PROCURA EM ESPAÇOS DE ESTADOS

Luís Morgado

2024

RACIOCÍNO ATRAVÉS DE PROCURA

O raciocínio automático através de procura em espaços de estados é um método de resolução de problemas, nomeadamente **problemas de planeamento**, nos quais se pretende encontrar uma sequência de situações e de acções que levem de uma situação inicial a uma situação final.

Um exemplo deste tipo de problemas é a navegação com base em mapas para determinar percursos entre localizações.

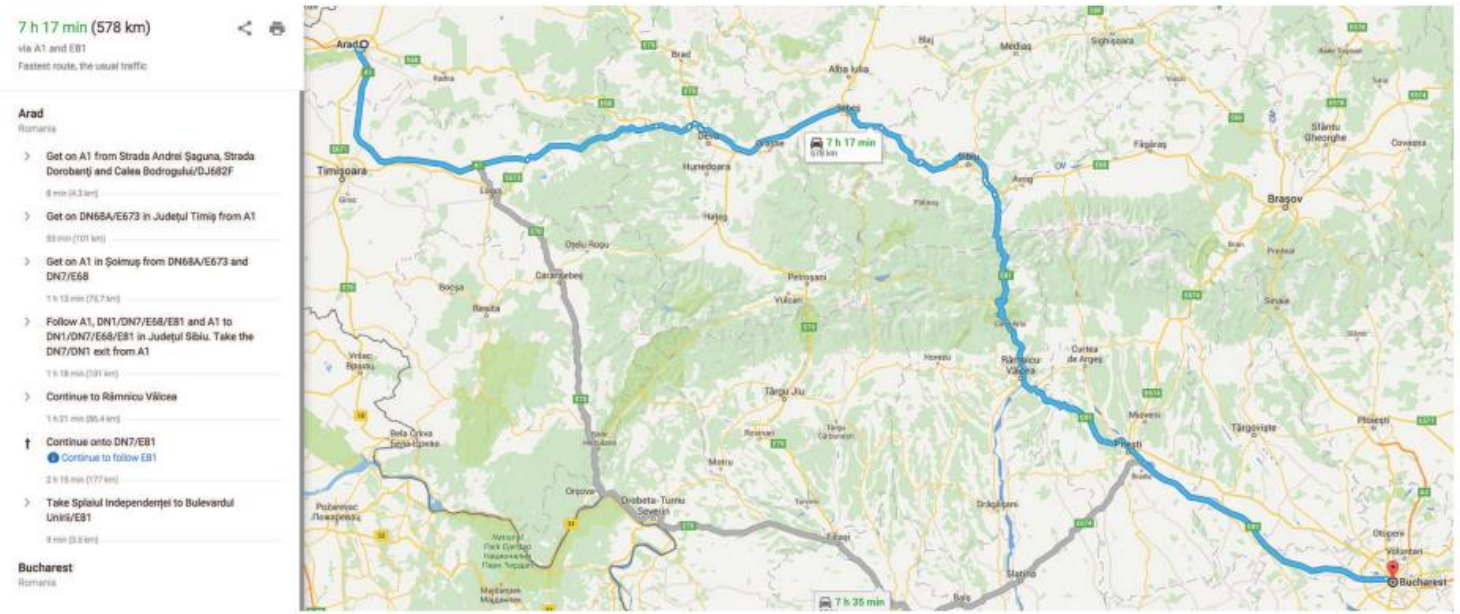


Figure 3.28 A Web service providing driving directions, computed by a search algorithm.

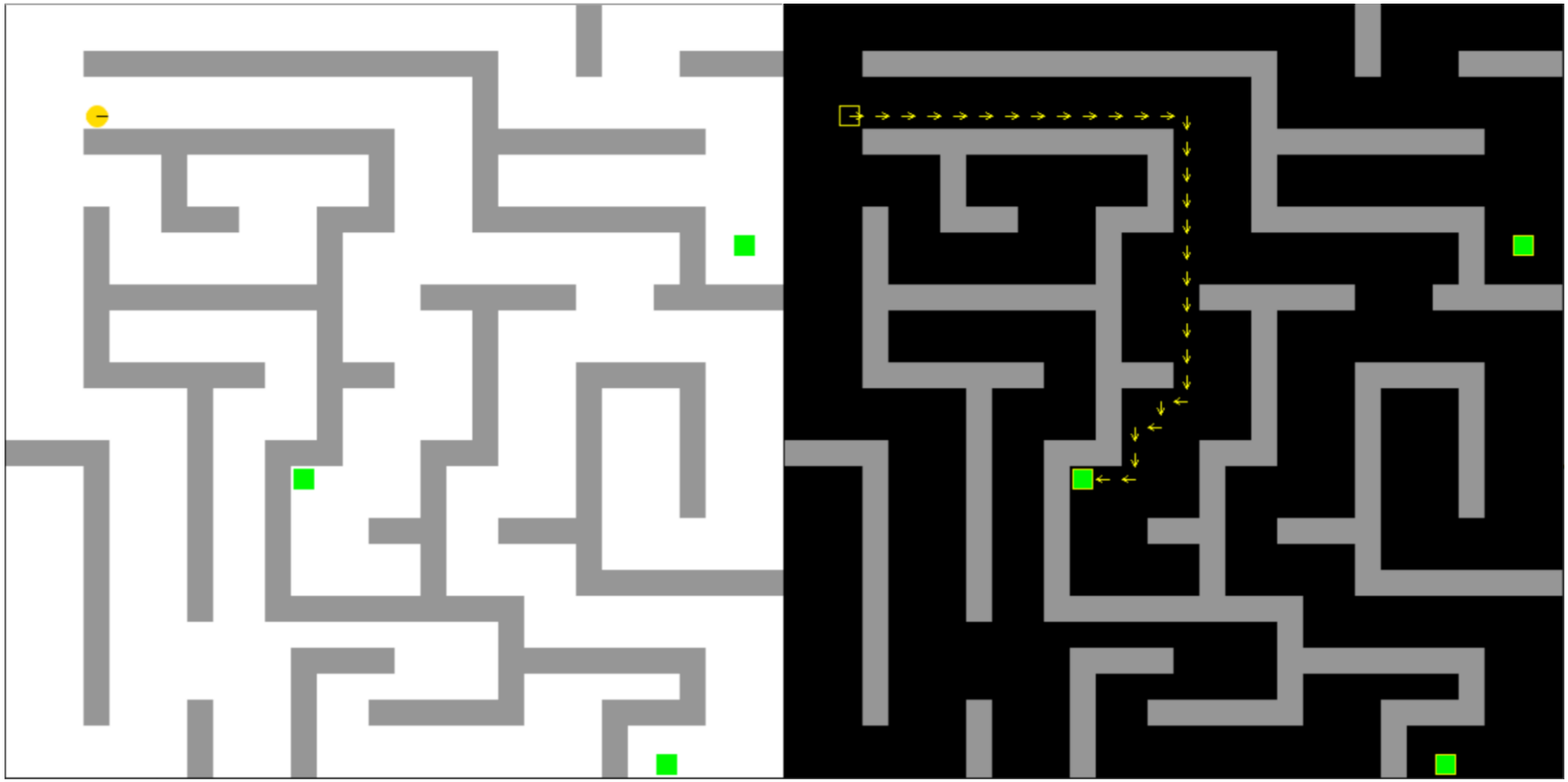
[Russel & Norvig, 2010]

RACIOCÍNO ATRAVÉS DE PROCURA

PROBLEMAS DE PLANEAMENTO

Problemas cuja solução consiste numa sequência de acções a realizar e de situações a percorrer para, partindo de uma situação inicial, atingir uma situação final (objectivo).

Exemplo na plataforma de simulação:

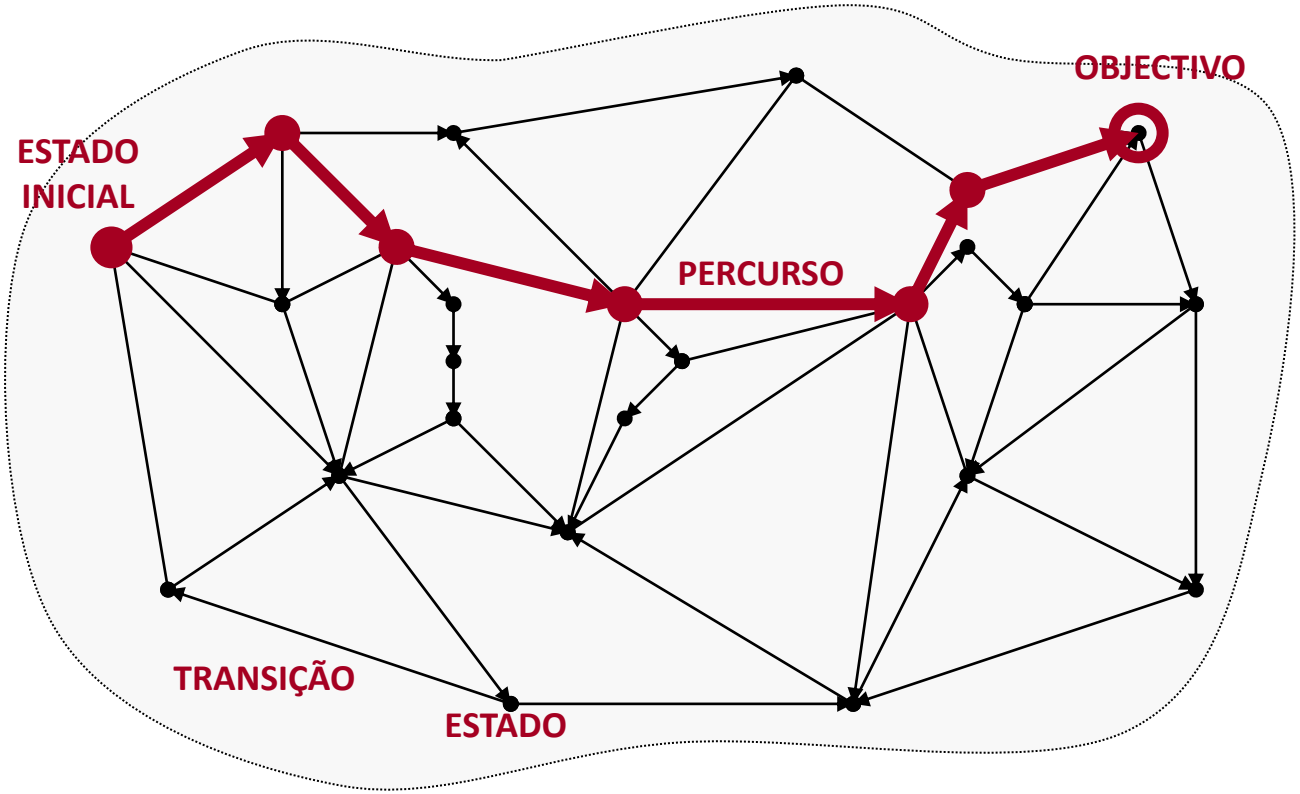


RACIOCÍNO ATRAVÉS DE PROCURA

PROBLEMAS DE PLANEAMENTO

O raciocínio automático através de procura, opera na resolução de problemas de planeamento tendo por base uma abstração do domínio do problema, designada **espaço de estados**, em que cada situação (configuração) possível na resolução do problema é representada como um **estado** no espaço de estados, e as várias acções que produzem a mudança (transformação) de uma configuração para outra, são representadas como **operadores** que quando aplicados originam transições de estado, gerando novos estados.

A solução de um problema corresponde a uma sequência de estados e operadores, ou seja, um **percurso** no espaço de estados, que liga um **estado inicial** a um estado **objectivo**.



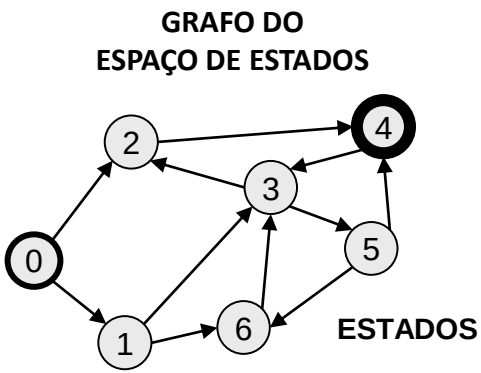
PROCESSO DE PROCURA

O processo de procura ocorre por **exploração do espaço de estados** a partir do estado inicial

O espaço de estados é representado como um grafo, onde cada vértice corresponde a um estado e cada arco corresponde a uma transição entre estados

Em cada passo de processamento:

- É verificado se o **estado actual corresponde ao *objectivo***
  - Se corresponder ao objectivo, o processamento termina e é retornado o percurso (sequência de estados e operadores) do estado inicial ao estado objectivo
- Não sendo o estado actual um objectivo, esse estado é ***expandido***, sendo gerados todos os ***estados sucessores*** por aplicação dos vários ***operadores*** possíveis
- Para cada estado sucessor é repetido o processo
- **Se não existirem estados sucessores, o processo de procura termina** com a indicação de que não existe solução
  - A não existência de estados sucessores resulta, por exemplo, da exploração de todo o espaço de estados sem ter sido encontrada solução



ÁRVORE DE PROCURA

Para **manter a informação gerada em cada passo de procura** é mantida uma estrutura de informação, designada ***árvore de procura***

Consiste numa **estrutura em árvore, organizada em *nós***, que mantém informação relativa a cada transição de estado explorada

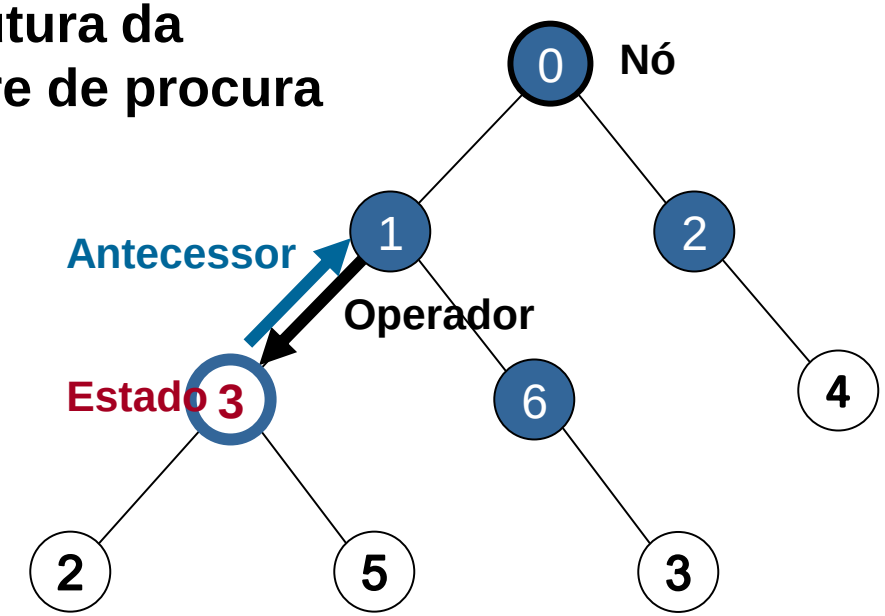
Relaciona cada nó com o seu antecessor e mantendo informação do estado correspondente ao nó e do operador que originou a transição de estado respectiva gerando esse novo estado

Nó:

- Estado
- Operador (que gerou o estado)
- Antecessor
- Profundidade (do nó)
- Custo (da raiz até ao nó)

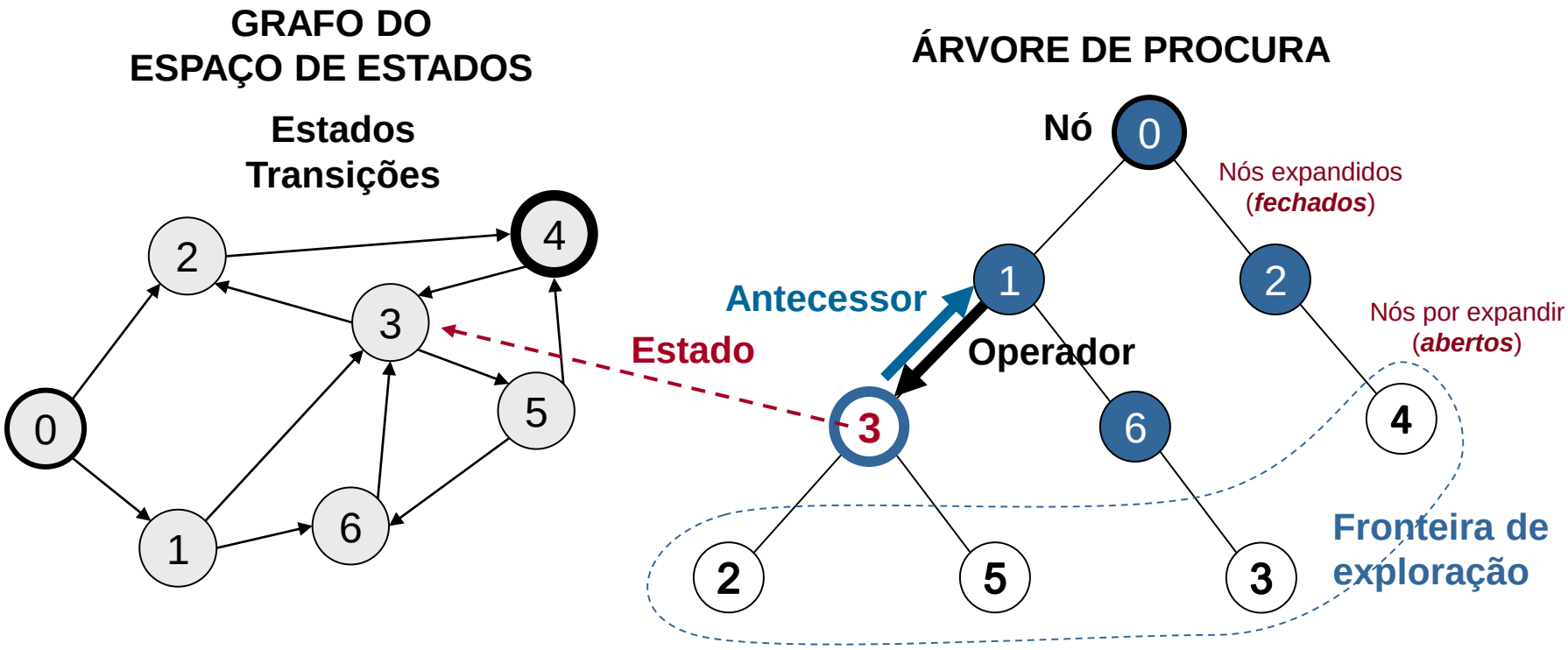
Profundidade e **custo**:
Informação complementar para controlo do processo de procura

Estrutura da árvore de procura



PROCESSO DE PROCURA

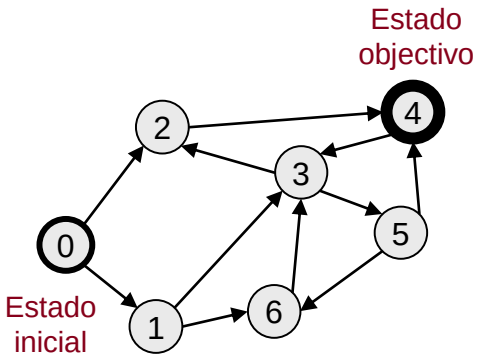
- Exploração sucessiva do espaço de estados
- Etapa de procura: **Nó**
- **Árvore de procura**
  - Raiz: Nó correspondente ao e*stado inicial*
- **Fronteira de exploração** (estrutura de dados com relação de ordem)
  - Critério de ordenação determina estratégia de controlo da procura



MÉTODOS DE PROCURA

Procura em profundidade (*Depth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **recentes**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no início** da fronteira



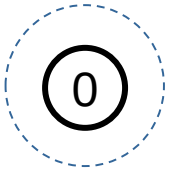
Grafo do Espaço de Estados

Fronteira de exploração [ ]

[0]

Fronteira de exploração mantida em memória como uma lista de nós

(Neste exemplo os nós estão representados pelo número do estado respectivo)



Fronteira de exploração

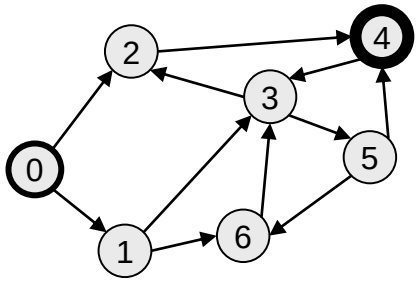
Árvore de Procura

No início da procura contém um nó correspondente ao estado inicial do problema

MÉTODOS DE PROCURA

Procura em profundidade (*Depth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **recentes**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no início** da fronteira



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

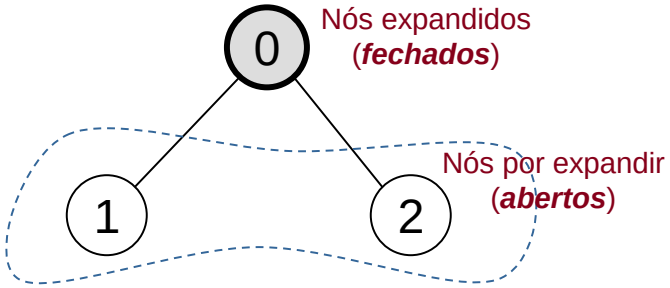
[1, 2]

Passo de procura

- Remover o primeiro nó da fronteira
- Verificar se estado do nó corresponde ao estado objectivo
- Expandir nó, inserindo os nós sucessores na fronteira de acordo com a estratégia de controlo da procura (no caso da procura em profundidade, os **nós mais recentes são inseridos no início** da fronteira, de modo a serem expandidos primeiro)

A ordem de geração dos nós sucessores depende da ordem de aplicação dos operadores

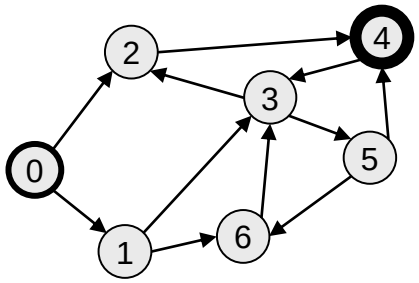
Árvore de Procura



MÉTODOS DE PROCURA

Procura em profundidade (*Depth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **recentes**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no início** da fronteira



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

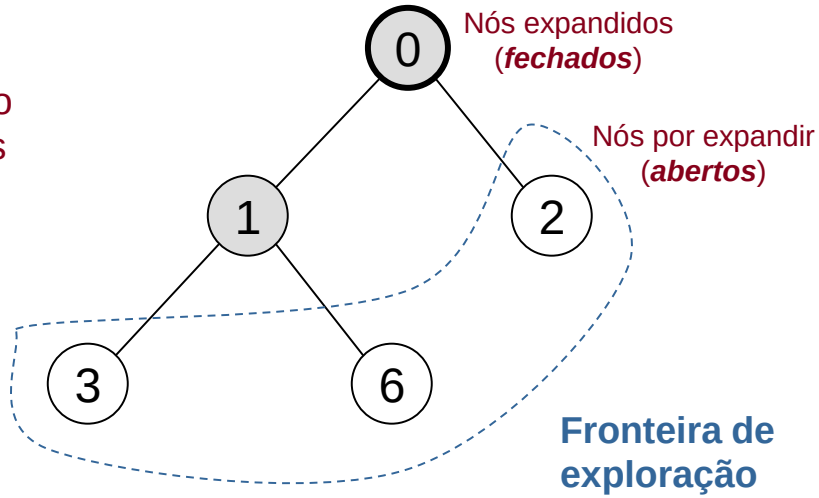
[1, 2]

1 [2]

[**3, 6**, 2]

A ordem de geração dos nós sucessores depende da ordem de aplicação dos operadores

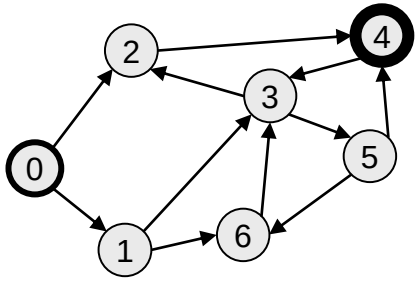
Árvore de Procura



MÉTODOS DE PROCURA

Procura em profundidade (*Depth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **recentes**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no início** da fronteira



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

[**1, 2**]

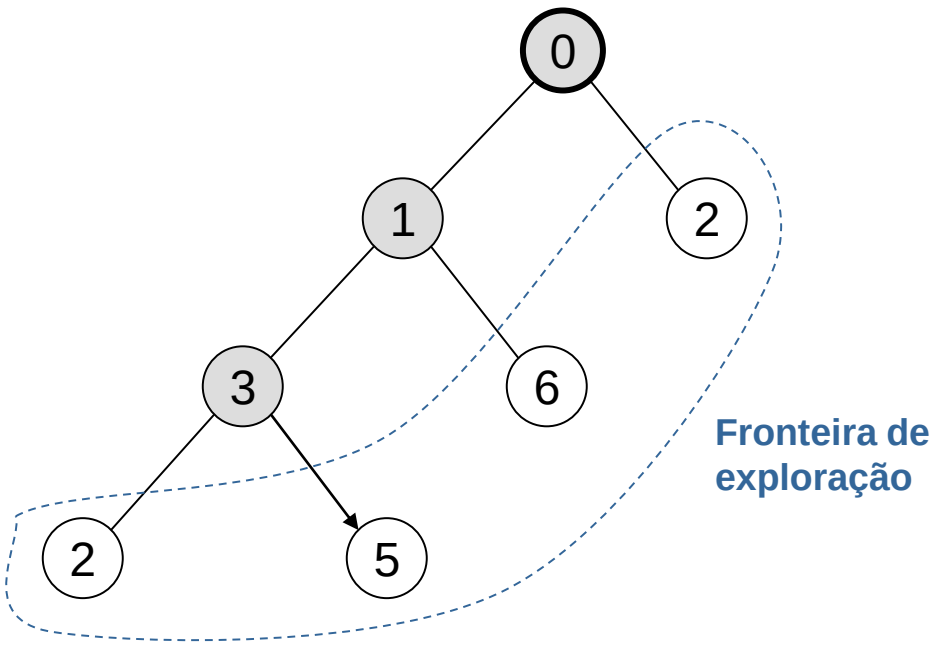
1 [2]

[**3, 6**, 2]

3 [6,2]

[**2, 5**, 6, 2]

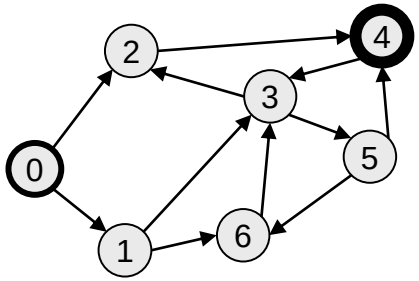
Árvore de Procura



MÉTODOS DE PROCURA

Procura em profundidade (*Depth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **recentes**



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

[**1, 2**]

1 [2]

[**3, 6**, 2]

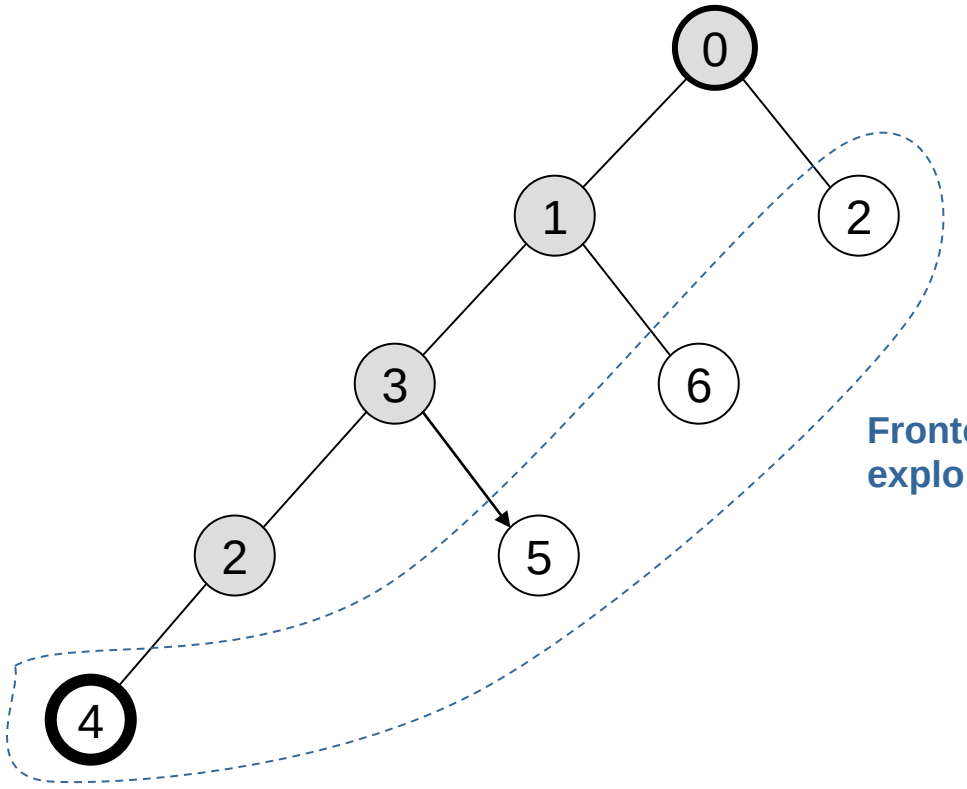
3 [6,2]

[**2, 5**, 6, 2]

2 [5,6,2]

[**4**, 5, 6, 2]

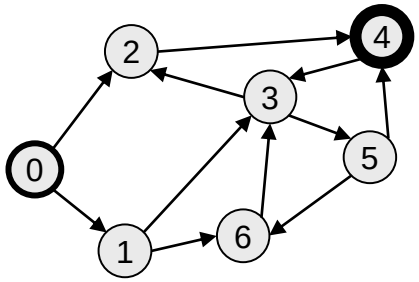
Árvore de Procura



MÉTODOS DE PROCURA

Procura em profundidade (*Depth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **recentes**



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

[**1, 2**]

1 [2]

[**3, 6**, 2]

3 [6,2]

[**2, 5**, 6, 2]

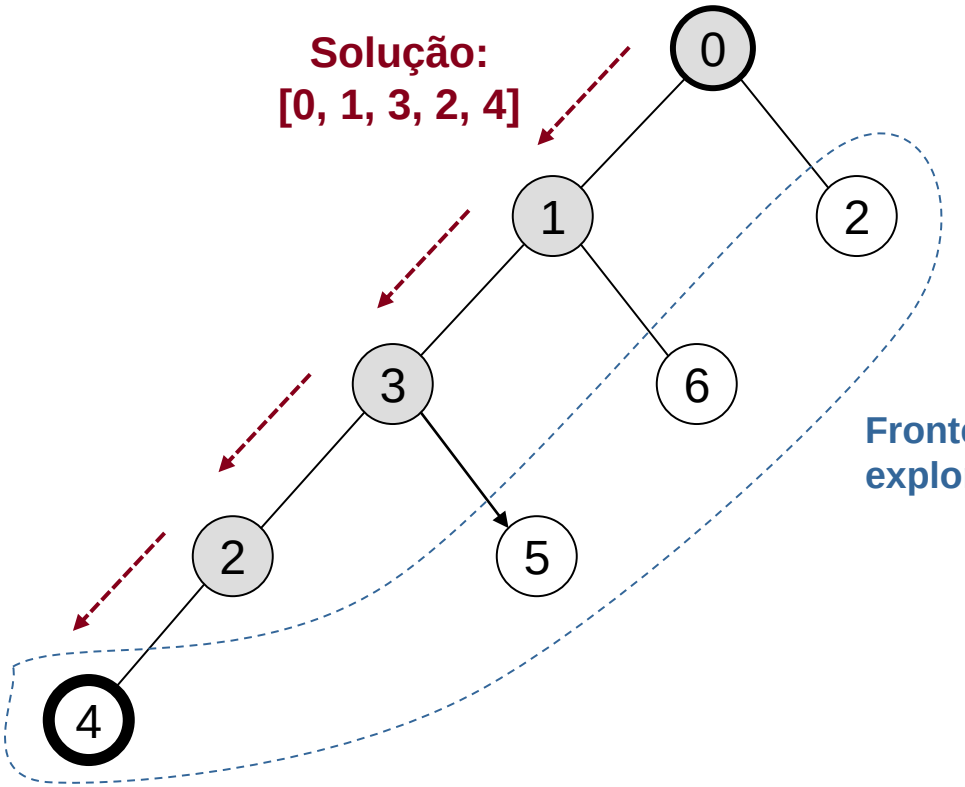
2 [5,6,2]

[**4**, 5, 6, 2]

Estado objectivo → **4** [5,6,2]

Árvore de Procura

Solução:
[0, 1, 3, 2, 4]

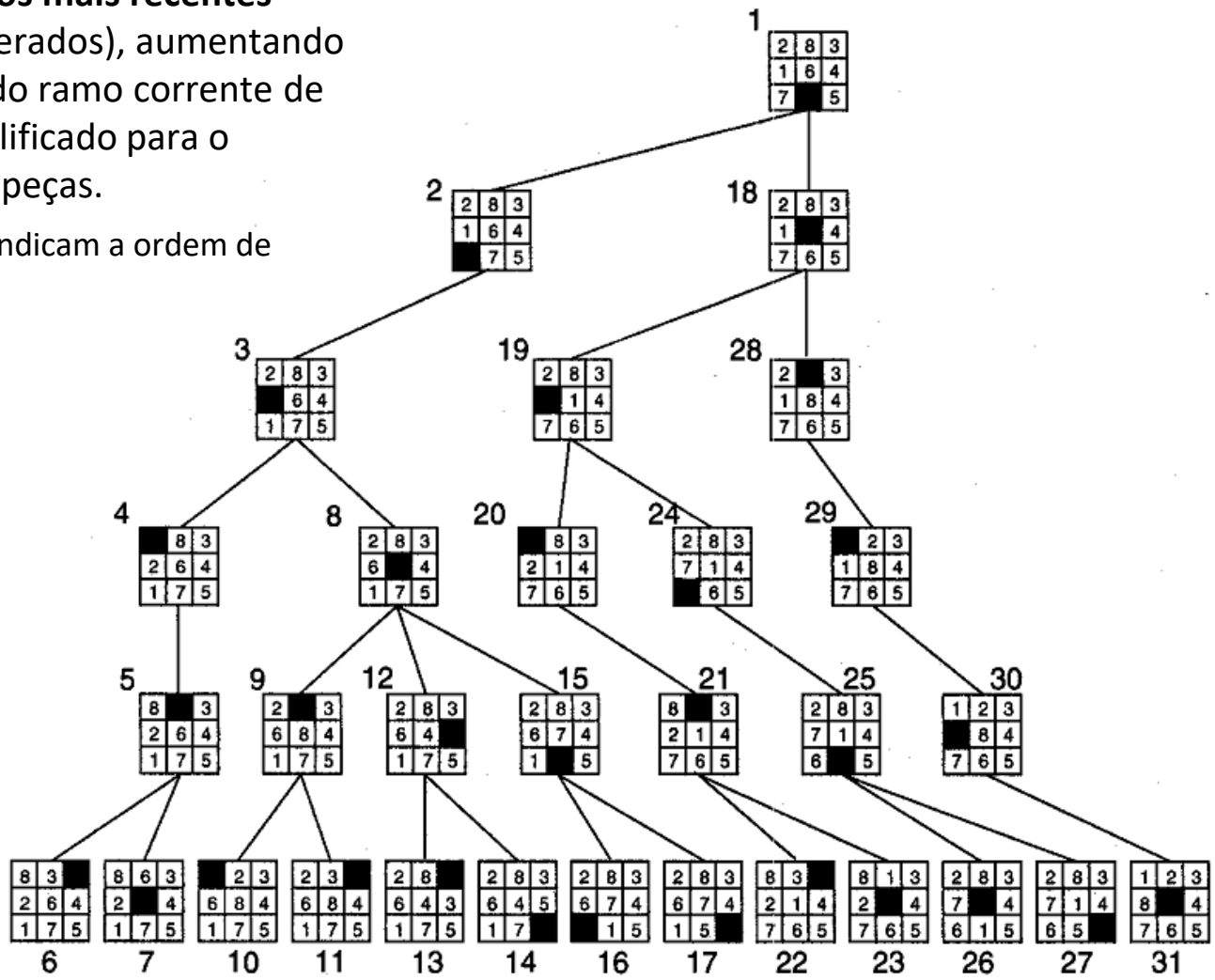


Procura termina: a solução corresponde à sequência de nós (estados e operadores) no ramo da árvore de procura que contém o nó correspondente ao estado objectivo

PROCURA EM PROFUNDIDADE

Na **procura em profundidade**, a procura decorre **explorando os nós mais recentes primeiro** (últimos a ser gerados), aumentando por isso a profundidade do ramo corrente de procura, tal como exemplificado para o problema do puzzle de 8 peças.

Nota: Os índices numéricos indicam a ordem de exploração.



[Luger, 2009]

PROCURA EM PROFUNDIDADE

Algoritmo de procura em profundidade

Representação do problema:
- Estado inicial
- Conjunto de operadores,
- Função de teste de objectivo

Representação de operador:
- Função que *aplica* o operador a um estado
- Função de *custo* de transição de estado

Processo de procura

- Criar nó inicial
- Iniciar fronteira LIFO com nó
- Enquanto fronteira não vazia
  - Remover primeiro nó da fronteira
  - Verificar se estado do nó é objectivo, se for retornar solução que termina no nó
  - Expandir o nó
  - Para cada nó sucessor
    - Inserir nó na fronteira
- Caso fronteira vazia indicar que não existe solução

```
1.  function procura_profundidade(problema) : Solucao
2.    no ← No(problema.estado_inicial)
3.    fronteira ← FronteiraLIFO(no)
4.    while not fronteira.vazia do
5.      no ← fronteira.remover()
6.      if problema.objectivo(no.estado) then
7.        return Solucao(no)
8.      for no_sucessor in expandir(problema, no) do
9.        fronteira.inserir(no_sucessor)
10.   return none
```

PROCURA EM PROFUNDIDADE

Expandir nó (gerar nós sucessores de um nó)

```
1.  function expandir(problema, no) : Lista<No>
2.     sucessores ← Lista<No>()
3.     estado ← no.estado
4.     for operador in problema.operadores do
5.        estado_suc ← operador.aplicar(estado)
6.        if estado_suc is not none then
7.           custo ← no.custo + operador.custo(estado, estado_suc)
8.           no_successor ← No(estado_suc, operador, no, custo)
9.           sucessores.juntar(no_successor)
10.    return sucessores
```

Processo de expansão de nós

- Iniciar lista de sucessores vazia
- Obter estado do nó
- Para cada operador do problema
  - Gerar estado sucessor aplicando o operador ao estado actual
  - Se nó sucessor existir
    - Calcular custo do nó sucessor (custo até ao nó antecessor + custo da transição para o estado sucessor)
    - Gerar nó sucessor
    - Juntar nó sucessor à lista de nós sucessores
- Retornar lista de nós sucessores

MÉTODOS DE PROCURA

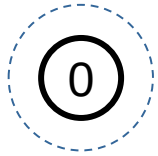
Procura em largura (*Breadth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no fim** da fronteira

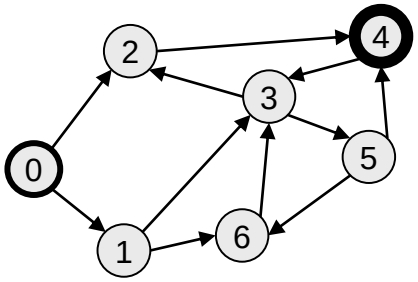
Fronteira de exploração [ ]

[0]

Árvore de Procura



Fronteira de exploração

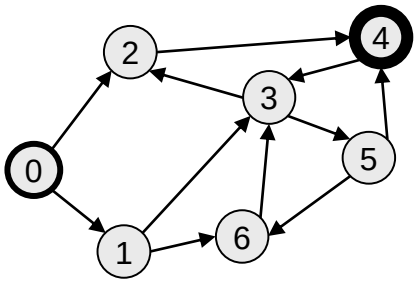


Grafo do
Espaço de Estados

MÉTODOS DE PROCURA

Procura em largura (*Breadth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no fim** da fronteira



Grafo do
Espaço de Estados

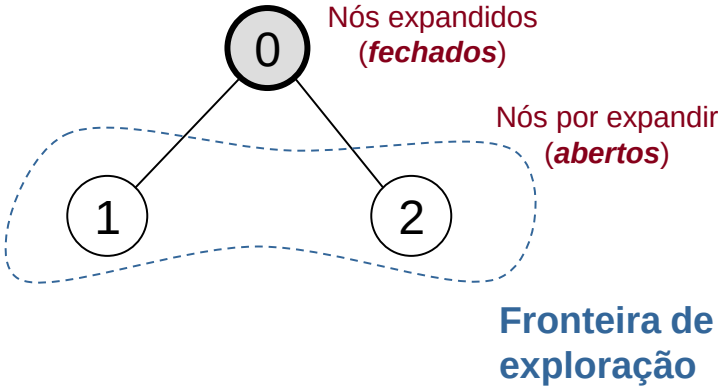
Fronteira de exploração [ ]

[0]

0 [ ]

[1, 2]

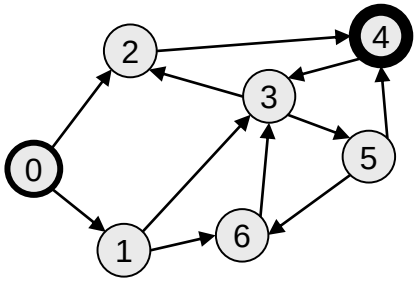
Árvore de Procura



MÉTODOS DE PROCURA

Procura em largura (*Breadth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no fim** da fronteira



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

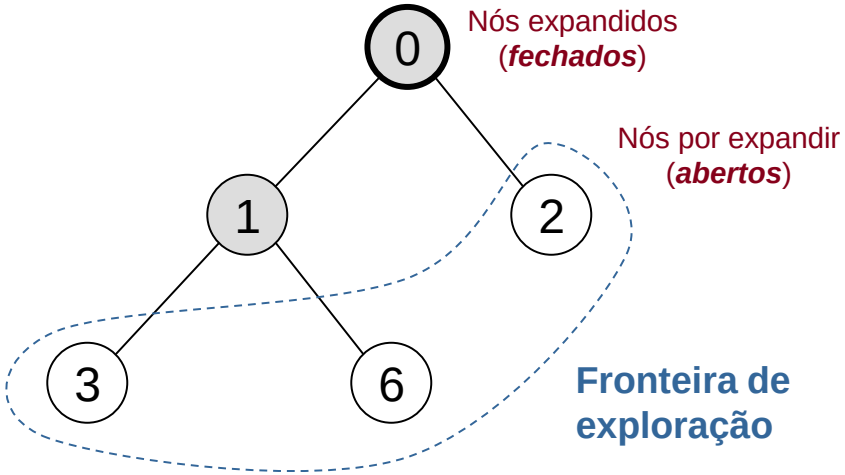
0 [ ]

[**1, 2**]

1 [2]

[2, **3, 6**]

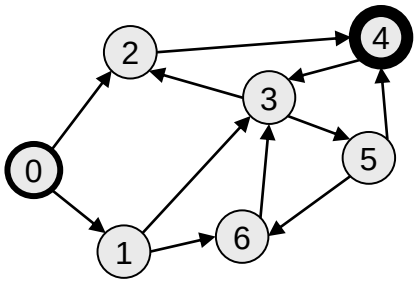
Árvore de Procura



MÉTODOS DE PROCURA

Procura em largura (*Breadth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no fim** da fronteira



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

[**1, 2**]

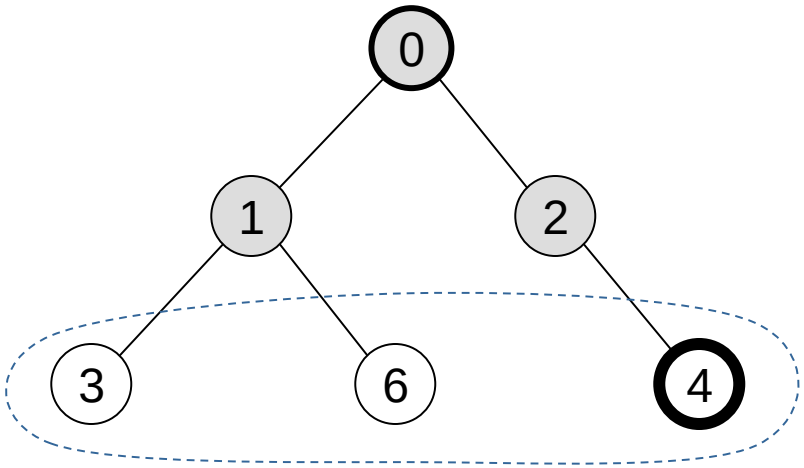
1 [2]

[2, **3, 6**]

2 [3,6]

[3, 6, **4**]

Árvore de Procura

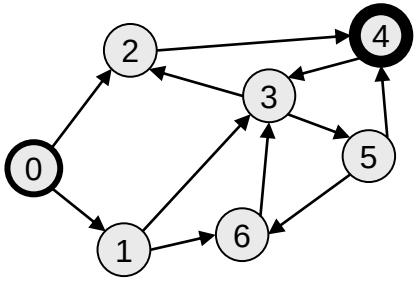


Fronteira de exploração

MÉTODOS DE PROCURA

Procura em largura (*Breadth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no fim** da fronteira



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

[**1, 2**]

1 [2]

[2, **3, 6**]

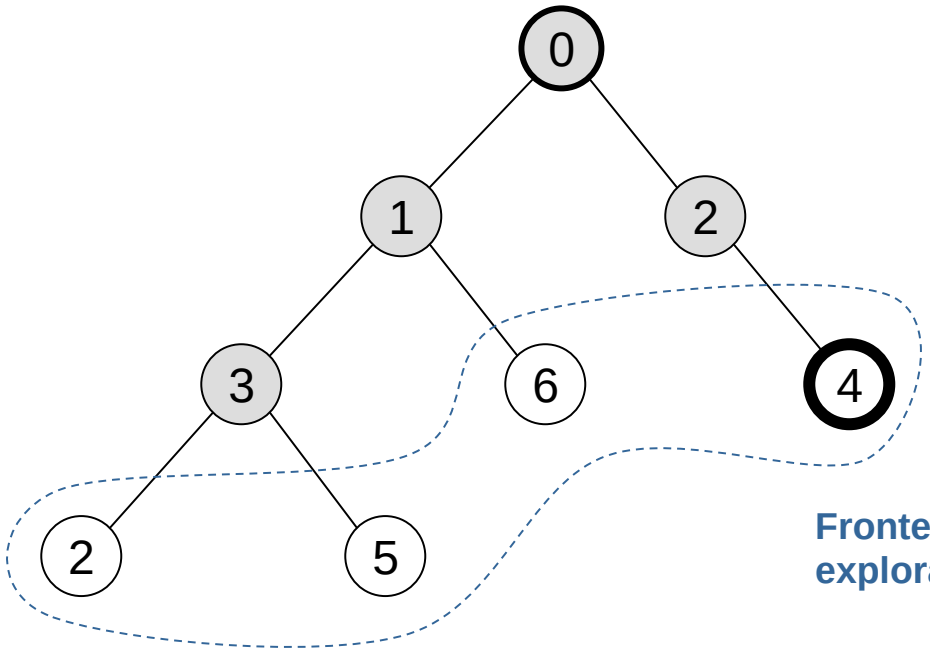
2 [3,6]

[3, 6, **4**]

3 [6,4]

[6,4,**2,5**]

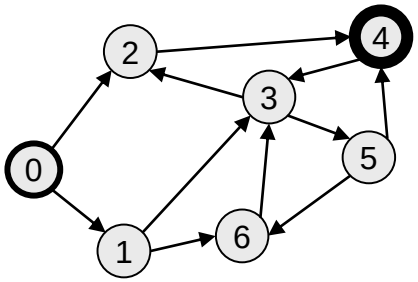
Árvore de Procura



MÉTODOS DE PROCURA

Procura em largura (*Breadth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**
    - Remoção no início da fronteira
    - Nós **mais recentes** são inseridos **no fim** da fronteira



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

[**1, 2**]

1 [2]

[2, **3, 6**]

2 [3,6]

[3, 6, **4**]

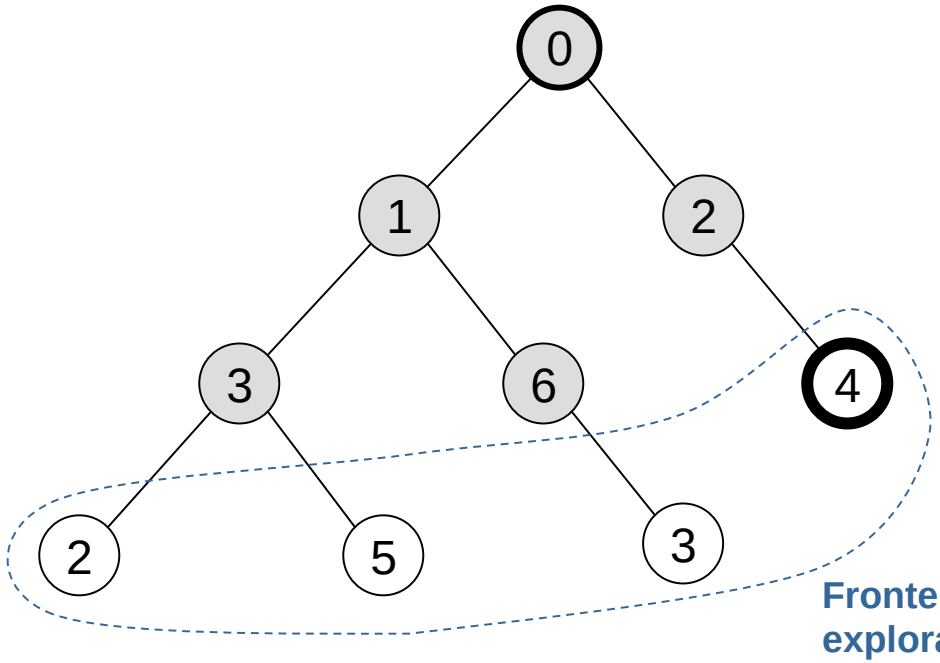
3 [6,4]

[6,4,**2,5**]

6 [4,2,5]

[4,2,5,**3**]

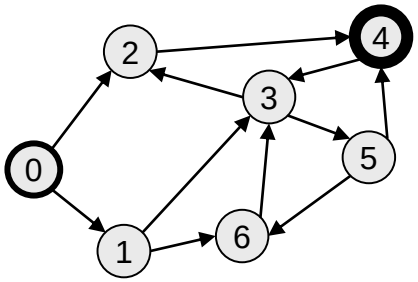
Árvore de Procura



MÉTODOS DE PROCURA

Procura em largura (*Breadth-First Search*)

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0]

0 [ ]

[**1, 2**]

1 [2]

[2, **3, 6**]

2 [3,6]

[3, 6, **4**]

3 [6,4]

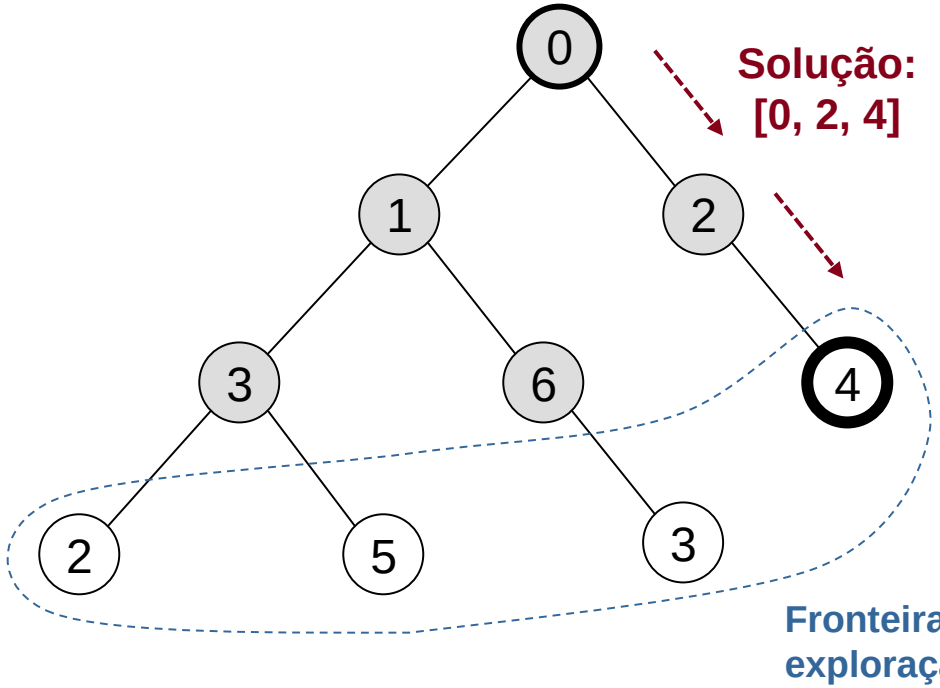
[6,4,**2,5**]

6 [4,2,5]

[4,2,5,**3**]

Estado objectivo → **4** [2,5,3]

Árvore de Procura

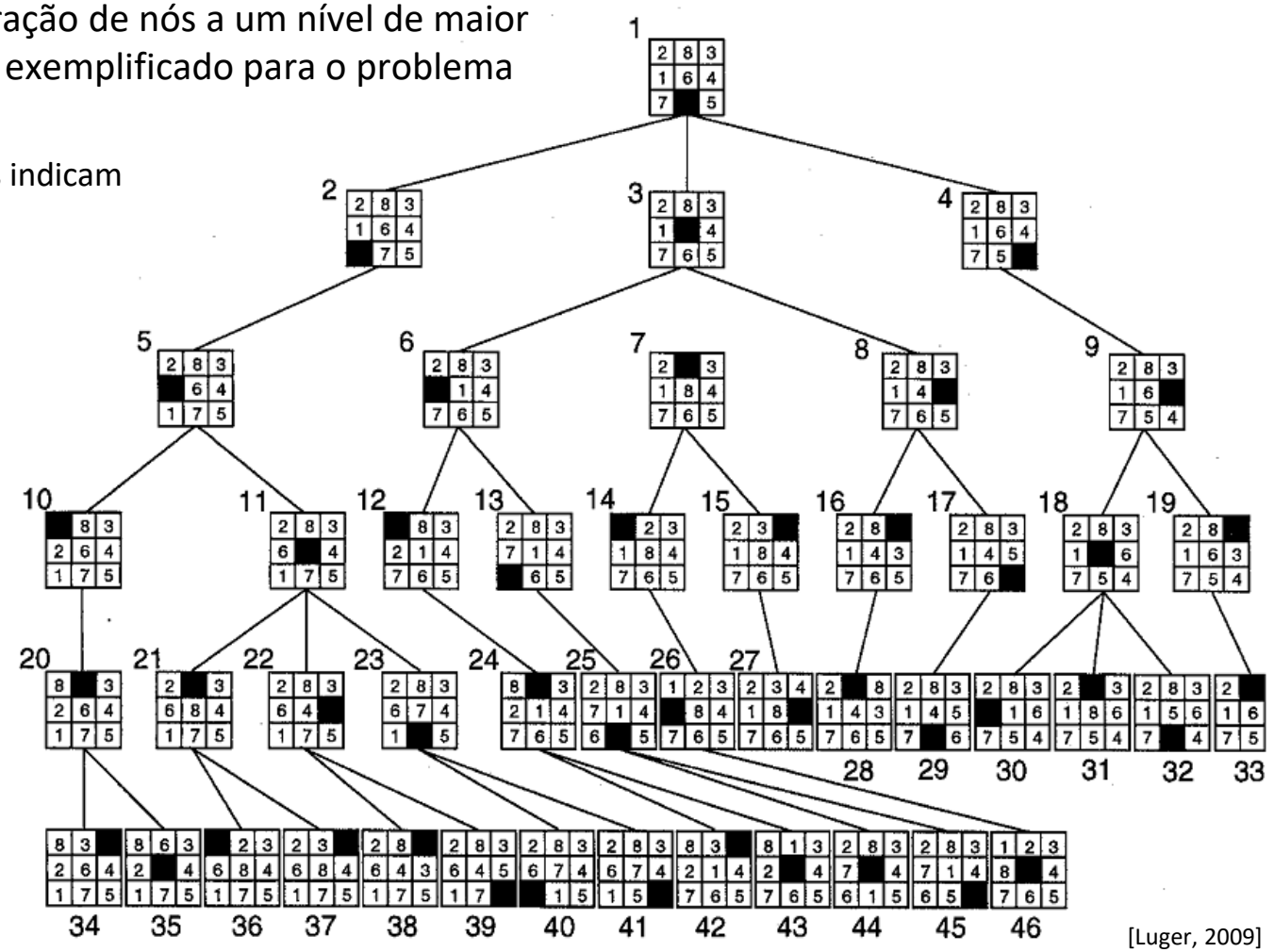


Procura termina: a solução corresponde à sequência de nós (estados e operadores) no ramo da árvore de procura que contém o nó correspondente ao estado objectivo

PROCURA EM LARGURA

Na procura em largura, a procura decorre **explorando os nós mais antigos primeiro** (primeiros a ser gerados), levando à **exploração exaustiva de cada nível de procura** antes da exploração de nós a um nível de maior profundidade, tal como exemplificado para o problema do puzzle de 8 peças.

Nota: Os índices numéricos indicam a ordem de exploração.



[Luger, 2009]

PROCURA EM GRAFOS COM CICLOS

ESTADOS REPETIDOS NA ÁRVORE DE PROCURA

- Grafo do espaço de estados apresenta ciclos
- Múltiplas transições para o mesmo estado
- Acções correspondentes às transições de estado são reversíveis

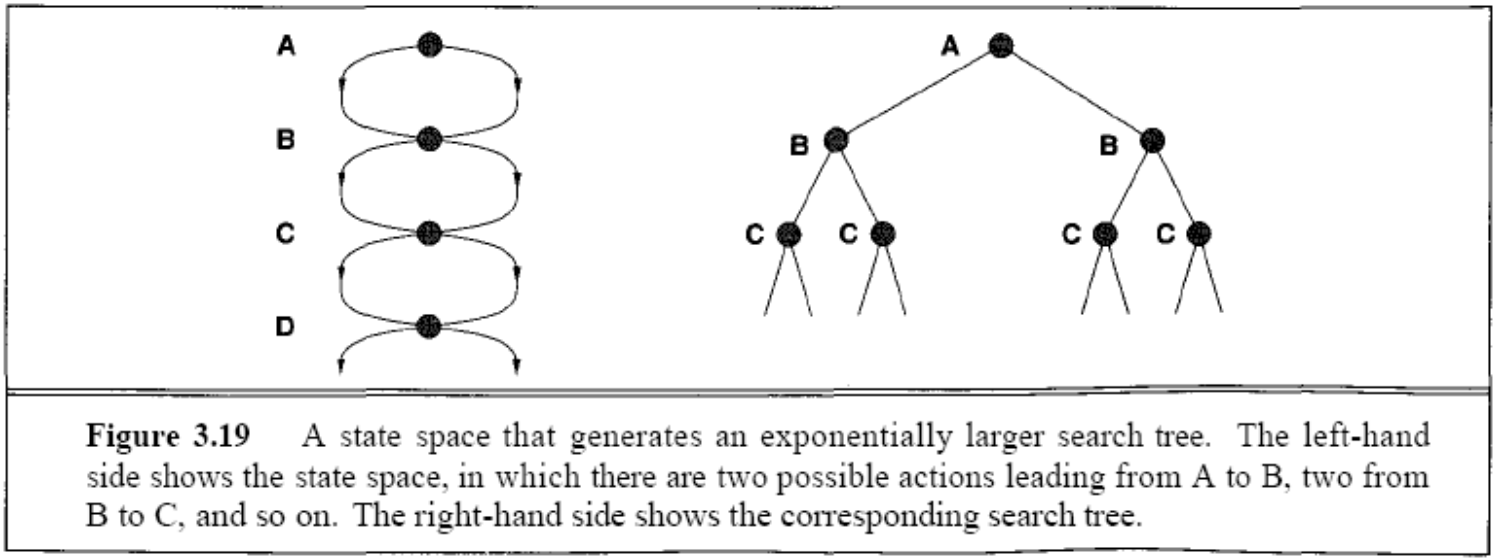


Figure 3.19 A state space that generates an exponentially larger search tree. The left-hand side shows the state space, in which there are two possible actions leading from A to B, two from B to C, and so on. The right-hand side shows the corresponding search tree.

[Russel & Norvig, 2003]

EXPANSÃO DE ESTADOS JÁ ANTERIORMENTE EXPLORADOS

- **Desperdício** de recursos (**tempo**, **memória**)

PROCURA GERAL EM GRAFOS

Para eliminação de nós correspondentes a estados repetidos, é necessário verificar se um novo *nó sucessor* corresponde a um estado que já foi anteriormente explorado, se isso acontecer, apenas o nó que corresponde ao percurso com **menor custo** deve **ser mantido**, o outro nó correspondente ao mesmo estado, mas num percurso com **maior custo** deve ser **eliminado**.

Para esta verificação, é necessário ter em conta que os nós podem estar pendentes para expansão (os nós existentes na fronteira de exploração), designados nós *abertos*, ou podem já ter sido expandidos, designados nós *fechados*.

- Nós ***Abertos***: nós gerados mas **não expandidos**
  - Fronteira de exploração
- Nós ***Fechados***: nós **expandidos**
- Ao gerar novo nó sucessor *noSuc* é necessário considerar:
  - $noSuc \notin Abertos \wedge noSuc \notin Fechados$
    - Inserir *noSuc* em *Abertos*
  - $noSuc \in Abertos$
    - Se *noSuc* foi atingido através de um caminho mais curto (com menor custo)
      - Remover nó anterior de *Abertos*
      - inserir *noSuc* em *Abertos*
  - $noSuc \in Fechados$
    - Se *noSuc* foi atingido através de um caminho mais curto (com menor custo)
      - Remover nó anterior de *Fechados*
      - inserir *noSuc* em *Abertos*

PROCURA EM GRAFOS COM CICLOS

Para facilitar o processamento dos nós repetidos, pode ser mantida uma única memória de nós **explorados** que inclui os nós abertos e os nós fechados.

Esta memória deve ser indexada por estado, de modo a possibilitar um acesso eficiente na verificação dos nós já explorados que, no caso geral, podem ser muito numerosos.

MEMÓRIA DE NÓS PROCESSADOS

- Nós gerados mas **não expandidos** (**fronteira** de exploração)
  - **ABERTOS**
- Nós **expandidos**
  - **FECHADOS**

EXPLORADOS

PROCURA EM LARGURA

Algoritmo de procura em largura

Processo de procura

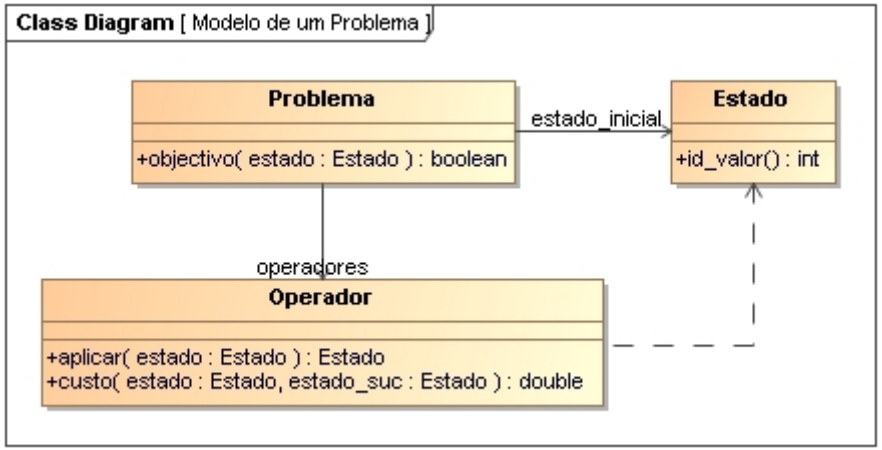
- Criar nó inicial
- Iniciar fronteira FIFO com nó
- Iniciar explorados
- Enquanto fronteira não vazia
  - Remover primeiro nó da fronteira
  - Verificar se estado do nó é objectivo, se for retornar solução que termina no nó
  - Expandir o nó
  - Por cada nó sucessor
    - Obter estado do nó
    - Se nó ainda não foi explorado
      - Juntar nó aos nós explorados
      - Inserir nó na fronteira
- Caso fronteira vazia indicar que não existe solução

```
1.  function procura_largura(problema) : Solucao
2.     no ← No(problema.estado_inicial)
3.     fronteira ← FronteiraFIFO(no)
4.     explorados ← {no.estado : no}
5.     while not fronteira.vazia do
6.        no ← fronteira.remover()
7.        if problema.objectivo(no.estado) then
8.           return Solucao(no)
9.        for no_sucessor in expandir(problema, no) do
10.          estado ← no_sucessor.estado
11.          if estado not in explorados then
12.             explorados[estado] ← no_sucessor
13.             fronteira.inserir(no_sucessor)
14.    return none
```

Perspectiva algorítmica → Realização como um mecanismo de procura

MODELO DE UM PROBLEMA

- **Estado**
  - Identificação única por valor (em função da informação de estado)
- **Operador**
  - Aplicado a um estado, gera um novo estado
  - Define custo de transição de estado
- **Problema**
  - Estado inicial
  - Operadores
  - Função de teste de objectivo

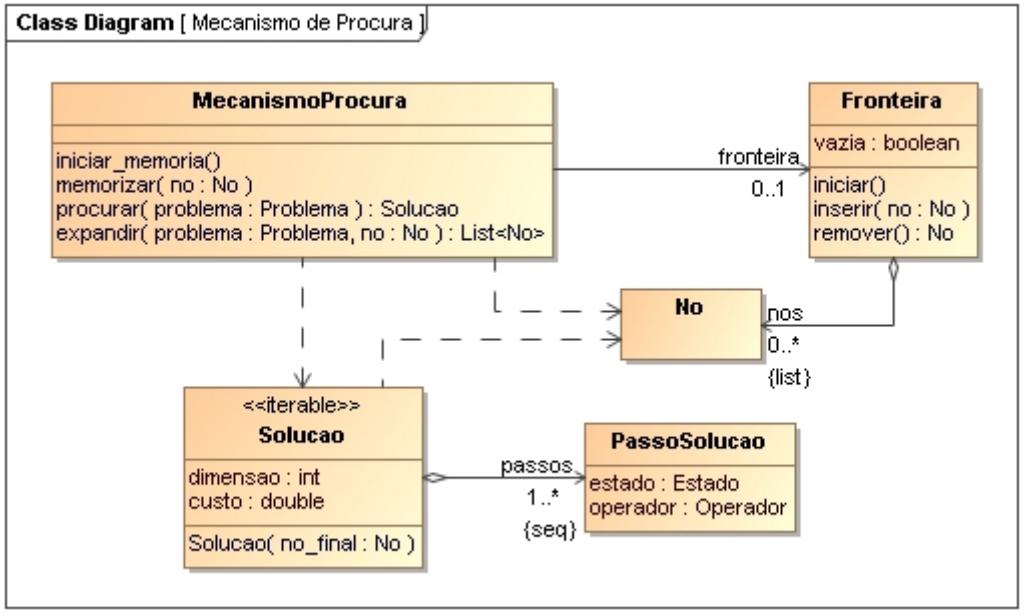


MECANISMO DE PROCURA

Permite procurar uma *solução* para um problema

Utiliza um *fronteira* de exploração para *memorizar* e gerir nós explorados

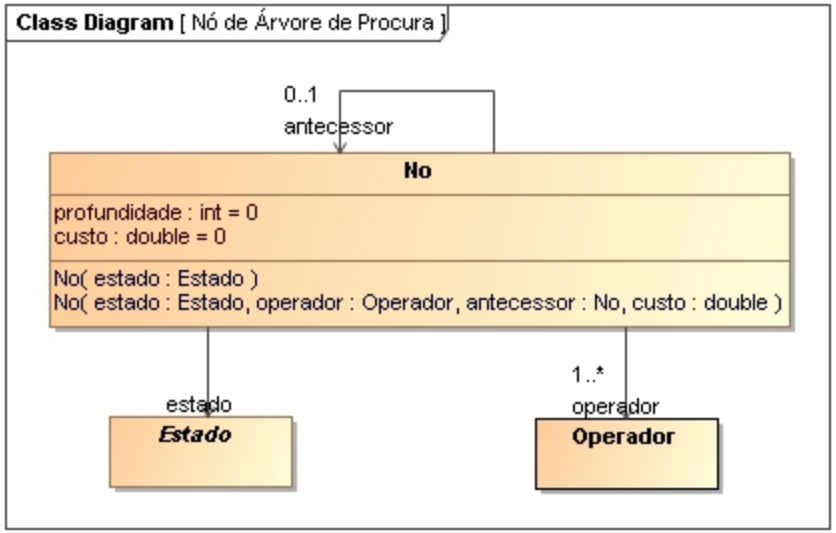
- **Fronteira**
  - Iniciar fronteira elimina todos os nós da fronteira
  - Permite inserir e remover nós de forma ordenada
  - Indica se a fronteira está vazia
- **Solução**
  - Representa um percurso correspondente a uma solução de um problema
    - Sequência de nós que representa um percurso no espaço de estados
    - Dimensão da solução (número de nós do percurso)
    - Permite acesso indexado e iteração sobre o percurso
    - Permite remover o primeiro nó do percurso



MECANISMO DE PROCURA

Árvore de procura organizada com base em *nós* de procura

- **Nó**
  - Elemento constituinte da árvore de procura, mantendo informação de:
    - **Estado**, a que corresponde o nó
    - **Operador**, que gerou o estado a que corresponde o nó (pode não existir)
    - **Antecessor**, nó antecessor na árvore de procura (pode não existir)
    - **Profundidade** do nó, na árvore de procura
    - **Custo** do percurso até ao nó
  - Comparável com outros nós em termos de custo

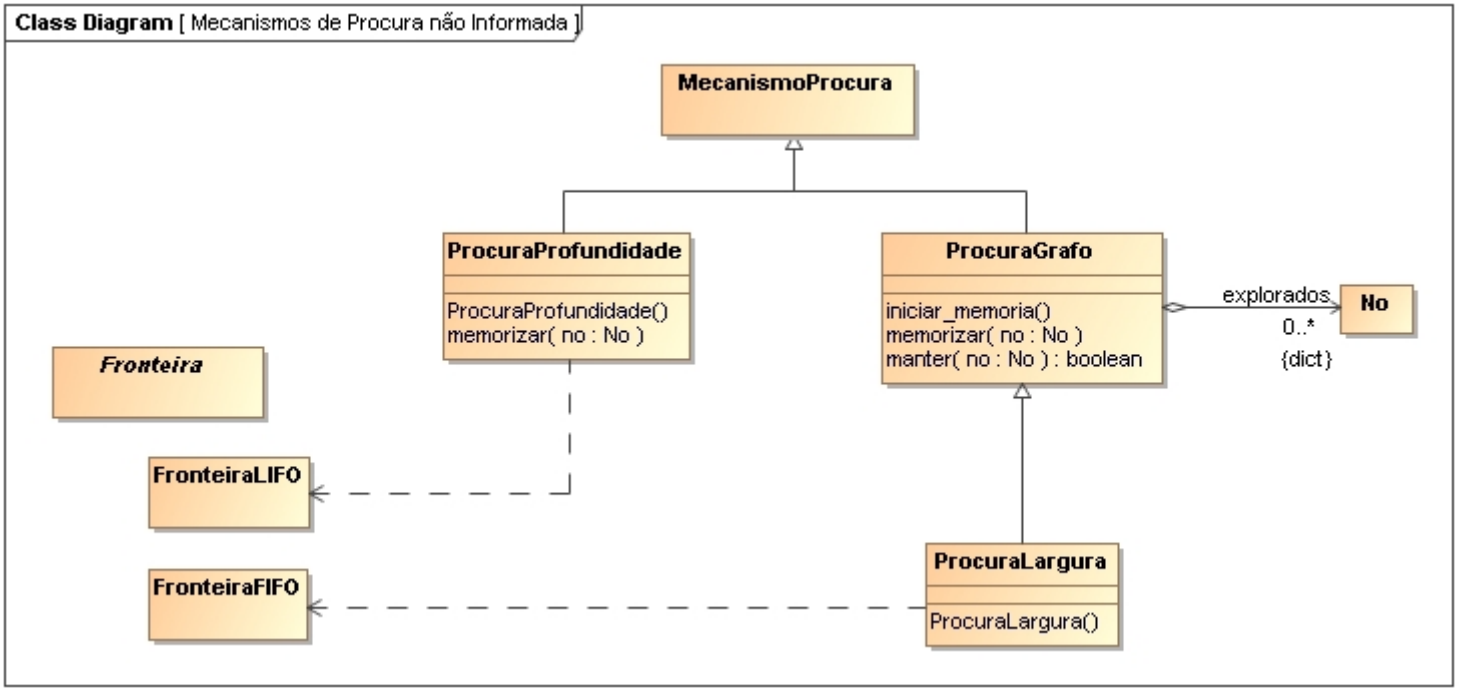


MECANISMO DE PROCURA

Mecanismos de procura não informada

(Não utilizam conhecimento do domínio do problema para guiar a procura)

- **Procura em profundidade**
  - Estratégia de controlo
    - Explorar primeiro os nós mais recentes
    - Utiliza uma fronteira do tipo LIFO
- **Procura em largura**
  - Estratégia de controlo
    - Explorar primeiro os nós mais antigos
    - Utiliza uma fronteira do tipo FIFO



BIBLIOGRAFIA

[Russel & Norvig, 2009]

S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd Edition, Prentice Hall, 2009

[Russel & Norvig, 2022]

S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th Edition, Pearson, 2022

[Nilsson, 1998]

N. Nilsson , *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann 1998

[Luger, 2009]

G. Luger , *Artificial Intelligence: Structures and Strategies for Complex Problem Solving* , Addison-Wesley, 2009

[Jaeger & Hamprecht, 2010]

M. Jaeger, F. Hamprecht, *Automatic Process Control for Laser Welding*, Heidelberg Collaboratory for Image Processing (HCI) , 2000