de 3 blocos que podem ser empilhados

# Busca como Solução de Problemas

## Exemplo

- **Mundo**
  - Conjunto de 3 blocos que podem ser empilhados
- **Estado do mundo**

# Busca como Solução de Problemas

## Exemplo

- **Mundo**
  - Conjunto de 3 blocos que podem ser empilhados
- **Estado do mundo**
  - Uma configuração de empilhamento dos 3 blocos

# Busca como Solução de Problemas

## Exemplo

- **Mundo**
  - Conjunto de 3 blocos que podem ser empilhados
- **Estado do mundo**
  - Uma configuração de empilhamento dos 3 blocos
- **Nós**



# Busca como Solução de Problemas

## Exemplo

- **Mundo**

- Conjunto de 3 blocos que podem ser empilhados

- **Estado do mundo**

- Uma configuração de empilhamento dos 3 blocos

- **Nós**

- Descrevem um estado do mundo (que blocos estão em cima de quais)

# Busca como Solução de Problemas

## Exemplo

- **Arestas**

# Busca como Solução de Problemas

## Exemplo

- **Arestas**
  - Ações que levam de um estado a outro (de uma configuração de blocos a outra)

# Busca como Solução de Problemas

## Exemplo

- **Arestas**
  - Ações que levam de um estado a outro (de uma configuração de blocos a outra)
  - Nesse caso, empilhamento de um bloco

# Busca como Solução de Problemas

## Exemplo

- **Arestas**
  - Ações que levam de um estado a outro (de uma configuração de blocos a outra)
  - Nesse caso, empilhamento de um bloco
- **Caminho no grafo**

# Busca como Solução de Problemas

## Exemplo

- **Arestas**

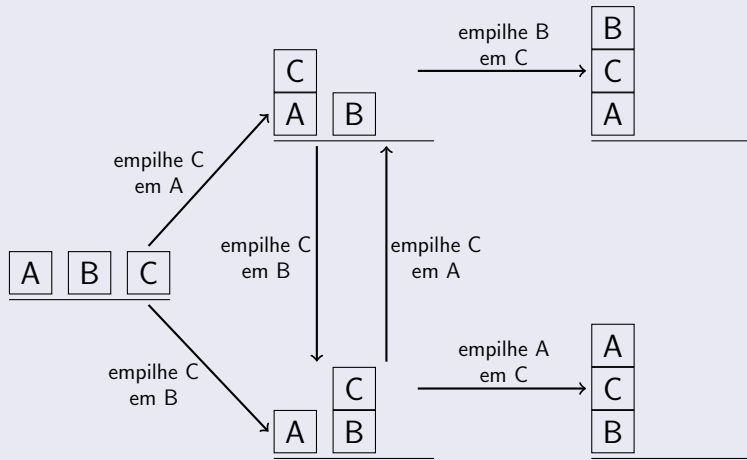
- Ações que levam de um estado a outro (de uma configuração de blocos a outra)
- Nesse caso, empilhamento de um bloco

- **Caminho no grafo**

- Um plano de ação para passar de um estado inicial a um final

# Busca como Solução de Problemas

## Exemplo (reduzido)



Fonte: Adaptado de MIT, 6-034 Spring-2005.

# Busca como Solução de Problemas

## Alguns Tipos de Problemas



# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Determinísticos, totalmente observáveis

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Determinísticos, totalmente observáveis
  - Conhecemos todos os estados

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Determinísticos, totalmente observáveis
  - Conhecemos todos os estados
  - Sabemos exatamente em que estados estamos

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Determinísticos, totalmente observáveis
  - Conhecemos todos os estados
  - Sabemos exatamente em que estados estamos
  - Sabemos exatamente em que estado estaremos se executarmos determinada ação

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Determinísticos, totalmente observáveis
  - Conhecemos todos os estados
  - Sabemos exatamente em que estados estamos
  - Sabemos exatamente em que estado estaremos se executarmos determinada ação
- Não observáveis

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Determinísticos, totalmente observáveis
  - Conhecemos todos os estados
  - Sabemos exatamente em que estados estamos
  - Sabemos exatamente em que estado estaremos se executarmos determinada ação
- Não observáveis
  - Não sabemos onde estamos

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Não determinísticos e/ou parcialmente observáveis:

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Não determinísticos e/ou parcialmente observáveis:
  - Continuamente obtemos informação adicional sobre o estado em que estamos



# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Não determinísticos e/ou parcialmente observáveis:
  - Continuamente obtemos informação adicional sobre o estado em que estamos
  - Abordados por meio de planos de contingência ou política de ação

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Não determinísticos e/ou parcialmente observáveis:
  - Continuamente obtemos informação adicional sobre o estado em que estamos
  - Abordados por meio de planos de contingência ou política de ação
- De espaço de estados desconhecido:

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Não determinísticos e/ou parcialmente observáveis:
  - Continuamente obtemos informação adicional sobre o estado em que estamos
  - Abordados por meio de planos de contingência ou política de ação
- De espaço de estados desconhecido:
  - Nada sabemos sobre nenhum estado

# Busca como Solução de Problemas

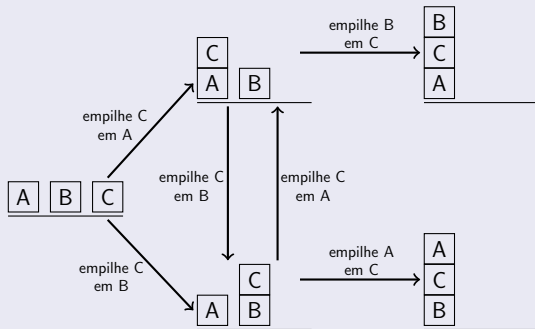
## Alguns Tipos de Problemas

- Não determinísticos e/ou parcialmente observáveis:
  - Continuamente obtemos informação adicional sobre o estado em que estamos
  - Abordados por meio de planos de contingência ou política de ação
- De espaço de estados desconhecido:
  - Nada sabemos sobre nenhum estado
  - Deve-se explorar o ambiente

# Busca como Solução de Problemas

## Alguns Tipos de Problemas

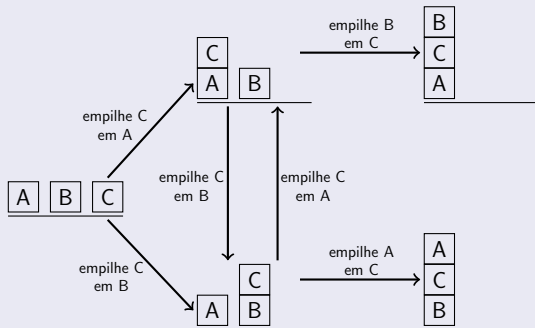
- Como esse problema pode ser classificado?



# Busca como Solução de Problemas

## Alguns Tipos de Problemas

- Como esse problema pode ser classificado?



Determinístico, totalmente observável

# Busca como Solução de Problemas

E quando apelamos à I.A.?

# Busca como Solução de Problemas

## E quando apelamos à I.A.?

- Quando o tamanho do grafo é inviável às técnicas exatas



# Busca como Solução de Problemas

## E quando apelamos à I.A.?

- Quando o tamanho do grafo é inviável às técnicas exatas
- Quando há outras variáveis sobre as quais não se tem controle. Ex:

# Busca como Solução de Problemas

## E quando apelamos à I.A.?

- Quando o tamanho do grafo é inviável às técnicas exatas
- Quando há outras variáveis sobre as quais não se tem controle. Ex:
  - Probabilidade de algo inesperado acontecer (ambiente não determinístico ou não observável)

# Busca como Solução de Problemas

## E quando apelamos à I.A.?

- Quando o tamanho do grafo é inviável às técnicas exatas
- Quando há outras variáveis sobre as quais não se tem controle. Ex:
  - Probabilidade de algo inesperado acontecer (ambiente não determinístico ou não observável)
  - Desconhecimento de todos os estados no grafo

# Busca como Solução de Problemas

## E quando apelamos à I.A.?

- Quando o tamanho do grafo é inviável às técnicas exatas
- Quando há outras variáveis sobre as quais não se tem controle. Ex:
  - Probabilidade de algo inesperado acontecer (ambiente não determinístico ou não observável)
  - Desconhecimento de todos os estados no grafo
  - Mudanças dinâmicas no próprio grafo

# Busca como Solução de Problemas

## E quando apelamos à I.A.?

- Quando o tamanho do grafo é inviável às técnicas exatas
- Quando há outras variáveis sobre as quais não se tem controle. Ex:
  - Probabilidade de algo inesperado acontecer (ambiente não determinístico ou não observável)
  - Desconhecimento de todos os estados no grafo
  - Mudanças dinâmicas no próprio grafo
  - Ou seja, nos 3 últimos tipos de problemas vistos

# Busca como Solução de Problemas

## Classificação dos Algoritmos

## Classificação dos Algoritmos

- Não informados

## Classificação dos Algoritmos

- Não informados
  - Não recebem qualquer outra informação sobre o problema, além de sua definição



## Classificação dos Algoritmos

- Não informados
  - Não recebem qualquer outra informação sobre o problema, além de sua definição
- Informados

# Busca como Solução de Problemas

## Classificação dos Algoritmos

- Não informados
  - Não recebem qualquer outra informação sobre o problema, além de sua definição
- Informados
  - Possuem pistas sobre onde encontrar a solução

# Busca como Solução de Problemas

## Classificação dos Algoritmos

- Não informados
  - Não recebem qualquer outra informação sobre o problema, além de sua definição
- Informados
  - Possuem pistas sobre onde encontrar a solução
  - Possuem acesso a alguma informação específica da tarefa, que os ajude a executá-la

## Classificação dos Algoritmos

- Não informados
  - Não recebem qualquer outra informação sobre o problema, além de sua definição
- Informados
  - Possuem pistas sobre onde encontrar a solução
  - Possuem acesso a alguma informação específica da tarefa, que os ajude a executá-la
    - Ex: saber que determinada via tem maior chance de engarrafamentos

# Busca como Solução de Problemas

## Classificação dos Algoritmos

- De caminho ótimo

# Busca como Solução de Problemas

## Classificação dos Algoritmos

- De caminho ótimo
  - Buscam o melhor meio para resolver o problema

# Busca como Solução de Problemas

## Classificação dos Algoritmos

- De caminho ótimo
  - Buscam o melhor meio para resolver o problema
  - Tentam otimizar a solução do problema

# Busca como Solução de Problemas

## Classificação dos Algoritmos

- De caminho ótimo
  - Buscam o melhor meio para resolver o problema
  - Tentam otimizar a solução do problema
- De qualquer caminho



# Busca como Solução de Problemas

## Classificação dos Algoritmos

- De caminho ótimo
  - Buscam o melhor meio para resolver o problema
  - Tentam otimizar a solução do problema
- De qualquer caminho
  - Buscam tão somente uma solução para o problema (um caminho qualquer até a solução)

# Busca como Solução de Problemas

## Passos para uma Solução

# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados

# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo

# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema

# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema
  - Estado inicial

# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema
  - Estado inicial
  - Ações possíveis em cada estado

# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema
  - Estado inicial
  - Ações possíveis em cada estado
  - Uma descrição do que cada ação faz (modelo de transição)



# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema
  - Estado inicial
  - Ações possíveis em cada estado
  - Uma descrição do que cada ação faz (modelo de transição)
  - Teste do objetivo, para ver se estamos no estado final

# Busca como Solução de Problemas

## Passos para uma Solução


- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema
  - Estado inicial
  - Ações possíveis em cada estado
  - Uma descrição do que cada ação faz (modelo de transição)
  - Teste do objetivo, para ver se estamos no estado final
  - Avaliação do custo

# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema
  - Estado inicial
  - Ações possíveis em cada estado
  - Uma descrição do que cada ação faz (modelo de transição)
  - Teste do objetivo, para ver se estamos no estado final
  - Avaliação do custo

Juntos, estado inicial, ações e modelo de transição definem o **espaço de estados** do problema



# Busca como Solução de Problemas

## Passos para uma Solução

- Definição dos possíveis estados
- Formulação do objetivo
- Formulação do problema
  - Estado inicial
  - Ações possíveis em cada estado
  - Uma descrição do que cada ação faz (modelo de transição)
  - Teste do objetivo, para ver se estamos no estado final
  - Avaliação do custo

**Espaço de estados:**  
conjunto de todos  
os estados atingíveis  
a partir do estado  
inicial por alguma  
sequência de ações

# Busca como Solução de Problemas

## Passos para uma Solução

- Busca de sequência de atos que solucionem o problema

# Busca como Solução de Problemas

## Passos para uma Solução

- Busca de sequência de atos que solucionem o problema
- Execução das ações

# Passos para uma Solução

## Definição dos Possíveis Estados

# Passos para uma Solução

## Definição dos Possíveis Estados

- Estados representam todos os aspectos relevantes do problema



# Passos para uma Solução

## Definição dos Possíveis Estados

- Estados representam todos os aspectos relevantes do problema
- São descrições do domínio em um determinado curso de ação (snapshot)

# Passos para uma Solução

## Definição dos Possíveis Estados

- Estados representam todos os aspectos relevantes do problema
- São descrições do domínio em um determinado curso de ação (snapshot)
- Exemplos:

# Passos para uma Solução

## Definição dos Possíveis Estados

- Estados representam todos os aspectos relevantes do problema
- São descrições do domínio em um determinado curso de ação (snapshot)
- Exemplos:
  - Posição de caixas em uma sala (empilhamento)

# Passos para uma Solução

## Definição dos Possíveis Estados

- Estados representam todos os aspectos relevantes do problema
- São descrições do domínio em um determinado curso de ação (snapshot)
- Exemplos:
  - Posição de caixas em uma sala (empilhamento)
  - Nomes de aeroportos (viagem pelo mundo)

# Passos para uma Solução

## Definição dos Possíveis Estados

- Estados representam todos os aspectos relevantes do problema
- São descrições do domínio em um determinado curso de ação (snapshot)
- Exemplos:
  - Posição de caixas em uma sala (empilhamento)
  - Nomes de aeroportos (viagem pelo mundo)
  - Endereço dos aeroportos (translado do hotel ao aeroporto)

# Passos para uma Solução

## Formulação do Objetivo

# Passos para uma Solução

## Formulação do Objetivo

- O objetivo é qualquer conjunto de estados desejáveis (não necessariamente um só)

# Passos para uma Solução

## Formulação do Objetivo

- O objetivo é qualquer conjunto de estados desejáveis (não necessariamente um só)
- Trata-se de identificar os estados em que se deseja estar



# Passos para uma Solução

## Formulação do Objetivo

- O objetivo é qualquer conjunto de estados desejáveis (não necessariamente um só)
- Trata-se de identificar os estados em que se deseja estar
- A tarefa será encontrar uma sequência de atos que leve a um ou mais estados objetivos

# Passos para uma Solução

## Formulação do Problema

# Passos para uma Solução

## Formulação do Problema

- Para tal, deve-se definir:

# Passos para uma Solução

## Formulação do Problema

- Para tal, deve-se definir:
  - Se o ambiente é estático ou dinâmico

# Passos para uma Solução

## Formulação do Problema

- Para tal, deve-se definir:
  - Se o ambiente é estático ou dinâmico
  - Se é observável (conhecemos todos os estados) ou não

# Passos para uma Solução

## Formulação do Problema

- Para tal, deve-se definir:
  - Se o ambiente é estático ou dinâmico
  - Se é observável (conhecemos todos os estados) ou não
  - Se determinístico (soluções são simples sequências de atos) ou não (deve-se lidar com o inesperado)

# Passos para uma Solução

## Formulação do Problema

- Para tal, deve-se definir:
  - Se o ambiente é estático ou dinâmico
  - Se é observável (conhecemos todos os estados) ou não
  - Se determinístico (soluções são simples sequências de atos) ou não (deve-se lidar com o inesperado)
  - O nível dos detalhes envolvidos. Ex: se consideramos “sair do estacionamento” uma ação, ou teremos que descrever cada um de seus passos

# Passos para uma Solução

## Formulação do Problema: Ações Possíveis



# Passos para uma Solução

## Formulação do Problema: Ações Possíveis

- Dado um estado particular, quais são as ações permitidas e a que estados nos levam?

# Passos para uma Solução

## Formulação do Problema: Ações Possíveis

- Dado um estado particular, quais são as ações permitidas e a que estados nos levam?
- Ações devem ser

# Passos para uma Solução

## Formulação do Problema: Ações Possíveis

- Dado um estado particular, quais são as ações permitidas e a que estados nos levam?
- Ações devem ser
  - Determinísticas: sabemos exatamente o estado em que o algoritmo estará após a execução de uma ação

# Passos para uma Solução

## Formulação do Problema: Ações Possíveis

- Dado um estado particular, quais são as ações permitidas e a que estados nos levam?
- Ações devem ser
  - Determinísticas: sabemos exatamente o estado em que o algoritmo estará após a execução de uma ação
  - Discretas: não precisamos representar o que acontece enquanto a ação está sendo executada.

# Passos para uma Solução

## Formulação do Problema: Ações Possíveis

- Dado um estado particular, quais são as ações permitidas e a que estados nos levam?
- Ações devem ser
  - Determinísticas: sabemos exatamente o estado em que o algoritmo estará após a execução de uma ação
  - Discretas: não precisamos representar o que acontece enquanto a ação está sendo executada.
    - Ex: assumimos que um voo nos leva ao destino, sem nos importarmos com o que acontece durante ele

# Passos para uma Solução

## Teste do Objetivo

# Passos para uma Solução

## Teste do Objetivo

- Determina se um dado estado é um estado-objetivo

# Passos para uma Solução

## Teste do Objetivo

- Determina se um dado estado é um estado-objetivo
- Exemplos:
  - Todos os blocos estão empilhados



# Passos para uma Solução

## Teste do Objetivo

- Determina se um dado estado é um estado-objetivo
- Exemplos:
  - Todos os blocos estão empilhados
  - Chegamos ao aeroporto de destino

# Passos para uma Solução

## Teste do Objetivo

- Determina se um dado estado é um estado-objetivo
- Exemplos:
  - Todos os blocos estão empilhados
  - Chegamos ao aeroporto de destino
  - Chegamos ao aeroporto em tempo para o voo

# Passos para uma Solução

## Avaliação do Custo

# Passos para uma Solução

## Avaliação do Custo

- Atribui um valor numérico a cada solução obtida

# Passos para uma Solução

## Avaliação do Custo

- Atribui um valor numérico a cada solução obtida
  - A cada caminho no grafo

# Passos para uma Solução

## Avaliação do Custo

- Atribui um valor numérico a cada solução obtida
  - A cada caminho no grafo
- Via de regra, o custo do caminho é a soma dos custos de cada ação dentro dele

# Passos para uma Solução

## Avaliação do Custo

- Atribui um valor numérico a cada solução obtida
  - A cada caminho no grafo
- Via de regra, o custo do caminho é a soma dos custos de cada ação dentro dele
  - Pode haver custos marginais, atrelados a eventos externos, por exemplo

# Passos para uma Solução

## Avaliação do Custo

- Atribui um valor numérico a cada solução obtida
  - A cada caminho no grafo
- Via de regra, o custo do caminho é a soma dos custos de cada ação dentro dele
  - Pode haver custos marginais, atrelados a eventos externos, por exemplo
- Ex:
  - Tempo de viagem



# Passos para uma Solução

## Avaliação do Custo

- Atribui um valor numérico a cada solução obtida
  - A cada caminho no grafo
- Via de regra, o custo do caminho é a soma dos custos de cada ação dentro dele
  - Pode haver custos marginais, atrelados a eventos externos, por exemplo
- Ex:
  - Tempo de viagem
  - Número de operações

# Passos para uma Solução

## Avaliação do Custo

- Atribui um valor numérico a cada solução obtida
  - A cada caminho no grafo
- Via de regra, o custo do caminho é a soma dos custos de cada ação dentro dele
  - Pode haver custos marginais, atrelados a eventos externos, por exemplo
- Ex:
  - Tempo de viagem
  - Número de operações
  - Etc.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:
  - Cada uma das possíveis localizações (inteiras) dos números

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:
  - Cada uma das possíveis localizações (inteiras) dos números
  - Ignoro localizações enquanto são movidos

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:
  - Cada uma das possíveis localizações (inteiras) dos números
  - Ignoro localizações enquanto são movidos
- Objetivo:

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:
  - Cada uma das possíveis localizações (inteiras) dos números
  - Ignoro localizações enquanto são movidos
- Objetivo:
  - Ordenar os números

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.



# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:
  - Cada uma das possíveis localizações (inteiras) dos números
  - Ignoro localizações enquanto são movidos
- Objetivo:
  - Ordenar os números
- Formulação:

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:
  - Cada uma das possíveis localizações (inteiras) dos números
  - Ignoro localizações enquanto são movidos
- Objetivo:
  - Ordenar os números
- Formulação:
  - Estado inicial: qualquer um

1	2	3
4	5	6
7	8	

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Estados:
  - Cada uma das possíveis localizações (inteiras) dos números
  - Ignoro localizações enquanto são movidos
- Objetivo:
  - Ordenar os números
- Formulação:
  - Estado inicial: qualquer um. Ex:

7	2	4
5		6
8	3	1

Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Formulação (cont.):
  - Ações possíveis:

1	2	3
4	5	6
7	8	

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Formulação (cont.):
  - Ações possíveis:
    - Ir à esquerda, direita, para cima ou para baixo, dependendo de onde estiver o número movido

1	2	3
4	5	6
7	8	

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Formulação (cont.):
  - Ações possíveis:
    - Ir à esquerda, direita, para cima ou para baixo, dependendo de onde estiver o número movido
    - Deve ser movido de modo a ocupar posição em branco

1	2	3
4	5	6
7	8	

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Formulação (cont.):
  - Ações possíveis:
    - Ir à esquerda, direita, para cima ou para baixo, dependendo de onde estiver o número movido
    - Deve ser movido de modo a ocupar posição em branco
  - Teste:

1	2	3
4	5	6
7	8	

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Formulação (cont.):
  - Ações possíveis:
    - Ir à esquerda, direita, para cima ou para baixo, dependendo de onde estiver o número movido
    - Deve ser movido de modo a ocupar posição em branco
  - Teste:
    - Se o estado atual corresponde ao objetivo

1	2	3
4	5	6
7	8	

7	2	4
5		6
8	3	1

?  
=

1	2	3
4	5	6
7	8	



# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Formulação (cont.):
  - Ações possíveis:
    - Ir à esquerda, direita, para cima ou para baixo, dependendo de onde estiver o número movido
    - Deve ser movido de modo a ocupar posição em branco
  - Teste:
    - Se o estado atual corresponde ao objetivo
  - Custo:

1	2	3
4	5	6
7	8	

7	2	4
5		6
8	3	1

?  
=

1	2	3
4	5	6
7	8	

# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça

- Formulação (cont.):
  - Ações possíveis:
    - Ir à esquerda, direita, para cima ou para baixo, dependendo de onde estiver o número movido
    - Deve ser movido de modo a ocupar posição em branco
  - Teste:
    - Se o estado atual corresponde ao objetivo
  - Custo:
    - O número total de movimentos

1	2	3
4	5	6
7	8	

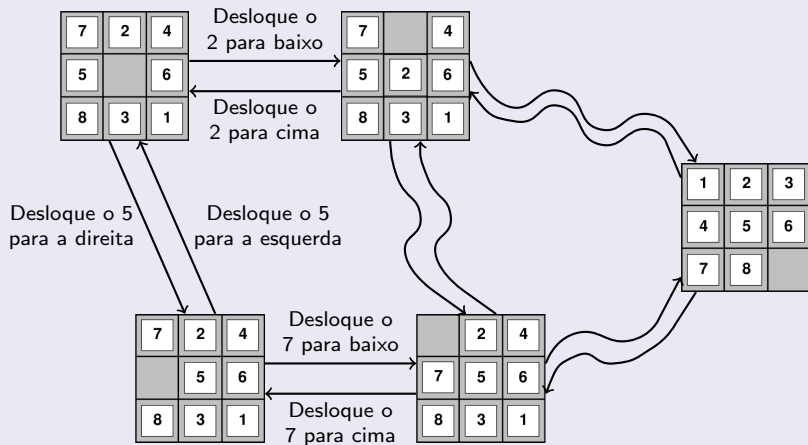
7	2	4
5		6
8	3	1

?  
=

1	2	3
4	5	6
7	8	

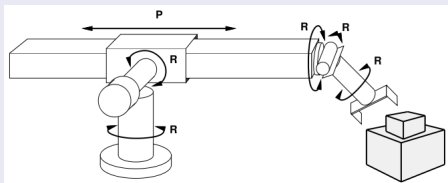
# Busca como Solução de Problemas

## Exemplo: Quebra-Cabeça – Grafo (parcial)



# Busca como Solução de Problemas

## Exemplo: Robô

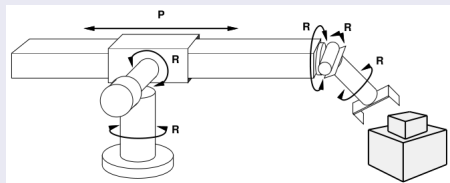


Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Estados:

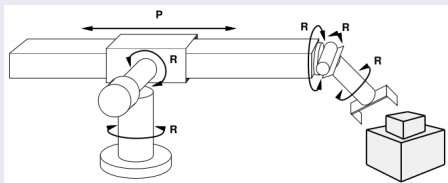


Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Estados:
  - Conjunto de coordenadas (ângulos) de cada parte do robô

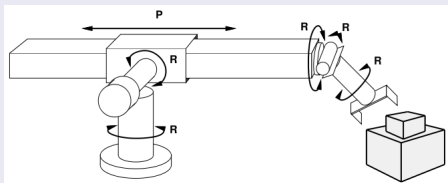


Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Estados:
  - Conjunto de coordenadas (ângulos) de cada parte do robô
  - Posição das partes do objeto a ser montado

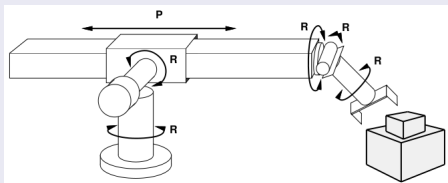


Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Estados:
  - Conjunto de coordenadas (ângulos) de cada parte do robô
  - Posição das partes do objeto a ser montado
- Objetivo:



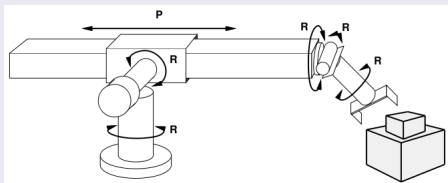
Fonte: AIMA. Russell & Norvig.



# Busca como Solução de Problemas

## Exemplo: Robô

- Estados:
  - Conjunto de coordenadas (ângulos) de cada parte do robô
  - Posição das partes do objeto a ser montado
- Objetivo:
  - Ter o objeto montado

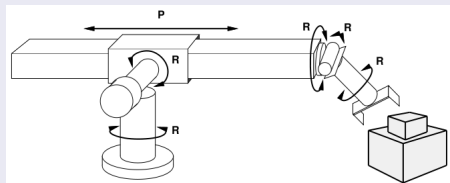


Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Formulação:

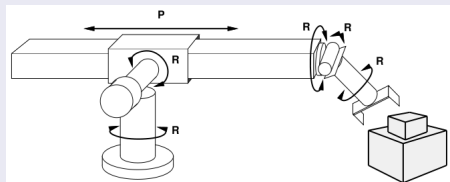


Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Formulação:
  - Estado inicial: posição atual do robô

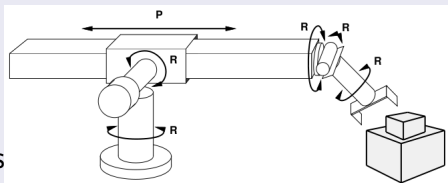


Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Formulação:
  - Estado inicial: posição atual do robô
  - Ações possíveis: Movimentos contínuos das juntas do robô (ângulos de movimento)



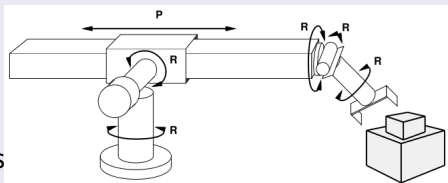
Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Formulação:

- Estado inicial: posição atual do robô
- Ações possíveis: Movimentos contínuos das juntas do robô (ângulos de movimento)
- Teste: Se o objeto está montado (sem levar em conta o robô!)



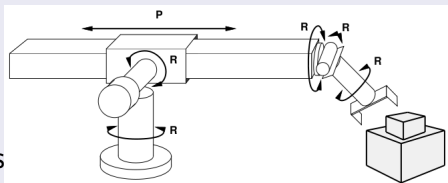
Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Robô

- Formulação:

- Estado inicial: posição atual do robô
- Ações possíveis: Movimentos contínuos das juntas do robô (ângulos de movimento)
- Teste: Se o objeto está montado (sem levar em conta o robô!)
- Custo: Tempo de execução



Fonte: AIMA. Russell & Norvig.

# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Definição:
  - 3 missionários e 3 canibais estão em uma margem de um rio. Há uma canoa para apenas 2 pessoas. Deve-se levar todos para o outro lado do rio, cuidando para que o número de canibais nunca ultrapasse o número de missionários.

# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

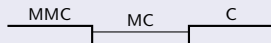
- Definição:
  - 3 missionários e 3 canibais estão em uma margem de um rio. Há uma canoa para apenas 2 pessoas. Deve-se levar todos para o outro lado do rio, cuidando para que o número de canibais nunca ultrapasse o número de missionários.
- Estados:



# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

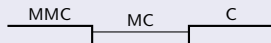
- Definição:
  - 3 missionários e 3 canibais estão em uma margem de um rio. Há uma canoa para apenas 2 pessoas. Deve-se levar todos para o outro lado do rio, cuidando para que o número de canibais nunca ultrapasse o número de missionários.
- Estados:
  - Qualquer combinação de canibais e missionários na margem esquerda, barco e margem direita



# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Definição:
  - 3 missionários e 3 canibais estão em uma margem de um rio. Há uma canoa para apenas 2 pessoas. Deve-se levar todos para o outro lado do rio, cuidando para que o número de canibais nunca ultrapasse o número de missionários.
- Estados:
  - Qualquer combinação de canibais e missionários na margem esquerda, barco e margem direita
  - Não há estado em que o número de canibais ultrapasse o de missionários



# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Objetivo:

# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Objetivo:

- Todos no lado direito do rio



# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Objetivo:

- Todos no lado direito do rio



- Formulação:

# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Objetivo:

- Todos no lado direito do rio



- Formulação:

- Estado inicial:



# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

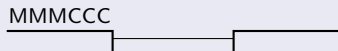
- Objetivo:

- Todos no lado direito do rio



- Formulação:

- Estado inicial:
- Ações possíveis:



# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

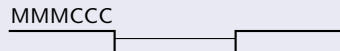
- Objetivo:

- Todos no lado direito do rio



- Formulação:

- Estado inicial:



- Ações possíveis:

- Transportar ou uma ou duas pessoas (canibais, missionários ou ambos) de um lado a outro do rio



# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

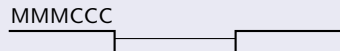
- Objetivo:

- Todos no lado direito do rio



- Formulação:

- Estado inicial:



- Ações possíveis:

- Transportar ou uma ou duas pessoas (canibais, missionários ou ambos) de um lado a outro do rio
- Ou seja, embarque + desembarque de uma ou duas pessoas

# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Objetivo:

- Todos no lado direito do rio



- Formulação:

- Estado inicial:



- Ações possíveis:

- Transportar ou uma ou duas pessoas (canibais, missionários ou ambos) de um lado a outro do rio
- Ou seja, embarque + desembarque de uma ou duas pessoas
- Teste: Compara se o estado atual é o estado-objetivo

# Busca como Solução de Problemas

## Exemplo: Missionários e Canibais

- Objetivo:

- Todos no lado direito do rio



- Formulação:

- Estado inicial:



- Ações possíveis:

- Transportar ou uma ou duas pessoas (canibais, missionários ou ambos) de um lado a outro do rio
- Ou seja, embarque + desembarque de uma ou duas pessoas
- Teste: Compara se o estado atual é o estado-objetivo
- Avaliação do custo: Número de viagens da canoa

# Busca por Soluções

- Soluciona o problema via uma busca

# Busca por Soluções

- Soluciona o problema via uma busca

## Procedimento Básico

# Busca por Soluções

- Soluciona o problema via uma busca

## Procedimento Básico

- Mapeiam-se os estados do problema a um grafo (redução do problema)

# Busca por Soluções

- Soluciona o problema via uma busca

## Procedimento Básico

- Mapeiam-se os estados do problema a um grafo (redução do problema)
- Busca-se um caminho que leve do estado inicial a um final (solução do problema)

# Busca por Soluções

- Soluciona o problema via uma busca

## Procedimento Básico

- Mapeiam-se os estados do problema a um grafo (redução do problema)
- Busca-se um caminho que leve do estado inicial a um final (solução do problema)
- O caminho representa um plano de ação para solução do problema



# Busca por Soluções

## Tipos de Busca

# Busca por Soluções

## Tipos de Busca

- Não informadas

# Busca por Soluções

## Tipos de Busca

- Não informadas
  - Mesmas características de soluções não-informadas

# Busca por Soluções

## Tipos de Busca

- Não informadas
  - Mesmas características de soluções não-informadas
  - Olham todos os nós em uma ordem específica

# Busca por Soluções

## Tipos de Busca

- Não informadas
  - Mesmas características de soluções não-informadas
  - Olham todos os nós em uma ordem específica
  - São as buscas “clássicas”: largura, profundidade etc

# Busca por Soluções

## Tipos de Busca

- Não informadas
  - Mesmas características de soluções não-informadas
  - Olham todos os nós em uma ordem específica
  - São as buscas “clássicas”: largura, profundidade etc
- Informadas

# Busca por Soluções

## Tipos de Busca

- Não informadas
  - Mesmas características de soluções não-informadas
  - Olham todos os nós em uma ordem específica
  - São as buscas “clássicas”: largura, profundidade etc
- Informadas
  - Mesmas características de soluções informadas

# Busca por Soluções

## Tipos de Busca

- Não informadas
  - Mesmas características de soluções não-informadas
  - Olham todos os nós em uma ordem específica
  - São as buscas “clássicas”: largura, profundidade etc
- Informadas
  - Mesmas características de soluções informadas
  - Usam informação específica do problema



# Busca por Soluções

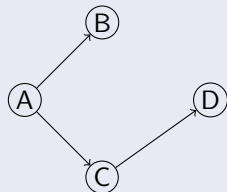
## Tipos de Busca

- Não informadas
  - Mesmas características de soluções não-informadas
  - Olham todos os nós em uma ordem específica
  - São as buscas “clássicas”: largura, profundidade etc
- Informadas
  - Mesmas características de soluções informadas
  - Usam informação específica do problema
  - São buscas com heurísticas: melhor primeiro,  $A^*$ , hill climbing etc

## Buscas Não Informadas

# Buscas Não Informadas

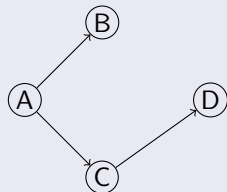
## Busca de Custo Uniforme



# Buscas Não Informadas

## Busca de Custo Uniforme

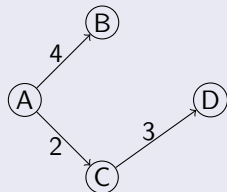
- Extensão à busca em largura



# Buscas Não Informadas

## Busca de Custo Uniforme

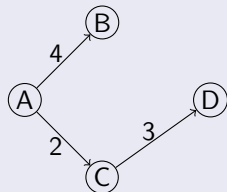
- Extensão à busca em largura
  - Os pesos (custos) nas arestas não são iguais



# Buscas Não Informadas

## Busca de Custo Uniforme

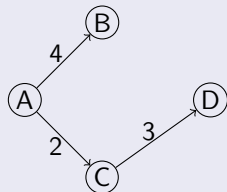
- Extensão à busca em largura
  - Os pesos (custos) nas arestas não são iguais
  - Sempre expande o nó  $n$  cujo **caminho até a raiz**  $g(n)$  é o menor (menor custo de caminho)



# Buscas Não Informadas

## Busca de Custo Uniforme

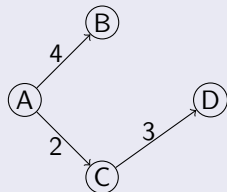
- Extensão à busca em largura
  - Os pesos (custos) nas arestas não são iguais
  - Sempre expande o nó  $n$  cujo **caminho até a raiz**  $g(n)$  é o menor (menor custo de caminho)
  - Não necessariamente faz a busca em largura



# Buscas Não Informadas

## Busca de Custo Uniforme

- Extensão à busca em largura
  - Os pesos (custos) nas arestas não são iguais
  - Sempre expande o nó  $n$  cujo **caminho até a raiz**  $g(n)$  é o menor (menor custo de caminho)
  - Não necessariamente faz a busca em largura
- Os custos devem ser  $> 0$

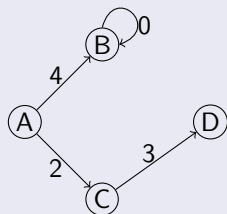




# Buscas Não Informadas

## Busca de Custo Uniforme

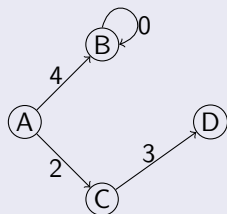
- Extensão à busca em largura
  - Os pesos (custos) nas arestas não são iguais
  - Sempre expande o nó  $n$  cujo **caminho até a raiz**  $g(n)$  é o menor (menor custo de caminho)
  - Não necessariamente faz a busca em largura
- Os custos devem ser  $> 0$ 
  - Do contrário, pode cair em um laço infinito, caso haja aresta de custo 0 levando ao próprio nó



# Buscas Não Informadas

## Busca de Custo Uniforme

- Extensão à busca em largura
  - Os pesos (custos) nas arestas não são iguais
  - Sempre expande o nó  $n$  cujo **caminho até a raiz**  $g(n)$  é o menor (menor custo de caminho)
  - Não necessariamente faz a busca em largura
- Os custos devem ser  $> 0$ 
  - Do contrário, pode cair em um laço infinito, caso haja aresta de custo 0 levando ao próprio nó
  - O custo do caminho deve sempre aumentar



# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

```
Q ← Nó inicial (custo = 0);  
enquanto Q não estiver vazia faça  
    C ← Retire de Q o caminho com  
    menor custo;  
    se cabeça(C) = objetivo então  
        └ Retorne C  
    para cada filho f de cabeça(C) faça  
        └ Adicione [f,C] a Q  
retorna falha
```

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

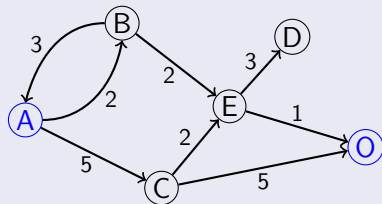
**Q**  $\leftarrow$  Nó inicial (custo = 0);  
**enquanto** *Q não estiver vazia* **faça**  
    C  $\leftarrow$  Retire de Q o caminho com  
        menor custo;  
    **se** *cabeça(C) = objetivo* **então**  
        └ Retorne C  
    **para cada** *filho f de cabeça(C)* **faça**  
        └ Adicione [f,C] a Q  
**retorna** *falha*

←  
Lista de caminhos  
(com seu custo)  
a partir da origem

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

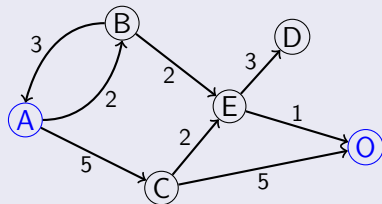
Q:

C:

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  **Nó inicial** (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q: 

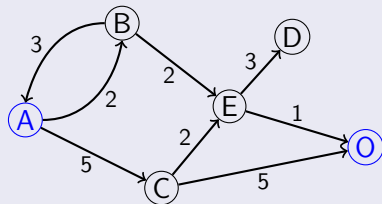
[A] (0)
---------

C:

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto** *Q não estiver vazia* **faça**  
     $C \leftarrow$  Retire de Q o caminho com menor custo;  
    **se**  $cabeça(C) = objetivo$  **então**  
        └ Retorne C  
    **para cada** filho  $f$  de  $cabeça(C)$  **faça**  
        └ Adicione  $[f, C]$  a Q  
**retorna** *falha*



**Caminho (custo)**

Q: 

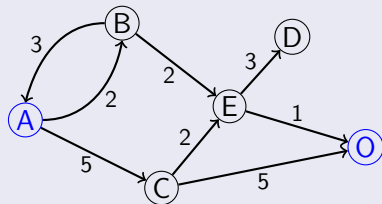
[A] (0)
---------

C:

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     **$C \leftarrow$  Retire de  $Q$  o caminho com menor custo;**  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q:

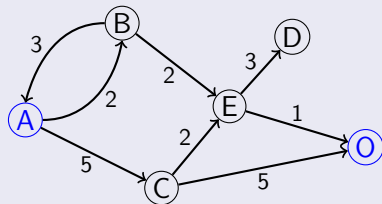
C:



# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



Caminho (custo)

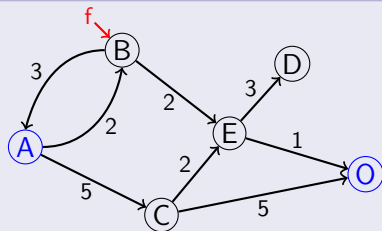
Q:

C:

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



Caminho (custo)

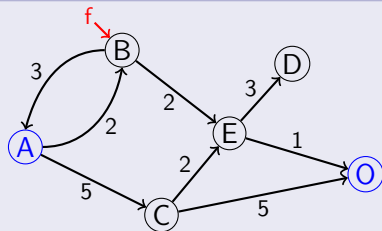
Q:

C:

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

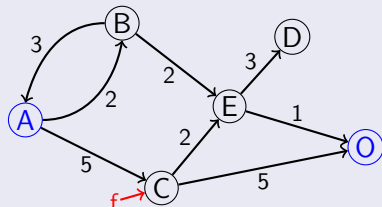
Q: [B,A] (2)

C: [A] (0)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

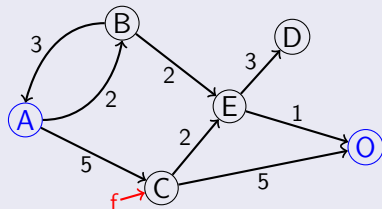
Q: [B,A] (2)

C: [A] (0)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q:

[B,A] (2)
[C,A] (5)

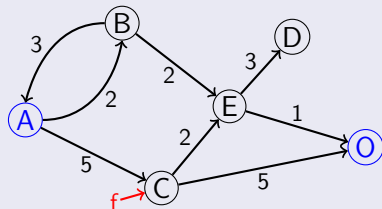
C:

[A] (0)
---------

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto** *Q não estiver vazia* **faça**  
     $C \leftarrow$  Retire de Q o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne C  
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a Q  
**retorna** *falha*



**Caminho (custo)**

Q:

[B,A] (2)
[C,A] (5)

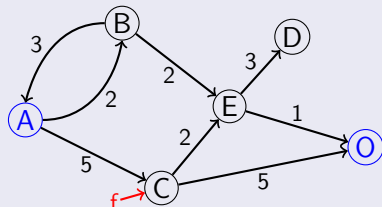
C:

[A] (0)
---------

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com  
    menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

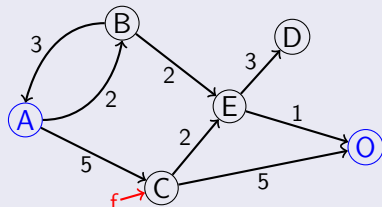
Q: [C,A] (5)

C: [B,A] (2)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q: [C,A] (5)

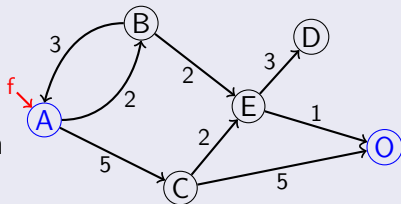
C: [B,A] (2)



# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

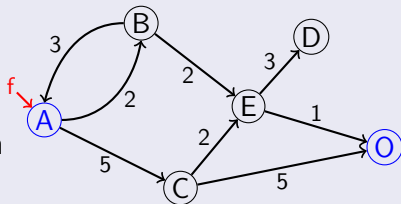
Q: [C,A] (5)

C: [B,A] (2)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q:

[C,A] (5)
[A,B,A] (5)

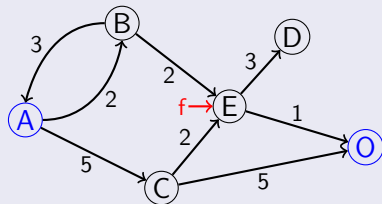
C: 

[B,A] (2)
-----------

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q:

[C,A] (5)
[A,B,A] (5)

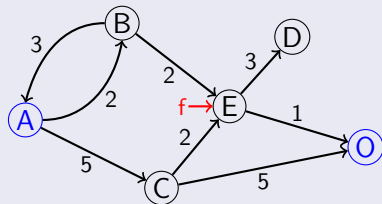
C: 

[B,A] (2)
-----------

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

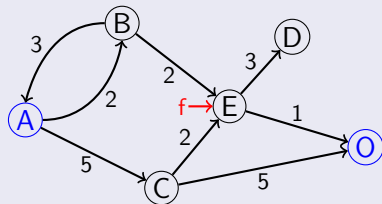
Q:	[C,A] (5)
	[A,B,A] (5)
	[E,B,A] (4)

C: [B,A] (2)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto** *Q não estiver vazia* **faça**  
     $C \leftarrow$  Retire de Q o caminho com menor custo;  
    **se**  $cabeça(C) = objetivo$  **então**  
        └ Retorne C  
    **para cada** filho  $f$  de  $cabeça(C)$  **faça**  
        └ Adicione  $[f, C]$  a Q  
**retorna** *falha*



**Caminho (custo)**

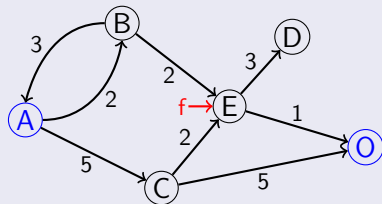
Q:	[C,A] (5)
	[A,B,A] (5)
	[E,B,A] (4)

C: [B,A] (2)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com  
    menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q:

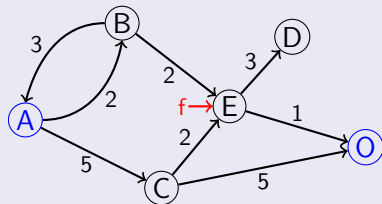
[C,A] (5)
[A,B,A] (5)

C: [E,B,A] (4)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q:

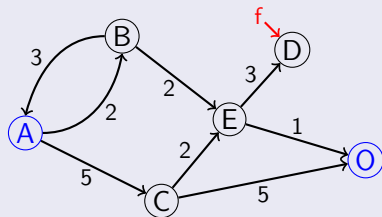
[C,A] (5)
[A,B,A] (5)

C: [E,B,A] (4)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada**  $f$  **filho**  $f$  **de**  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

Q:

[C,A] (5)
[A,B,A] (5)

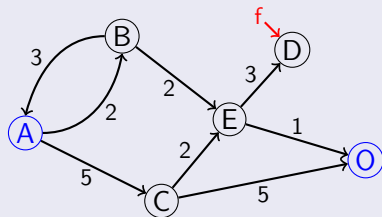
C: [E,B,A] (4)



# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** falha



**Caminho (custo)**

Q:

[C,A] (5)
[A,B,A] (5)
[D,E,B,A] (7)

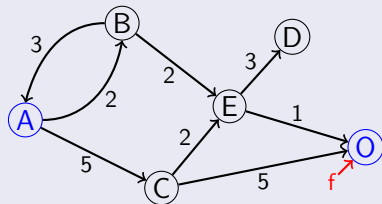
C:

[E,B,A] (4)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



**Caminho (custo)**

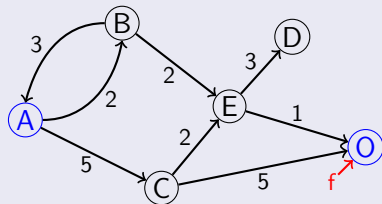
Q:	[C,A] (5)
	[A,B,A] (5)
	[D,E,B,A] (7)

C: [E,B,A] (4)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

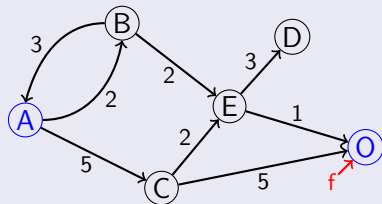
Q:	[C,A] (5)
	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)

C: [E,B,A] (4)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto** *Q não estiver vazia* **faça**  
     $C \leftarrow$  Retire de Q o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne C  
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a Q  
**retorna** *falha*



### Caminho (custo)

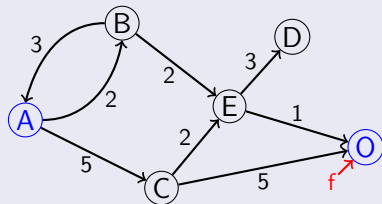
Q:	[C,A] (5)
	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)

C: [E,B,A] (4)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com  
    menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

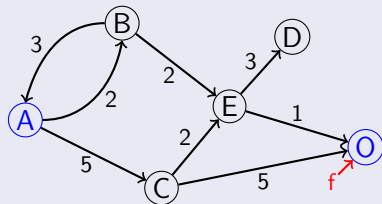
Q:	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)

C: [C,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

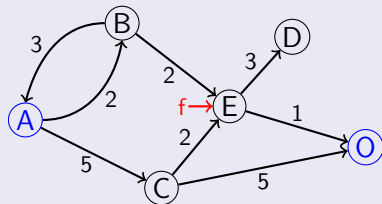
Q:	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)

C: [C,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

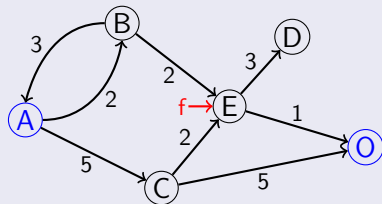
Q:	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)

C: [C,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

Q:	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)

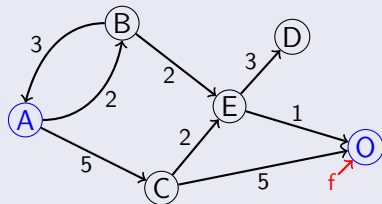
C: [C,A] (5)



# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

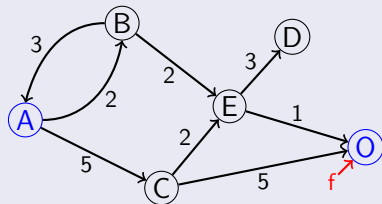
Q:	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)

C: [C,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

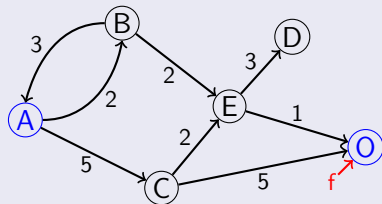
Q:	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)

C: [C,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto** *Q não estiver vazia* **faça**  
     $C \leftarrow$  Retire de Q o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne C  
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a Q  
**retorna** *falha*



### Caminho (custo)

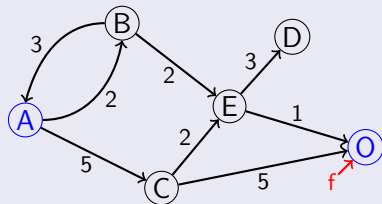
Q:	[A,B,A] (5)
	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)

C: [C,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com  
    menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

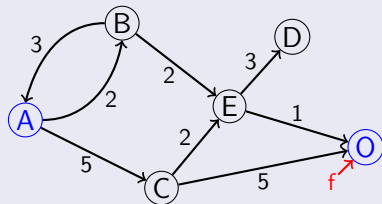
Q:	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)

C: [A,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

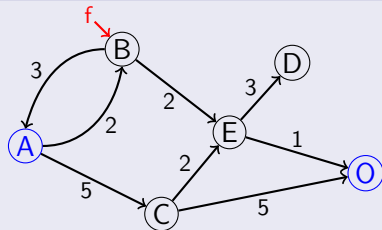
Q:	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)

C: [A,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada**  $f$  **filho**  $f$  **de**  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

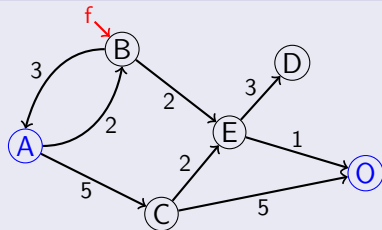
Q:	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)

C: [A,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

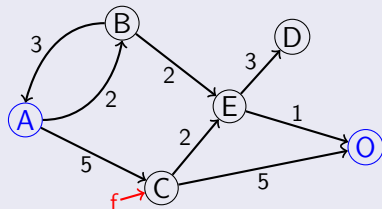
Q:	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)
	[B,A,B,A] (7)

C: [A,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** *filho  $f$  de  $\text{cabeça}(C)$*  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

Q:	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)
	[B,A,B,A] (7)

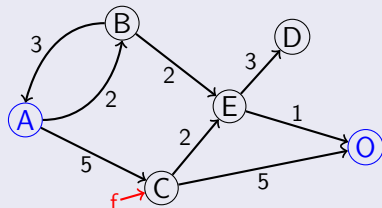
C: [A,B,A] (5)



# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

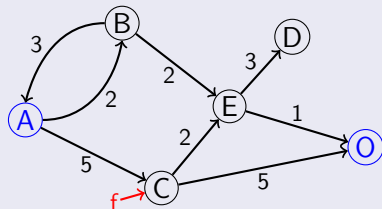
Q:	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)
	[B,A,B,A] (7)
	[C,A,B,A] (10)

C: [A,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto** *Q não estiver vazia* **faça**  
     $C \leftarrow$  Retire de Q o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne C  
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a Q  
**retorna** *falha*



### Caminho (custo)

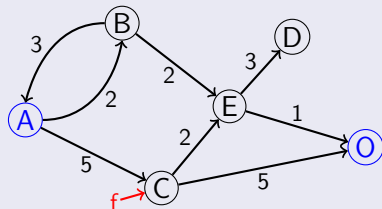
Q:	[D,E,B,A] (7)
	[O,E,B,A] (5)
	[E,C,A] (7)
	[O,C,A] (10)
	[B,A,B,A] (7)
	[C,A,B,A] (10)

C: [A,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com  
        menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

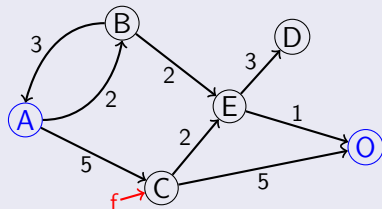
Q:	[D,E,B,A] (7)
	[E,C,A] (7)
	[O,C,A] (10)
	[B,A,B,A] (7)
	[C,A,B,A] (10)

C: [O,E,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ Retorne  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

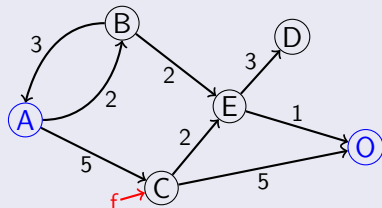
Q:	[D,E,B,A] (7)
	[E,C,A] (7)
	[O,C,A] (10)
	[B,A,B,A] (7)
	[C,A,B,A] (10)

C: [O,E,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme – Algoritmo

$Q \leftarrow$  Nó inicial (custo = 0);  
**enquanto**  $Q$  não estiver vazia **faça**  
     $C \leftarrow$  Retire de  $Q$  o caminho com menor custo;  
    **se**  $\text{cabeça}(C) = \text{objetivo}$  **então**  
        └ **Retorne**  $C$   
    **para cada** filho  $f$  de  $\text{cabeça}(C)$  **faça**  
        └ Adicione  $[f, C]$  a  $Q$   
**retorna** *falha*



### Caminho (custo)

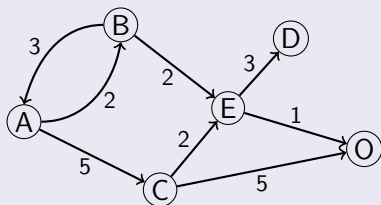
Q:	[D,E,B,A] (7)
	[E,C,A] (7)
	[O,C,A] (10)
	[B,A,B,A] (7)
	[C,A,B,A] (10)

C: [O,E,B,A] (5)

# Buscas Não Informadas

## Busca de Custo Uniforme

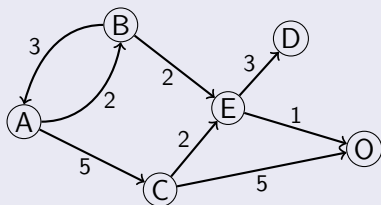
- BCU dá o melhor caminho entre 2 vértices



# Buscas Não Informadas

## Busca de Custo Uniforme

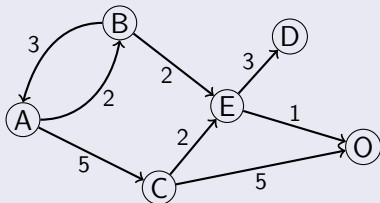
- BCU dá o melhor caminho entre 2 vértices
- E mesmo havendo laços, o algoritmo consegue prosseguir



# Buscas Não Informadas

## Busca de Custo Uniforme

- BCU dá o melhor caminho entre 2 vértices
- E mesmo havendo laços, o algoritmo consegue prosseguir
- Uma hora o laço cria um caminho grande demais para ser considerado

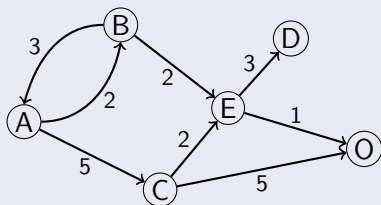




# Buscas Não Informadas

## Busca de Custo Uniforme

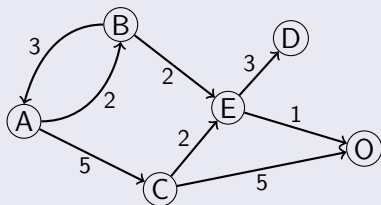
- BCU dá o melhor caminho entre 2 vértices
- E mesmo havendo laços, o algoritmo consegue prosseguir
- Uma hora o laço cria um caminho grande demais para ser considerado
- Mas isso, claro, somente se os pesos das arestas forem  $> 0$



# Buscas Não Informadas

## Busca de Custo Uniforme

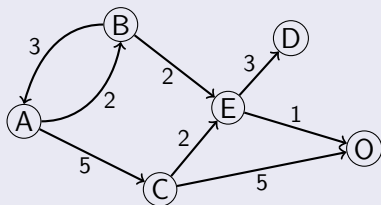
- Ainda assim, seguir o laço gasta processamento



# Buscas Não Informadas

## Busca de Custo Uniforme

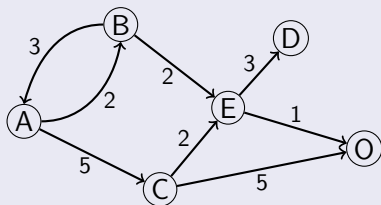
- Ainda assim, seguir o laço gasta processamento
- Uma alternativa é marcar os nós já visitados



# Buscas Não Informadas

## Busca de Custo Uniforme

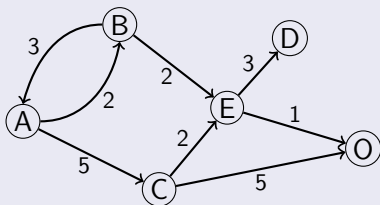
- Ainda assim, seguir o laço gasta processamento
- Uma alternativa é marcar os nós já visitados
  - E não mais considerá-los em futuros caminhos



# Buscas Não Informadas

## Busca de Custo Uniforme

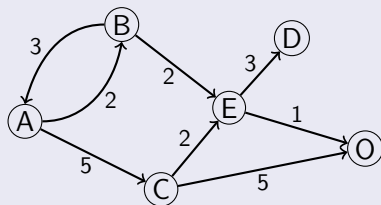
- Ainda assim, seguir o laço gasta processamento
- Uma alternativa é marcar os nós já visitados
  - E não mais considerá-los em futuros caminhos
- Mas aí o resultado não é mais ótimo



# Buscas Não Informadas

## Busca de Custo Uniforme

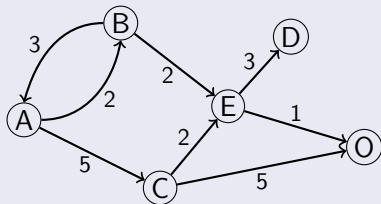
- Ainda assim, seguir o laço gasta processamento
- Uma alternativa é marcar os nós já visitados
  - E não mais considerá-los em futuros caminhos
- Mas aí o resultado não é mais ótimo
  - Isso porque podemos chegar a um nó já visitado por um caminho menor, que acabará sendo ignorado



# Buscas Não Informadas

## Busca de Custo Uniforme

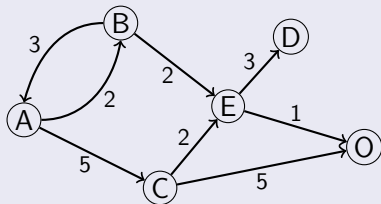
- Para remediar isso, podemos, sempre que chegarmos a um nó já visitado, ver se o caminho até ele é menor do que aquele que está em Q



# Buscas Não Informadas

## Busca de Custo Uniforme

- Para remediar isso, podemos, sempre que chegarmos a um nó já visitado, ver se o caminho até ele é menor do que aquele que está em Q
- Se for menor, substituímos o caminho em Q por esse menor

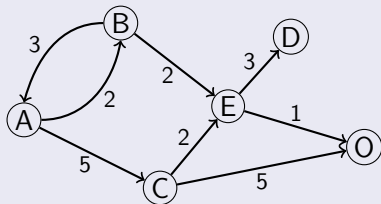




# Buscas Não Informadas

## Busca de Custo Uniforme

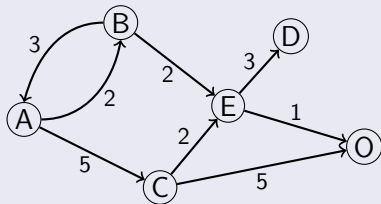
- Para remediar isso, podemos, sempre que chegarmos a um nó já visitado, ver se o caminho até ele é menor do que aquele que está em Q
- Se for menor, substituímos o caminho em Q por esse menor
- Assim consideraremos sempre o caminho mais curto



# Buscas Não Informadas

## Busca de Custo Uniforme

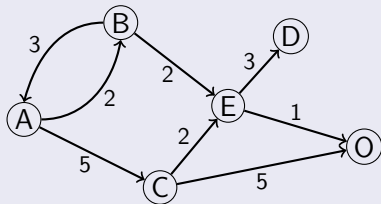
- É uma busca guiada pelo custo dos caminhos, não sua profundidade (número de arestas)



# Buscas Não Informadas

## Busca de Custo Uniforme

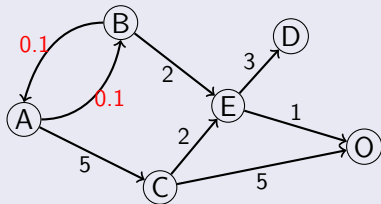
- É uma busca guiada pelo custo dos caminhos, não sua profundidade (número de arestas)
- Pode preferir explorar longos caminhos de passos pequenos antes de explorar caminhos maiores e potencialmente mais úteis



# Buscas Não Informadas

## Busca de Custo Uniforme

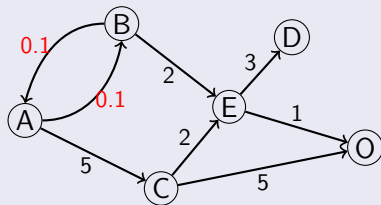
- É uma busca guiada pelo custo dos caminhos, não sua profundidade (número de arestas)
- Pode preferir explorar longos caminhos de passos pequenos antes de explorar caminhos maiores e potencialmente mais úteis



# Buscas Não Informadas

## Busca de Custo Uniforme

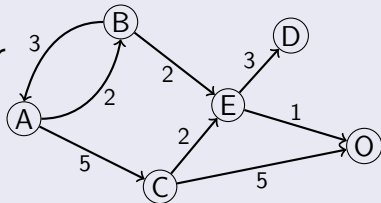
- É uma busca guiada pelo custo dos caminhos, não sua profundidade (número de arestas)
- Pode preferir explorar longos caminhos de passos pequenos antes de explorar caminhos maiores e potencialmente mais úteis
- Não muito adequada se a profundidade do caminho importar



# Buscas Não Informadas

## Busca de Custo Uniforme

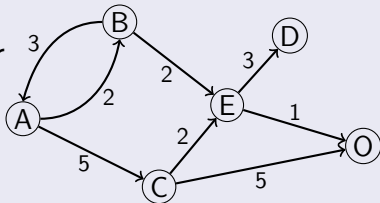
- Quando todos os custos são iguais, a busca é similar à em largura



# Buscas Não Informadas

## Busca de Custo Uniforme

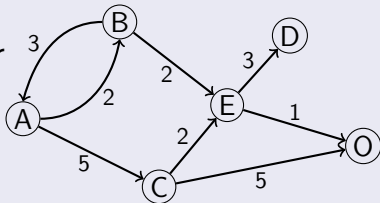
- Quando todos os custos são iguais, a busca é similar à em largura
- Exceto que ela não para assim que encontra o objetivo, como na busca em largura



# Buscas Não Informadas

## Busca de Custo Uniforme

- Quando todos os custos são iguais, a busca é similar à em largura
- Exceto que ela não para assim que encontra o objetivo, como na busca em largura
- Em vez disso, examina todos os nós na profundidade do objetivo para ver se algum tem custo menor

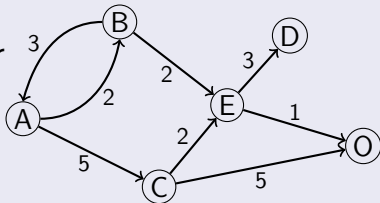




# Buscas Não Informadas

## Busca de Custo Uniforme

- Quando todos os custos são iguais, a busca é similar à em largura
- Exceto que ela não para assim que encontra o objetivo, como na busca em largura
- Em vez disso, examina todos os nós na profundidade do objetivo para ver se algum tem custo menor
- Gasta mais tempo com um procedimento inútil (pelos pesos serem iguais)



# Referências

- Russell, S.; Norvig P. (2010): Artificial Intelligence: A Modern Approach. Prentice Hall. 3a ed.
- [https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-410-principles-of-autonomy-and-decision-making-fall-2010/lecture-notes/MIT16\\_410F10\\_lec14.pdf](https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-410-principles-of-autonomy-and-decision-making-fall-2010/lecture-notes/MIT16_410F10_lec14.pdf)
- <http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-034Spring-2005/LectureNotes/index.htm>