Busca Informada & Não-Informada

Profa. Flavia Cristina Bernardini

Busca

- Imagine que você está em uma nova cidade e deseja encontrar um bar para beber um chopp
  - Se você tem um mapa, você pode descobrir como sair de onde você está para chegar ao bar
  - Se você não tem um mapa, você poderia caminhar sem rumo até encontrar um bar
  - Ou você poderia procurar sistematicamente por um bar

Busca

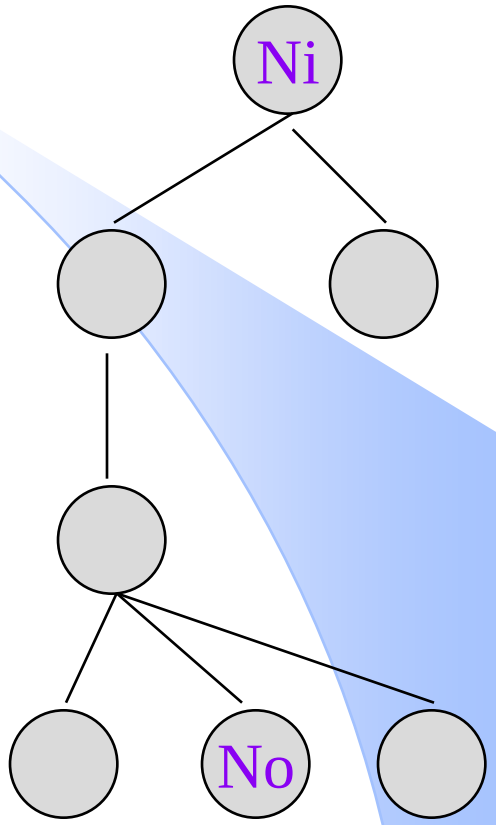
- Você poderia, por exemplo, utilizar o seguinte algoritmo de “encontrar bar”
  - 1. Procure um bar
  - 2. Caso não encontrou um bar, vá para um local não visitado por você e repita o passo 1
  - 3. Se você encontrou um bar, vá e beba um chop
  - 4. Após o décimo chopp, saia e caia na sarjeta

Busca

- Um sistema de IA pode resolver problemas da seguinte forma:
  - Ele sabe onde ele está (conjunto de informações inicial)
  - Ele sabe onde deseja ir (estado objetivo)
- Resolver problema em IA envolve busca pelo estado objetivo
- Simples sistemas de IA reduzem raciocínio a busca

Busca

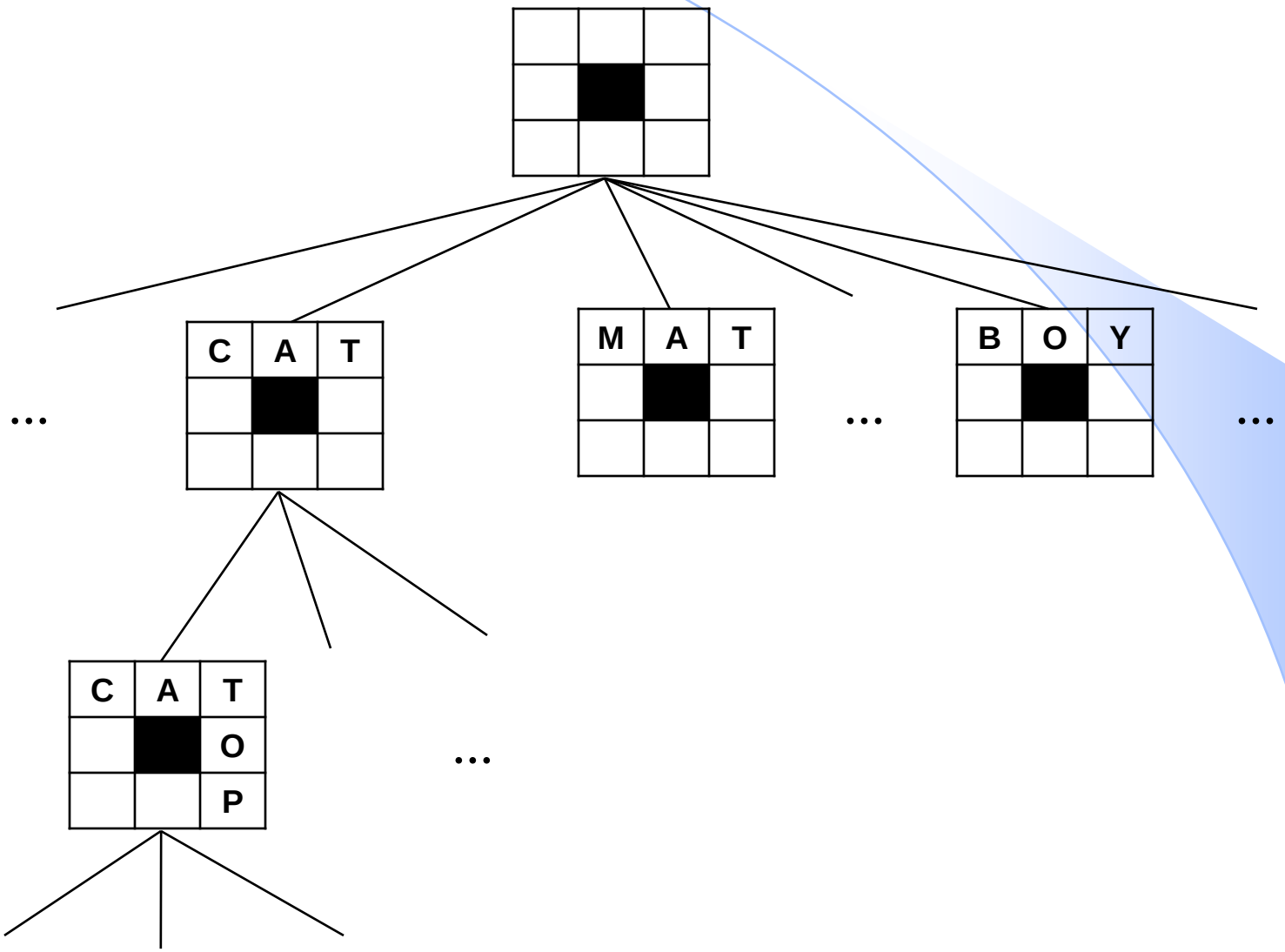
- Problemas de busca são freqüentemente descritos utilizando diagramas de árvores
  - Nó inicial = onde a busca começa
  - Nó objetivo = onde ela termina
- Objetivo: Encontrar um caminho que ligue o nó inicial a um nó objetivo



Busca

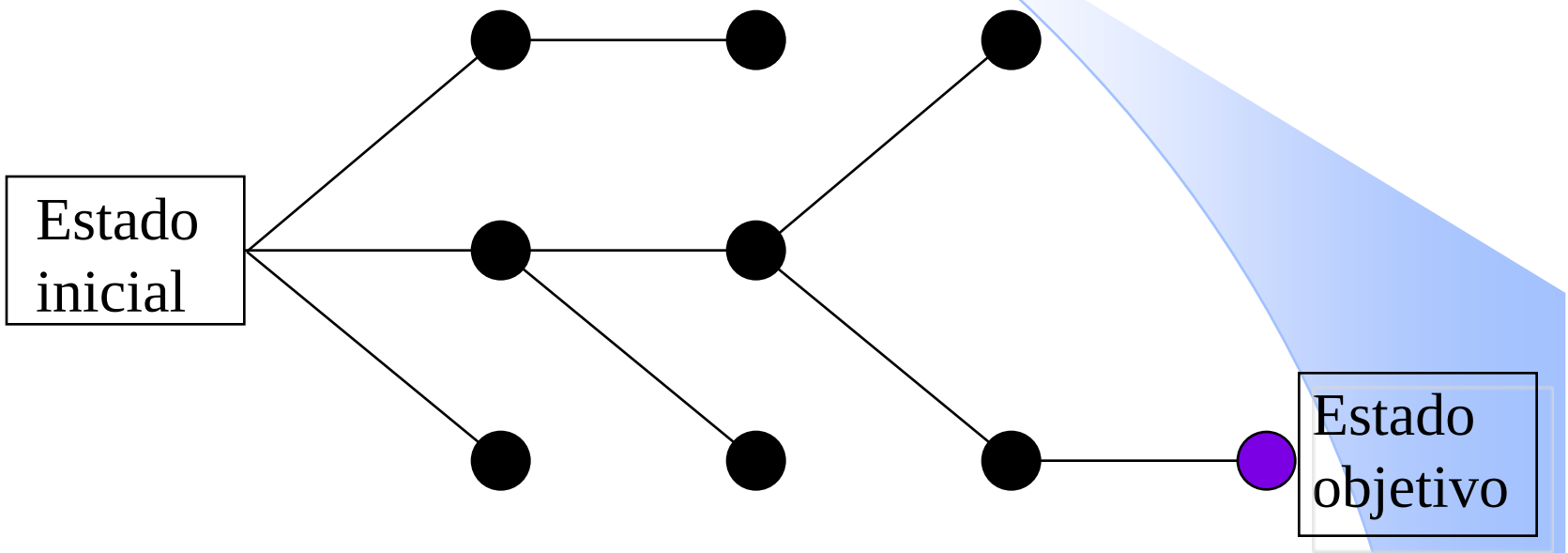
- Entrada:
  - Descrição dos nós inicial e objetivo
  - Procedimento que produz os sucessores de um nó arbitrário
- Saída:
  - Seqüência legal de nós iniciando com o nó inicial e terminando com o nó objetivo
  - Exemplo: palavras cruzadas

Busca



Uma árvore de busca

- Uma busca pode ser definida graficamente:

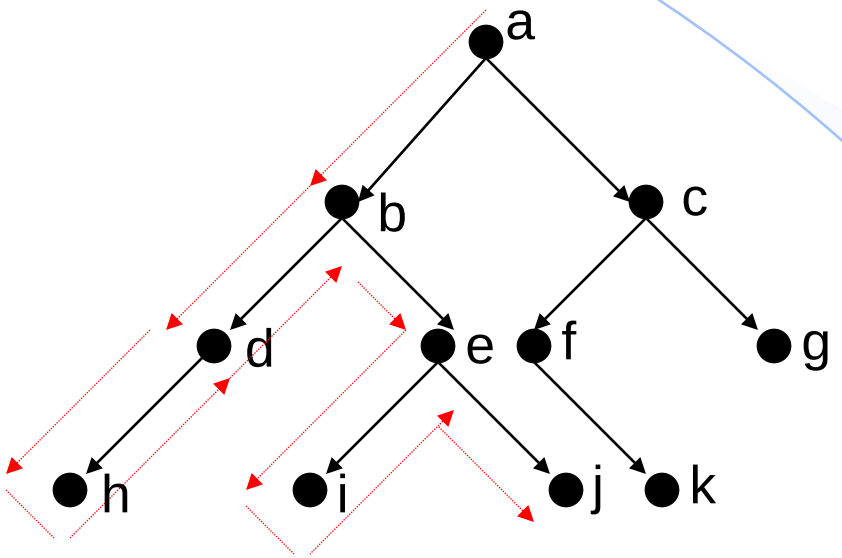


- O objetivo é atravessar a árvore partindo do estado inicial até o estado objetivo

Estratégias Básicas de Busca

- Busca em Profundidade
- Busca em Largura

Busca em Profundidade

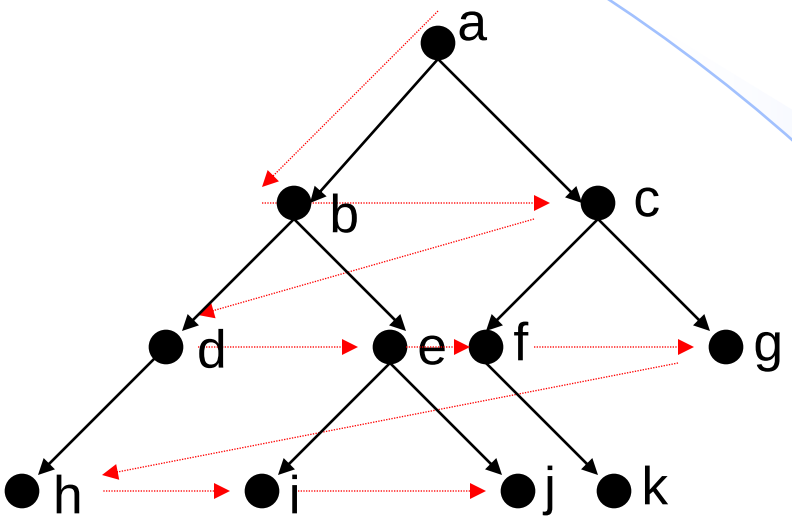


- Nó escolhido: Sempre o mais ***distante*** do nó inicial

Busca em Profundidade - Algoritmo

- Se N é um nó solução então Sol = [N], ou
- Se há um nó adjacente N1 a N, tal que existe um caminho Sol1 partindo de N1 até o nó meta, então Sol = [N | Sol1]

Busca em Largura



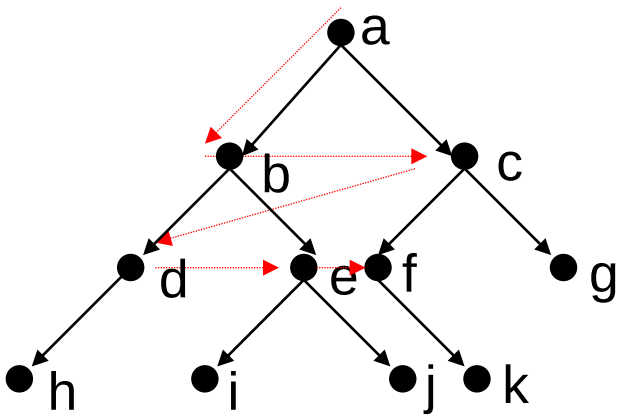
- Nó escolhido: Sempre o mais ***próximo*** do nó inicial

Busca em Largura - Algoritmo

Dado um conjunto de caminhos candidatos:

- Se o primeiro caminho contém o nó meta como primeiro elemento da fila, este é uma solução, ou
- Remova o primeiro caminho do conjunto de candidatos e gere um conjunto de todas as possíveis extensões de um nó deste caminho, adicione este conjunto de extensões ao final do conjunto de candidatos e execute busca em largura no conjunto restante

Execução do Algoritmo



1 - [[a]]

2 - [[b,a] , [c,a]]

2a - [[d,b,a] , [e,b,a]]

3 - [[c,a] , [d,b,a] , [e,b,a]]

4 - [[d,b,a] , [e,b,a] , [f,c,a] , [g,c,a]]

5 - [[e,b,a] , [f,c,a] , [g,c,a] , [h,d,b,a]]

6 - [[f,c,a] , [g,c,a] , [h,d,b,a] , [i,e,b,a] , [j,e,b,a]]

Problemas da busca

- Com o aumento da árvore de decisão e do número de possíveis caminhos, o tempo de busca aumenta
- Existem várias formas de reduzir o tempo de busca, alguns dos quais serão discutidos mais adiante

Possíveis situações

- Mais de um nó objetivo
- Mais de um nó inicial
- Nestas situações
  - Encontrar qualquer caminho de um nó inicial para um nó objetivo
  - Encontrar melhor caminho

Definições importantes

- Profundidade: número de ligações entre um dado nó e o nó inicial
- Largura: número de sucessores (filhos) de um nó

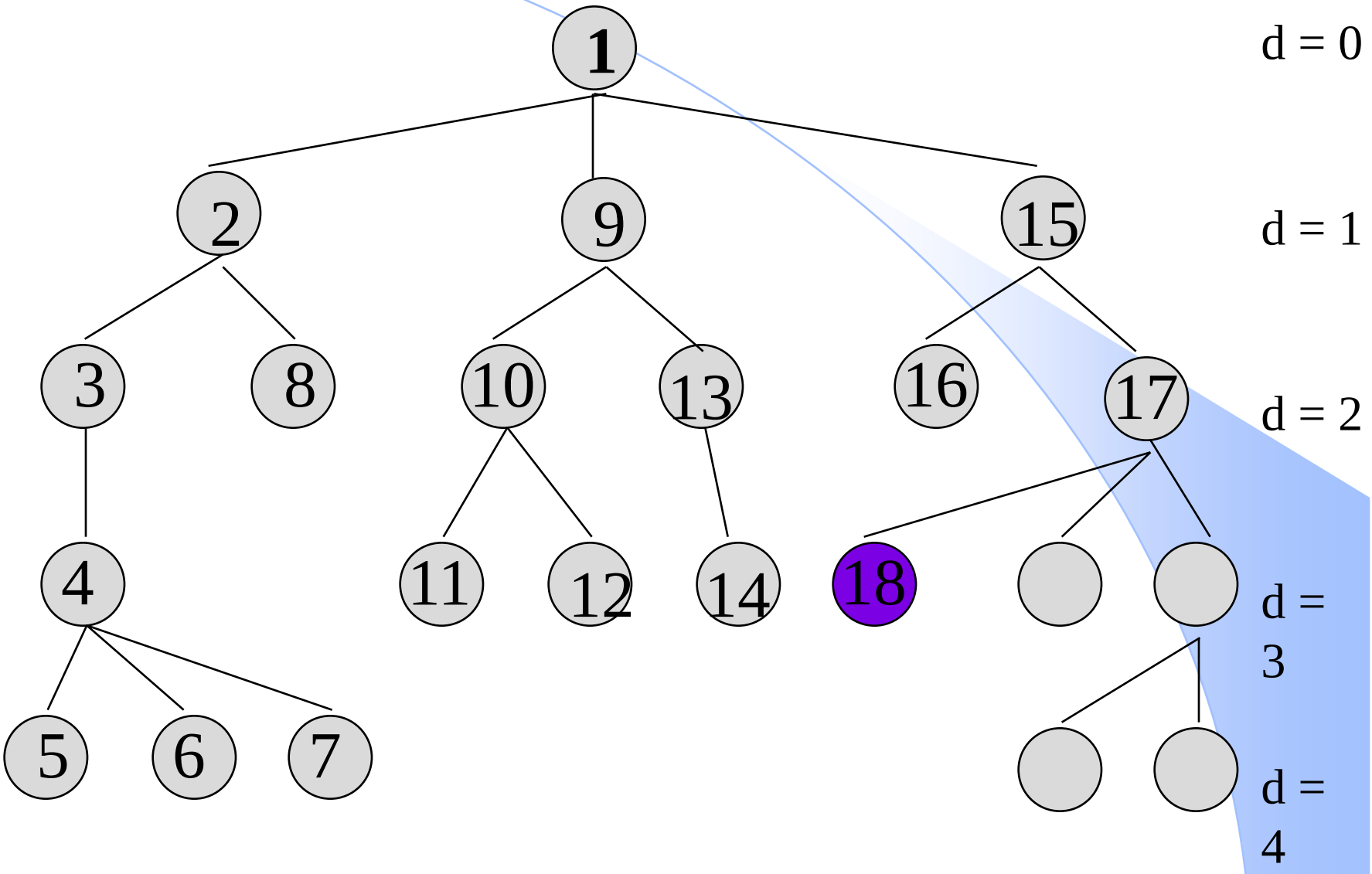
Algoritmos de Busca

- Existem vários algoritmos de busca diferentes, o que os distingue é a maneira como o nó *n* é escolhido no passo 2
- Métodos de busca
  - Busca cega: a escolha depende da posição do nó na árvore de busca
  - Busca heurística: A escolha utiliza informações específicas do domínio para ajudar na decisão

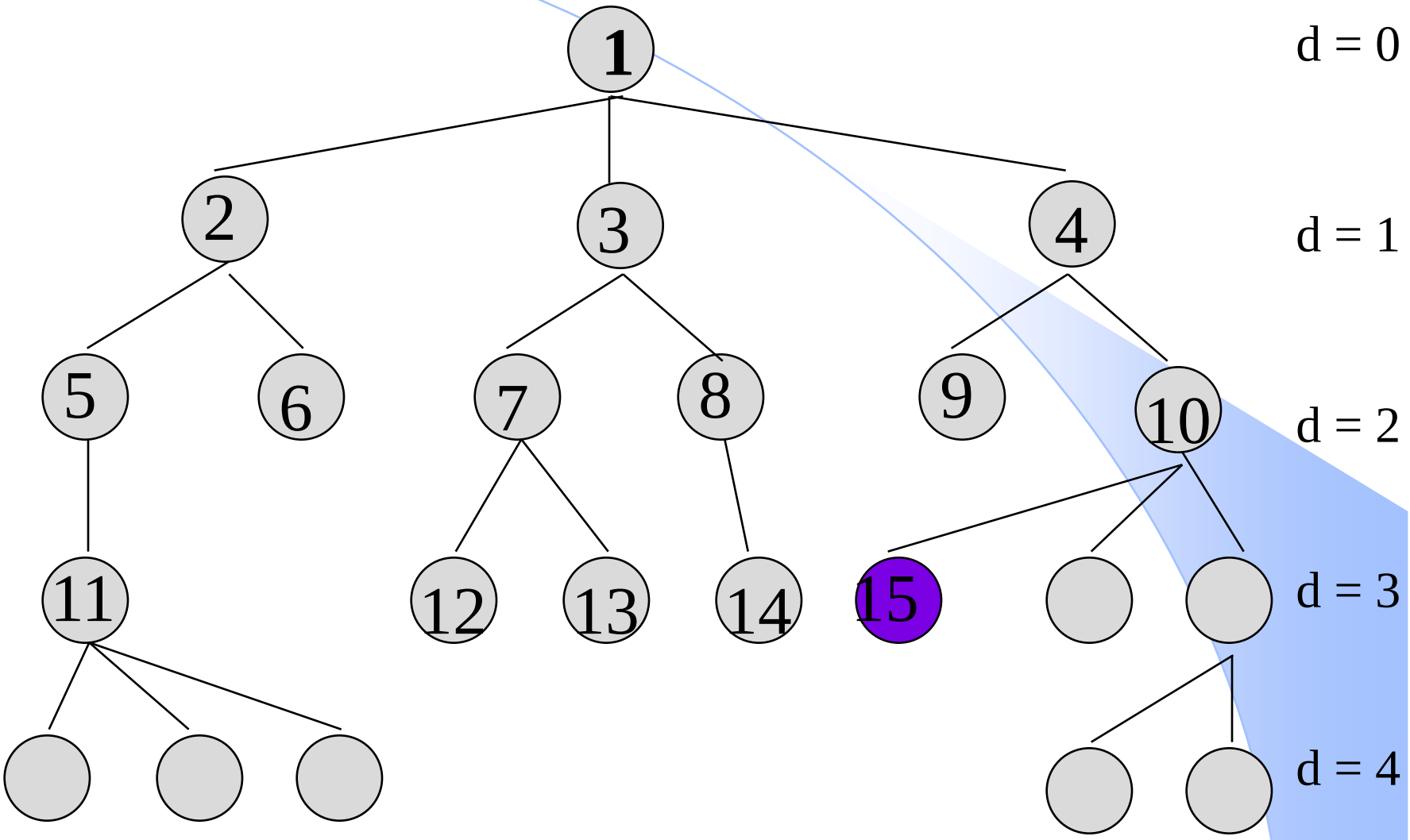
Técnicas de busca cega

- Busca em Profundidade (BP)
  - A árvore é examinada de cima para baixo
  - Aconselhável nos casos onde os caminhos improdutivos não são muito longos
- Busca em Largura (BL)
  - A árvore é examinada da esquerda para a direita
  - Aconselhável quando o número de ramos não é muito grande

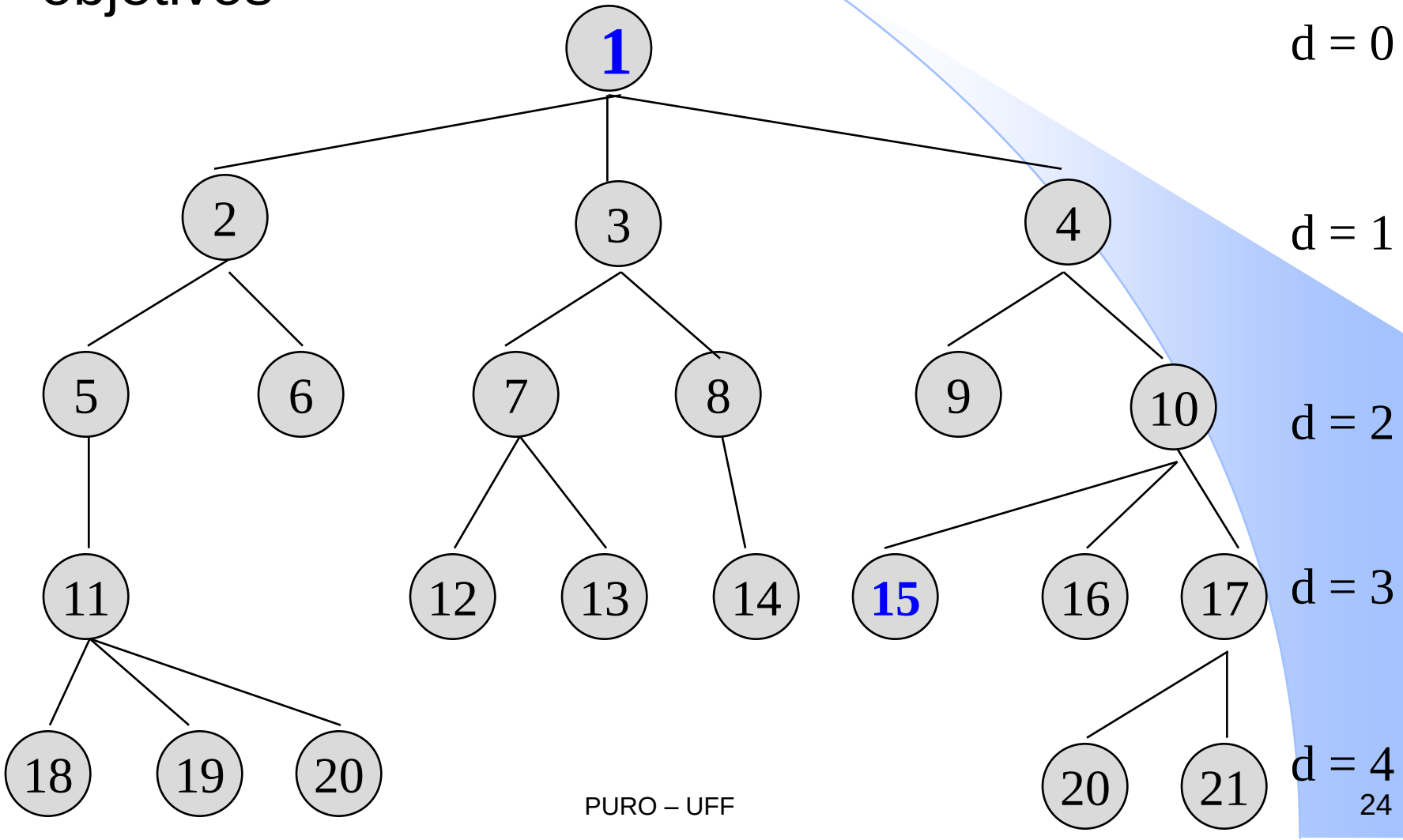
Busca em profundidade



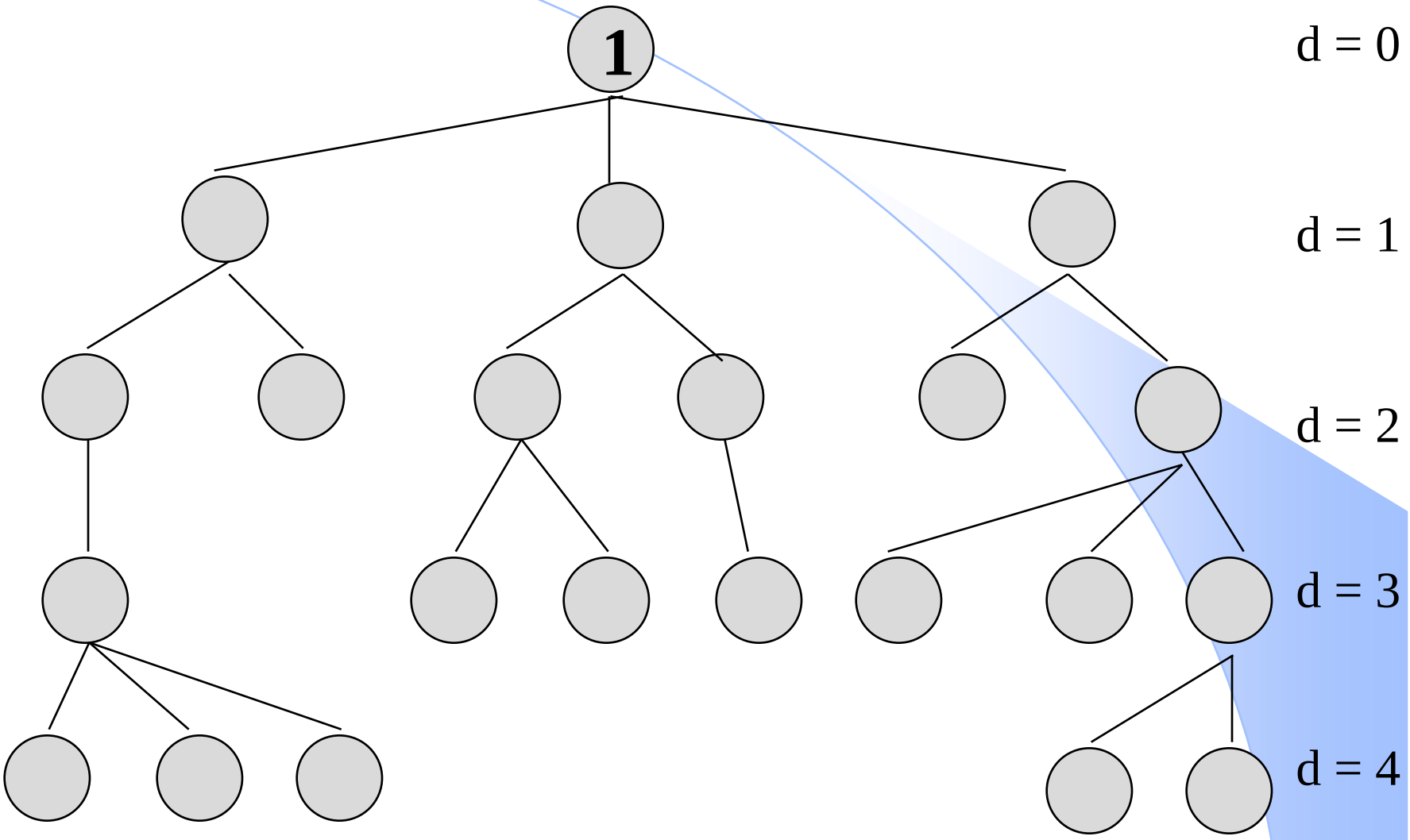
Busca em largura



Exemplo 1: Dada a árvore abaixo, utilizando BP, indique: a) Memória máxima e b) Número mínimo de passos necessários para atingir um dos nós objetivos



Exercício



Resposta ao exemplo 1

a) 1 L = {1}

2 L = {2_1, 3_1,4_1}

3 L = {5_{21},6_{21},3_1,4_1}

4 L = {11_{521}, 6_{21},3_1,4_1}

5 L = {18_{11521}, 19_{11521}, 20_{11521} 6_{21},3_1,4_1}

6 L = {19_{11521}, 20_{11521} 6_{21},3_1,4_1}

7 L = {20_{11521}, 6_{21},3_1,4_1}

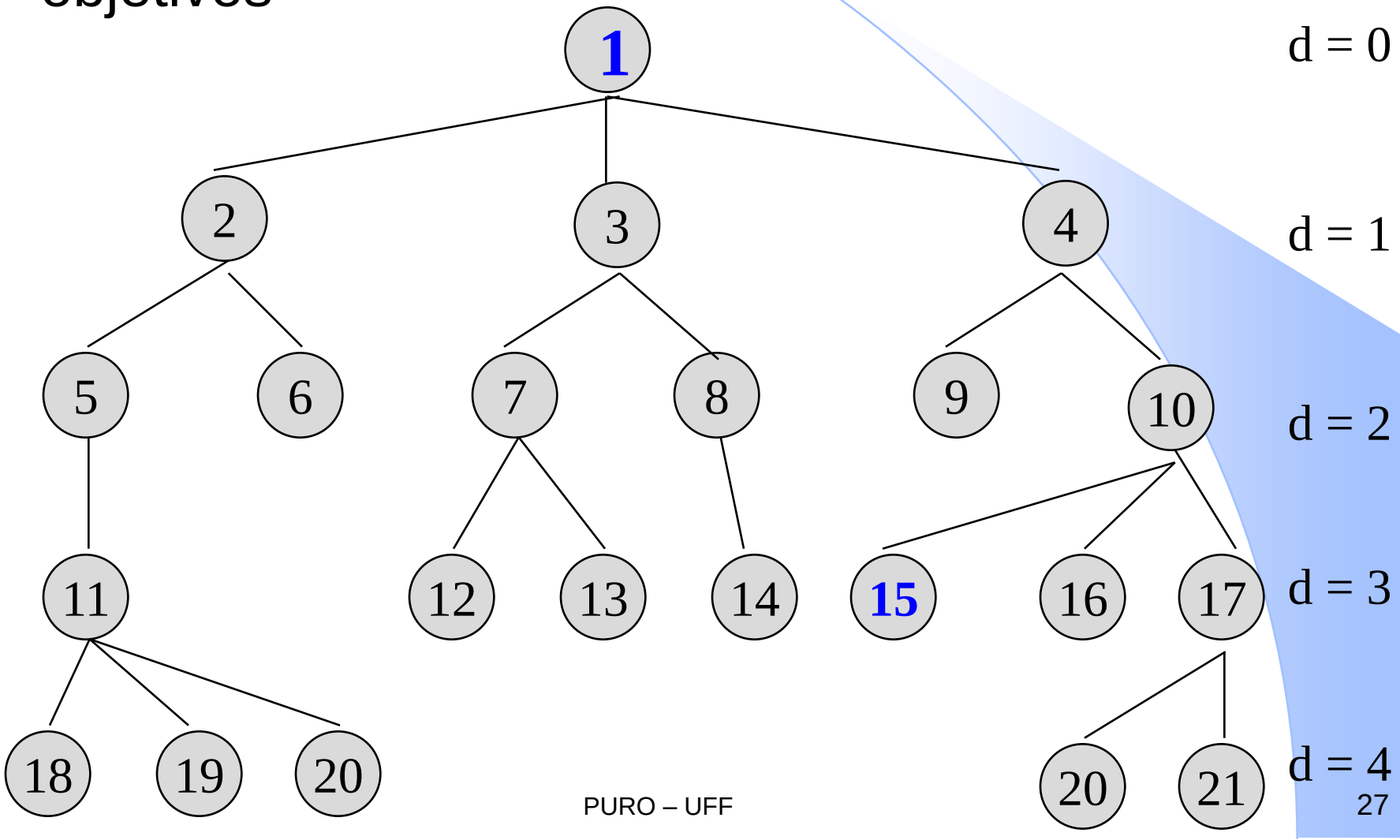
8 L = {6_{21},3_1,4_1}

9 L = {3_1,4_1}

....

18 L = {15_{1041},16_{1041},17_{1041}}

- Exemplo 2: Dada a árvore abaixo, utilizando BL, indique: a) Memória máxima e b) Número mínimo de passos necessários para atingir um dos nós objetivos



Observações

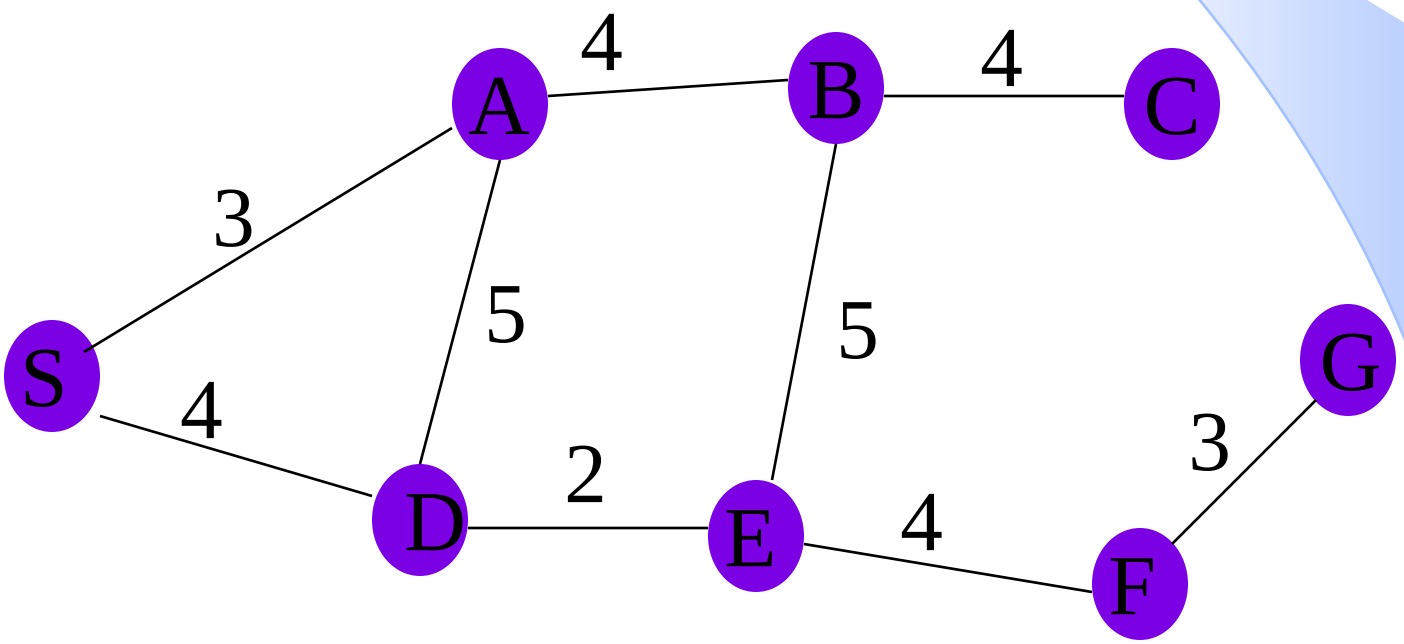
- BP e BL não precisam ser realizadas em uma ordem específica
- Memória utilizada pelas duas técnicas
  - BP: precisa armazenar todos os filhos não visitados de cada nó entre nó atual e nó inicial
  - BL: antes de examinar nó a uma profundidade d, é necessário examinar e armazenar todos os nós a uma profundidade d - 1
  - BP utiliza menos memória

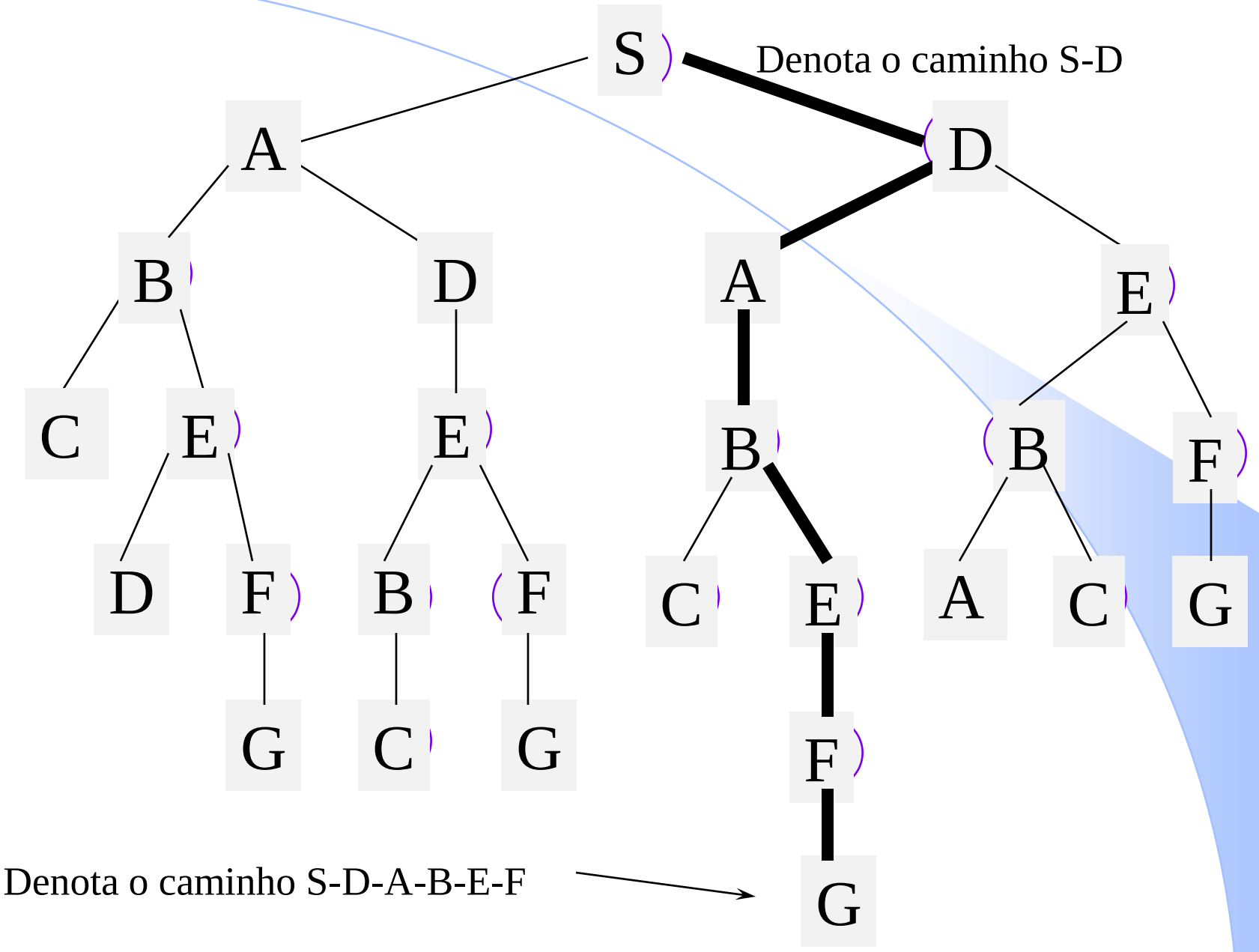
Observações

- Quanto ao tempo
  - BP é geralmente mais rápida
  - Métodos de busca cega não examinam a árvore de forma ótima, o que poderia minimizar o tempo gasto para resolver o problema

Busca em grafo – Exemplo

- Dado o grafo abaixo, encontrar a menor distância de S a G




S
Denota o caminho S-D
A
D
B
D
A
E
C
E
E
B
B
F
D
F
B
F
C
E
A
C
G
G
C
G
F
G
Denota o caminho S-D-A-B-E-F

Problemas da busca

- Com o aumento da árvore de decisão e do número de possíveis caminhos, o tempo de busca aumenta
- Existem várias formas de reduzir o tempo de busca, alguns dos quais serão discutidos mais adiante

Busca Heurística

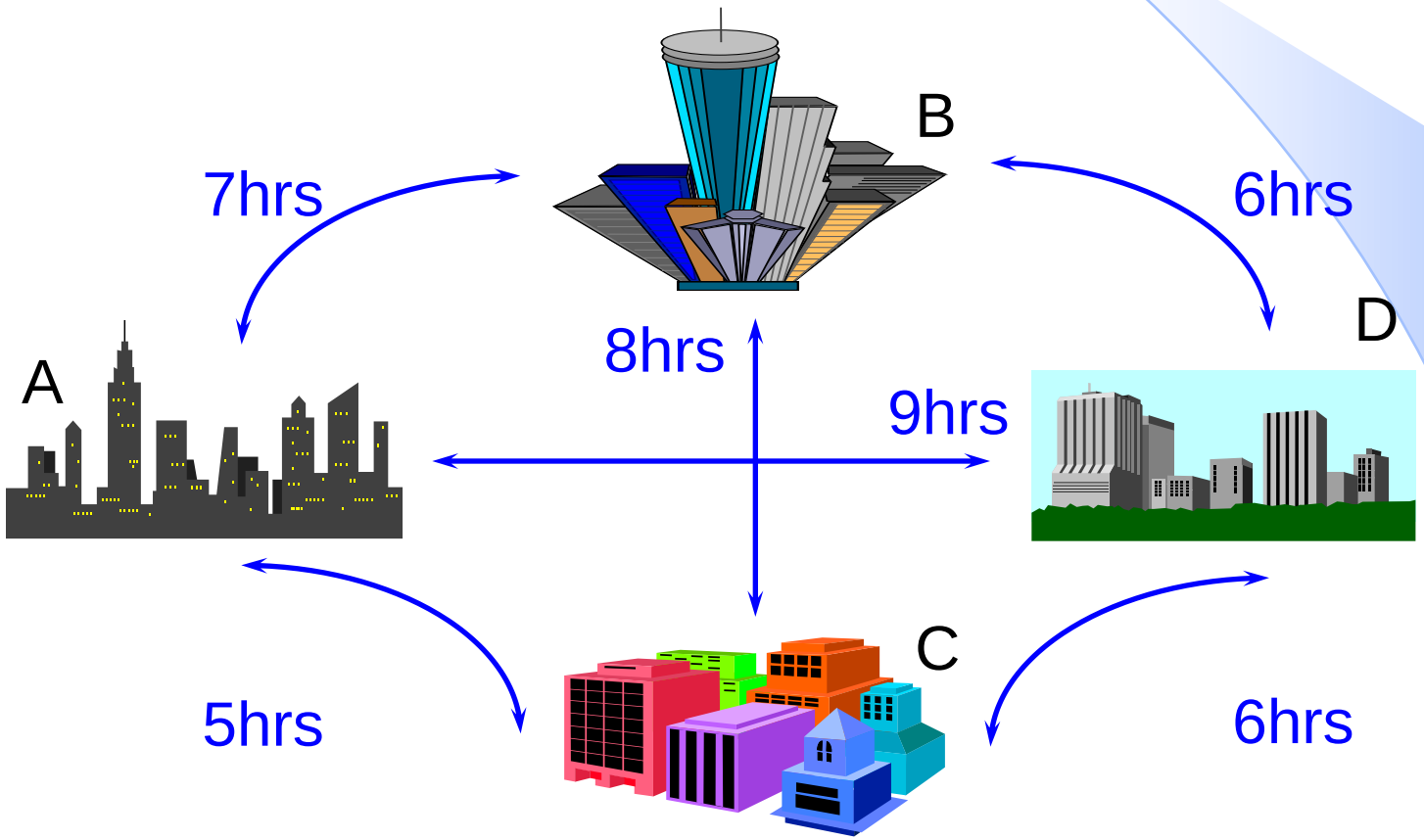
- Digamos que você está numa Cidade, e quer pegar um trem para casa, mas não sabe qual deve pegar.
- Se você morasse na zona Norte, naturalmente ignoraria todos os trens que fossem para o sul.
- Se você morasse na zona Sul, naturalmente ignoraria todos os trens que fossem para o Norte.

Exemplo: problema do caixeiro viajante (TSP)

- Um caixeiro viajante deve visitar N cidades em sua área de vendas
- O caixeiro começa de uma base, visita cada cidade uma única vez e retorna à sua cidade no final
- A cada viagem está associado um custo
  - O caixeiro deve percorrer a rota mais curta

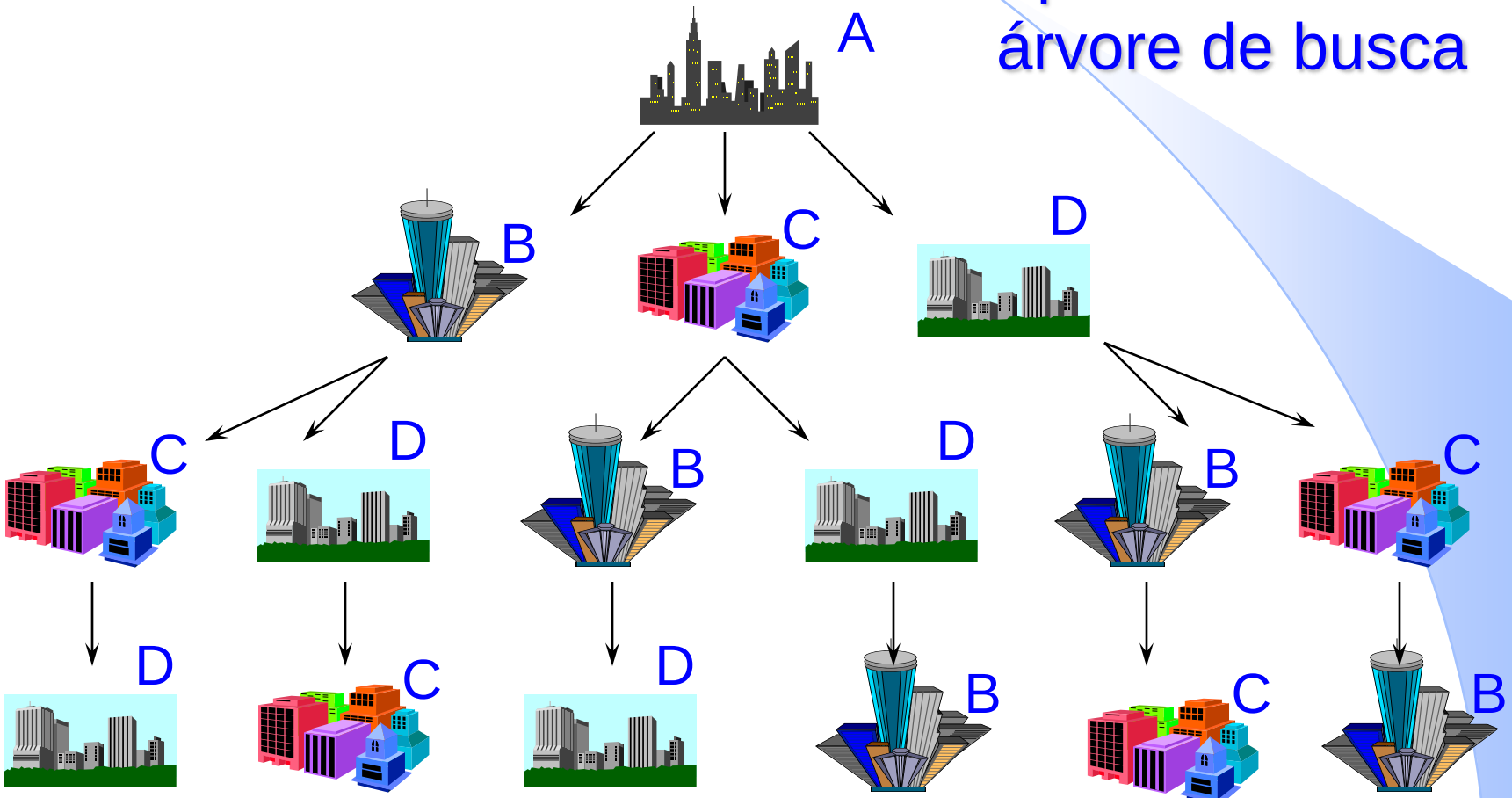
O problema TSP

Considere as rotas definidas entre estas quatro cidades:



O problema TSP

O problema do TSP representado como árvore de busca



Explosão Combinatória

- Com quatro cidades, temos 6 caminhos possíveis.
- Com dez cidades, temos 362.880 caminhos possíveis.
- Quanto mais cidades adicionarmos ao TSP, mais caminhos possíveis há.
- O que nos leva a uma explosão combinatória.
- Como prevenir ou pelo menos limitar isto?

Problemas Clássicos

- Encontrar um caminho para um objetivo
- Missionários e canibais
- N-rainhas
- Jogos
- Xadrez
- Gamão
- Torres de Hanói
- Simplesmente encontrar um objetivo
- Problema do tabuleiro de xadrez danificado

Observações

- Perguntas a serem feitas antes de utilizar métodos de busca:
  - Busca é a melhor maneira para resolver o problema?
  - Quais métodos de busca resolvem o problema?
  - Qual deles é o mais eficiente para este problema?

Heurísticas

- O TSP e outros problemas de IA são basicamente problemas de busca.
- Precisamos limitar de alguma forma o espaço de busca, e assim tornar o processo de busca mais rápido e eficiente.
- Humanos utilizariam “macetes”.
- Em IA são chamados de Heurísticas.
- Estas heurísticas ajudam a limitar a busca.

Busca Heurística

- Melhores estratégias de Busca para resolução de problemas complexos
- Heurística: “conselhos” não numéricos que determinam o sucessor de um dado estado

Busca Heurística (cont)

- Possui efeito “local” - oferece um conselho de escolha do sucessor de um estado específico mas não referente à toda estratégia de busca
- Principal papel: eliminar ou podar ramos de busca

Busca Heurística (cont)

- Problema - duas heurísticas podem fazer recomendações contraditórias
  - Se uma é melhor que a outra, então a melhor deve ter prioridade
- Podem ser utilizadas:
  - Funções de Custo
  - Funções de Avaliação

Funções de Custo

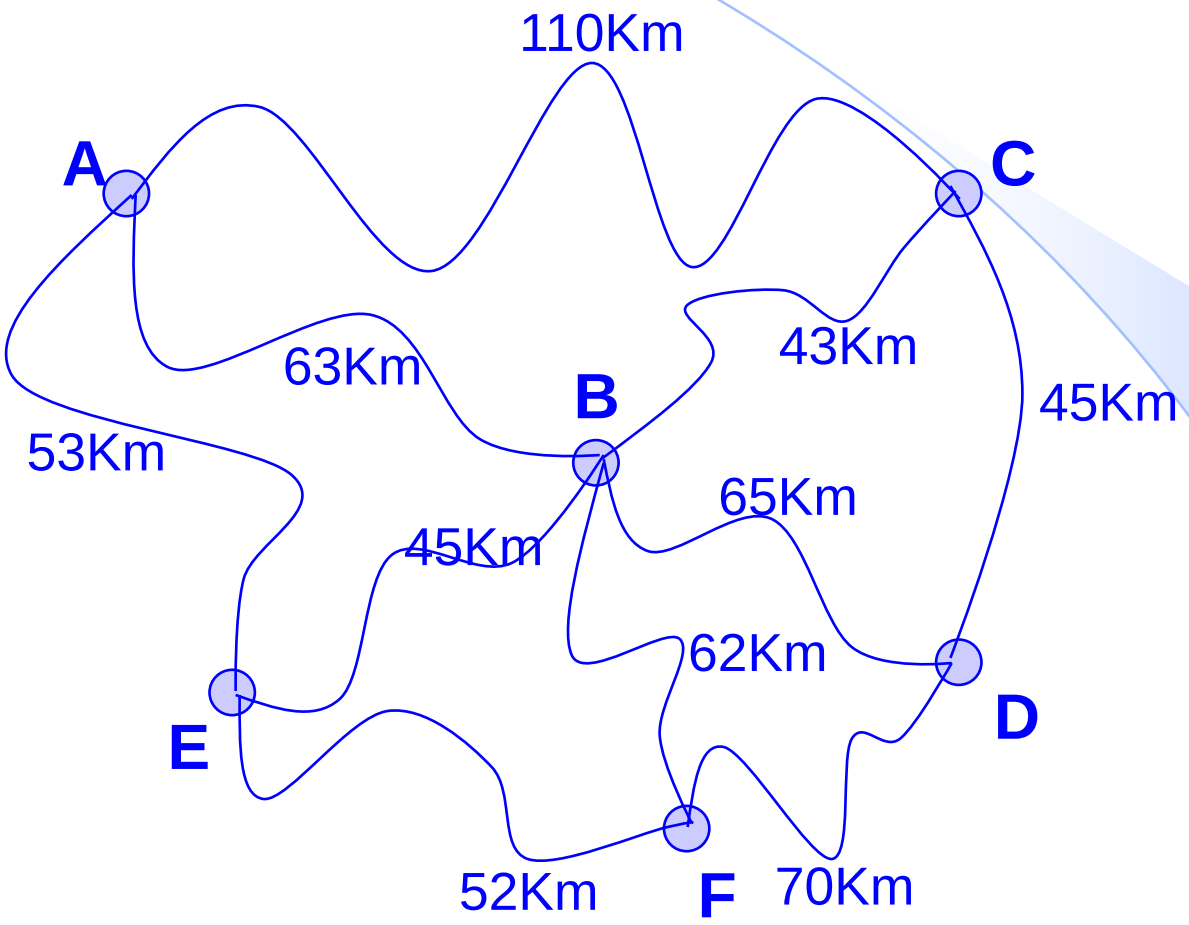
- Funções não negativas que medem a dificuldade de ir de um estado para o outro
- Usando-as, é possível encontrar um bom ou ainda o melhor caminho para alcançar uma dada meta
- Referência ao **passado**
- **g(n): Custo para ter chegado até o nó do grafo**

Funções de Avaliação

- Método para:
  - calcular um valor numérico para os estados sucessores de um dado estado
  - decidir pelo sucessor que tem o “melhor” valor
- Valores:
  - números não negativos
  - o menor valor é o mais promissor
  - o estado meta é 0 (zero)
- Referência ao **futuro**

- **h(n): Custo estimado para chegar ao nó meta**

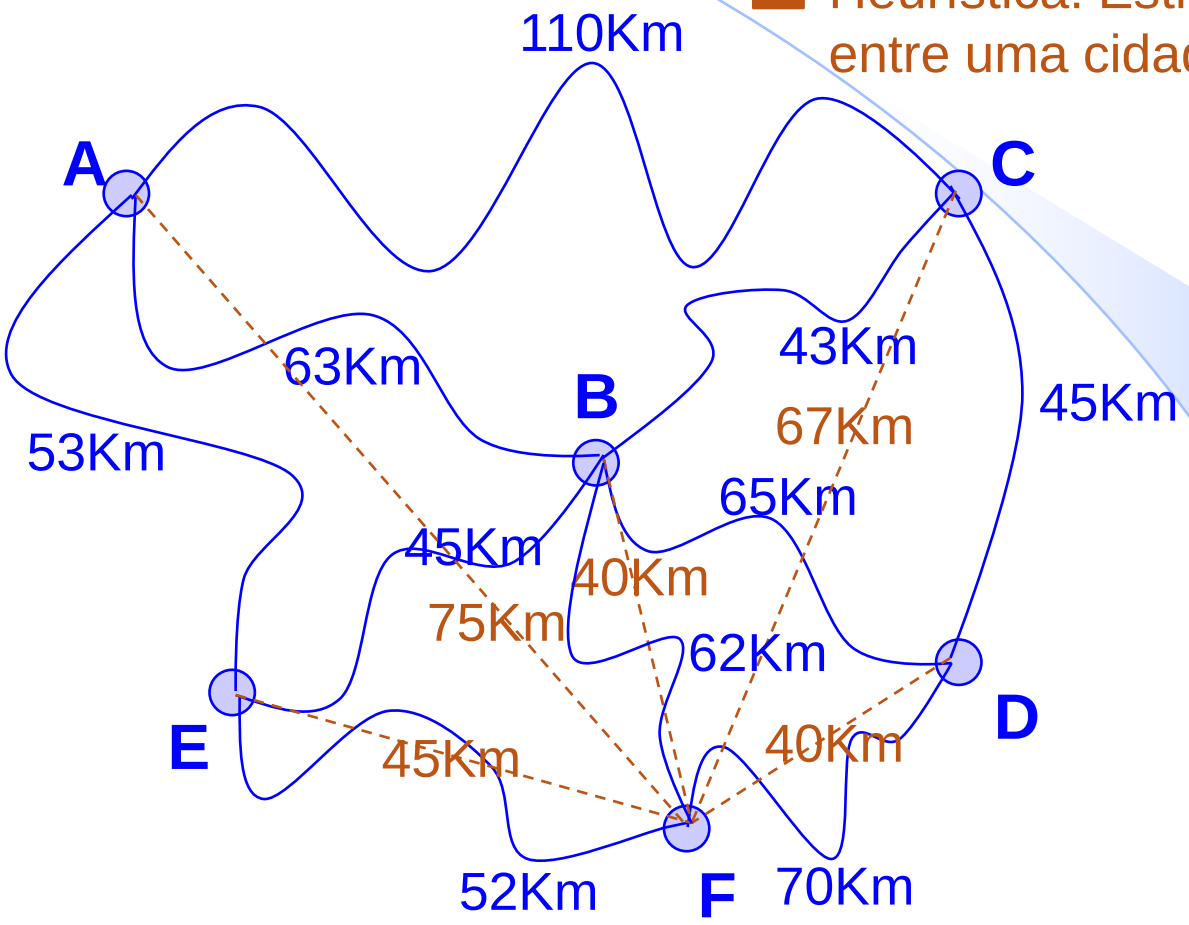
Exemplo – Busca Heurística



- Custo real entre uma cidade e outra

Exemplo – Busca Heurística

- Heurística: Estimativa de custo entre uma cidade e outra

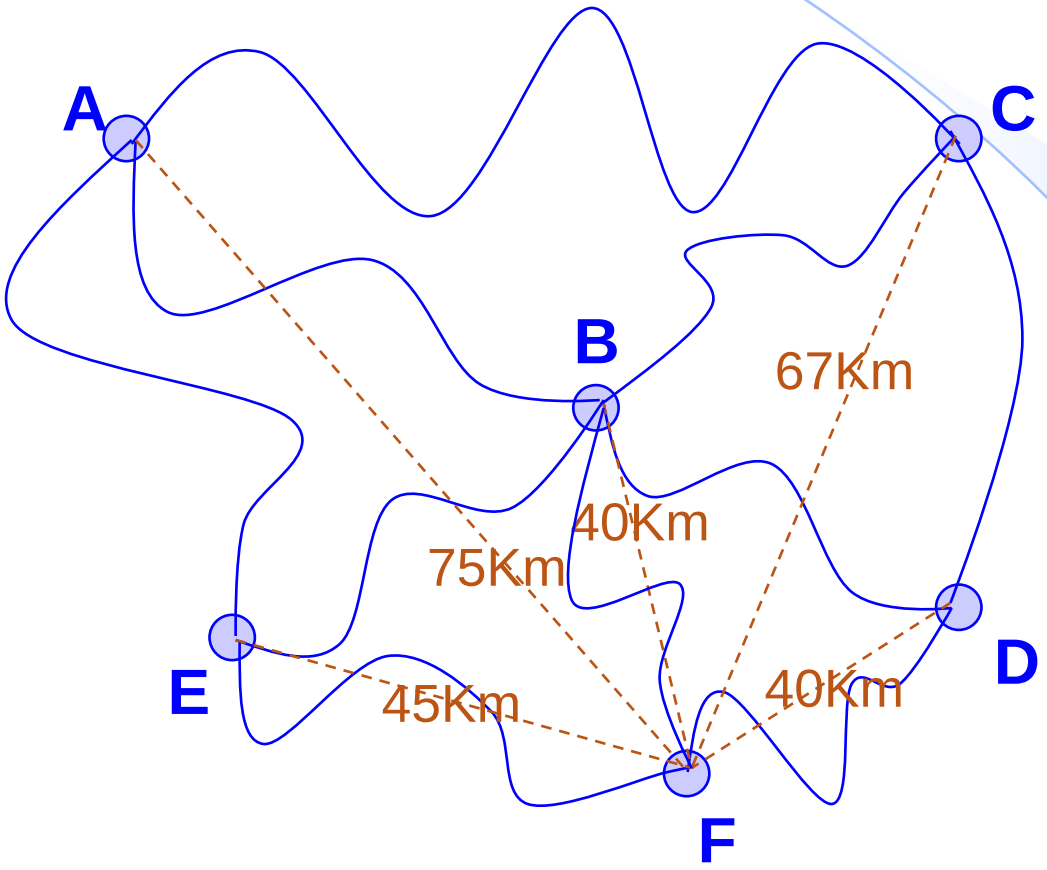


- Custo real entre uma cidade e outra

Estratégias de Busca usando Funções de Avaliação

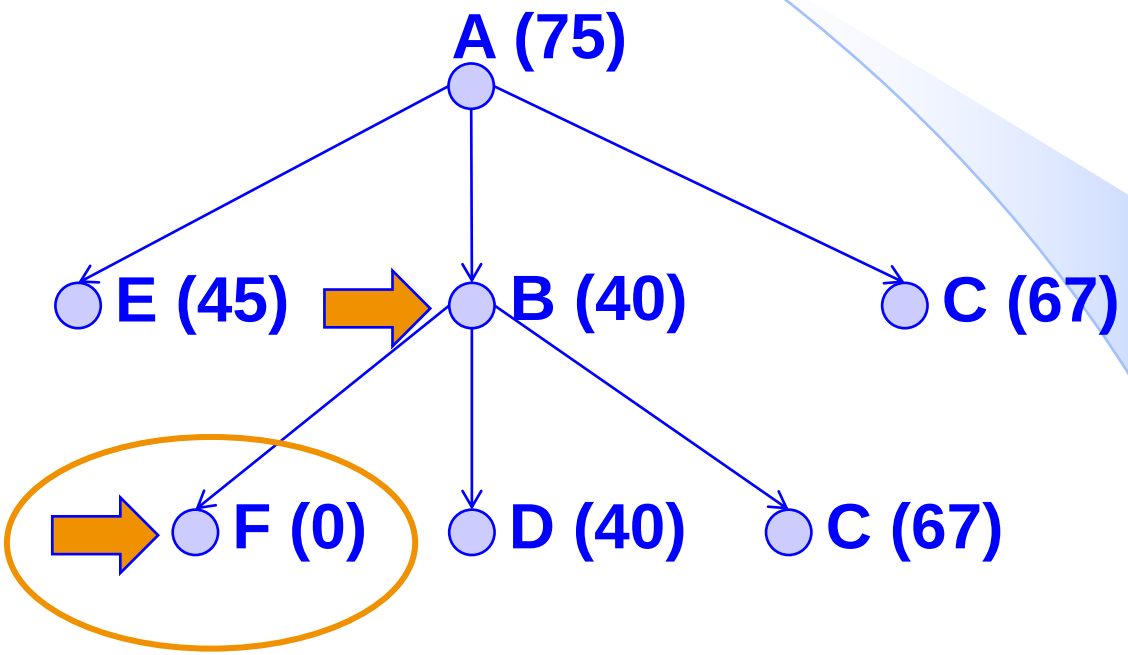
- Hill-Climbing (ou otimização discreta)
  - consiste de uma busca em ***profundidade*** usando funções de avaliação – $h(n)$
- Best-First
  - consiste de uma busca em ***largura*** usando funções de avaliação – $h(n)$

Execução do Algoritmo Hill-Climbing

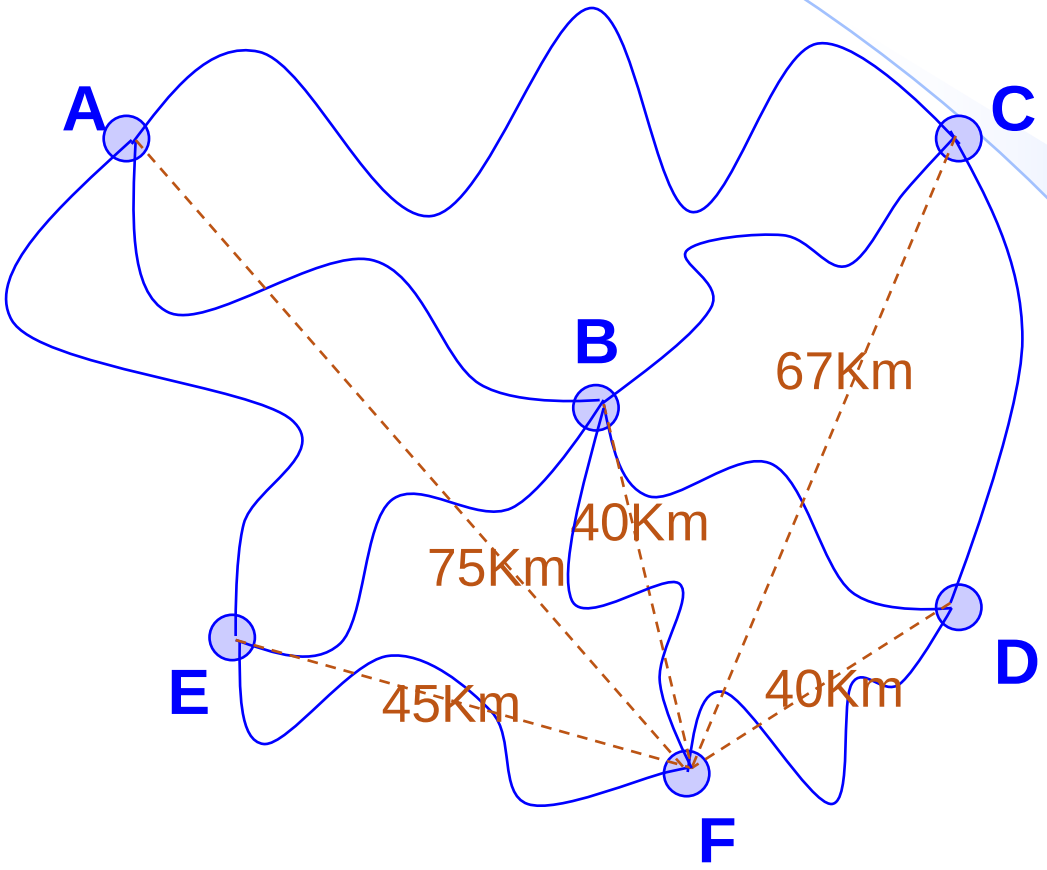


- Como fica a solução?
  - Profundidade + h(n)

Execução do Algoritmo Hill-Climbing – Solução

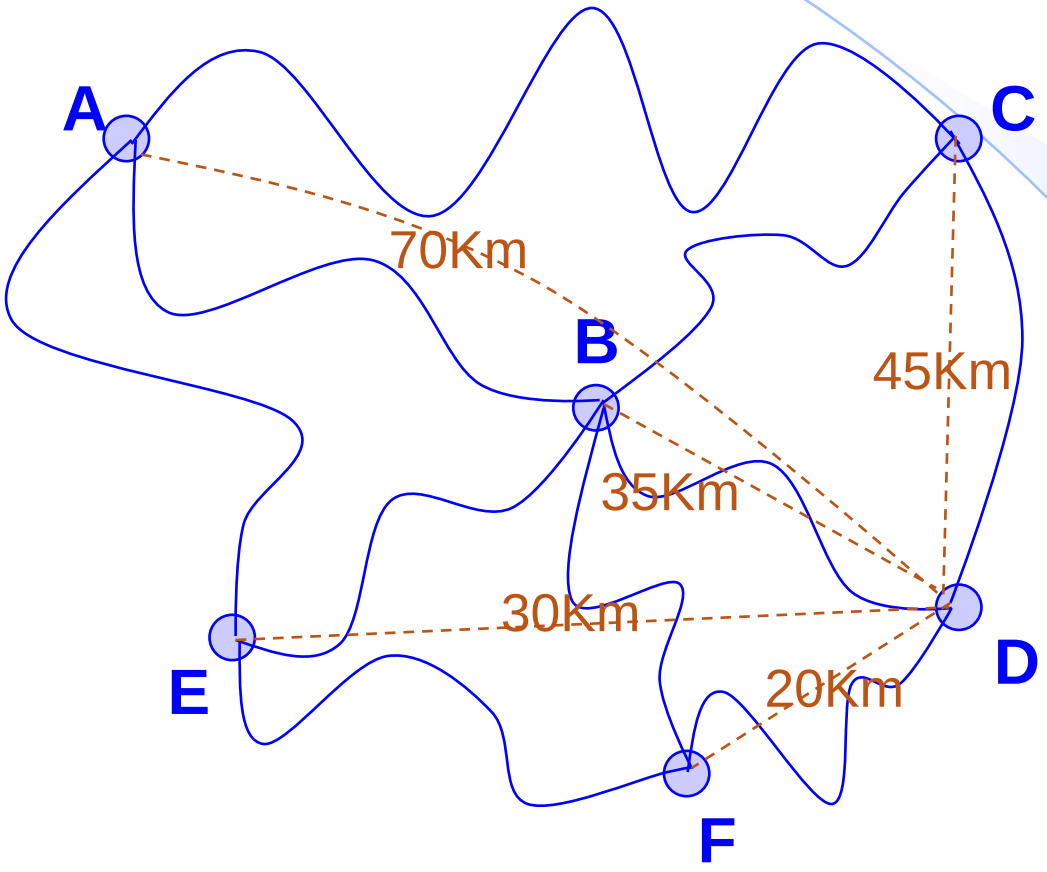


Execução do Algoritmo Best-First



- Como fica a solução?
  - Largura + h(n)

Exercício – de A para D



Execução do Algoritmo Best-First – Solução

1 - [[75, A]]

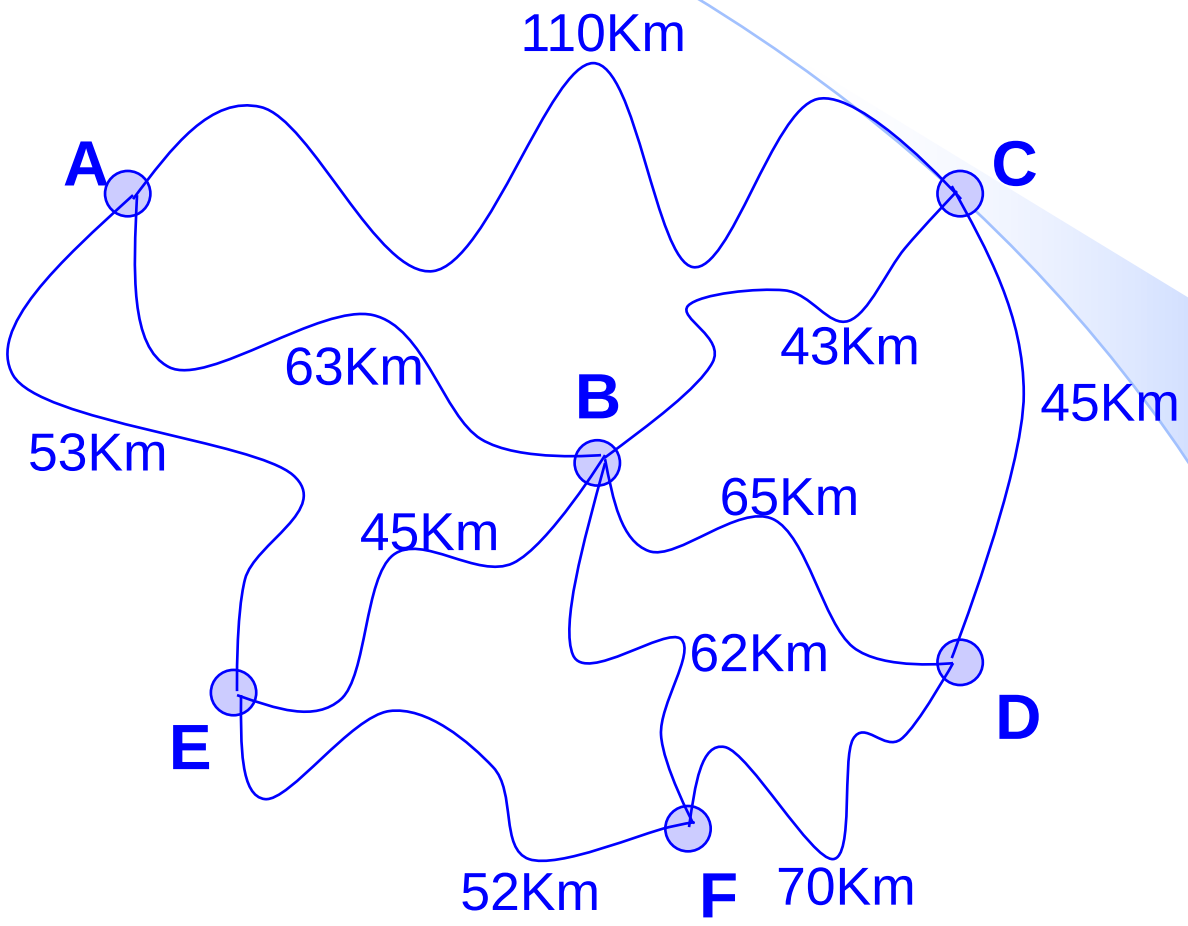
2 - [[45,E,A] , [40,B,A] , [67,C,A]] – Varrer a lista e buscar o elemento com menor valor

3 - [[0,F,B,A] , [40,D,B,A] , [67,C,B,A] , [45,E,A] , [67,C,A]]

Estratégias de Busca usando Funções de Custo

- Branch-and-Bound
  - Consiste de uma busca em ***largura*** usando funções de custo
  - Esse método constrói os candidatos à solução passo a passo
  - A cada passo o candidato é avaliado
  - Se num determinado instante esta avaliação permite concluir que aquele candidato não pode levar a uma solução então aquela sequência de geração é interrompida
  - O processo retrocede ao passo anterior e uma nova seqüência de geração é tentada

Execução do Algoritmo Branch and Bound



- Como fica a solução?
  - Largura + g(n)

Execução do Algoritmo Branch and Bound

1 - [[A]]

2 - [[53,E,A] , [63,B,A] , [110,C,A]] – Varrer a lista e buscar o elemento com menor valor

3 - [[105,F,E,A] , [98,B,E,A] , [63,B,A] , [110,C,A]]

4 - [[108,E,B,A] , [125,F,B,A] , [128,D,B,A] , [106,C,B,A] , [105,F,E,A] , [98,B,E,A] , [110,C,A]]

5 - [~~[98,E,B,E,A]~~ , [160,F,B,E,A] , [163,D,B,E,A] , [136,C,B,E,A] , [108,E,B,A] , [125,F,B,A] , [128,D,B,A] , [106,C,B,A] , [105,F,E,A] , [110,C,A]]

Busca Ótima (A*)

- Faz uso tanto da função de avaliação quanto da de custo
- Atribui valores de custo e avaliação aos sucessores de um estado a fim de selecionar aquele mais promissor
- Importante: os valores possuem mesmas unidades!

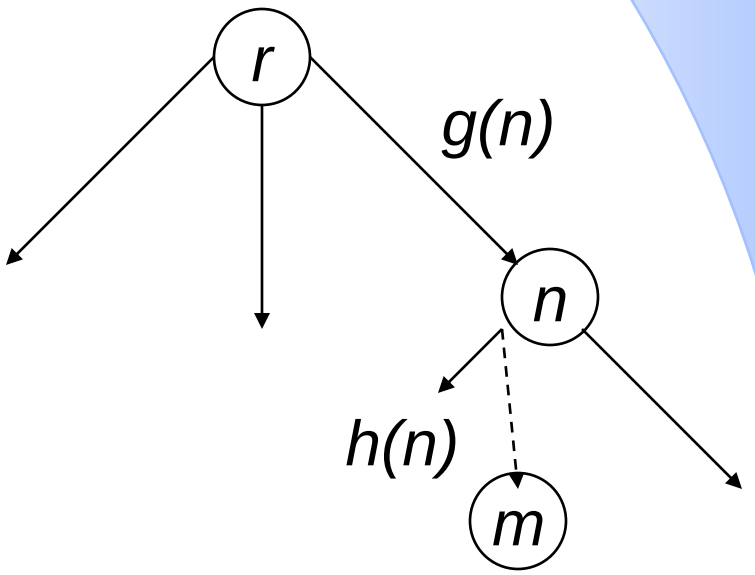
Busca Ótima (A*) (cont)

- Estimativa de custo de ir da raiz (*r*) até a meta (*m*) passando pelo nó *n:*

$$f(n) = g(n) + h(n)$$

onde:

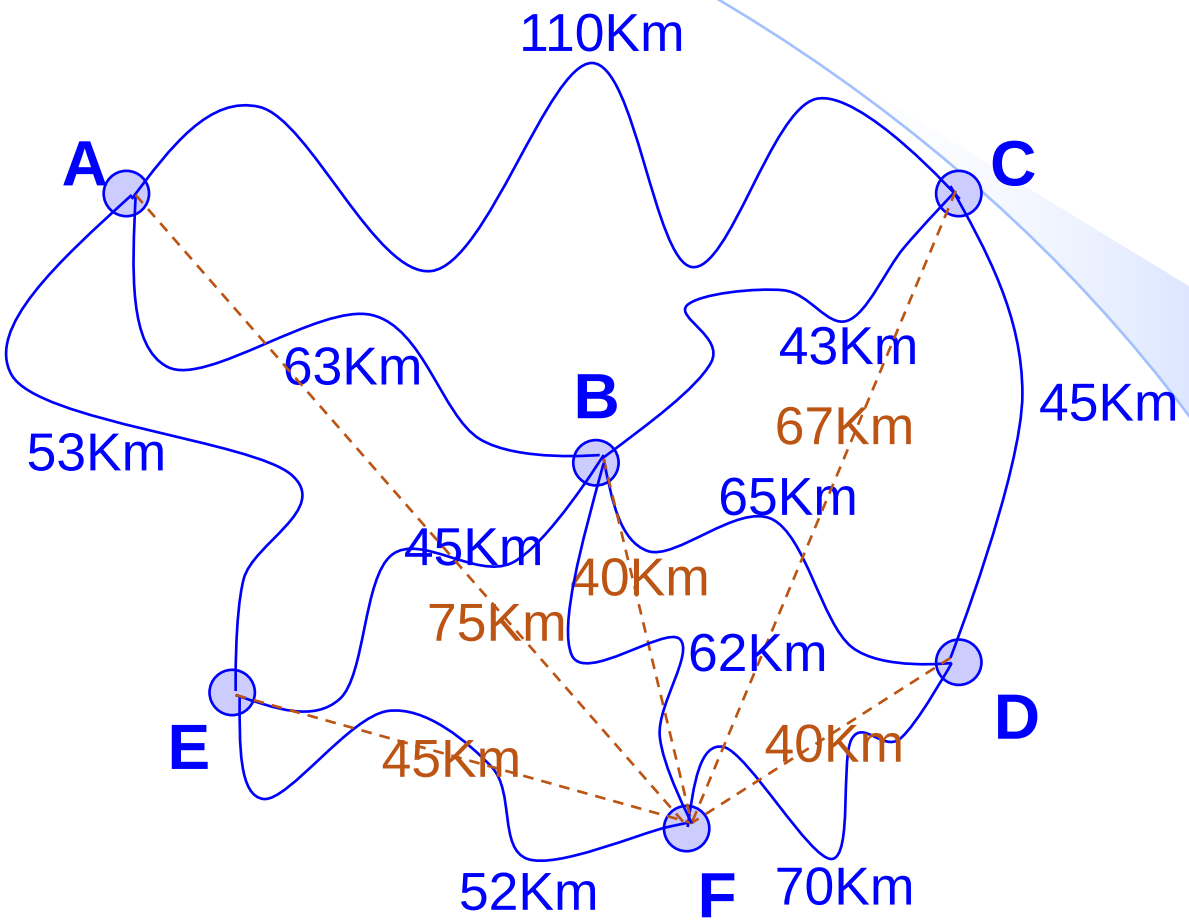
- g(n) = função de custo do nó *n*
- h(n) = função de avaliação do nó *n*



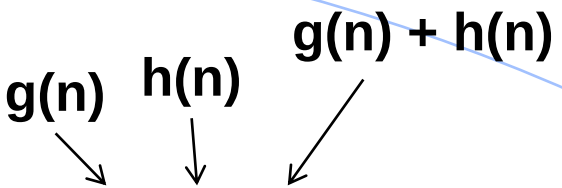
Busca Ótima (A*) (cont2)

Se a função de avaliação para qualquer estado e_i é sempre menor ou igual que o custo real de e_i para a meta, então o primeiro caminho encontrado pela estratégia de busca A* é o caminho de custo mínimo **(ótimo)**

Execução do Algoritmo A*



- Como fica a solução?
  - Largura + g(n) + h(n)



1 - [[0,75,75,A]]

2 - [[53,45,98,E,A] , [63,40,103,B,A] ,
[110,67,177,C,A]]

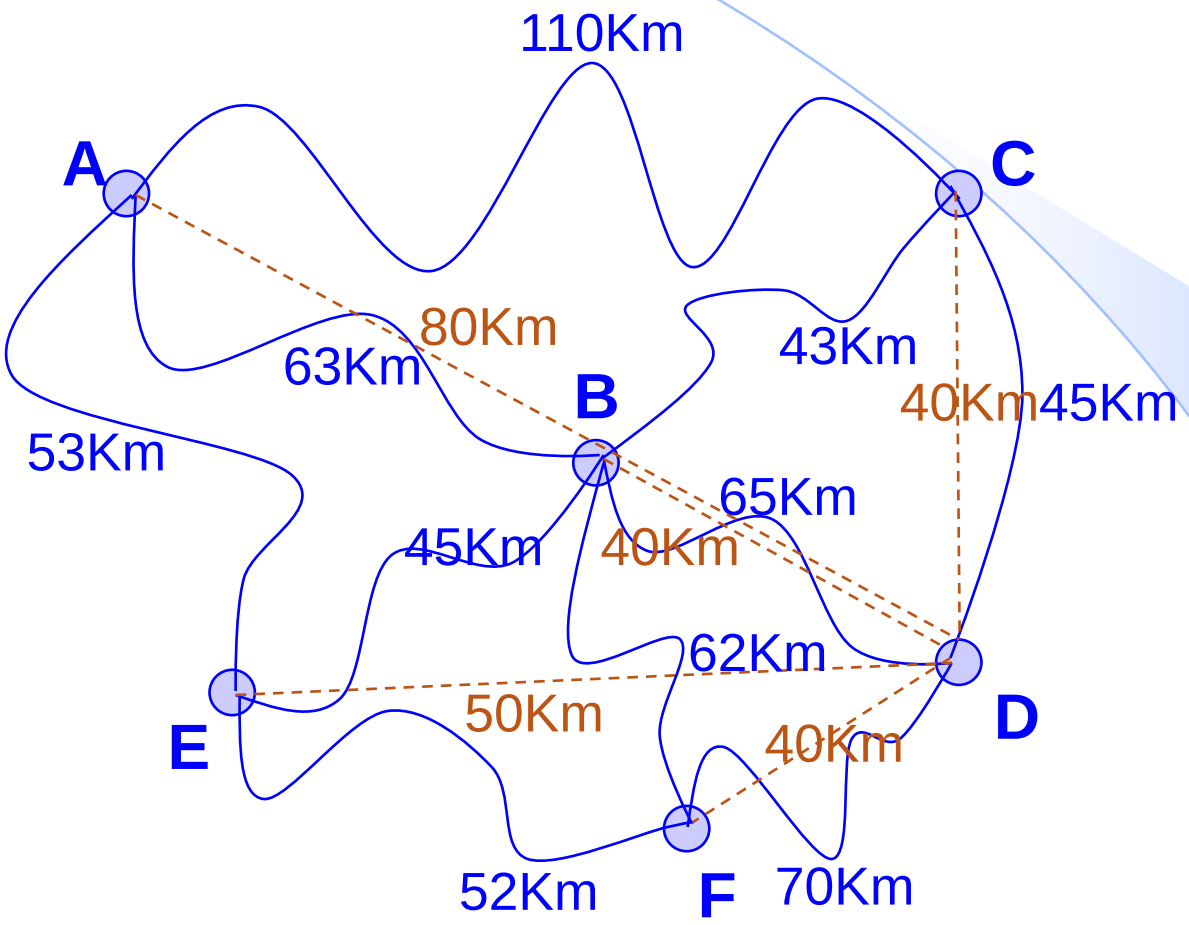
3 - [[105,0,105,F,E,A] , [128,40,168,B,E,A] ,
[63,40,103,B,A] , [110,67,177,C,A]]

4 - [[138,45,183,E,B,A] , [125,0,125,F,B,A] ,
[128,40,168,D,B,A] , [130,67,197,C,B,A] ,
[105,0,105,F,E,A] , [128,40,168,B,E,A] ,
[110,67,177,C,A]]

Organização das Estratégias de Busca

Estratégias de Busca	Usa Agenda?	Usa função de avaliação	Usa função de custo	Próximo estado
Profundidade	*Não*	*Não*	*Não*	o sucessor do último estado, caso contrario o sucessor do predecessor
Largura	*Sim*	*Não*	*Não*	o estado mais longe na agenda(fila)
Hill-Climbing	*Não*	*Sim*	*Não*	o sucessor com mínimo valor de função de avaliação
Best-First	*Sim*	*Sim*	*Não*	o estado na agenda com mínimo valor de função de avaliação
Branch-and-Bound	*Sim*	*Não*	*Sim*	o estado na agenda com mínimo valor de função de custo total
A*	*Sim*	*sim*	*Sim*	o estado na agenda com mínimo valor da soma da função de avaliação e custo total

Exercício



- Execute o algoritmo Branch and Bound e A*

Bibliografia

- **Busca Informada e Não Informada: Aplicações.** Nota Didática. M.C.Monard - ICMC-USP
- **Introdução a Programação PROLOG**. UCPel - Universidade Católica de Pelotas, caps. 14 e 15
- **Artificial Inteligence Techniques in Prolog**. YOAV SHOHAM, Stanford University