Escola Politécnica da USP - Depto. de Enga. Mecatrônica

PMR-3510 Inteligência Artificial

Aula 9- Busca X Inferência Lógica

Prof. José Reinaldo Silva

reinaldo@usp.br

d-l@b

Cronograma de entrega do trabalho em grupo

O cronograma de trabalho será dividido em "milestones": **o primeiro trabalho consiste em desenvolver um programa em Prolog para resolver o mundo de blocos.**

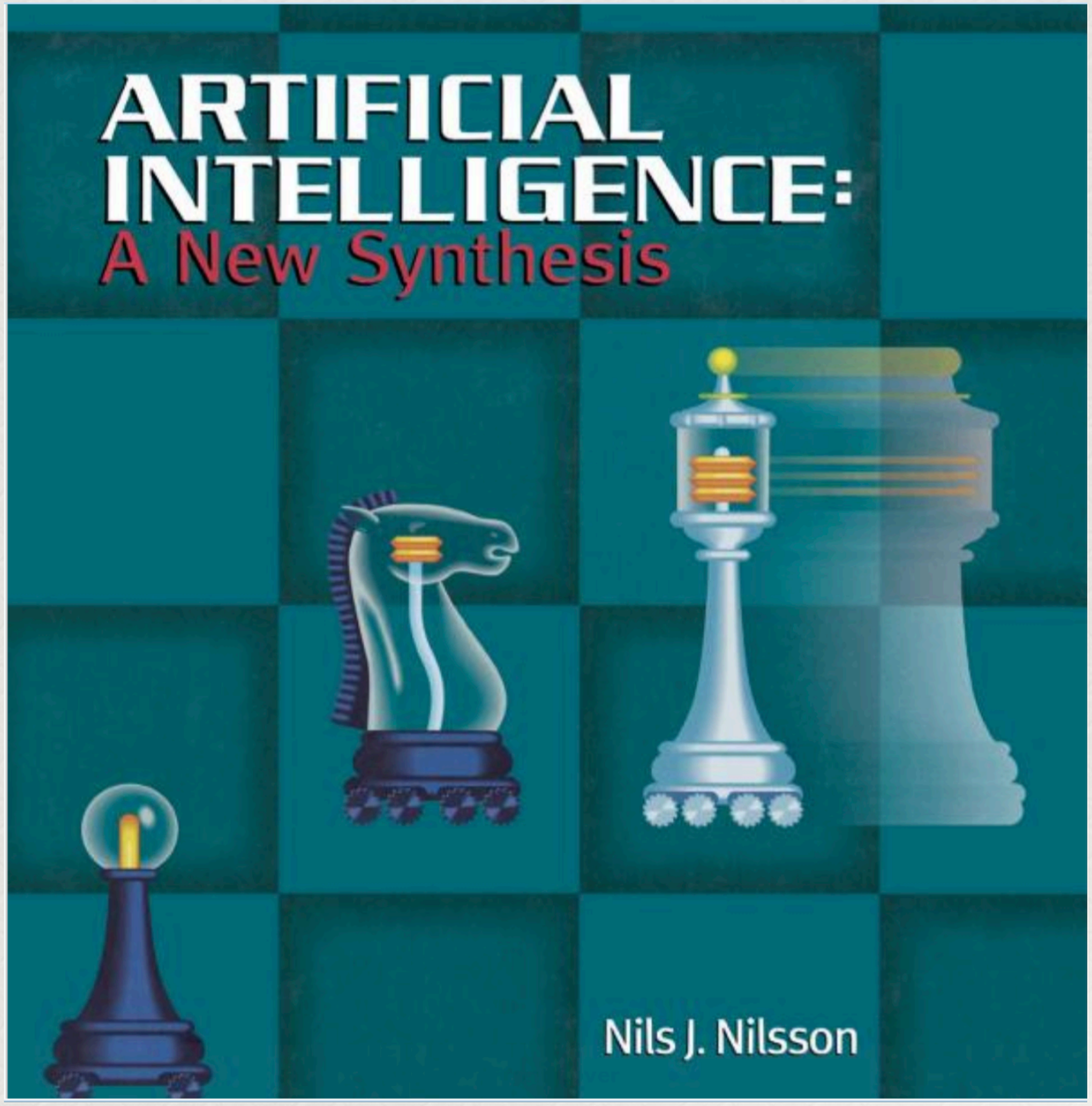
25/09 - início do processo, definição do domínio

02/10 - definição do algoritmo e implementação

16/10 - entrega do programa final

23/10 - competição

Onde estamos?



Onde estamos?

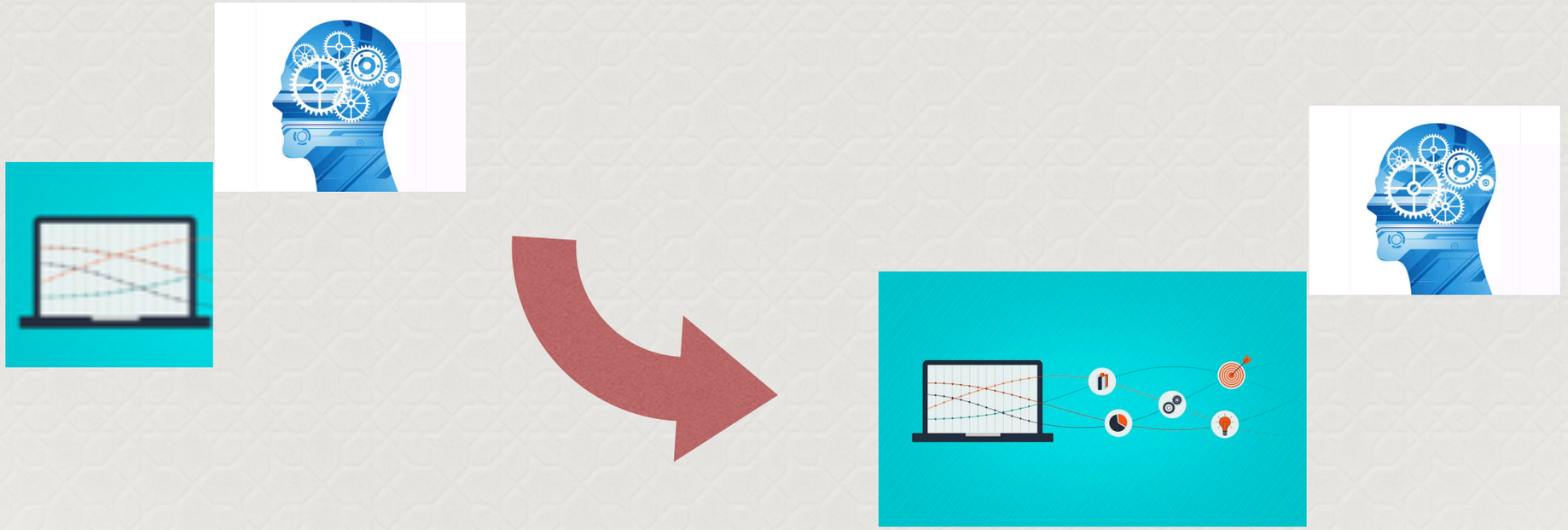
Onde estamos?

O que vem depois?

Cap. 10 e parte dos capítulos 13 e 14

Resolução de problemas com IA

Existem duas maneiras de resolver problemas usando IA: uma é algorítmica, usando busca (clássica ou informada); a outra é usando inferência lógica.



Robô Valkyrie mede 1,8 metro e pesa 125 quilos. (Foto: Nasa)

métodos de busca clássica e informada

programação lógica

métodos de inferência

Estratégias de Busca Informada

Análise de custo

$f(x) = g(x)$

Heurística

$f(x) = h(x)$

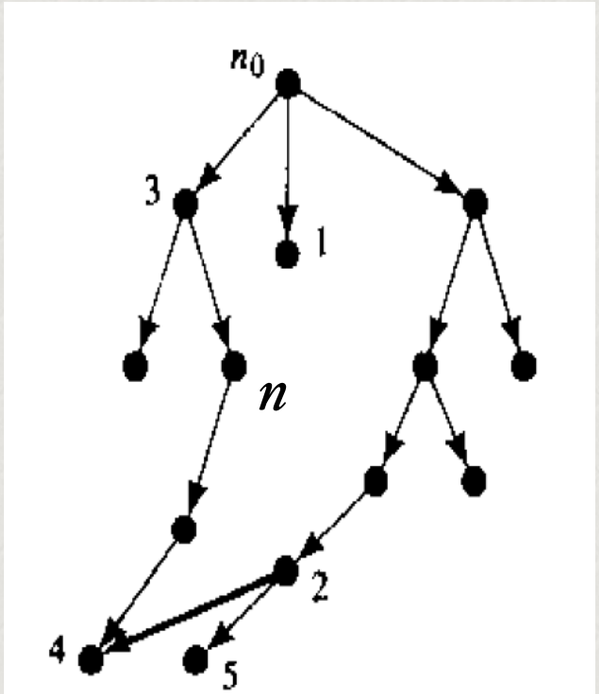
Algorítmo A*

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

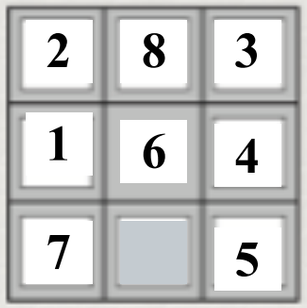
Análise de custo	Heurística
f(x) = g(x)	f(x) = h(x)

Algoritmo A*

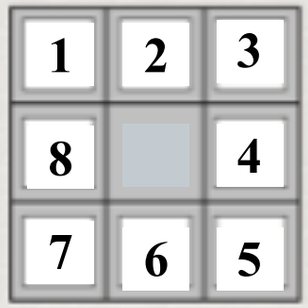
$$f(x) = g(x) + h(x)$$



A função heurística em um estado n atingível a partir de n_0 é a composição do custo g(x) para atingir o nó n, e do custo estimado h(x) para chegar ao estado objetivo.



f(0) = 0 + 4

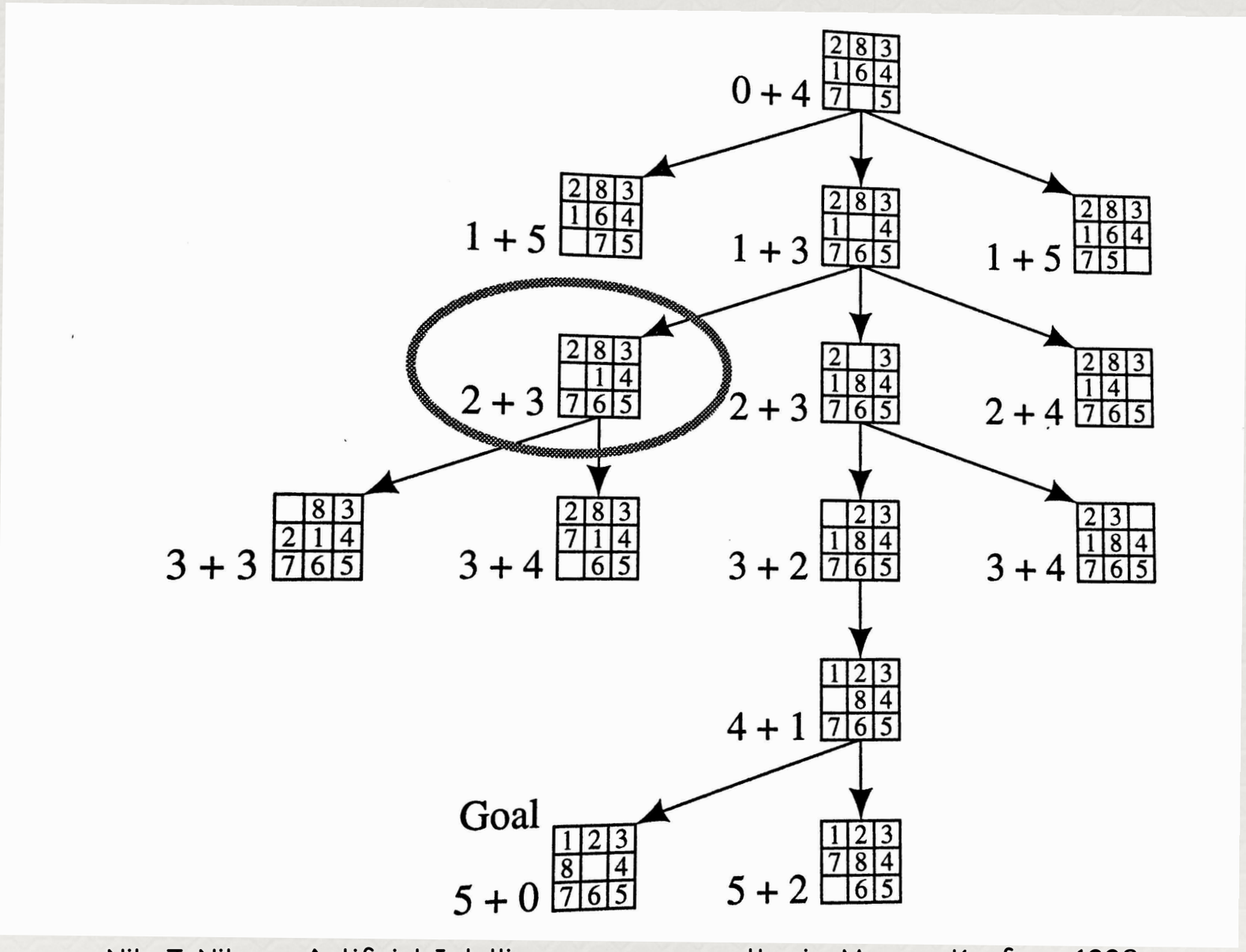


estado final

comparando o estado inicial (a raiz da árvore de busca) com o estado final mostrado à direita, temos que os tiles 1, 2, 8 e 6 estão fora do lugar, portanto neste estado (o nível zero ou raiz, está a uma distância avaliada pelo número mínimo de movimentos para atingir o estado final) h(0) = 4, e a função de avaliação é f(0) = 0 + 4 = 4

Nils J. Nilsson, Artificial Intelligence: a new synthesis, Morgan Kaufann, 1998

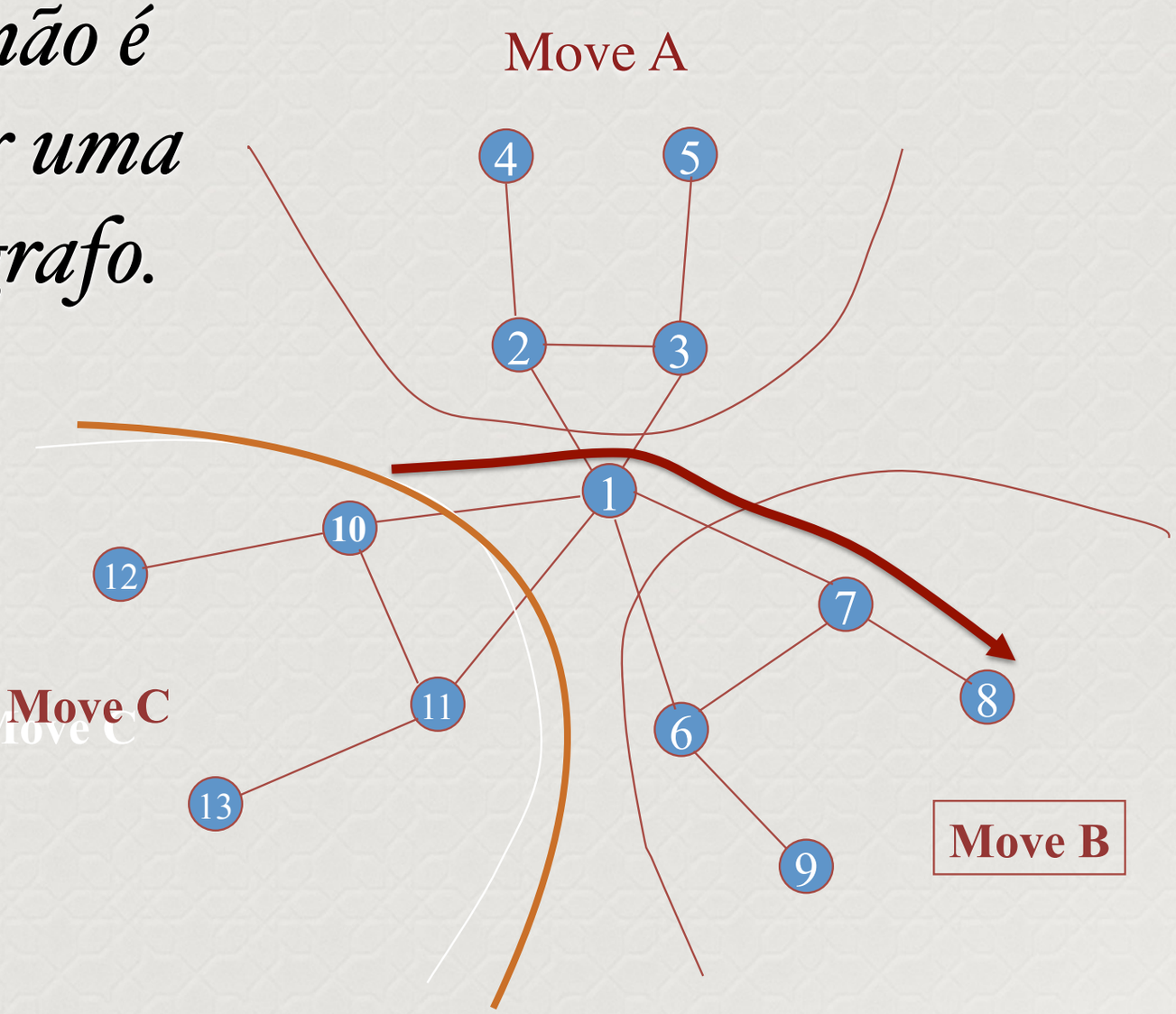
Árvore de busca e a busca informada



Nils J. Nilsson, Artificial Intelligence: a new synthesis, Morgan Kaufann, 1998

Até aqui estamos supondo que a busca é sempre feita sobre uma árvore, como no exemplo acima. Entretanto, para o mundo de blocos...

...o espaço de estados não é mais caracterizado por uma árvore e sim por um grafo.



*Vamos portanto generalizar o algoritmo A**

Vamos construir uma árvore de busca que pode servir para todos os algoritmos de busca que vimos até aqui.

1. Crie uma árvore de busca Tr que inicialmente contém apenas um nó representando o estado inicial n_0. Coloque n_0 em uma lista chamada OPEN;
2. Crie uma outra lista chamada CLOSED inicialmente vazia;
3. Se OPEN estiver vazia então a busca falha e o processo se encerra.
4. Selecione o primeiro nó em OPEN, retire da list e coloque em CLOSED. Este nó n será usado para expandir a árvore de busca;
5. Se o nó n é o estado alvo a busca se encerra com sucesso. O "plano" pode ser visto resgatando o "stack" CLOSED.

6. Expanda o nó n gerando o conjunto de sucessores, que deve ser colocado em OPEN;

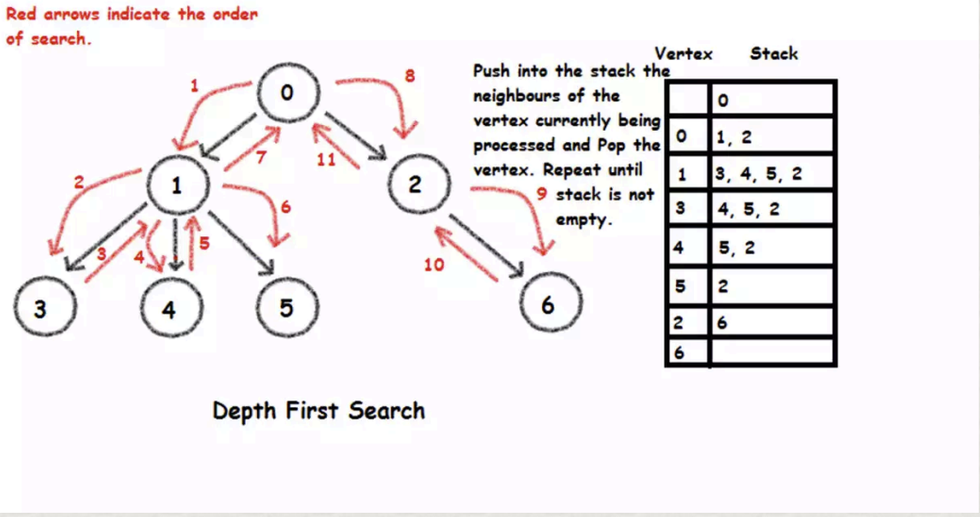
7. Ordene a lista em OPEN de acordo com algum critério ou com a avaliação da heurística;

8. Vá para o passo 3.

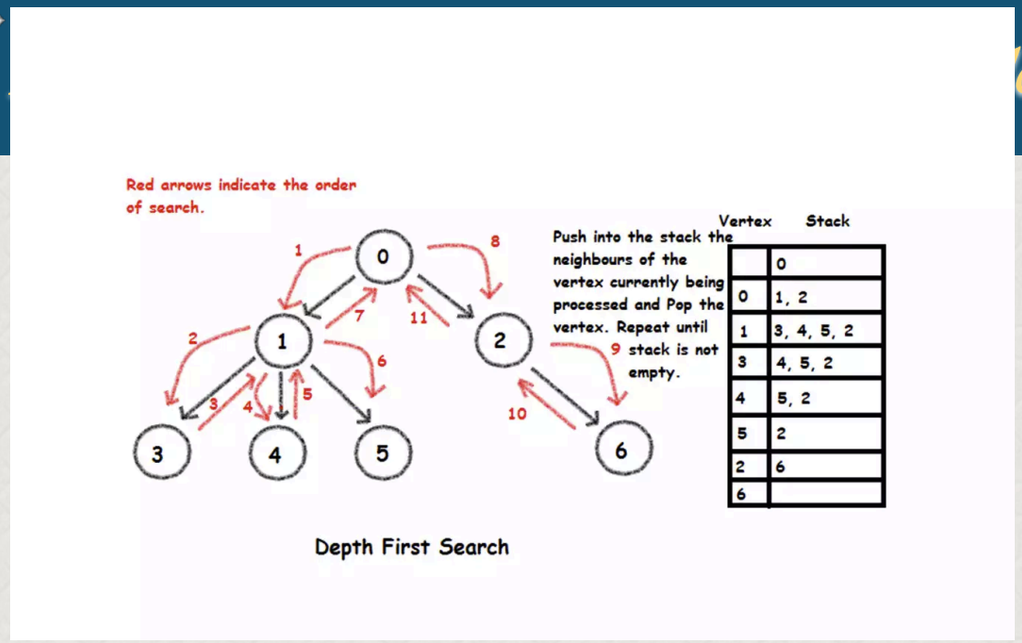
Com pequenas mudanças este mesmo algoritmo pode ser usado para os principais métodos de busca que vimos até agora: busca em profundidade (deph first), busca em largura (breadth first), e o A (best first).*

Busca em profundidade

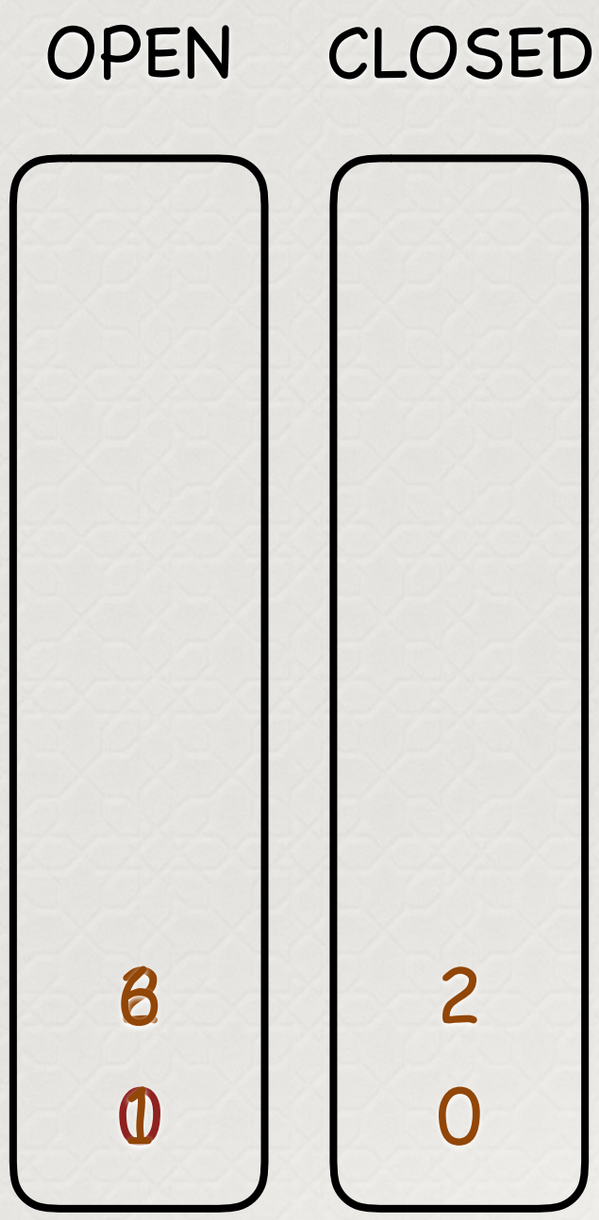
7. Ordene a lista em OPEN de acordo com algum critério ou com a avaliação da heurística;



No caso da busca em profundidade basta remover o passo 7 para não ordenar a lista OPEN.

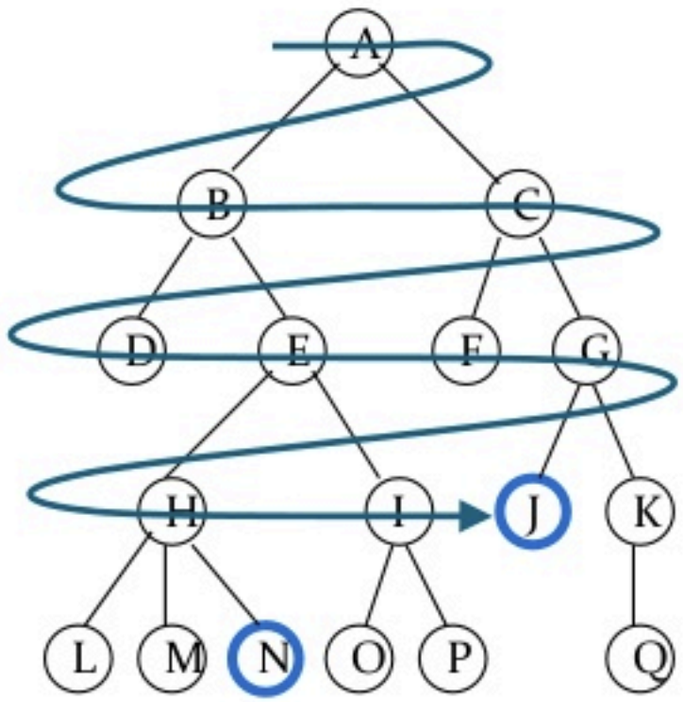


1. Crie uma árvore de busca Tr que inicialmente contém apenas um nó representando o estado inicial n_0. Coloque n_0 em uma lista chamada OPEN;
2. Crie uma outra lista chamada CLOSED inicialmente vazia;
3. Se OPEN estiver vazia então a busca falha e o processo se encerra.
4. Selecione o primeiro nó em OPEN, retire da list e coloque em CLOSED. Este nó n será usado para expandir a árvore de busca;
5. Se o nó n é o estado alvo a busca se encerra com sucesso. O "plano" pode ser visto resgatando o "stack" CLOSED.
6. Expanda o nó n gerando o conjunto de sucessores, que deve ser colocado em OPEN;
7. Ordene a lista em OPEN de acordo com algum critério ou com a avaliação da heurística;
8. Vá para o passo 3.

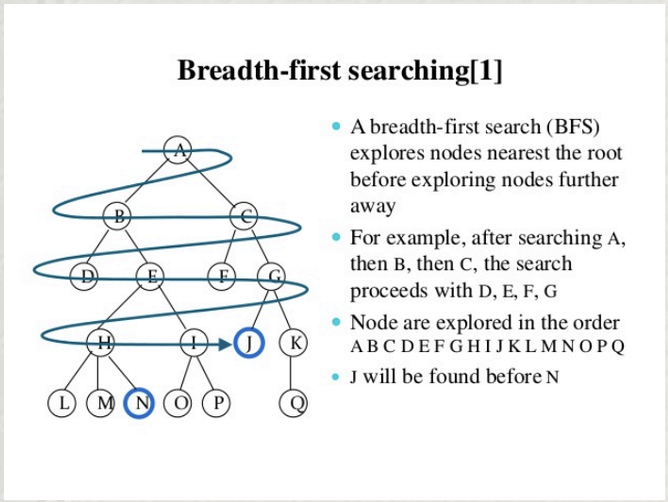


Busca em largura

Breadth-first searching[1]



- A breadth-first search (BFS) explores nodes nearest the root before exploring nodes further away
- For example, after searching A, then B, then C, the search proceeds with D, E, F, G
- Node are explored in the order A B C D E F G H I J K L M N O P Q
- J will be found before N



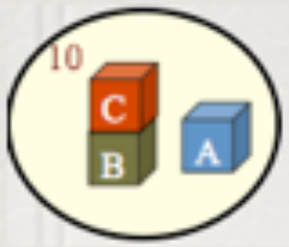
Busca em largura

1. Crie uma árvore de busca Tr que inicialmente contém apenas um nó representando o estado inicial n_0. Coloque n_0 em uma lista chamada OPEN;
2. Crie uma outra lista chamada CLOSED inicialmente vazia;
3. Se OPEN estiver vazia então a busca falha e o processo se encerra.
4. Selecione o primeiro nó em OPEN, retire da list e coloque em CLOSED. Este nó n será usado para expandir a árvore de busca;
5. Se o estado alvo é um dos nós em CLOSED a busca se encerra com sucesso. O "plano" pode ser visto resgatando o "stack" CLOSED.
6. Expanda o nó n gerando o conjunto de sucessores, que deve ser colocado em OPEN;
7. Ordene a lista em OPEN de acordo com algum critério ou com a avaliação da heurística;
8. Vá para o passo 3.

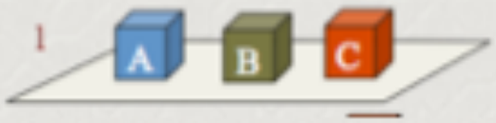
A*

1. Crie uma árvore de busca Tr que inicialmente contém apenas um nó representando o estado inicial n_0. Coloque n_0 em uma lista chamada OPEN;
2. Crie uma outra lista chamada CLOSED inicialmente vazia;
3. Se OPEN estiver vazia então a busca falha e o processo se encerra.
4. Selecione o primeiro nó em OPEN, retire da list e coloque em CLOSED. Este nó n será usado para expandir a árvore de busca;
5. Se o nó n é o estado alvo a busca se encerra com sucesso. O "plano" pode ser visto resgatando o "stack" CLOSED.
6. Expanda o nó n gerando o conjunto de sucessores, que deve ser colocado em OPEN;
7. Ordene a lista em OPEN de acordo com algum critério ou com a avaliação da heurística;
8. Vá para o passo 3.

No caso do mundo de blocos cada "estado" é representado por uma descrição da configuração e não por um número.



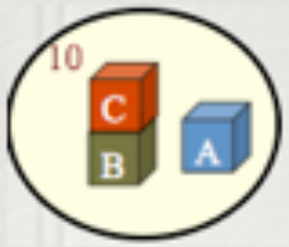
sobre(c, b)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(a)

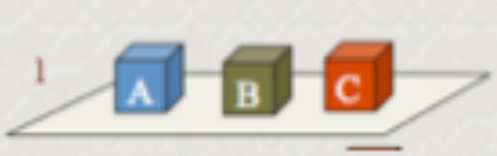


sobre(c, mesa)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(b)
livre(a)

Portanto, "expandir" o nó 10 mostrado abaixo significa verificar a aplicabilidade dos operadores, como o move(c,mesa), mostrado abaixo.

move(c, mesa)

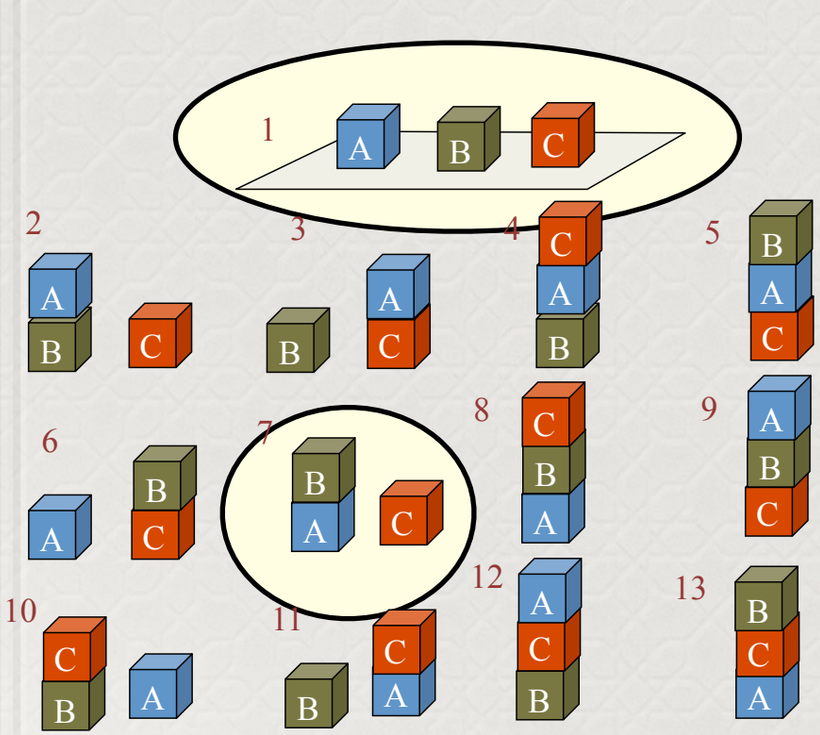


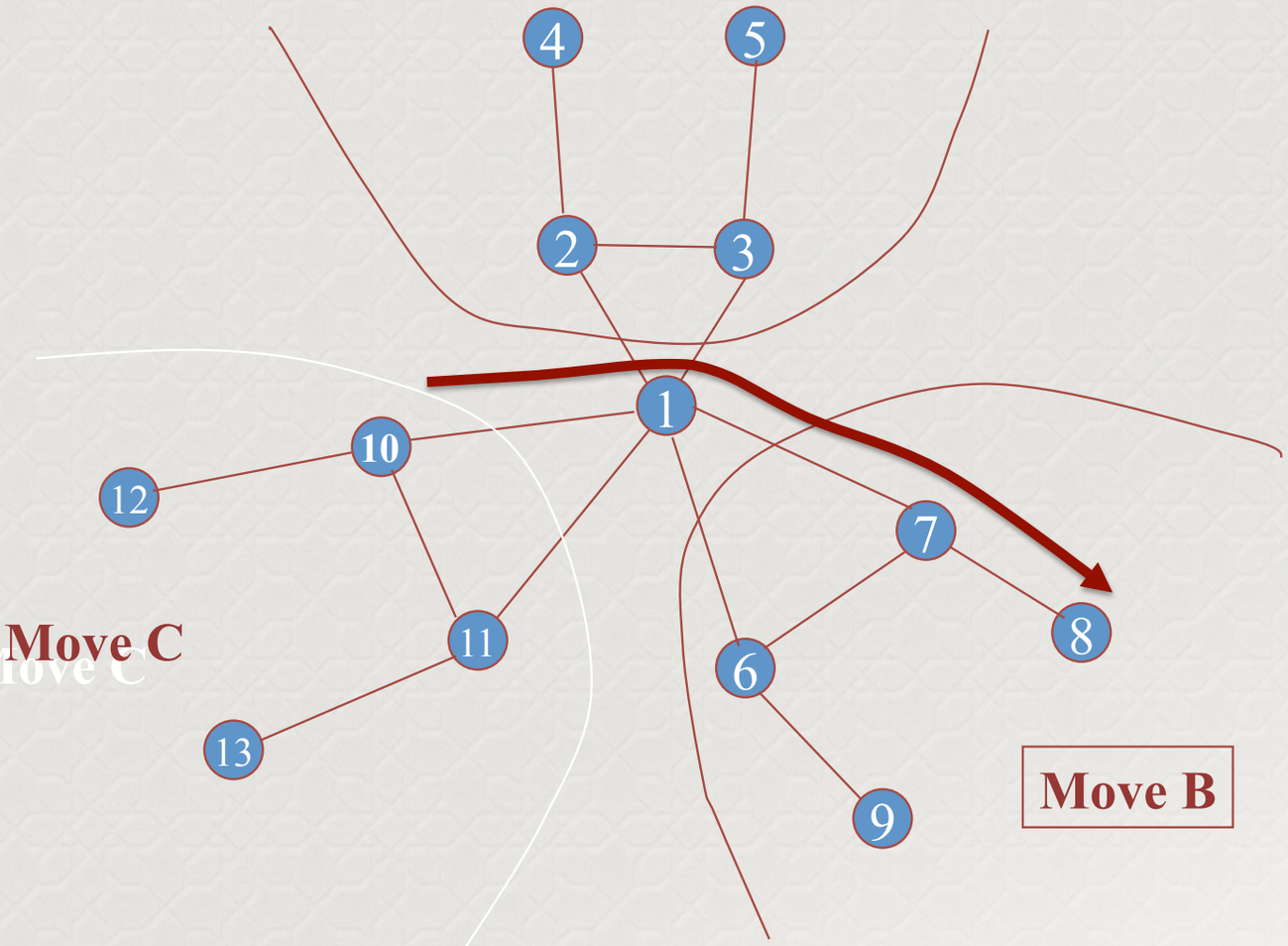


sobre(c, b)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(a)

sobre(c, mesa)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(b)
livre(a)

Na representação dos estados estamos fazendo um passagem do estado 7 para o estado 1 do mundo de blocos.






```
init_state([on(b,a), on(a,table), on(c,table), free(b), free(c)]).

stack_plan([]).

move(X,table,Z,S):- subset([on(X, R)],Z),
    remove([on(X,R)],Z,W),
    add([on(X,table),free(R)], W, S).
move(X,Y,Z,S):- subset([free(X)],Z), subset([free(Y)],Z),
    subset([on(X,R)],Z),
    remove([free(Y),on(X,R)],Z,W),
    add([on(X,Y), free(R)], W, S).

make_plan(X,X).
make_plan(P1,P2):-  next_state(P1,Z), ..., make_plan(Z,P2).

remove(M,N,K):- subtract(N,M,K).
add(M,N,K) :- append(M,N,K).
```

swish.swi-prolog.org/example/examples.swinb

AliExpress | Booking.com | Dafiti | Americanas | Facebook | Getting Started

SWISH File Edit Examples Help

134 users online

Search

examples | Program

```
1  init_state([on(b,a), on(a,table), on(c,table), free(b), free(c)]).
2
3  stack_plan([]).
4
5  move(X,table,Z,S):- subset([on(X, R)],Z),
6      remove([on(X,R)],Z,W),
7      add([on(X,table),free(R)], W, S).
8  move(X,Y,Z,S):- subset([free(X)],Z), subset([free(Y)],Z),
9      subset([on(X,R)],Z),
10     remove([free(Y),on(X,R)],Z,W),
11     add([on(X,Y), free(R)], W, S).
12
13 make_plan(X,X).
14 make_plan(P1,P2):-  next_state(P1,Z), ..., make_plan(Z,P2).
15
16 remove(M,N,K):- subtract(N,M,K).
17 add(M,N,K) :- append(M,N,K).
18
19
```

init_state(X), *move*(b,table,X,S).

S = [*on*(b, (table)), *free*(a), *on*(a, (table)), *on*(c, (table)), *free*(b), *free*(c)],
X = [*on*(b, a), *on*(a, (table)), *on*(c, (table)), *free*(b), *free*(c)]

Next | 10 | 100 | 1,000 | Stop

?- `init_state(X), move(b,table,X,S).`

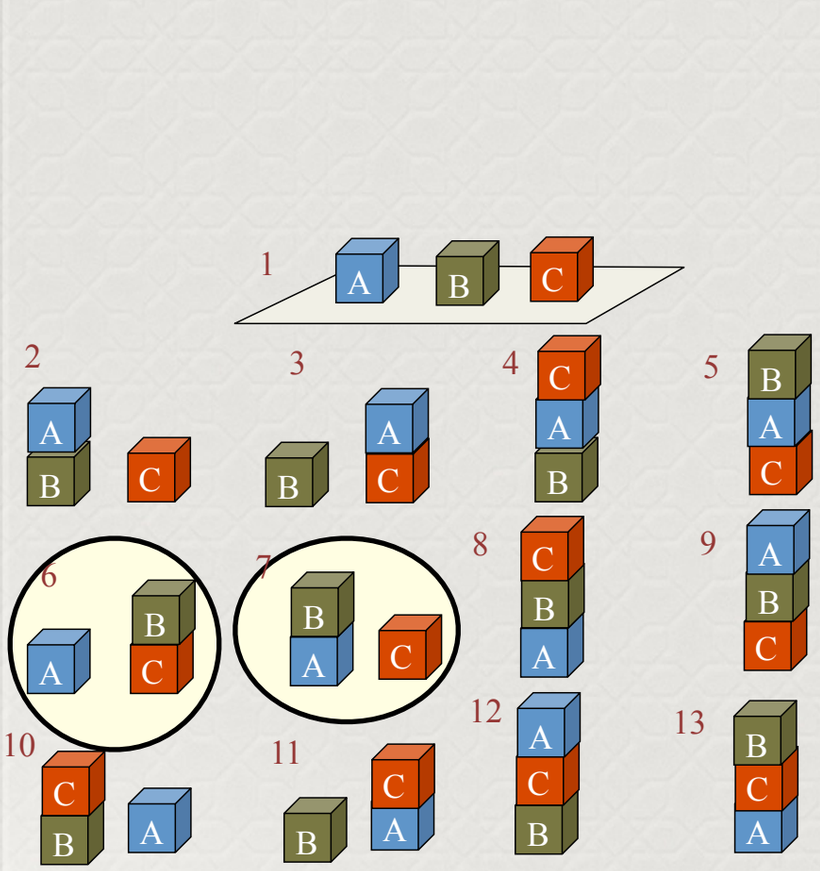
Examples | History | Solutions | table results | Run!

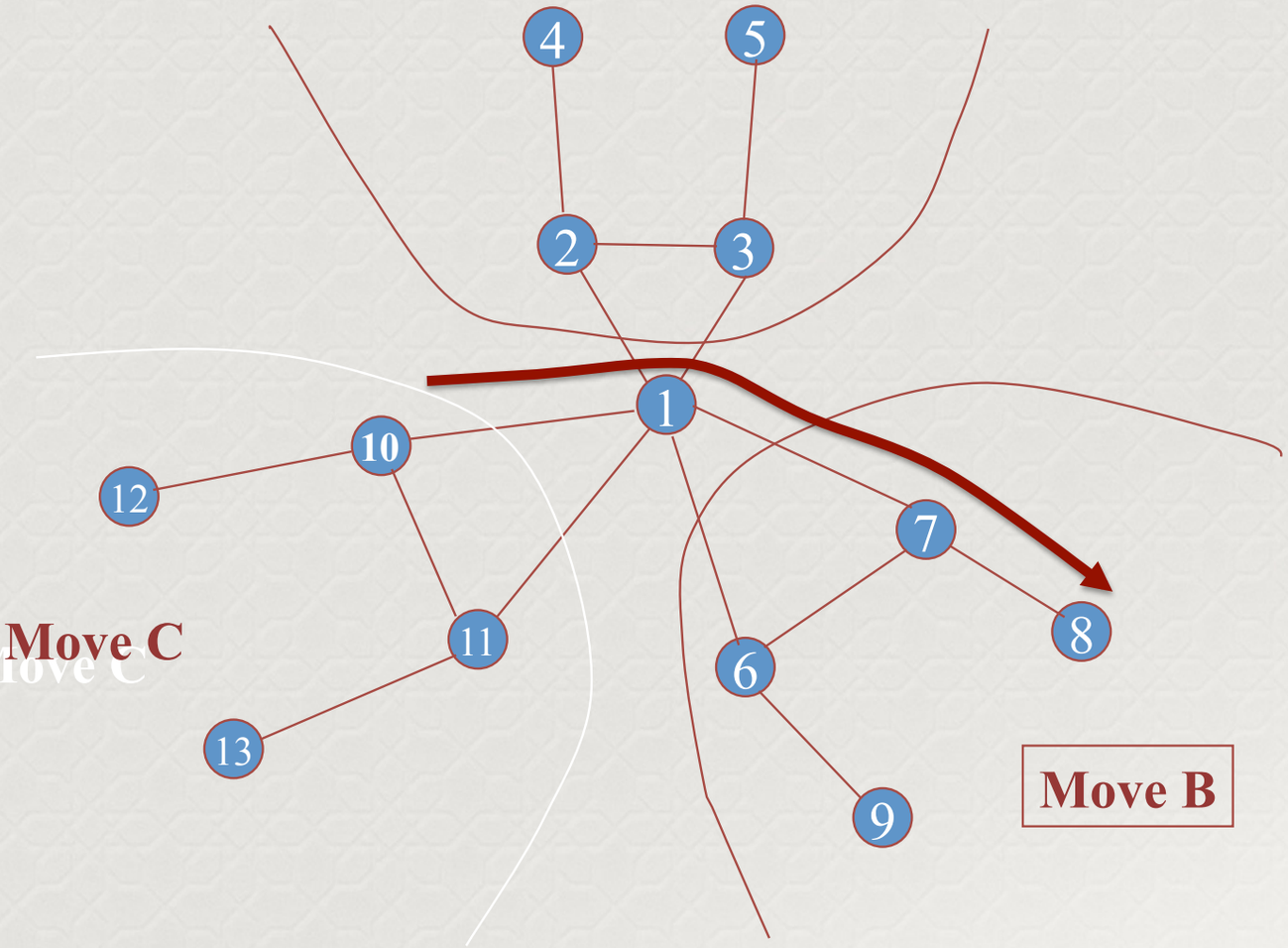
init_state(X), *move*(b,table,X,S).

S = [*on*(b, (table)), *free*(a), *on*(a, (table)), *on*(c, (table)), *free*(b), *free*(c)],
X = [*on*(b, a), *on*(a, (table)), *on*(c, (table)), *free*(b), *free*(c)]

```
?- init_state(X), move(b,table,X,S).
```

Na representação dos estados estamos fazendo um passagem do estado 7 para o estado 6 do mundo de blocos.






```
init_state([on(b,a), on(a,table), on(c,table), free(b), free(c)]).

stack_plan([]).

move(X,table,Z,S):- subset([on(X, R)],Z),
    remove([on(X,R)],Z,W),
    add([on(X,table),free(R)], W, S).
move(X,Y,Z,S):- subset([free(X)],Z), subset([free(Y)],Z),
    subset([on(X,R)],Z),
    remove([free(Y),on(X,R)],Z,W),
    add([on(X,Y), free(R)], W, S).

make_plan(X,X).
make_plan(P1,P2):-  next_state(P1,Z), ..., make_plan(Z,P2).

remove(M,N,K):- subtract(N,M,K).
add(M,N,K) :- append(M,N,K).
```

swish.swi-prolog.org/example/examples.swinb

AliExpress | Booking.com | Dafiti | Americanas | Facebook | Getting Started

SWISH File Edit Examples Help

134 users online

Search

examples | Program

```
1  init_state([on(b,a), on(a,table), on(c,table), free(b), free(c)]).
2
3  stack_plan([]).
4
5  move(X,table,Z,S):- subset([on(X, R)],Z),
6      remove([on(X,R)],Z,W),
7      add([on(X,table),free(R)], W, S).
8  move(X,Y,Z,S):- subset([free(X)],Z), subset([free(Y)],Z),
9      subset([on(X,R)],Z),
10     remove([free(Y),on(X,R)],Z,W),
11     add([on(X,Y), free(R)], W, S).
12
13 make_plan(X,X).
14 make_plan(P1,P2):-  next_state(P1,Z), ..., make_plan(Z,P2).
15
16 remove(M,N,K):- subtract(N,M,K).
17 add(M,N,K) :- append(M,N,K).
18
19
```

init_state(X), *move*(b,c,X,S).

S = [*on*(b, c), *free*(a), *on*(a, (table)), *on*(c, (table)), *free*(b)],
X = [*on*(b, a), *on*(a, (table)), *on*(c, (table)), *free*(b), *free*(c)]

?- init_state(X), move(b,c,X,S).

Examples | History | Solutions | table results | Run!

init_state(X), *move*(b,c,X,S).

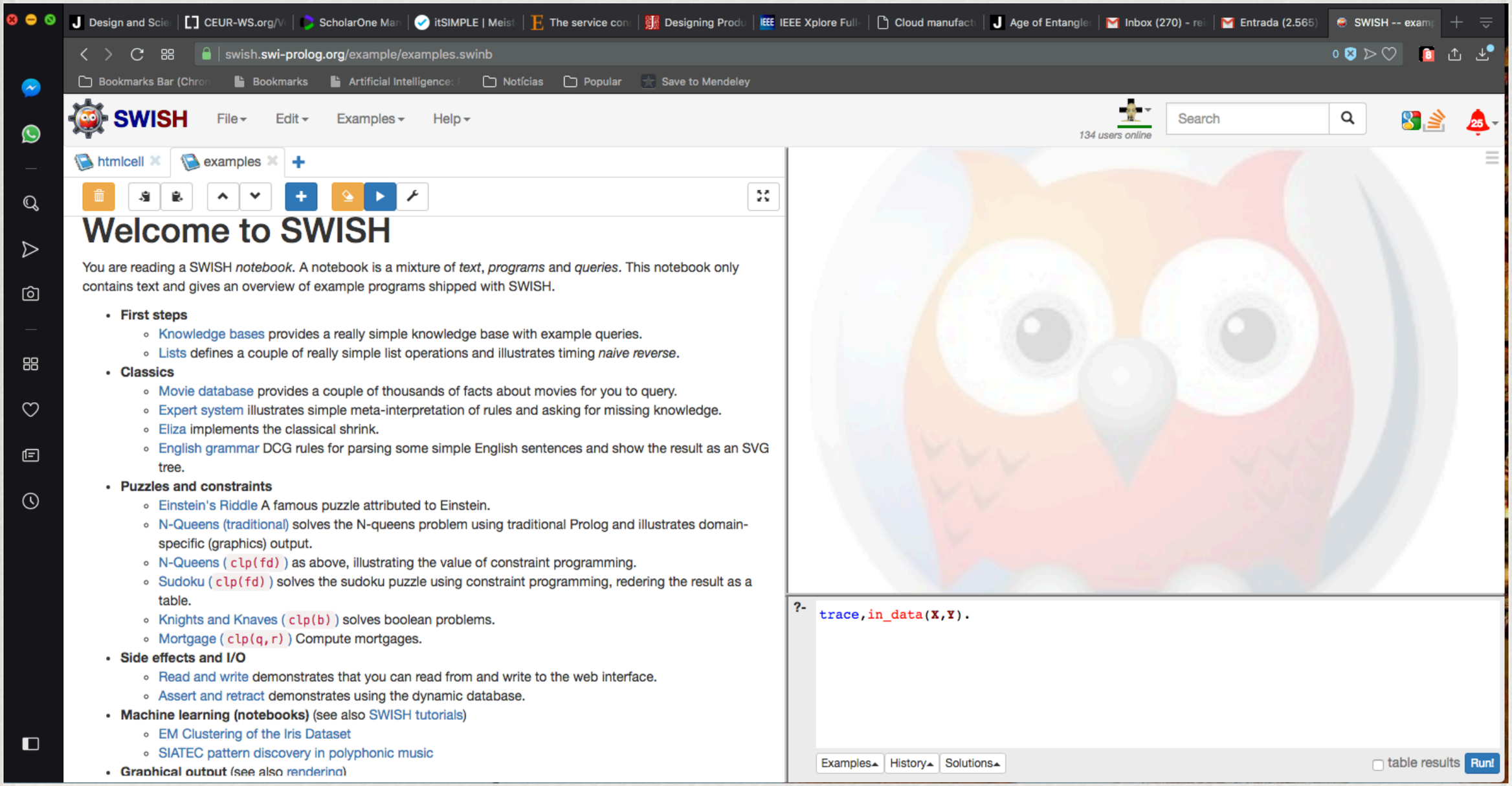
S = [*on*(b, c), *free*(a), *on*(a, (table)), *on*(c, (table)), *free*(b)],
X = [*on*(b, a), *on*(a, (table)), *on*(c, (table)), *free*(b), *free*(c)]

```
?- init_state(X), move(b,c,X,S).
```

Na aula que vem vamos discutir mais sobre "planning" e suas aplicações generalizando o processo à partir do STIPS.

A competição entre equipes

No dia 23, teremos a competição onde todos os grupos deverão usar os respectivos programas para resolver 3 problemas apresentados durante a aula. Vale a ordem (e portanto o tempo) de resolução e a solução encontrada. Todos deverão usar o sistema online WISH Prolog.


SWISH
File
Edit
Examples
Help
Search
134 users online
htmlcell
examples
Welcome to SWISH
You are reading a SWISH notebook. A notebook is a mixture of text, programs and queries. This notebook only contains text and gives an overview of example programs shipped with SWISH.
First steps
Knowledge bases provides a really simple knowledge base with example queries.
Lists defines a couple of really simple list operations and illustrates timing naive reverse.
Classics
Movie database provides a couple of thousands of facts about movies for you to query.
Expert system illustrates simple meta-interpretation of rules and asking for missing knowledge.
Eliza implements the classical shrink.
English grammar DCG rules for parsing some simple English sentences and show the result as an SVG tree.
Puzzles and constraints
Einstein's Riddle A famous puzzle attributed to Einstein.
N-Queens (traditional) solves the N-queens problem using traditional Prolog and illustrates domain-specific (graphics) output.
N-Queens (clp(fd)) as above, illustrating the value of constraint programming.
Sudoku (clp(fd)) solves the sudoku puzzle using constraint programming, redering the result as a table.
Knights and Knaves (clp(b)) solves boolean problems.
Mortgage (clp(q,r)) Compute mortgages.
Side effects and I/O
Read and write demonstrates that you can read from and write to the web interface.
Assert and retract demonstrates using the dynamic database.
Machine learning (notebooks) (see also SWISH tutorials)
EM Clustering of the Iris Dataset
SIATEC pattern discovery in polyphonic music
Graphical output (see also rendering)
?-
trace,in_data(X,Y).
Examples
History
Solutions
table results
Run!

O procedimento

- Todos devem estar preparados com o Swish e o seu algoritmo, mas sem tocar no teclado ou no mouse;
- Os estados inicial e final serão colocados em um slide;
- Todos esperam o "sinal de largada" para começar o processo;
- Devem inserir os estado inicial e final no programa;
- Chegando ao estado final o programa deve descrever este estado e colocar na tela o custo da busca.
- com esta informação na saída do programa a equipe faz um sinal de termino e o tempo é anotado (posição no ranking).

Após a competição

Após a competição cada equipe deve fazer o upload no sistema e-disciplinas (na página de PMR 3510) de um arquivo PDF que descreve a heurística (f(x)= g(x) e h(x)), suas propriedades, e uma listagem do código Prolog. A nota final será a média da nota atribuída a este documento e a nota obtida na competição.

A nota da competição

A nota da competição será a composição da classificação por tempo (de zero a cinco), e cinco pontos se a equipe conseguiu resolver o plano mesmo que em um tempo mais longo, e zero se não fechou o processo de busca (sistema em loop infinito).

As equipes

Grupo 1:

Fernando Vicente Grando Monteiro 8992919

Marcos Menon José 8989112

Sverker Fabian Hugert 11462480

Vitor Augusto Martin 8993100

Grupo 2:

David Calil Spindola Pedro - 8989384

Diego Augusto Vieira Rodrigues - 8989276

Felipe Cominato Nemr - 9345662

Guilherme Sugahara Faustino - 9348971

Grupo 3:

Lucas Hideki Sakurai 8989193

Lucas Pereira Cotrim 8989092

Matheus Torres Guinezi 9345679

Monize Bessa Arabadgi 7961944

Renan Masashi Yamaguchi 8989151

Grupo 4:

Alexandre Zamora Zerbini Denigres - 8583072

Dylan Kim Heleno - 8586072

Henrique Yda Yamamoto - 9349502

Vinicius Augusto Carnevali Miquelin - 8988410

Vinicius Takiuti Miura - 9345874

Grupo 5:

Guilherme Dello Russo - 9345895

Nathan Géraud PERRIN- 10935360

Natália Thoma Ricardo - 9344806

Grupo 6:

Beatriz Santin de Araujo Pinho - 8533851

Bianca Faria Silva - 8991599

Bruna Sayuri de Souza Suzuki - 7987501

Murillo José Almeida Faria de Oliveira - 9436785

Grupo 7:

Daniel Tsutsumi - 9349005

Gabriel Pinto - 8988017

Juliana Lopes - 8512600

Kaio Takase - 9345690

Matheus Ramalho - 9345710

Grupo 8(?):

Danilo Polidoro - 8582982

Até a próxima aula!