Unit - I

AI - A modern approach } Russell I Norvig

AI → CS → Automation of intelligent s/m facts like humans.

Approaches:

Acting Humanity. . Acting Rationally

Thinking Humanity Thinking Rationally.

Acting Humanity

Turning Test

capatutities.

* Natural language processing
* Knowledge rep
* Automated reasoning
* Machine learning.

Thinking Humanity

Cognitive modeling approach.

Thinking Rationally.

Logical reasoning.

Ram is man.

Acting Rationally.

Agent acts

Problem solving Agent → sensor → Actuators.

Robot → sensors & motors

S/w agent → keystrokes, screen.

Internet shopping Agent → Activator → display.

HTML code.

Taminology:

percept: → I/p ~~Introduction to AI~~

→ Birds are flying.

persept seq :-

Agent function → map

Problem solving. Agent

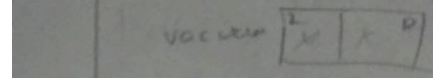
1. problem definition.
2. problem Analysis.
3. Knowledge Representation.
4. Problem solving.

Agent → set of action → succesfull status

Steps:

1. Goal setting
2. Goal formation.
   i) current state
   ii) Tabulet performance.
3. Problem formation. seq of action
4. Seach in unknown environment
5. Execute.

Vacuum World



Percept:

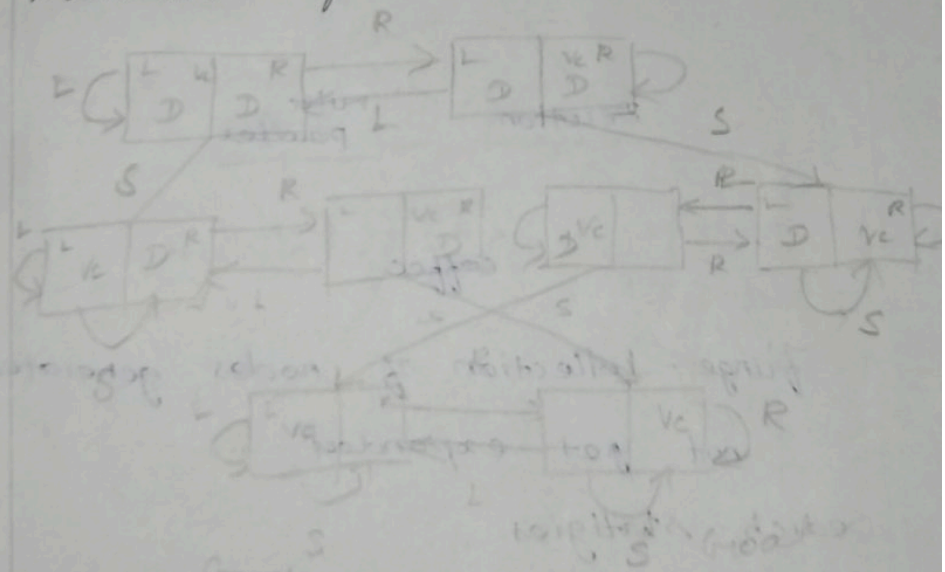
Percept: (L, Dirty) (or) (R, Dirty)

Actions

1. Left
2. Right
3. Suck (S)
4. NOP

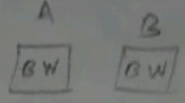
1. Initial state
2. Succesor function
3. Goal list
4. path cost.

Transition diagram:



Uniformed search strategies.

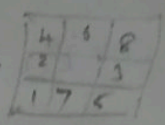
Toy Problem.



Puzzle problem

$2^3 = 8$

1. L
2. R
3. Black pickup



Goal test

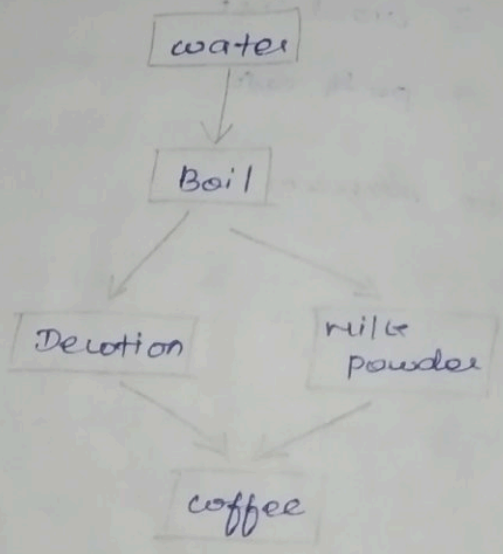
path cost.

Cryptarithmetic

→ Numeralagi

Searching for solution.

state space representation of a problem.



Fringe - collection of nodes generated not yet exponded.

Seach Statigies.

1. Uninformed seach (blind seach)
2. Informed seach (Hemistic seach

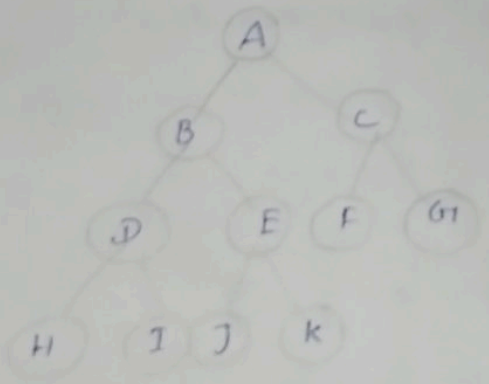
y. x → 15

Uniformed search.

1. BFS Breadth first search
2. DFS Depth first search
3. Depth limited
4. Iterative Deepening DFS
5. Bidirectional
6. Uniform cost.

Performance level

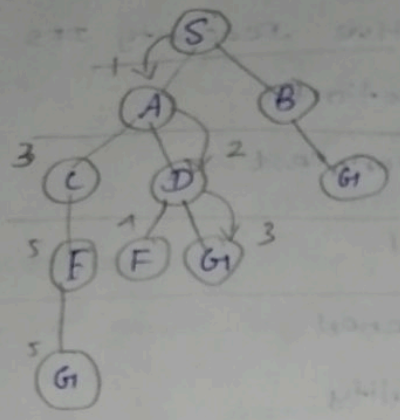
1. Completeness
2. Optimality
3. Time complexity
4. Space



Uniformed cost search (UCS)

weighted tree.

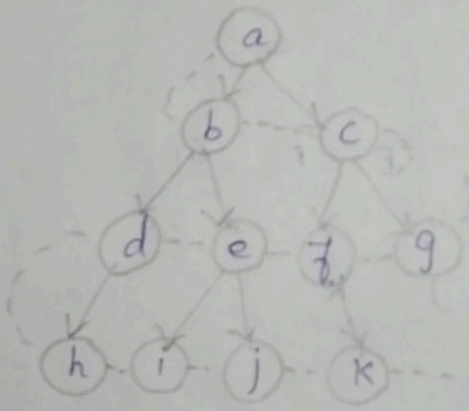
Priority queue.



Optime searching algorithm is minimum cost.

IDDFS

Breadth first search.



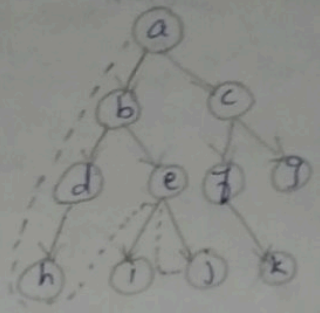
IDDFS – It means combination of

BFS & DFS

BFS & DFS is blind search (or)

Informed search

Best in DFS

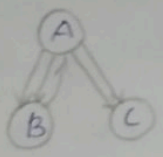


Let us see how?

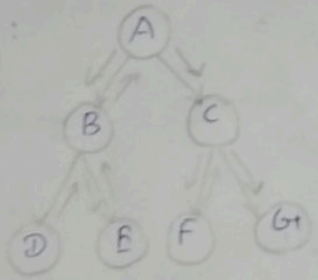
eg:

Depth

Step1: (A) 0 (A)

Step2: 

1 (A, B, C)

Step 3: 

2 (A, B, D, E, C, F, G)

Algorithm:

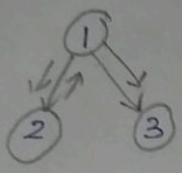
Step 1: set search_depth = 0

Step 2: Conduct a depth first search to a depth of SEARCH_DEPTH if the solution part is found then it returns it & exit

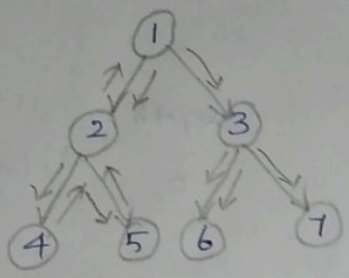
Step 3: otherwise increament 'SEARCH_DEPTH' by 1 and go to step 2.

goal node 11.

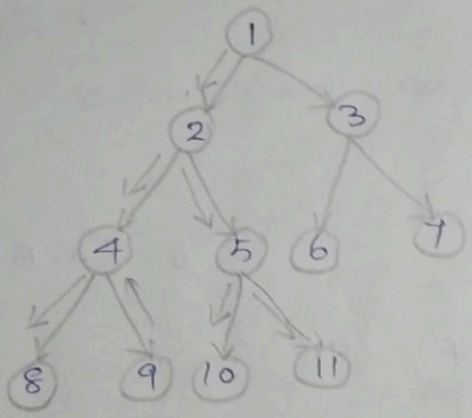
		Depth	Itration
step 1:	(1)	0	1st - (1)



1 2nd (1, 2, 3)



2 3rd (1, 2, 4, 5, 3, 6



3 4th

(1, 2, 4, 8, 9, 5,

10, 11, 3, 6, 7

So here we have compared two types of uninformed search BFS & DFS.

* DFS is efficient in terms of space by required some cut of depth in other two force depth packing.

* when the "solution was not found then it goes into the infinite rule.

* And BFS is get into the shortest solution path But it required large amount of space because all leaf node have to kept into the memory.

* So by taking that advantages of both BFS & DFS the IDDFS gives us the obtain solution compare to BFS and DFS.

Bidirectional search

Definition:

The idea is two run start and goal simultaneous.

BFS Informed.

It always select the path which appears best at that moment

It is combination of DFS & BFS.

It uses the heuristic function.

$h(n) \leq h^*(n) \rightarrow$ estimated cost.

The generally greedy best first algorithm is implemented by priority que

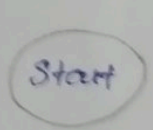
Bidirectional:

Bi- two

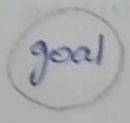
directional - way.

Two types.

* forward
* backword.



forward search

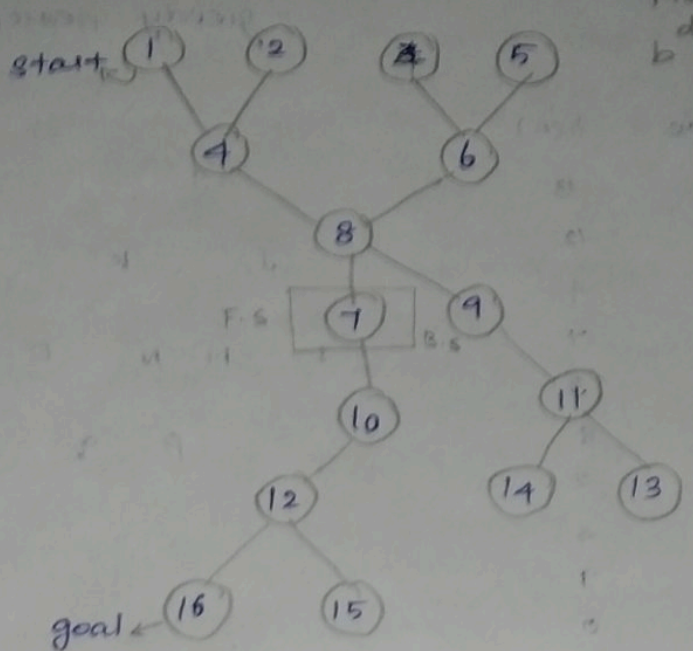


node.

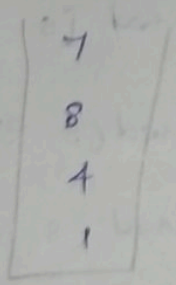
eg: Algorithm IDFS

Motivation

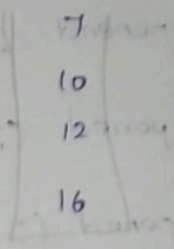
$b^{d/2} + b^{d/2} < b^{d}$



Forward stack status



Backword stack status



output: 1 4 8 7 10 12 16

Performance Evaluation.

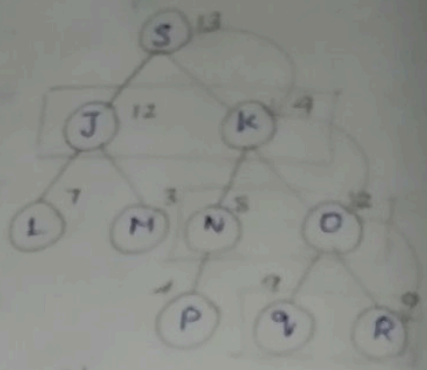
- Completeness
- Optimality.
- Time Complexity - $O(b^{d/2})$

where, b → branding factor (or) of nodes.

d → depth of the trees.

BFS - Best first search - (informed)
- greedy search

node	h(n)
S	13
J	12
K	4
L	7
M	3
N	8
O	2
P	4
Q	9
R	0



Intil = S

1st openend [J, K] . closed [S]

2nd opened (J) closed (S R]

3rd opened (J, N) closed (S, K, O]

opened (J, N, Q) closed (S, K, O, R]

A* Algorithm.

best form by search.

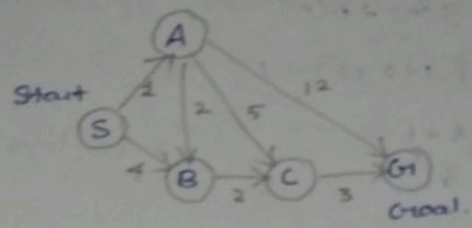
$f(n) = g(n) + h(n)$

$f(n)$ - estimated total path to reach the go

$g(n)$ - cost to reach the node.

$h(n)$ - Heuristic value.

Eg:



node	h(n)
S	7
A	6
B	2
C	1
G1	0

S → A

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

$= 1 + 6$

$= 7$

S → B

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

$= 4 + 2$

$= 6$

S → A → B

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

$= (1+2) + 2$

$= 3 + 2$

$= 5$

S → A → C

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

$= (1+5) + 1$

$= 6 + 1$

$= 7$

S → B → C

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

$= (4+2) + 1$

$= 7$

$S \rightarrow A \rightarrow B \rightarrow C$

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

$= (1+2+2)+1$

$= 5+1$

$= 6$

$S \rightarrow A \rightarrow G_1$

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

$= (1+12)+(0)$

$= 13$

$S \rightarrow A \rightarrow C \rightarrow G_1$

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

$= (1+5+05)+0$

$= 9$

$S \rightarrow B \rightarrow C \rightarrow G_1$

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

$= (4+2+3)+0$

$= 9$

$S \rightarrow A \rightarrow B \rightarrow C \rightarrow G_1$

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

$= (1+2+2+3)+0$

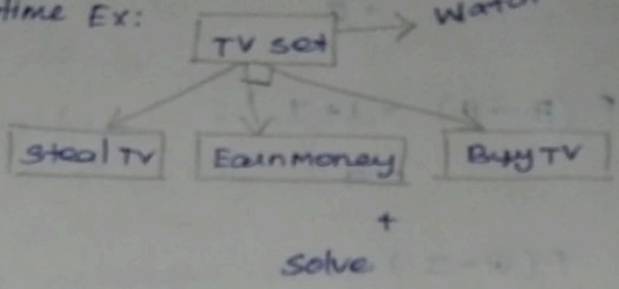
$= 8.$

Informed search (Heuristic search)

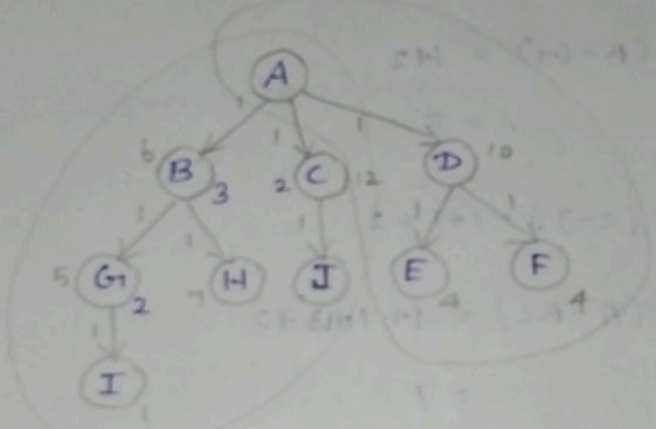
AO* search Algorithm.

And - OR

Real-time Ex:



Ex:



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

$g(n)$ - actual value

$h(n)$ - heuristic value.

F(A-D) / F(A-BC)

F(A-D)

= 1+10 = 11

F(A-BC)

= 1+1+6+12

= 20.

$F(A-D-EF) = 1+1+4+4$

$= 10$

$F(B-G) = 1+5$

$= 6$

$F(B-H) = 1+7$

$= 8$

$F(G-I) = 1+1$

$= 2$

$F(B-G) = 1+2$

$= 3$

$F(C-J) = 1+1 = 2$

$F(A-BC) = 1+1+3+2$

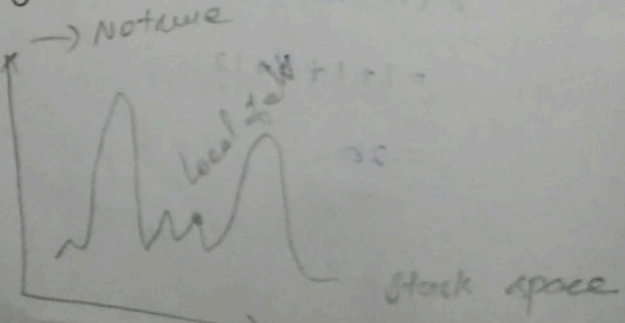
$= 7$

Local search and opt problems.

Heuristic. Advantage: Reduce memory.

1. Hill Climbing
2. Simulation Annealing
3. Local Beam search.
4. Genetic Algm (GA), Genetic algorithm

Hill climbing search



Function Hill climbing (problem) returns a state is local max.

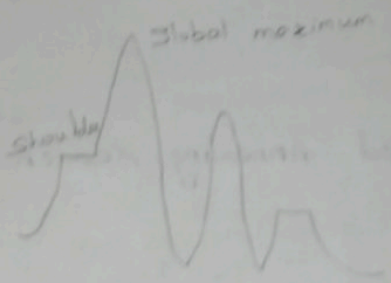
current ← make ← node (problem - Initial - state)

loop do

neighbour ← a highest valued successor of current

if neighbour value ≤ current value then return current state

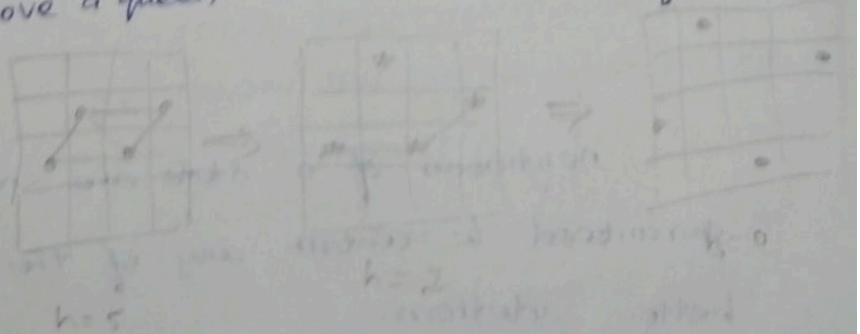
current ← neighbour.



Ex. n-queens

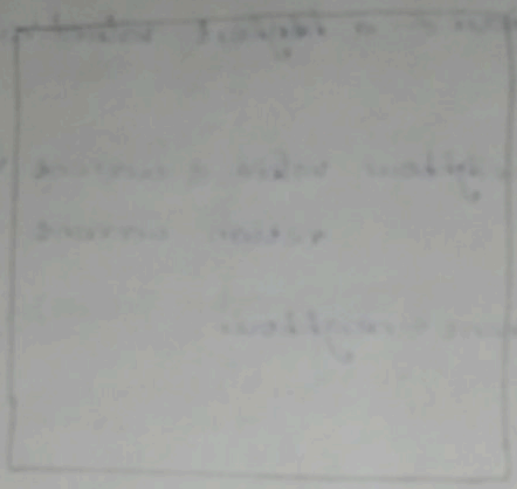
Put n queens on an n x n board with no two queens on the same row, column, or diagonal.

Move a queen to reduce number of conflicts

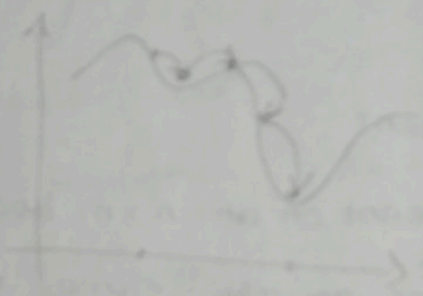


Ex: 8 queens

h = number of pairs of queens that are attacking each other either directly or indirectly (h=17 for the above state)



Simulated Annealing search



To avoid being stuck in a local maxima

neighbours of a state are not guaranteed to contain any of the existing better solutions.

Genetic Algorithms

Inspired by evolutionary biology and natural selection such as inheritance.

Evolves toward better solutions.

A successor state is generated by combining two parent states rather by modifying a single state.

Start with k randomly generated states (population). Each state is an individual

Algorithm
function
GENETIC-ALGORITHM (Population, FITNESS-FN)
return an individual

Inputs
population, a set of individuals
FITNESS-FN, a function that measures the fitness of an individual

repeat
new population ← empty set
for i = 1 to SIZE(population) do
x ← RANDOM SELECTION (population, FITNESS-FN)
y ← RANDOM SELECTION(population, FITNESS-FN)
child ← REPRODUCE (x, y)
if (small random probability) then child ← MUTATE (child)
and child to new-population

population ← new_population

until some individual is fit enough, or
enough time has

return the best individual in population,
according to FITNESS-FN

function REPRODUCE(x, y) returns an individual
inputs x, y parent individuals

n ← LENGTH(x); c ← random number from 1

return APPEND SUBSTRING(x, 1, c), SUBSTRING
(y, c+1, n))

Fitness function:

(min = 0, max = $8 \times 7/2 = 28$)

$$\frac{n(n-1)}{2}$$

no of non attacking pairs of queen

$\therefore = 24 | (24+23+20+11)$