15/9/22

Principles of Artificial Intelligence (PAI)

AI is a part of computer science concerns with the designing intelligent computer systems that is the systems which exhibit the characteristics associate with intelligence in human behaviour.

AI contains two terms :-

1) Intelligence 2) Artificial devices

A system with intelligence is expected to behave as intelligently as a human being. Secondly, A system with intelligence is expected to behave in best possible manners.

19/09/22

	conventional computer	AI computer
Processing		
Nature of Input		
explanation		
Focus		
Maintainance update		
Reasoning capability		

Application of AI

Artificial Intelligence

- General Problem solving
- expert system
- Natural Lang Processing
- computer vision
- Robotics
- computer Aided ~~Instruction~~ Instruction
- Planning

CHARACTERSTICS OF AI

20/9/22

a) Symbolic Representation

b) Knowledge Representation
- Knowledge
- Incomplete data
- conflicting data

c) Ability to learn

Symbolic representation - AI programs mainly deals with non numerical symbols and contrast strongly with commonly accepting view that computer can be used.
AI program can perform numerical calculations also where necessary but their significance or symbolic value will generally enter into reining process performed by program
for eg:- In AI
The importance are:-

1) It is the significant of theory — of nature of human intelligence and as such is of great interest to
2) It forms the basic beliefs that it is possible to built the program which can perform intelligent task as performed by human.

__/__/____

Knowledge
Knowledge can present as a modular. There is a coresponding external and symbolic system.
This knowledge can be studied and understand in human terms for its representation are numerical.
for eg:-

AI program has capability to provide some solution even if all data relevant to the problem is not available at the time.
consequence of data being incom— simply leads to conclusion are less certain. In real world, a decision may be wrong in the absence of relevant data. It can happen that the absence of complete data is inherent in the problem as in the game of the bridge.

conflicting data
Data items can even be contradictry to each other. This type of data is called conflicting data or data ~~connected~~ corrupted by arrow.

→ ~~for~~ Ability to learn >
Intelligent computers are amplitude to learn from their mistakes so that they can improve their performance by taking accounts of past errors. This is related to capacity for generalizing to drawing analogy and discarding selected information. The problem of given computers are learning similar to that of human beings is that of stimulating the machine process. The working of women mines can never be

programmed into machine so that no computer can ever have a mind of its own.

History of AI

→ First Generation late - 1950

1) Can a self repairing machine be built which will locate and repair errors.
2) Can a digital computer be programmed to program itself.
3) Using historical concepts can ~~be~~ a computer program to learn.

22/9/22 Agent :- It can be a program or ~~of~~ ~~self~~ tool.

Program → → → Output

There are 3 parameter for an intelligent system :-
1) Knowledge base
2) Operator
3) Search technique

State search Problems :-

A combination of all possible state for a given problem is called state space of problem.

22/9
Mon Tues Thurs

Initial

2	1	3	4
5	6	5	9
13	8	11	12
14	15	10	x

Goal

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	x

N = 4, N(N-1) = 4X3 = 12

conditions

1) N is odd, then puzzle is solvable

2) N is even, puzzle is solvable in two conditions.

a)

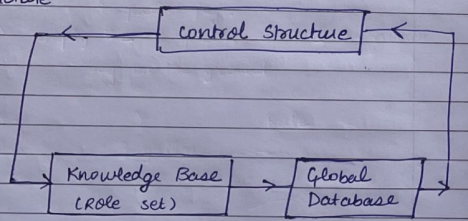
b)

Production System

- Powerfull Knowledge representation
- Bridge b/w AI Research & expert system.
- Heuristic Model for human behaviour.
- strong data-driven nature

New Rules can easily added into system Architecture of PRODUCTION SYSTEM;

i) Knowledge Base
ii) Global DataBase
iii) Control structure



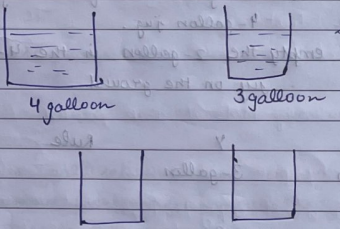
Problem Representation

Statement: Transforming a problem into component of production system is called problem representation.

State Space Representation

- Initial state
- Goal state
- Rules

Ex: water jug problem



1) (x, y) if $x < 4 \rightarrow (4, y)$ → fill 4-gallon jug.
2) (x, y) if $y < 3 \rightarrow (x, 3)$ → Full 3-gallon jug.
3) (x, y) if $X = d * d > 0 \rightarrow (X-d, y)$ Pour some water out of 4 gallon jug.
4) (x, y) if $y = d * d > 0 \rightarrow (x, y-d)$ Pour some water out of 3 galloon jug.
5) (x, y) if $x > 0 \rightarrow (0, y)$ empty the 4-gallon waters on the ground.
6) (x, y) if $y > 0 \rightarrow (x, 0)$ empty the 3-galloon water on the ground.

__/__/____

7) (x, y) , if x+y ≤ 4 and y>0 → 4, (y-(4+x)) — Pour water from 3 gallon jug into 4 gallon until jug is full.

8) (x, y) if x+y ≥ 3 and x>0 → (x-(3, y), 3) — Pour water from 4-gallon jug into 3-gallon jug until it is full.

9) (x, y) if x+y ≤ 4 and y>0 → (x+y, 0) — Pour all water of 3-gallon into 4-gallon.

10) (x, y) if x+y ≤ 3 and x>0 → (0, x+y) — Pour all water from 4-gallon to 3-gallon

11) (0, 2) → (2, 0) Pour 2-gallon water from 3-gallon jug to 4-gallon jug.

12) (2, 4) → (0, 4) empty the 2-gallon in the 4-gallon jug on the ground

X 4-gallon	Y 3-gallon	Rule
0	0	
0	3	→ 2
3	0	→ 9
3	3	→ 2
4	2	→ 7
0	2	→ 5 or 12
2	0	→ 9 or 11

27/9/22

State search Representation

Q) water jug problem

X (4 gallon)	Y (3 gallon)	Rule applied
0	0	
4	0	1
1	3	8
1	0	6
0	1	10
4	1	1
2	3	8

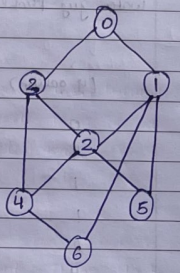
Algorithm for breadth first search :-

1) Create a variable called NODE list and set it to initial state.
2) until a goal state is found or NODE list is empty do.
3) 2.1 Remove first element from NODE list and call it E. If Node list empty quit.
   2.2 For each way that each rule can match the state described in E do.
   2.2.1 Apply the new rule to generate a new state.
   2.2.2 If the new state is a goal state quit and return this state.
   2.2.3 Otherwise add new list to end of NODE list.

Time/space complexity = $O(b^d)$ d = depth, b = branching factor

path is terminated :-

→ Reaches dead end
→ Produces Previous state
→ Becomes larger than limit



Backtracking :-

29/9/22 DFS → Depth First Search

TC = $O(b^d)$
SC = $O(d)$

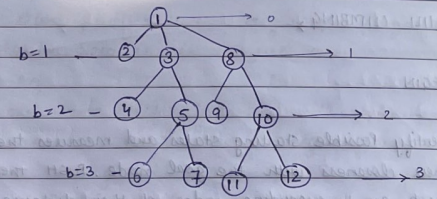
Begin
1) PUSH the starting Node at stack pointed to by the stack - top.

2) while stack is not empty do Begin
Pop stack to get stack top element

If stack topelement = goal
Return sucess & stop
Else
PUSH the children of stack top element in any order into the stack

End while
End

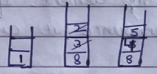
BFS



space complexity = $d(b-1)+1$

d → depth
b → branching factor

If goal = 5 ?



Note
No. of Nodes = $(d+1)$

No. of de nodes by branching factor = $\frac{b^{d-1}}{b-1}$

Time complexity of DFS = $d \rightarrow O(n)$

Time complexity of BFS → $\frac{b^d(b^d+1)}{2(b-1)}$

space complexity = b^d

BFS & DFS → both uninformed techniques

BFS → Breadth First Search

__/__/____

HILL CLIMBING

BEGIN

1) Identify possible starting states and measures the distance of their closeness with the goal node. PUSH them in a stack acc. to ascending order of their distance.

2) Repeat

Pop stack to get the stack top element. If the stack top element is the goal, announce it and exit.
else
PUSH its children into the stack in the ascending order of distance.
until stack empty
END

Pop climbing is simply a loop which continuously moves in the direction of increasing value that is uphill. It terminate when it reaches at the peak where no ~~robot~~ neighbour has a higher value.

The algorithm does not maintain a search Tree. so the ~~no~~ current node data structure need only record the state and its objective function. Hill ~~to~~ climbing does not look ahead beyond ~~a~~ ~~and~~ ~~neighbo~~ immediate neighbours of the current state. Hill climbing is sometimes called Greedy local search because it grabs a good neighbour state without thinking about where to go next.

__/__/____

Hill climbing often makes very rapid progress towards a solution because it is quiet easy to improve into a back state.
There are some difficulties in hill climbing.

1) A local maxstream ~~bt speak~~ which is higher than its neighbour but lower than global maxima. so it is very difficult for greedy algorithm to navigate all node.
2) Plateau is an area of the statesearch where evaluation of the function is flat. It can be flat to local maxima from which no uphill exists.
A hill climbing is not able to find to exit from plateau.
3) Ridge:-
It is an area of the search space which is higher than the corresponding areas and than itself as a slope but the orientation of high regions compare to the set of available nodes and direction in which they move make it impossible to traverse. To overcome this node apply ~~to a more~~ two or ~~not~~ more nodes rules before applying the test.

4/10/22

(3/10)

__/__/____

Informed search or Heuristic search

→ Hill climbing
→ Best First Search

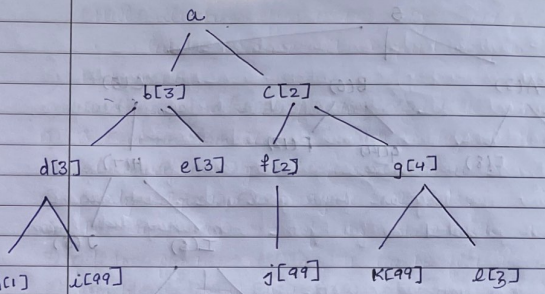
Breadth — BFS - Queue
Depth — DFS - Stack

} uninformed search

⇓

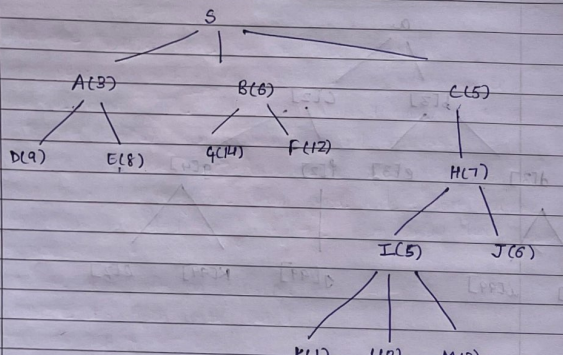
when we blindly look for the data in the provided data structure and stop at its first occurence.

For informed we Generate & Test :-



Hill climbing	Best First Search
→ Start at a	→ Start a
→ children of a [b(3), c(2)]	→ children of a [b(3), c(2)]
→ Best child c[2]	→ Best First → c
→ children of f [j(99)]	→ children of c → [f(2), g(4)]
Not goal	→ consider [b(3), f(2), g(4)]
→ Backtrack to 'g' through 'c'	→ Best first → f(2)
→ children of g →	→ children of f → j[99]
[K[99], l[3]	→ consider [b[3], g[4], j[99]]
→ Best child	→ Best b(3)
g → goal state	→ children of b = [d[3], e[3]]
→ children of c [f(2), g(4)]	→ consider [d(3), e(3), j(99), g(4)]
→ children of f [j[99]]	→ Best first d(3)
	→ children of d(3) → h(1), i(99)
	→ consider (h(1), i(99), e(3), j(99), g(4))
	→ Best h → goal state

Q)



Hill climbing:

1) start at S
2) Children of S [A(3), B(6), C(5)]
3) Best child A(3)
4) children of A [D(9), E(8)]
5) Best child E.
   E → Not goal state
6) Backtrack to C(5)
7) children of C → H(7)
8) children of H → [I(5), J(6)]
9) Best child I(5)
10) Children of I(5) → [K(1), L(0), M(2)]
11) Best child L(L(0)).
12) L → goal state

Backtrack to B(6)
children of B → [G(14), F(12)]
Best child F(12)
↳ Not goal state

Best first search →

1) start at S
2) children of S [A(3), B(6), C(5)]
3) Best ~~child~~ First A(3)
4) children of A(3) → [D(9), E(8)]
5) consider [D(9), E(8), B(6), C(5)]
6) Best first C(5).
7) children of C → H(7).
8) consider [D(9), E(8), B(6), H(7)]
9) Best First B → (6)
10) children of B → [G(14), F(12)]
11) consider [D(9), E(8), G(14), F(12), H(7)]
12) Best ~~child~~ → First → H(7)
13) children of H → I(5), J(6)
14) consider [D(9), E(8), G(14), F(12), I(5), J(6)]
15) Best First I(5).
16) children of I → K(1), L(0), M(2).
17) consider [D(9), E(8), G(14), F(12), K(1), L(0), M(2), J(6)]
18) Best First → L(0)
    ↳ goal state

__/__/____

Algorithm for best first search

1) Put the initial node on a list OPEN.
2) If (OPEN is empty) or (OPEN = GOAL) terminate.
3) Remove best Node from OPEN and call this Node a.
4) If (a = GOAL) then terminate and success.
5) else if node has successors generate all of them find out how far they are from goal node. Sort all the children generated so far by the remaining dist from goal.
6) Name this list as CLOSED.
7) Replace it. open with closed.
8) Go to step 2.

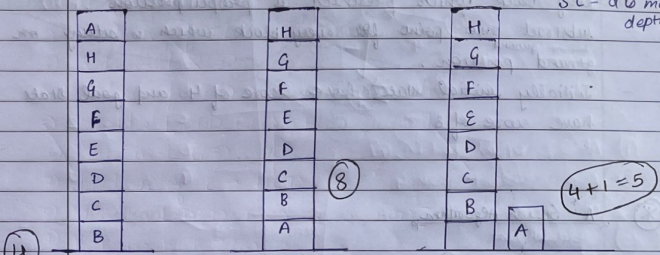
6/10/22

UNIT-01

Example of Hill climbing

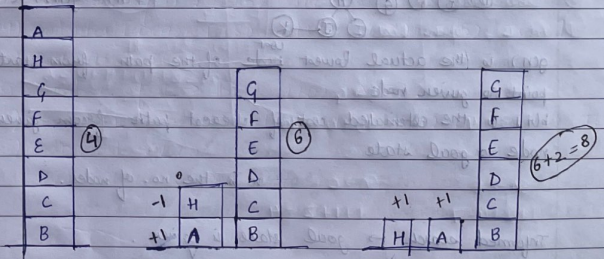
$TC = O(b^d)$

$SC = d$ is max depth



Initial state Goal state Intermediate state

Three Possible moves



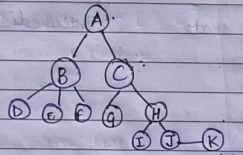
we are using two operators in the given example:-

R_1 operator - pick up one block and put it to the table

R_2 operator - pick up one block and put it on other.

- For the calculation of ~~add~~ at heuristic function one point for every block which presented at proper position.
- substract one point for every block which is sitting on wrong ground position.
- Initially initial state have score of 4 and goal state have score of 8

(*) A* Search Algorithm



K is the goal state

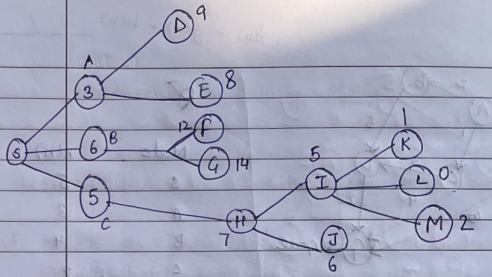
$$f(n) = g(n) + h(n)$$

g(n) is the actual lowest ~~node~~ cost of the path from start point to given node.

h(n) is the estimated cost of cheapest path from given node to goal state

n is the no. of nodes.

Informed search → goal state is given.



From (S → to K)

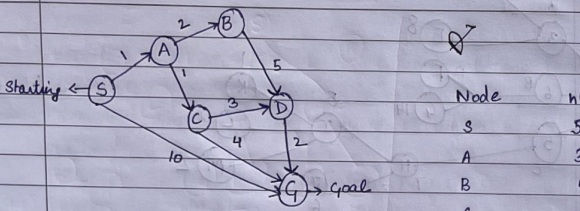
H(n) = 17

g(n) = 5

A* algorithm

1) Place starting Node in OPEN LIST.
2) check if the OPEN LIST is empty or not. If list is empty then return failure & stop.
3) ~~Select~~ If OPEN ~~is~~ LIST is not empty then select node from the OPEN LIST which has smallest value of heuristic function. (g(n) + h(n)) if node n is goal node then return success & stop.
4) otherwise expand node n and generate all of the successors and put n into CLOSED LOOP for each successor of n check whether n is already in Open or closed list. If not then compute function for n and place into OPEN LIST.
5) else if node n is already in OPEN/LIST OR CLOSED then it should be attached to back pointer which reflect lowest value of g(n) node.

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



Node	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

① At S, open list [A, G]
② A is smallest than G
Move to A
③ closed list
④ At A, open list [B, C]
⑤ C is smaller than B
Move to C
⑥ close list at C
⑦ At C, open list

S → A: 1+3 = 4
S → G: 10+0 = 10
SA → B: 1+2+4 = 7
SA → C: 1+1+2 = 4
SAC → G: 1+1+4+0 = 6
SAC → D: 1+1+3+6 = 11

goal reached → (6)

S Start →
A or G
f(A) = 1+3 = 4
f(G) = 10+0 = 10
B or C
f(B) = 1+2+4 = 7
f(C) = 1+1+2 = 4
D or G
f(D) = 3+6 = 11
f(G) = 2(1+1)+4+0 = 6

__/__/__

Advantages & Disadvantages of A^K :-

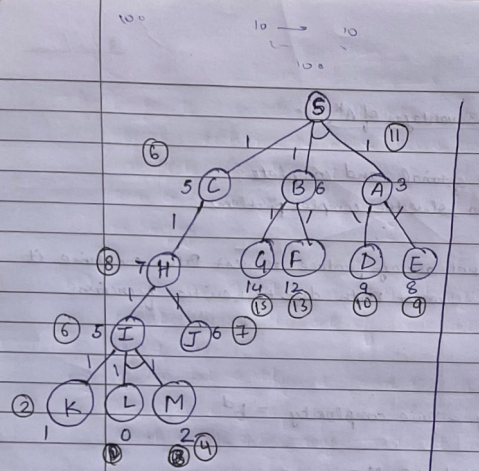
Ad i) A^* algorithm is optimal and complete.
2) This will help to solve the complex problems.

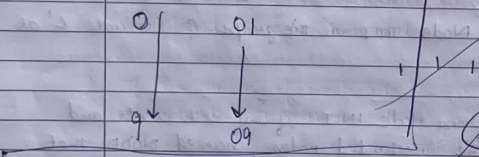
Disad i) It ~~always~~ does not always produce shortest path sometime it will not give shortest path due to heuristic function evaluation.

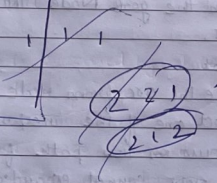
space complexity = Time complexity = b^d.

AO* And OR Algo

1) Given the goal Node INIT in the graph G Evaluate h' at INIT.
2) Repeat
(a) Trace the arc from node INIT if any such exists and select one of the unexpanded nodes, named NODE that occurs on this path for expansion.
(b) If NODE cannot expanded. Then assign FUTILITY as h' value of NODE indicating not solvable. else go to successor.
Begin
i) Append successor to go graph G.
ii) If successor is a terminal node. Then label it solved and set its h' value 0.
iii) If SUCCESSOR is not terminal node. Then estimate h'.







Constraint satisfaction

2	7	6
9	5	1
4	3	8

$\rightarrow m_{11} + m_{12} + m_{13} = 15$

$\rightarrow m_{21} + m_{22} + m_{23} = 15$

$\rightarrow m_{31} + m_{32} + m_{33} = 15$

Magic Square

m_{11}	m_{12}	m_{13}
m_{21}	m_{22}	m_{23}
m_{31}	m_{32}	m_{33}

$\downarrow m_{11} + m_{21} + m_{31} = 15$

$\downarrow m_{12} + m_{22} + m_{32} = 15$

$\downarrow m_{13} + m_{23} + m_{33} = 15$

$\searrow m_{11} + m_{22} + m_{33} = 15$

$\swarrow m_{13} + m_{22} + m_{31} = 15$

→ constraints can be used to

- check the correctness of a partial solution.
- calculate some parameters using given constraint
- choose some parameters and fix the value of that for further evaluation.
- The first two constraint can be evaluating using gener software methods i.e called constraint propogation.

If we evaluate first two constraint then third constrain automatically goes powerful.

The constraint may be two types:-

1) First those given as a part of problem domain.
2) Those which are assumed and maybe changed later using backtracking.

← Algorithm →

__/__/____

→ 1) Propagate available constraints. First SET OPEN to set all object which must have value assigned to them in a complete solution.
Then do until an inconsistency is detected or until OPEN is empty.

1.1) Select an object from OPEN & strengthen as much as possible with the set of constraints which apply on object.

1.2) If this set is different from the set which has assigned the last time to object was examined then add to OPEN all object which share any constraint with object.

2) If the union of the constraints discovered by above solution is defined solution then quit.

3) If the union of constraints discovered by above solution is contradiction then to return failure.

4) If neither of above action occur then it is neccessary to make a guess at something in order to proceed.

To do this loop until a solution is found or all possible solution.

__/__/____

← Statement of constraints →

```
  S E N D
  M O R E
M O N E Y
```

1) Values are to be assigned from 0 to 9.
2) No two letters should assign same value.
3) If some letter occurs more than once, it must be assigned some value.
4) The sum of digits must be acc. to given problem.

$M=1$

$S = 8$ or 9

$O = 0$ or $1 \rightarrow O = 0$

$N = E$ or $E+1 \rightarrow N = E+1$

$C_2 = 1$

$N+R > 8$

$E < 9$

$C_1 = 0$ / $C_1 = 1$

$E = 2$

$N = 3$
$R = 8$ or 9
$2+D = Y$ $\quad 2+D = 10+Y$

$C_1 = 0$

$2+D = Y$
$N+R = 10+E$
$R = 9$
$N = E$

$C_1 = 1$

$2+D = 10+Y$
$D = 8+Y$
$D = 8$ or 9

$D = 9 \quad Y = 1$

$D = 8 \quad Y = 0$

Mean end Analysis

Till here, the searching strategies only search in one direction either forward or backward.

Mean end analysis using bi-directional search:-
It means in this technique part of the problems are independently solved using forward and backward search simultaneously then the sub problems solutions are joined to obtain complete solutions.

It detects diff b/w current state and goal state and once such a diff is identified an operator that can reduce the difference must be found.
we apply backward chaining in which operators are selected than sub goals are setup to establish pre condition of the operator.
This is called operator sub-volume.

There are certain typical initial condition on which these rules or operators can be applied. These are called pre-condition. The newly generated states after rules are applied are called post condition.

The table that keeps track of the difference between current state and goal state is called difference table.

example :- GPS
STRIPS

Steps in MEA

1) Involves detection of difference b/w current and goal state.
2) Once difference identified then found an operator for reducing difference.
3) Some time operator cannot applied directly to current state.
4) Sub-problem of getting to state where operator applied.
5) Operator may not give result in goal state.
6) 2nd sub-problem of getting new state from goal state.
7) MEA process applied recursively.

Three primary goals of MEA

1) Transform
2) Reduce
3) Apply