Prolog

Artificial intelligence

Prolog

Prolog (Programming in Logic) is a programming language for AI and non-numerical programming in general.

Prolog

"John owns the book"

Owns (john,book)

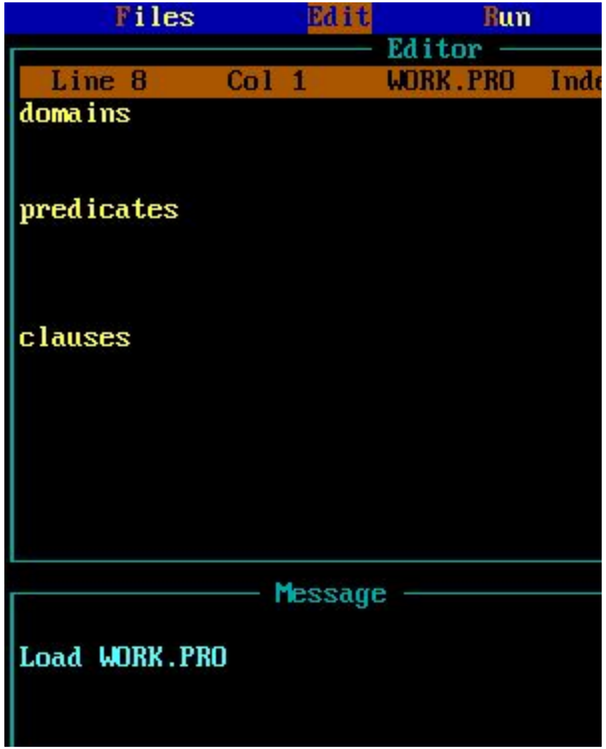
relationship(object1,object2)

Why Prolog

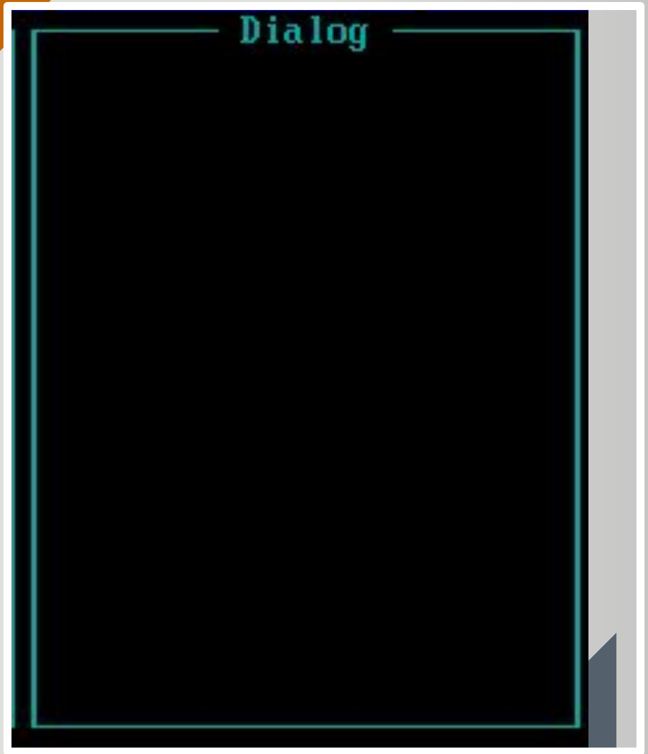
- Syntax
- back tracking
- multi directional reasoning
- Fact & Rule

Run Prolog

Files Edit Run Compile Options Setup
Editor
Dialog
Line
WORK.PRO Indent Insert
Load
Pick
New file
Save
Write to
Directory
Change dir
OS shell
Quit
domain
list =
predic
link(s
l).
go (sy
list)
clause
link
link (h, f).
Message
Trace
Load WORK.PRO
F2-Save F3-Load F6-Switch F9-Compile Alt-X-Exit

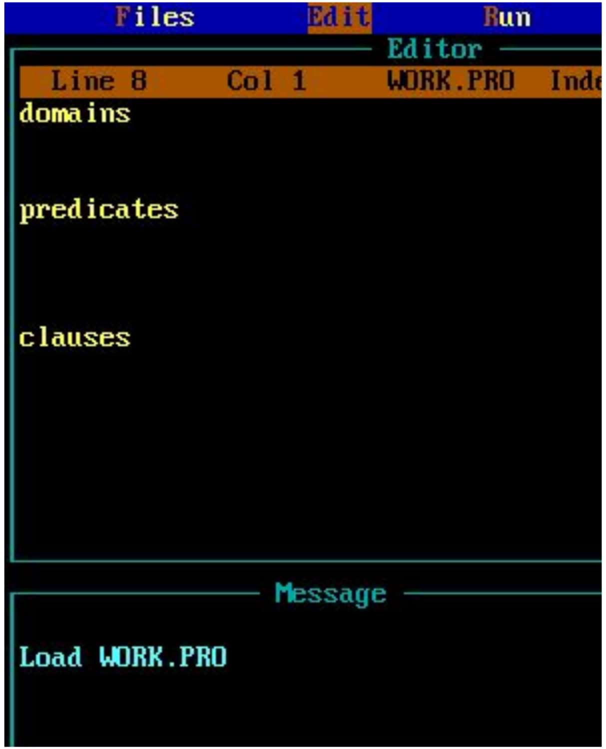


Structured Program Prolog



Run

alt+r

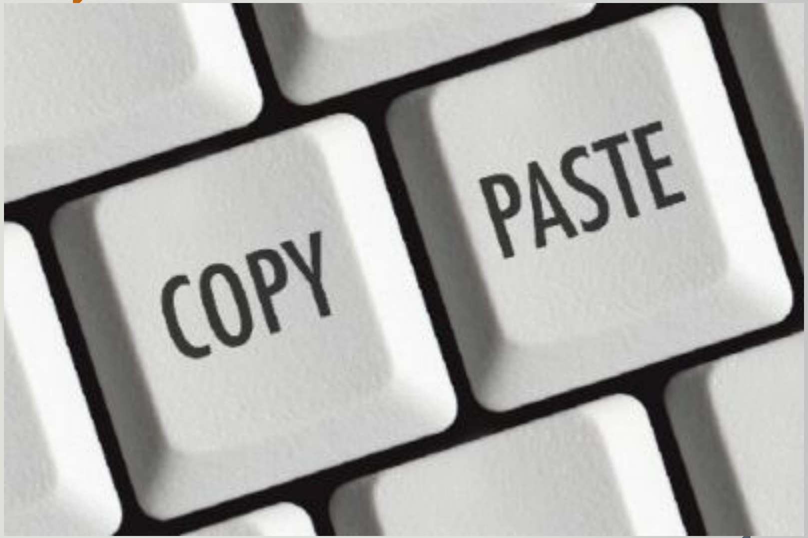

Files
Edit
Run
Editor
Line 8
Col 1
WORK.PRO
domains
predicates
clauses
Message
Load WORK.PRO

Edit

alt+e

Run Prolog


Files Edit Run Compile Options Setup
Editor
Dialog
Line
WORK.PRO Indent Insert
Load
Pick
New file
Save
Write to
Directory
Change dir
OS shell
Quit
domain
list =
predic
link(s
l).
go (sy
list)
clause
link
link (h, f).
Message
Trace
Load WORK.PRO
F2-Save F3-Load F6-Switch F9-Compile
Alt-X-Exit



Select

Ctrl+k+b

Ctrl+k+k

Copy

Ctrl+k+c

Cut

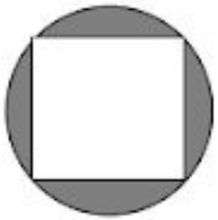
Ctrl+k+v

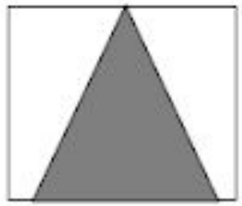
Fact

likes (ali, youssef).

Example

Circle color is gray.	color(circle,gray).
Square color is white.	color(square, white).
Triangle color is gray.	color(triangle,gray).
Rectangle color is white.	color(rectangle,white).
The square is inside the circle.	inside(square,circle).
The triangle is inside the rectangle.	inside(triangle,rectangle).





More Example

likes(john, susie).	/* John likes Susie */
likes(X, susie).	/* Everyone likes Susie */
likes(john, Y).	/* John likes everybody */
likes(john, Y), likes(Y, john).	/* John likes everybody and everybody likes John */
likes(john, susie); likes(john,mary).	/* John likes Susie or John likes Mary */
not(likes(john,pizza)).	/* John does not like pizza */
likes(john,susie) :- likes(john,mary).	/* John likes Susie if John likes Mary.

quiz

Fact:

Like (john, mary).

Like (john, flower).

Like (ali, mary).

Goal:

Like (john,X) ?

like(X, mary)?

Like(X, Y) ?

Symbols

English	Predicate Calculus	PROLOG
and	^	,
or	v	;
if	→	:-
not	~	not

Variables and Names

mother_of

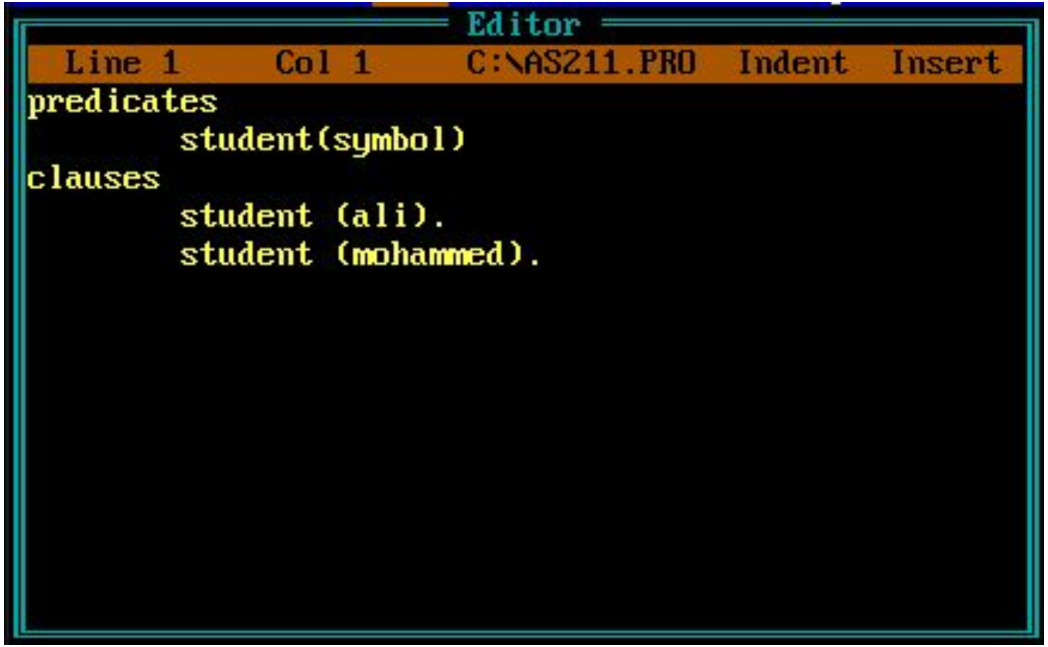
male

female

greater_than

socrates

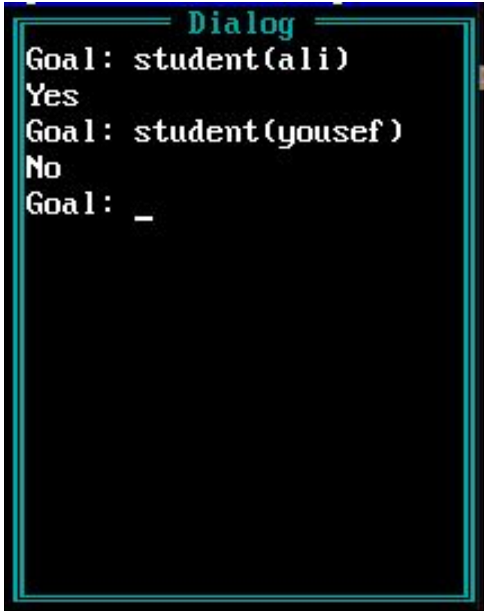
Example




```
predicates
        student(symbol)
clauses
        student (ali).
        student (mohammed).
```

run the code

1

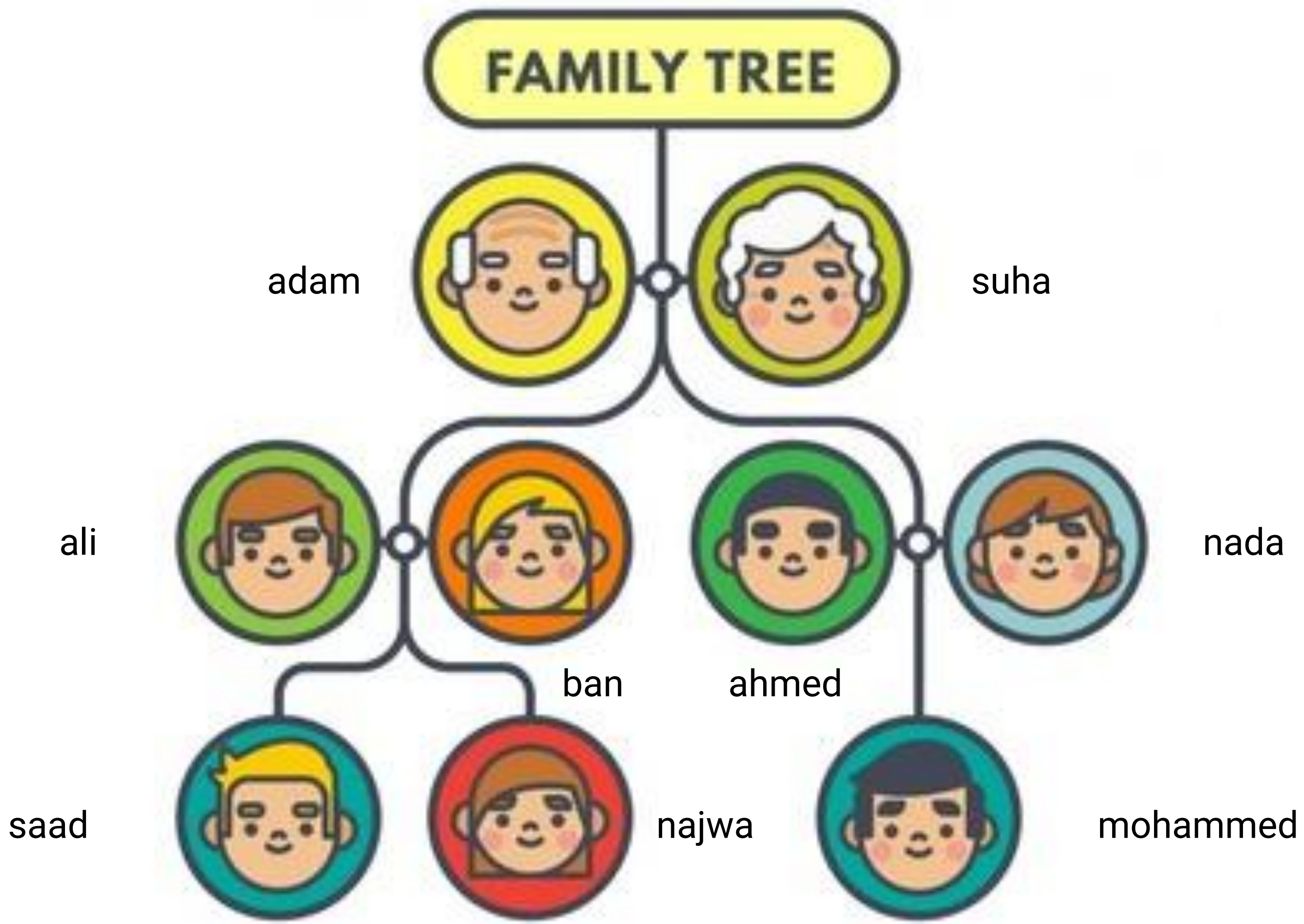


2

Capital letters □ variable

Rule

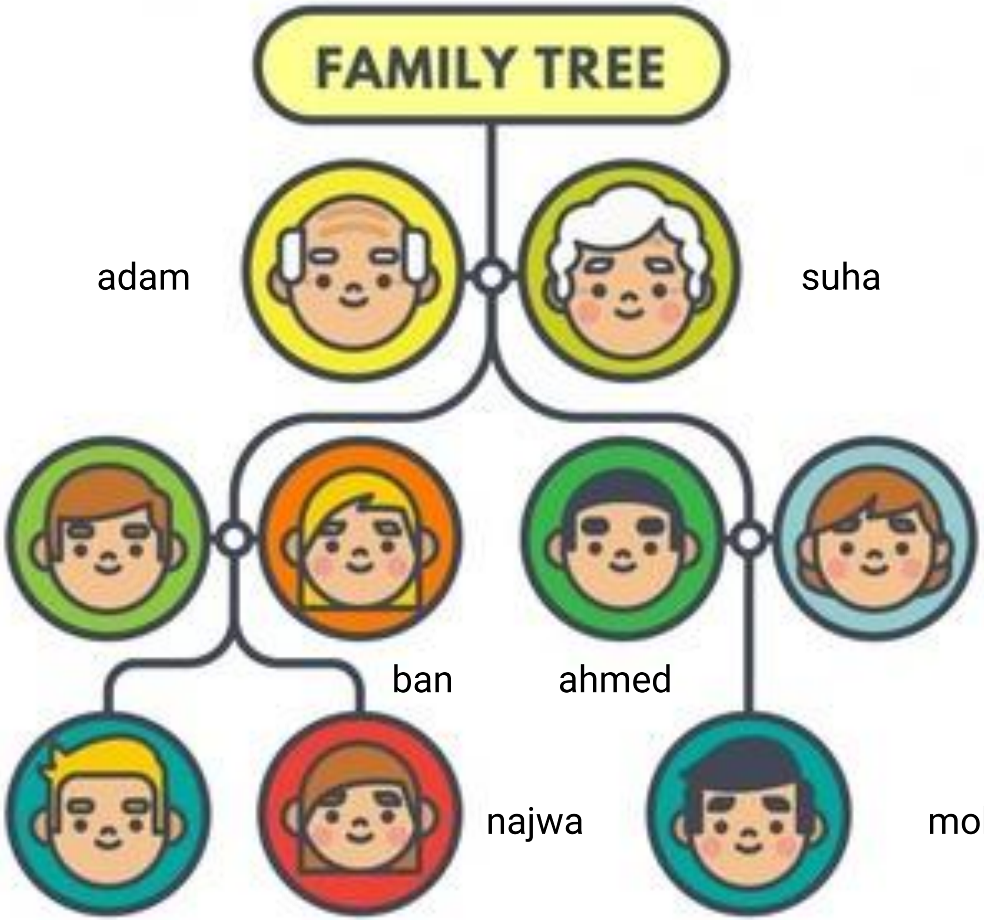
Rule name (parameters) :-
condition on fact1 (, ;) condition on fact2.


FAMILY TREE
adam
suha
ali
nada
ban
ahmed
saad
najwa
mohammed

father

male X

Y



Rule

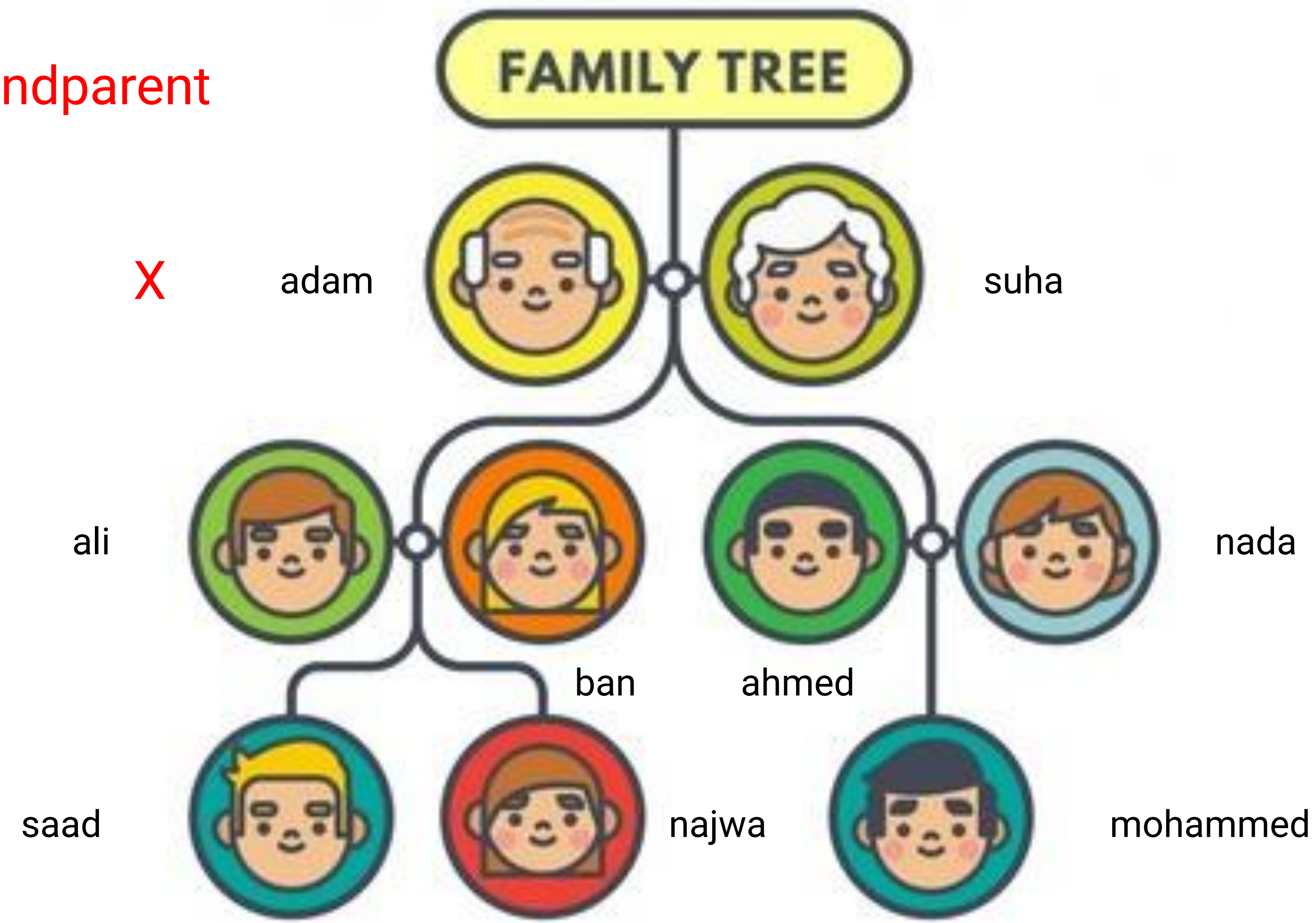
```
father(X,Y):-parent(X,Y),male(X).
```

grandparent

X

Y

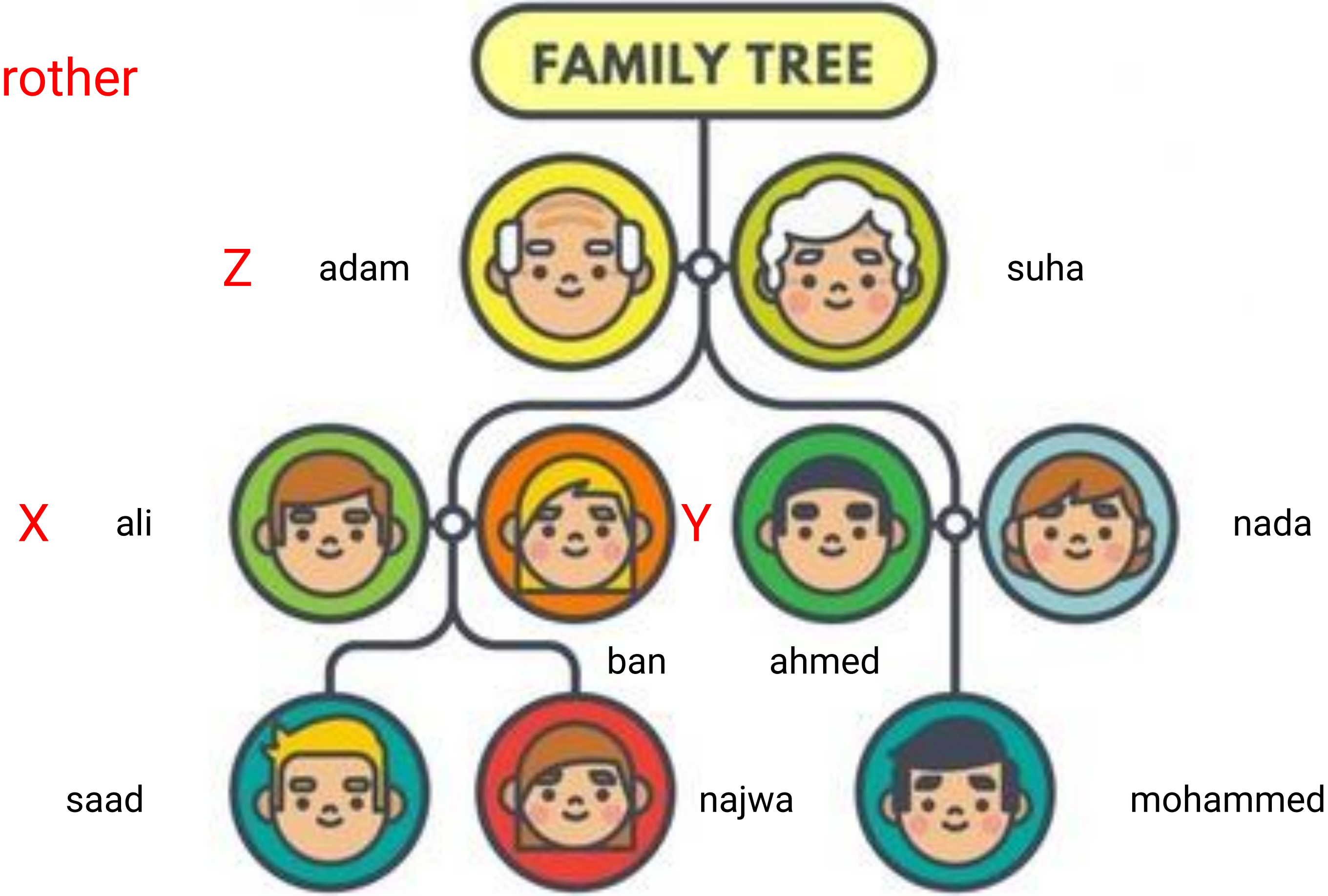
Z



Rule

```
grandparent(X,Z):-parent(X,Y),parent(Y,Z).
```

brother


FAMILY TREE
Z
adam
suha
X
ali
Y
nada
ban
ahmed
saad
najwa
mohammed

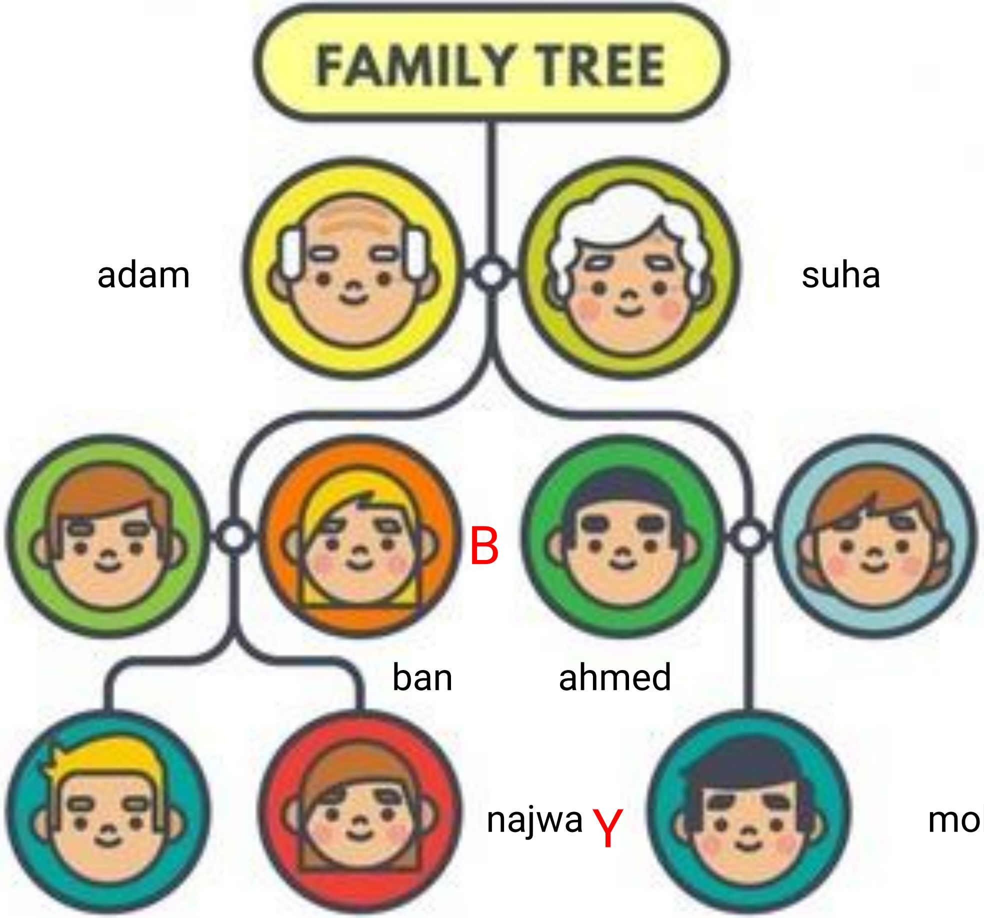
Rule

```
brother(X,Y):-
parent(Z,X),parent(Z,Y),male(X),X<>Y.
```

cousin

Father

brother

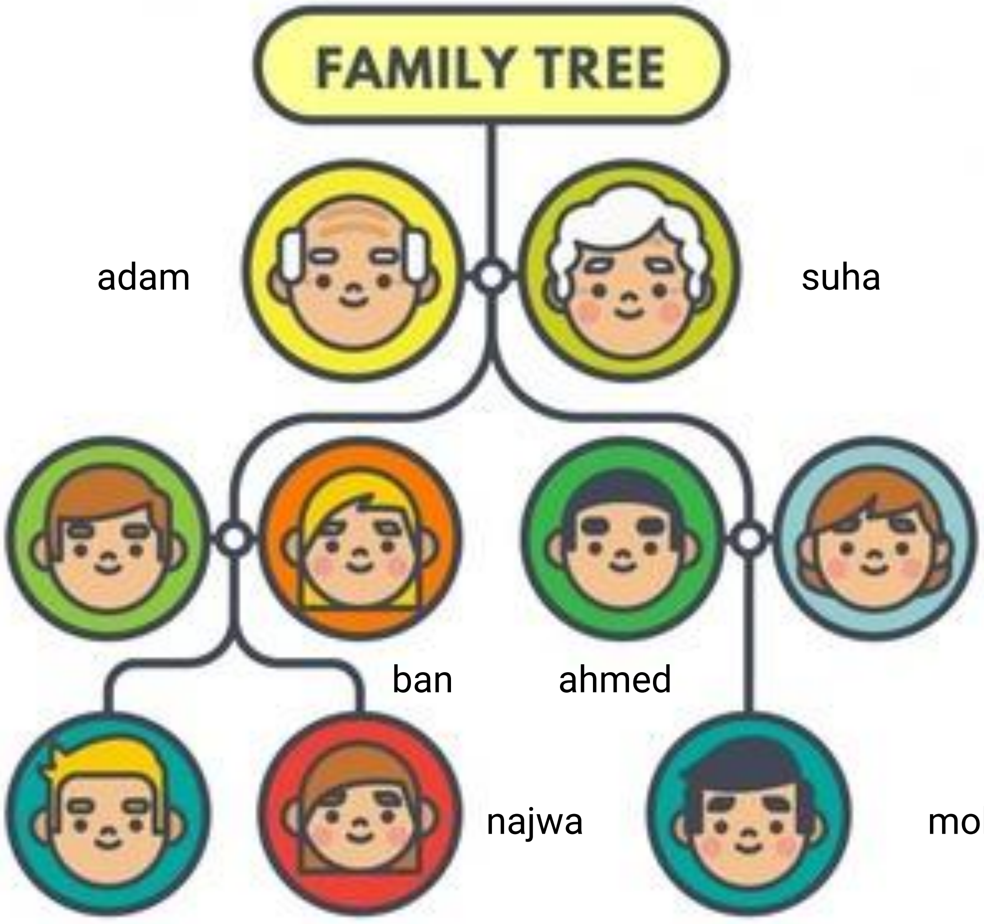


Rule

cousin(X,Y):-
father(A,X),father(B,Y),brother(A,B),A<>B.

H.W

1.Grandma
2. Mother
3. Sister
4. Aunt
5. Wife
6. Daughter



Data Type

Data Type	like
Integer	7 ، 23 ، 100 ، -25 ، -9
Real	2.3 ، 7.0 ، -8.8
Char	‘A’ ، ‘M’ ، ‘y’
string	"helloo" ، "Ali" ، "SAUC"
Symbol	helloo، ali ، sauc

mathematical operation

operation	symbol
addition	+
subtraction	-
multiplication	*
Integer part of division	div
Remainder of division	mod

logical operation

operation	symbol	operation	symbol
greater	>	Greater or equal	>=
Less than	<	Less than or equal	<=
Equal	=	Not equal	<>

mathematical function

Function name	
Exp(X)	Round(X)
Ln(X)	Abs(X)
Sqrt(X)	

Read and write function

Read function:

readint( )

Readchar( )

Readreal( )

Readln( )

Write function

Write( )

nl

Using IF THEN ELSE in PROLOG

If condition then statement else statement

Rule (X,Y) :- condition, then statement .

Rule (X,Y) :- Opposite condition , else statement

Using IF THEN ELSE in PROLOG

1- write prolog program that take two integer input us integer and print the greater.

2- write prolog program to check if the given number is positive or negative.

3- write prolog program to check if a given number is odd or even.

Repetition and Recursion

- who start using Visual Prolog are often dismayed to find that the language has no FOR, WHILE, or REPEAT statements.

- There is no direct way to express iteration.

- Prolog allows only two kinds

  - repetition--backtracking

  - recursion

Backtracking Revisited

when looks for another solution to a goal that has already been satisfied. It does this by retreating to the most recent subgoal that has an untried alternative

Implementing Backtracking with Loops

Simply define the two-clause predicate

```
repeat.
repeat :- repeat.
```

The purpose of repeat is to allow backtracking ad Infinitum.

Recursive Procedures

The other way to express repetition is through recursion. A recursive procedure is one that calls itself.

Recursion is the natural way to describe any problem that contains within itself another problem of the same kind

Factorial

N! = N * (N-1) * (n-2) *... * 3 * 2 * 1

3! = 3 * 2 * 1

Tail Recursion

The other way to express repetition is through recursion. A recursive procedure is one that calls itself.

Recursion is the natural way to describe any problem that contains within itself another problem of the same kind

list in prolog

List processing is a powerful technique in Prolog.

In prolog, a list is an object that contains an arbitrary number of other objects within it. Lists correspond roughly to array in other languages but unlike an array, a list does not require you to how big it will be before using it.

list in prolog

syntax of list

Domains
list = integer*

Heads and Tails = [H|T]

list = [1, 2, 3].

H = 1 T =[2,3]

H =2 T =[3]

H =3 T=[ ]

list in prolog

syntax of list

Domains

list = integer*

Heads and Tails = [H|T]

the head of [a, b, c] is a

the tail of [a, b, c] is [b, c]

Using Lists

Because a list is really a recursive compound data structure, you need recursive algorithms to process it. The most basic way to process a list is to work through it, doing something to each element until you reach the end.

Using Lists

An algorithm of this kind usually needs two clauses. One of them says what to do with an ordinary list (one that can be divided into a head and a tail). The other says what to do with an empty list.

Thank you for listening

any questions … ?

Systematic Search
Basic Graph Concepts

Artificial Intelligence

Search

```
Procedure Generate & Test

Begin

    Repeat

      Generate a new state and call it current-state;

    Until current-state = Goal;

End.
```

Search

1. Describe the search problem

- State
- Operator
- Conditions

2. Choose the search method

State-space

We centered around a general scheme called state space, for representing problems. A state space is a graph whose node corresponds to the problem situation and a given problem is reduced to finding a path in this graph.

Algorithm

- Nodes
- Arc
- Goal
- Current

Search Methods

1- Blind search

- Breadth First Search
- Depth First Search

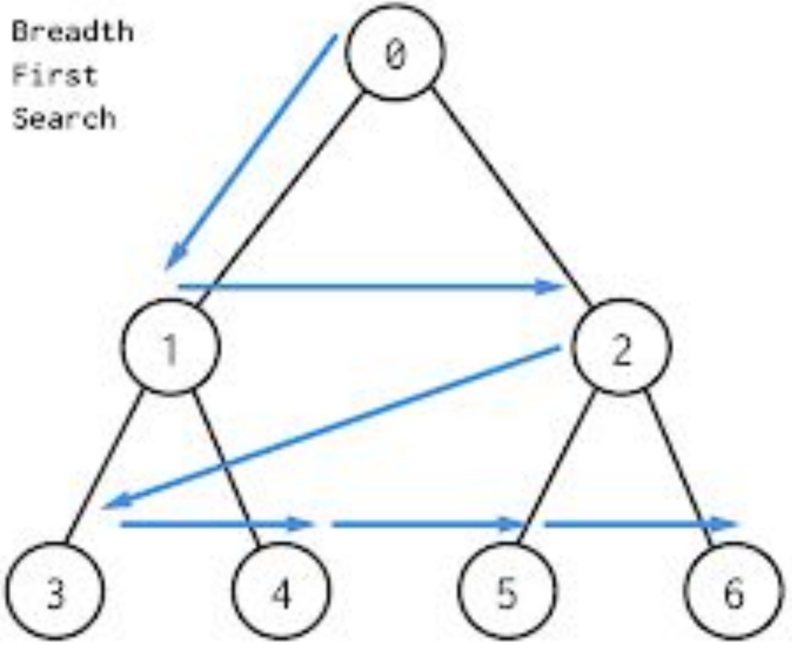
2- Heuristic search

- Hill Climbing Search
- Best First Search
- A algorithm.
- A* algorithm.

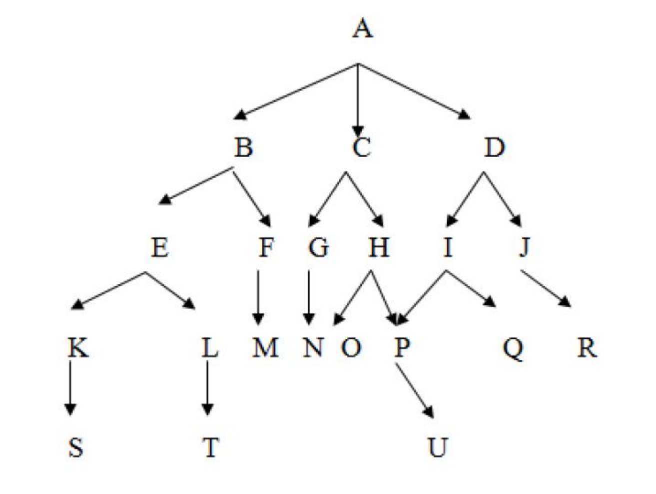
Search Methods

1- Blind search

- Breadth First Search



Breadth First Search



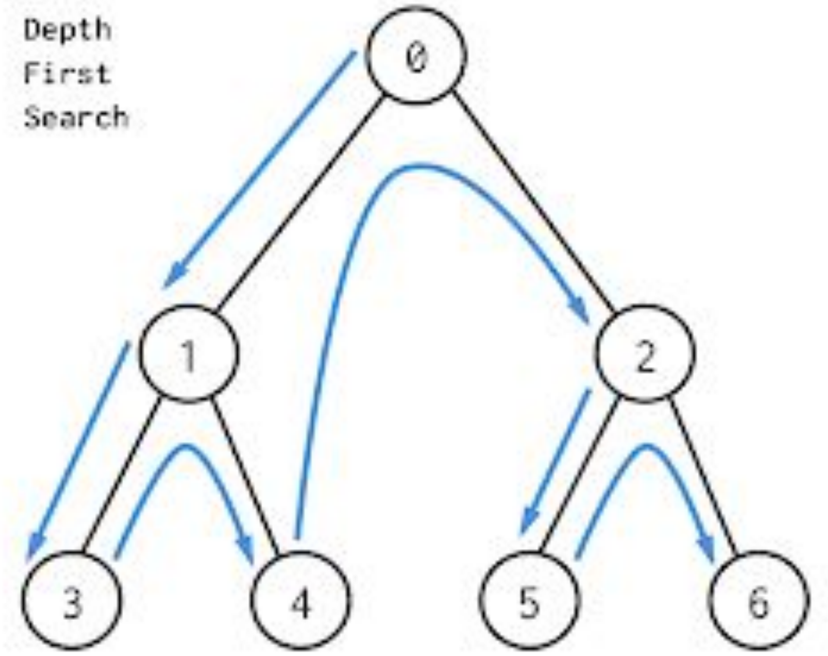
Open	closed
1 –Open= [A];	closed = [ ].
2 –Open= [B, C, D];	closed = [A].
3 –Open= [C, D, E, F];	closed = [B, A].
4 –Open= [D, E, F, G, H];	closed = [C, B, A].
5 –Open= [E, F, G, H, I, J];	closed = [D, C, B, A].
6 –Open= [F, G, H, I, J, K, L];	closed = [E, D, C, B, A].
7 –Open= [G, H, I, J, K, L, M];	closed = [F, E, D, C, B, A].

path = [A, B, C, D, E, F, G].

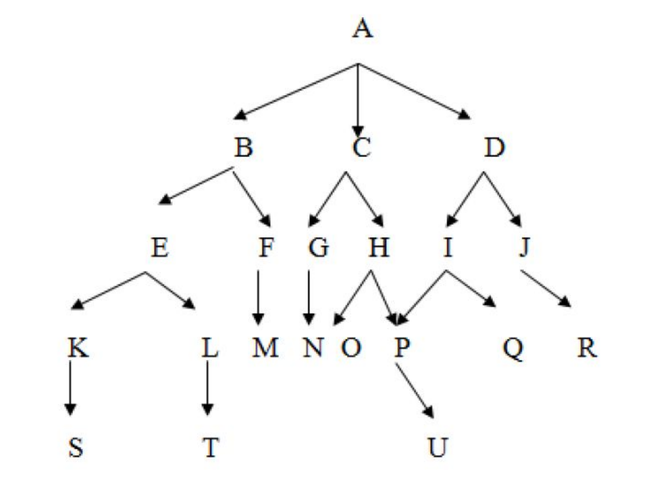
Search Methods

1- Blind search

- Depth First Search



Depth First Search



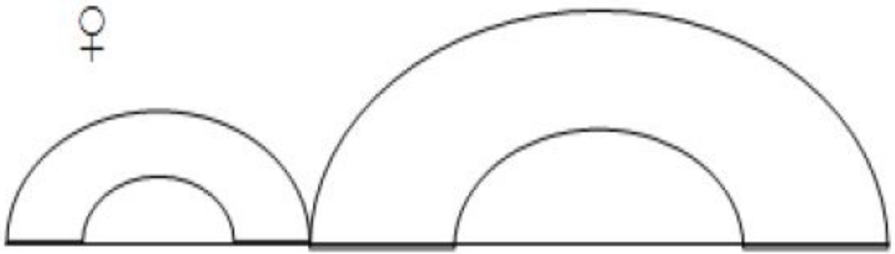
Open	closed
1. open = [A];	closed = [ ]
2. open = [B,C,D];	closed = [A]
3. open = [E,F,C,D];	closed = [B,A]
4. open = [K,L,F,C,D];	closed = [E,B,A]
5. open = [S,L,F,C,D];	closed = [K,E,B,A]
6. open = [L,F,C,D];	closed = [S,K,E,B,A]
7. open = [T,F,C,D];	closed = [L,S,K,E,B,A]

path = [A, B, E, K, S, L, T]

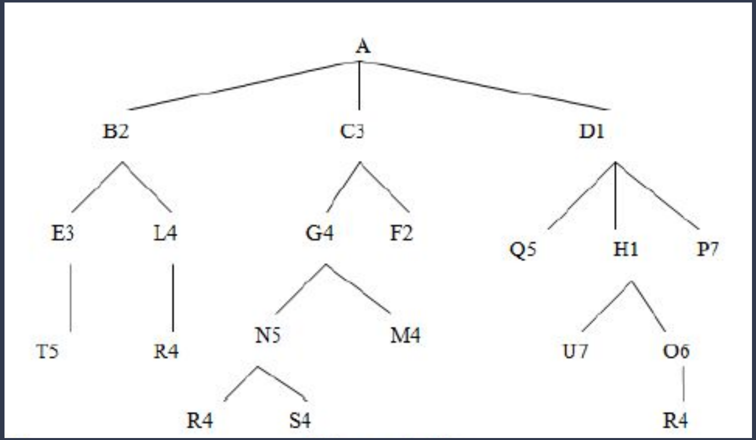
Search Methods

2- Heuristic search

- Hill Climbing Search



Hill Climbing Search



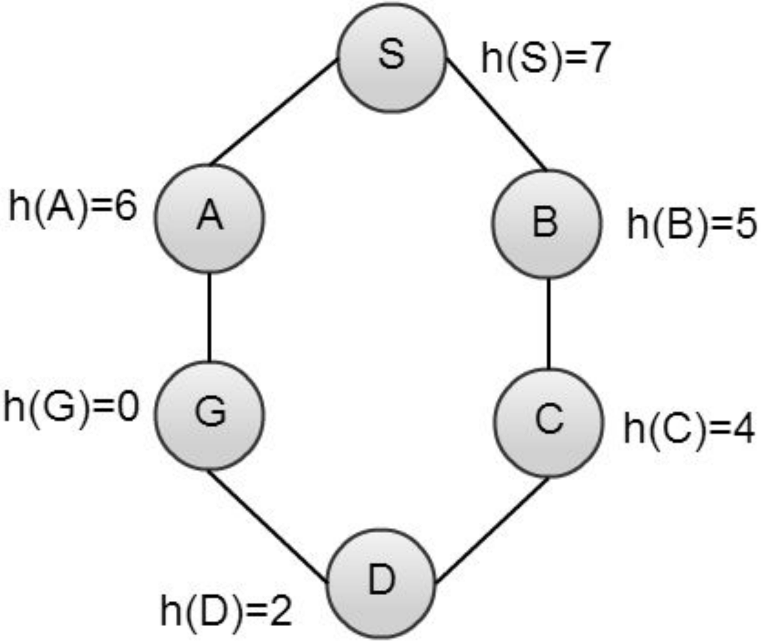
Open	Closed	X
1. Open=[A]	Closed=[]	A
2. Open=[D1,B2,C3]	Closed=[A]	D1
3. Open=[H1,Q5,P7]	Closed=[A, D1]	H1
4. Open=[O6,U7]	Closed=[A,D1,H1]	O6
5. Open=[R4]	Closed=[A,D1,H1,O6]	R4

The solution path is: A-D1-H1-O6-R4

Search Methods

2- Heuristic search

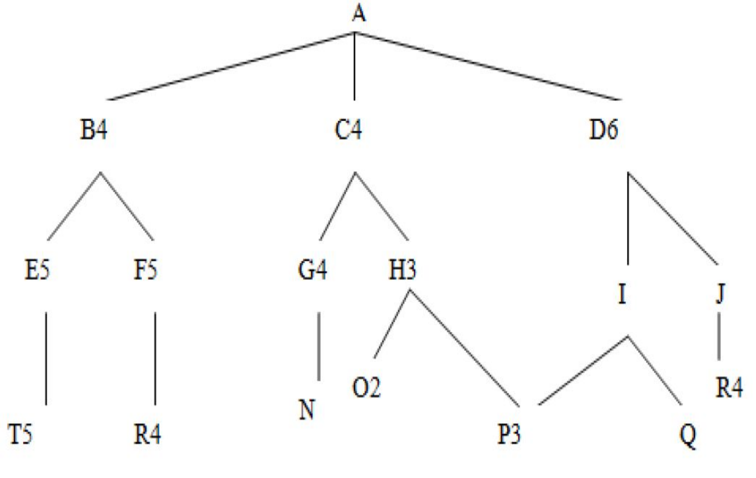
- Best-First-Search



The heuristic function

(h(n)) as : f(n) = h(n)

Best-First-Search



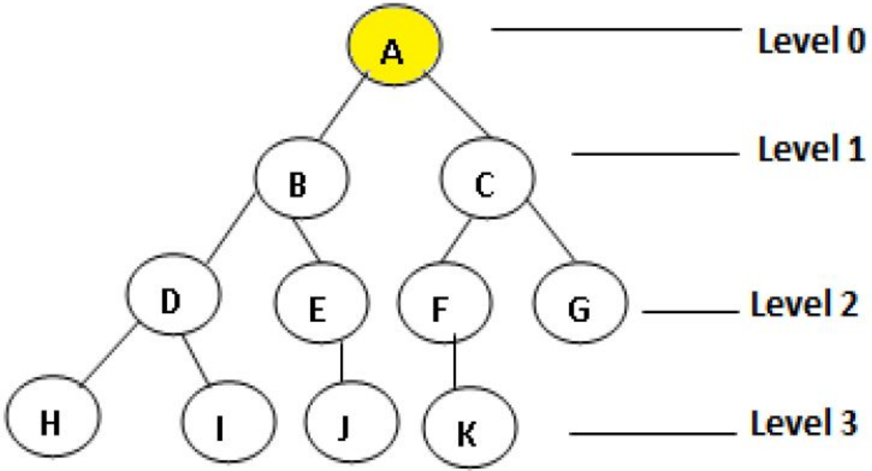
Open	closed
1. Open=[A5]	Closed=[]
2. Open=[B4,C4,D6]	Closed=[A5]
3. Open=[C4,E5,F5,D6]	Closed=[B4,A5]
4. Open=[H3,G4,E5,F5,D6]	Closed=[C4,B4,A5]
5. Open=[O2,P3,G4,E5,F5,D6]	Closed=[H3,C4,B4,A5]
6. Open=[P3,G4,E5,F5,D6]	Closed=[O2,H3,C4,B4,A5]
7. Open=[G4,E5,F5,D6]	Closed=[P3,O2,H3,C4,B4,A5]

The solution path is: A5 -B4 -C4 -H3 –O2-P3

Search Methods

2- Heuristic search

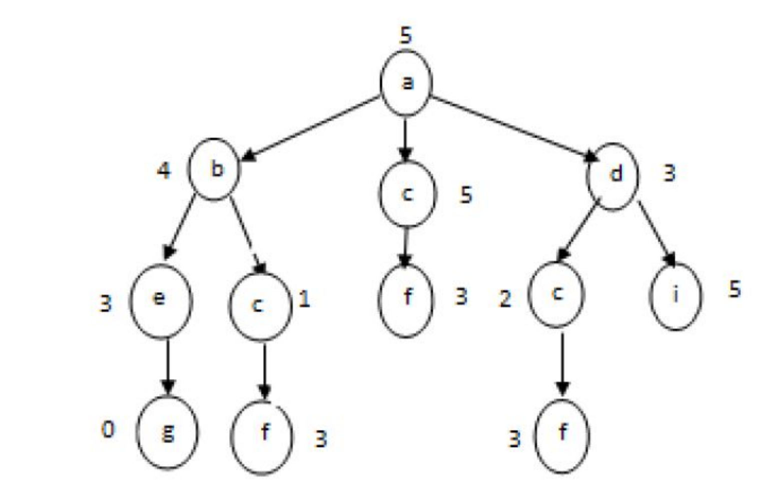
- A Search



The heuristic function

F (n) = h (n) + g (n)

A Search



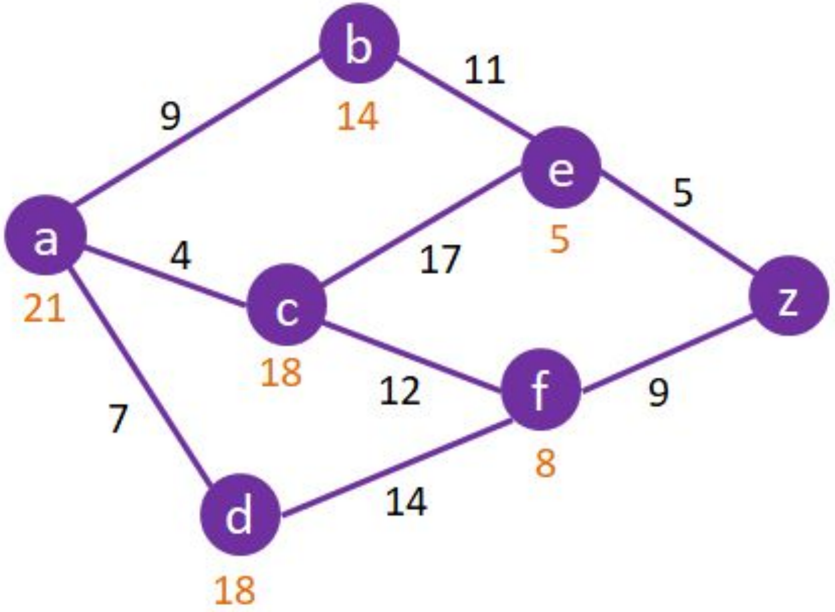
Open	closed
1. Open=[A5]	closed=[ ]
2. Open=[D4 , B5 , C6]	closed=[A5]
3. Open=[C4 , B5,I7]	closed=[A5 , D4]
4. Open=[B5 , F6,I7]	closed=[A5 , D4 , C4]
5. Open=[C3 , E5 , F6,I7]	closed=[A5 , D4 ,C4, B5]
6. Open=[E5 , F6,I7]	closed=[A5 , D4 , B5 , C3]
7. Open=[G3 , F6,I7]	closed=[A5 , D4 , B5 , C3 , E5]

the resulted path is : A5 ->D3 -> B4 -> C1 -> E3 -> G0 = 16

Search Methods

2- Heuristic search

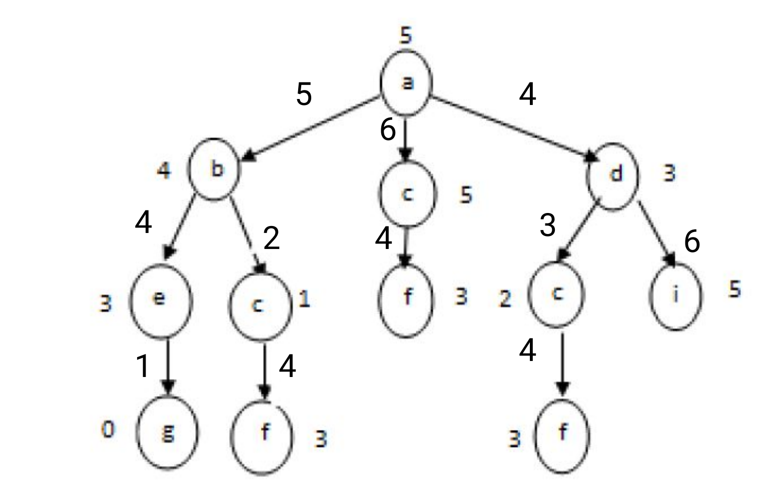
- A* Search



The heuristic function

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

A* Search

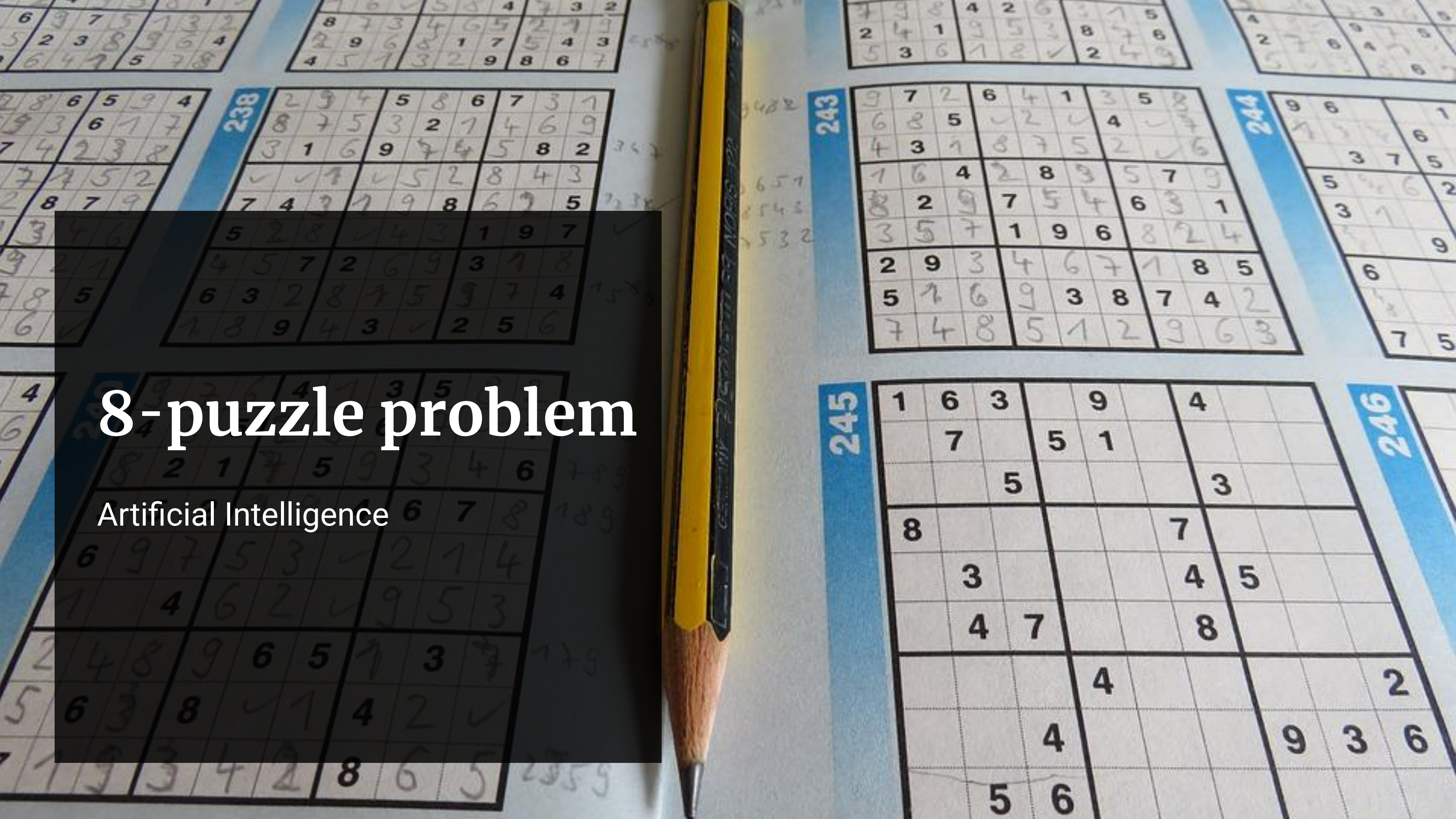


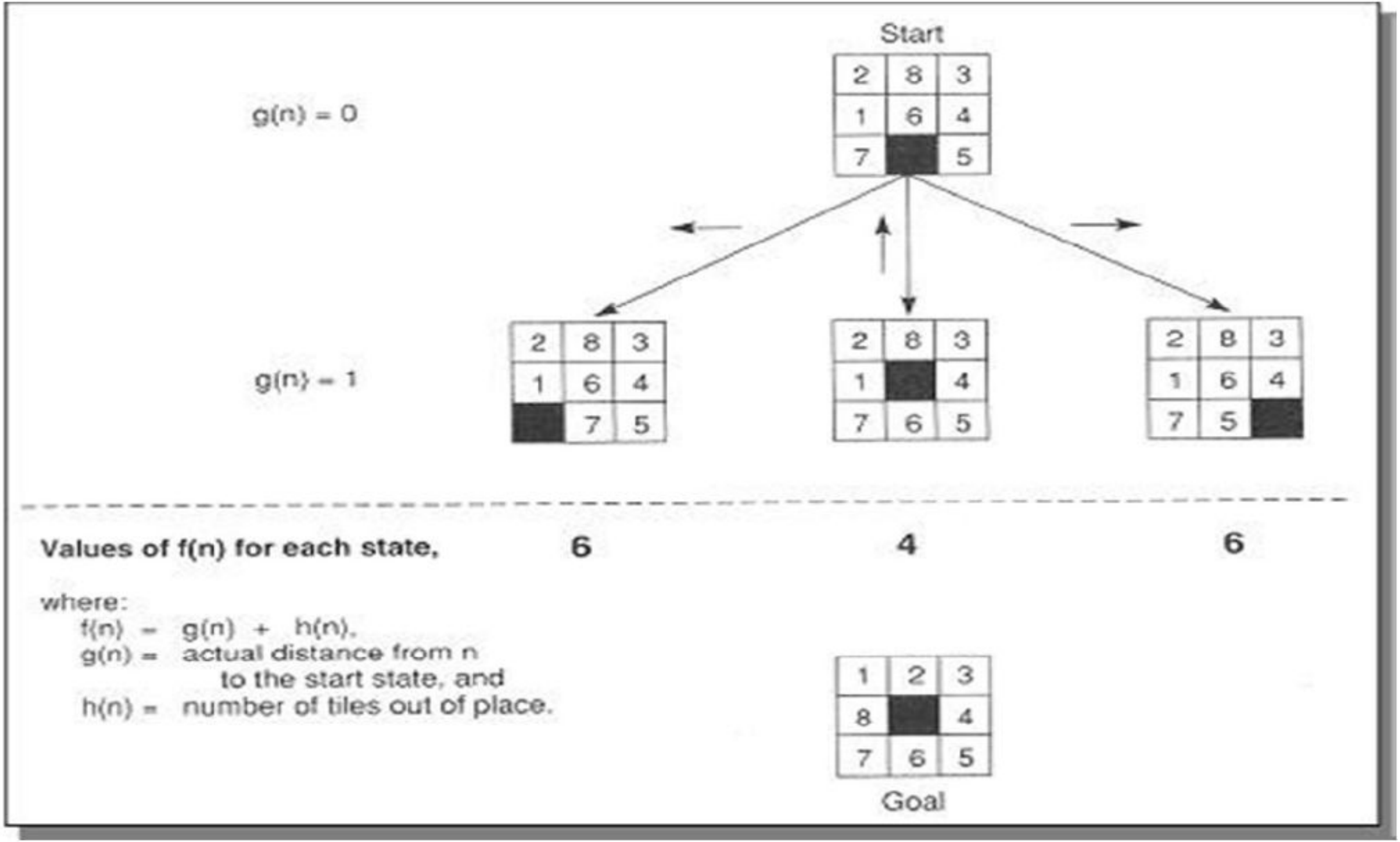
Open	closed
Open=[a5]	closed=[]
Open= [d7,b9,c11]	closed= [a5]
Open= [b9 , c9,i15]	closed= [a5 , d7]
Open= [c8 , e12 , i15]	closed= [a5 , d7 , b9]
Open= [e12 , f14 , i15]	closed= [a5 , d7 , b9, c8]
Open= [g10,f14,i15]	closed= [a5,d7,b9,c8,e12,g10]

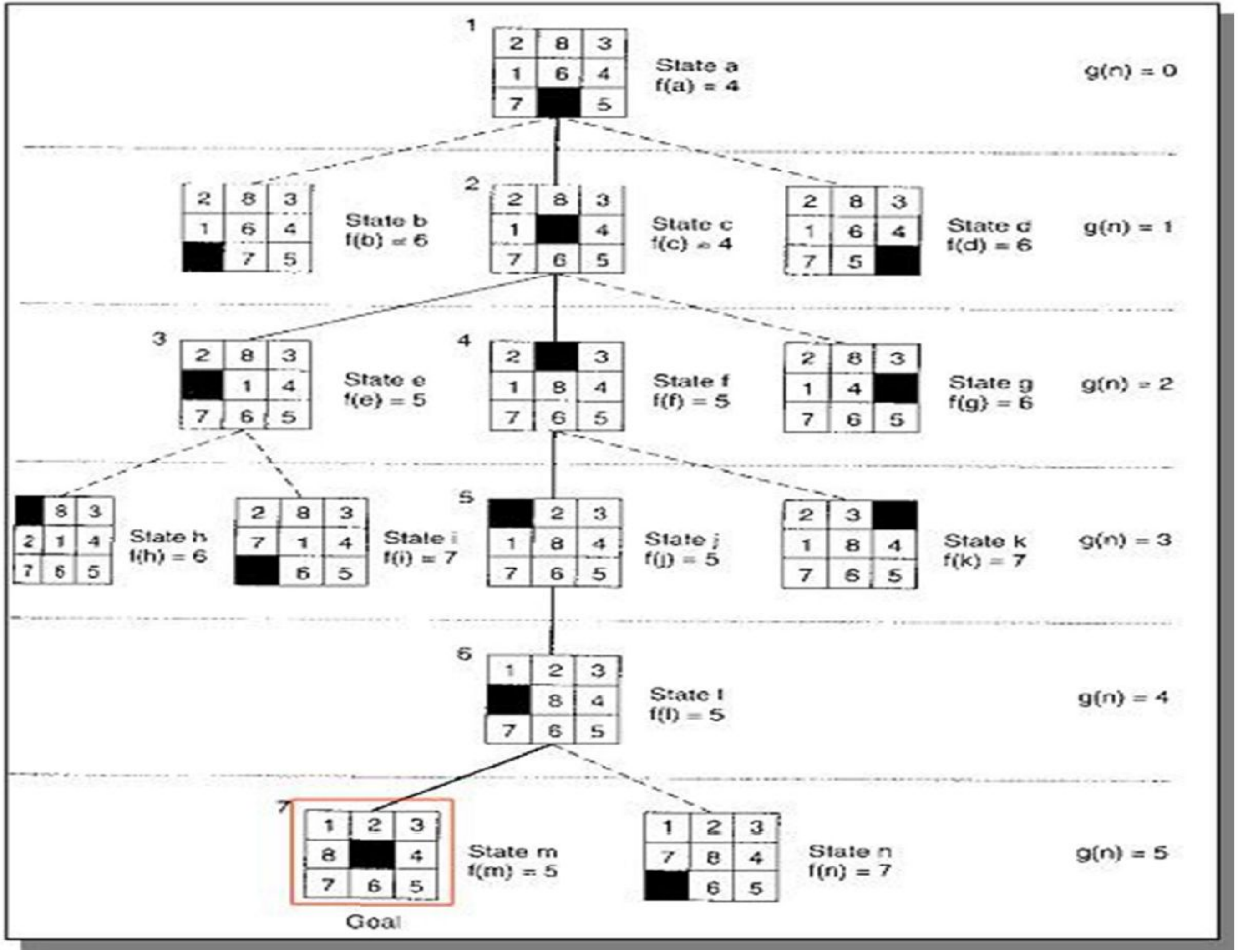
The goal is found & the resulted path is: A0 d4 b9 c2 e6 g1 =22

8-puzzle problem

Artificial Intelligence




Start
2 8 3
1 6 4
7 5
g(n) = 0
g(n) = 1
2 8 3
1 6 4
7 5
2 8 3
1 4
7 6 5
2 8 3
1 6 4
7 5
Values of f(n) for each state,
6
4
6
where:
f(n) = g(n) + h(n),
g(n) = actual distance from n to the start state, and
h(n) = number of tiles out of place.
1 2 3
8 4
7 6 5
Goal



1
State a
f(a) = 4
g(n) = 0
State b
f(b) = 6
2
State c
f(c) = 4
State d
f(d) = 6
g(n) = 1
3
State e
f(e) = 5
4
State f
f(f) = 5
State g
f(g) = 6
g(n) = 2
State h
f(h) = 6
State i
f(i) = 7
5
State j
f(j) = 5
State k
f(k) = 7
g(n) = 3
6
State l
f(l) = 5
g(n) = 4
7
State m
f(m) = 5
Goal
State n
f(n) = 7
g(n) = 5

After implementation of **A** algorithm, the Open and Closed is shown as follows:

1. Open=[a4], Closed=[]

2. Open=[c4,b6,d6],Closed=[a4]

3. Open=[e5,f5,b6,d6,g6],Closed=[a4,c4]

4. Open=[f5,b6,d6,g6,h6,i7],Closed=[a4,c4,e5]

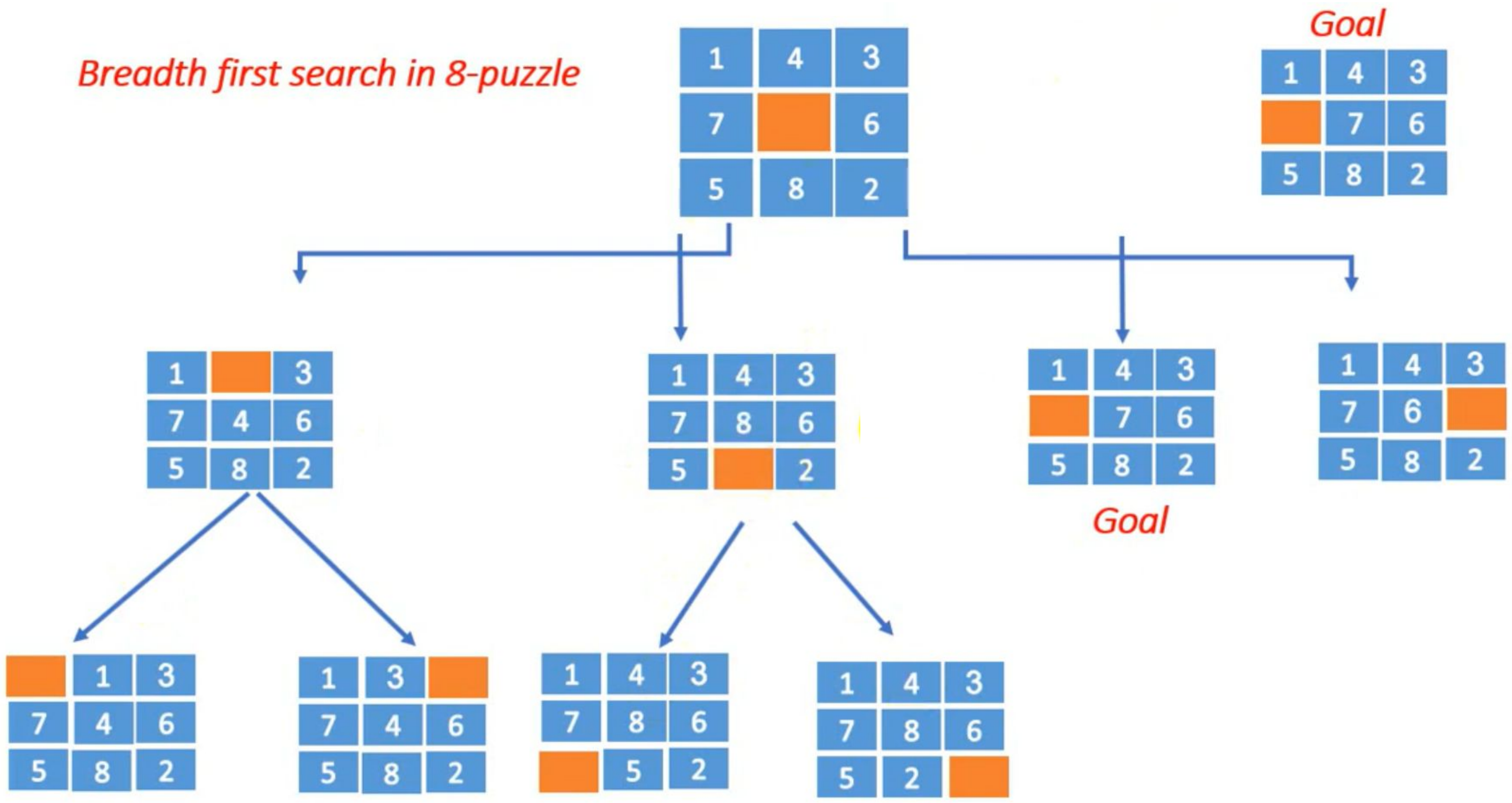
5. Open=[j5,b6,d6,g6,h6,j7,k7], Closed=[a4,c4,e5,f5]

6. Open=[l5, b6,d6,g6,h6,j7,k7],Closed=[a4,c4,e5,f5,j5]

7. Open=[m5, b6,d6,g6,h6,j7,k7,n7],Closed=[a4,c4,e5,f5,j5,l5]

8. Success, m=goal!!

Breadth first search in 8-puzzle



Thank you for listening

any questions