CS3EA01 Artificial Intelligence

Hitesh Kag

Syllabus

Unit I:

Introduction to artificial intelligence, various types of production systems, Characteristics of production systems, Study and comparison of breadth first search and depth first search techniques.

Unit II:

Optimization Problems: Hill-climbing search Simulated annealing like hill Climbing, Best first Search. A* algorithm, AO* algorithms etc and various types of control strategies, Heuristic Functions, Constraint Satisfaction Problem.

Unit III:

Knowledge Representation, structures, Predicate Logic, Resolution, Refutation, Deduction, Theorem proving, Inferencing, Semantic networks, Scripts, Schemas, Frames, Conceptual dependency.

Unit IV:

Uncertain Knowledge and Reasoning, forward and backward reasoning, monotonic and non-monotonic reasoning, Probabilistic reasoning, Baye's theorem, Decision Tree, Understanding, Common sense, Planning

Unit V:

Game playing techniques like minimax procedure, alpha-beta cut-offs etc, Study of the block world problem in robotics.

Artificial Intelligence

Learning
Reasoning
Understanding
Self Correction
Mental Ability

```
if (a%2==0)
    even
else
    odd
```

Def
Intelligence exhibited by the machines (Comp / Algo / System / Sw)

Simulation of Intelligent behaviours.

Sanfoundary
Quiz 1, 2, 3
Asg - 1, 2

Applications | Components

Machine Learning ⇒ NN ⇒ Set of Programs → Algo

Speech Recognition
Pattern Recognition

Supervised Learning

Hidden Layers

I/p Layer

O/p Layer

Perception

touching
smelling
listning

Unsupervised Learning

Cat / Dog

2 → 10-9 O/p

2
2

Class labels

Reinforcement

Natural language processing

Robot ⇒ Action

Pattern Recognition → Computer Vision

2D & 3D

Regular shapes
Irregular shapes

Digital Images Processing

Camera Sensor ⇒ Agent

2013 Deep Learning

Complex

Facebook

tag

9 hidden layer

Game Playing | Problem Solving ⇒ Searching

DFS BFS

Theorem Proving

Decision making

Behavior

Expert System

Chatbots

Medical Consultants (fever)

CADET

Applications

Finance

Education / Design

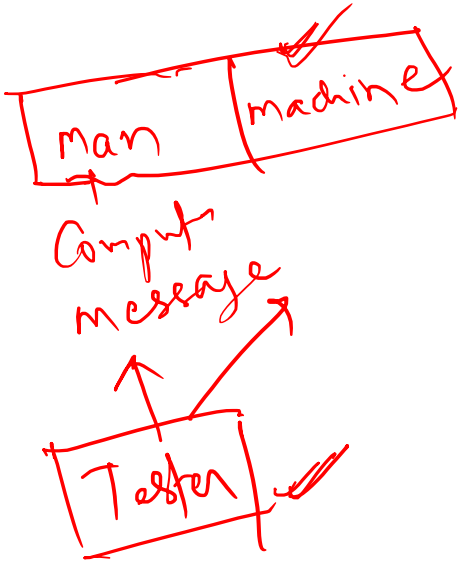
Health / Diagnosis

1. 2 Searching
3. Knowledge Representation
4. Prob. Reasoning
5. Game Playing

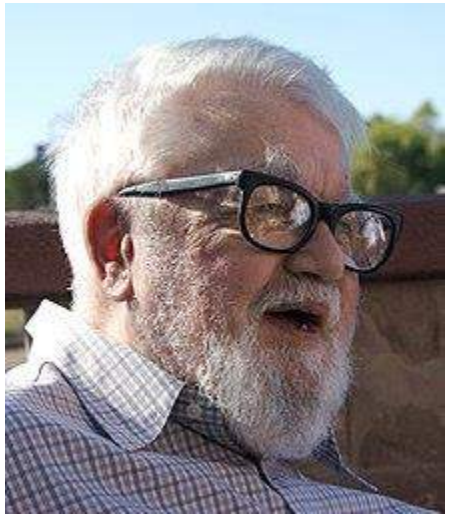
1950 Alan Turing

Can m/c think

Turing Test



Intelligent Behaviour is an ability to achieve human level performance in all cognitive tasks & sufficient to fool the interrogator

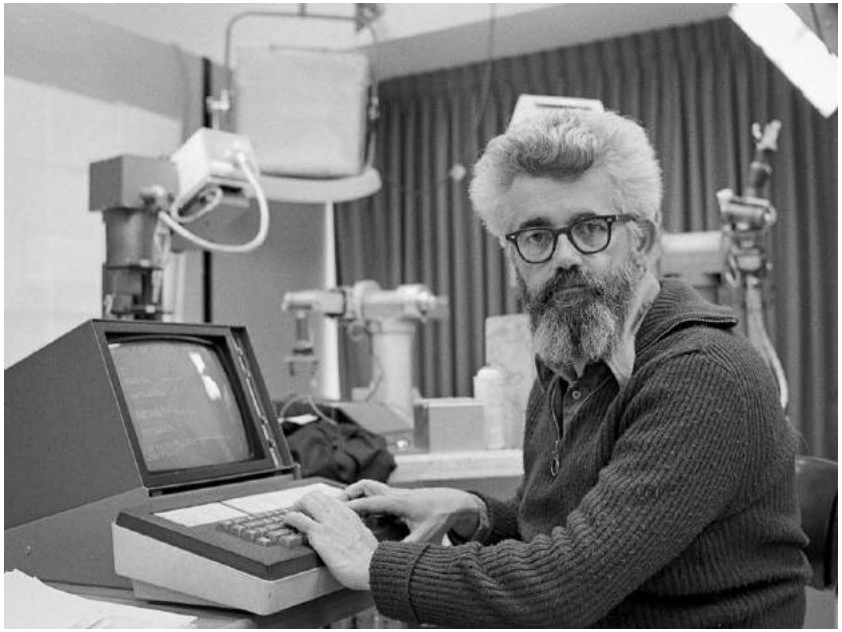


1956 father of AI John McCarthy

AI

- Algorithms
- Computing Power
- Data

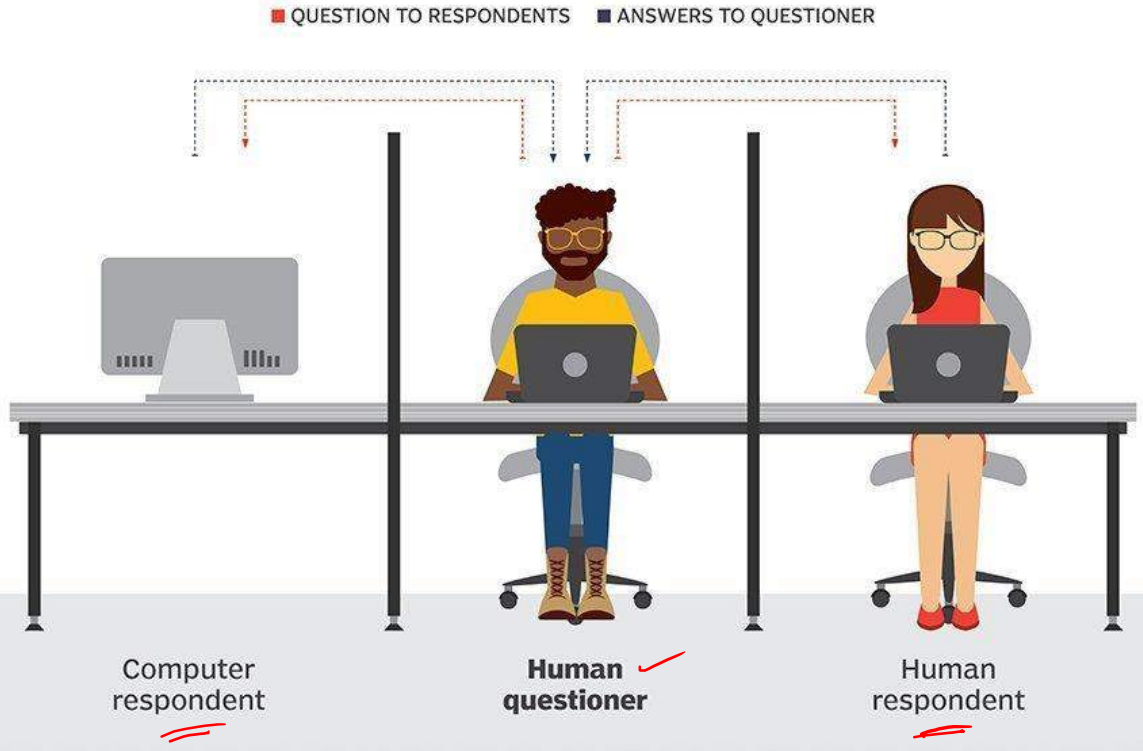
2000 chess



1956 father of AI John McCarthy

Turing test

During the Turing test, the human questioner asks a series of questions to both respondents. After the specified time, the questioner tries to decide which terminal is operated by the human respondent and which terminal is operated by the computer.



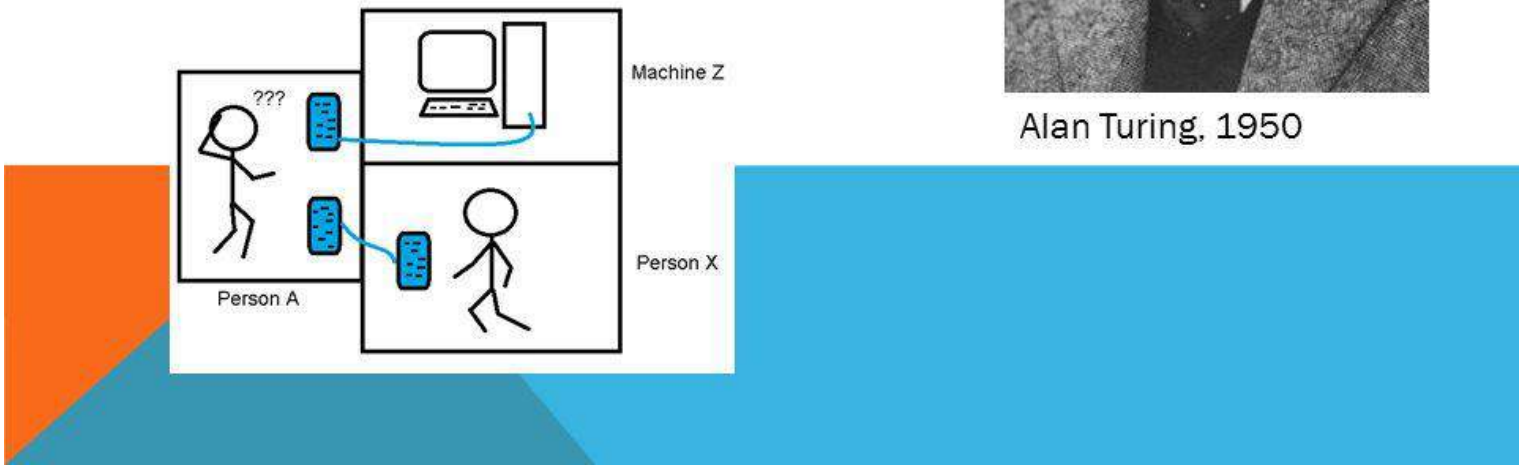
"CAN MACHINES THINK?"

What is the Turing Test?

- Hypothetical test called the imitation game to determine if a machine has artificial intelligence.

- Turing predicted that by 2000 it would be possible to fool an average interrogator with the probability of at least 30%

Alan Turing, 1950



Reasoning

Logic 1950

A* 1966

Prob. Reas 90-2010 ⇒ Algo + Comput Power + Data

AI

Google Deep mind 2016 19×19 Board

Formal Cognitive task

Game - chess
checker

Math. - Logic
- Calculus
- T-P.

Expert task

Engg. - Design
- fault find
- manuf

Medical - Diagnosis

- Stock market Predict

perceptual task

C. Vision

Speech

NLP Robot Control

Agent :- Anything that can perciere its env. through sensor, effectors.

Rational Agent . .

Logic

Rational thinking is the ability to consider the relevant variables of the situation & to access, organize & analyse relevant info (fact, opinion, judgement) to arrive at a sound conclusion

Rational Agent - An agent should strive to do right things, based on what it can percieve & the action it can perfo

(Ideal)

Performance

Agent type ⇒ Car Driver

Percepts ⇒ Speedom, GPS, Microphone, Camera

Action ⇒ Steer, Control, Accelerate, Brake

Goal: Safe

Envi: Road, traffic

Weak AI — machine could act as if they are intelligent

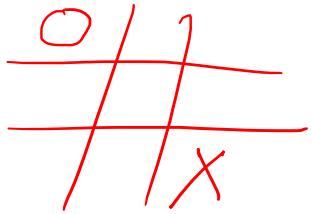
Strong AI — m/c that act intelligent have to think like intelligent too

Problem Solving

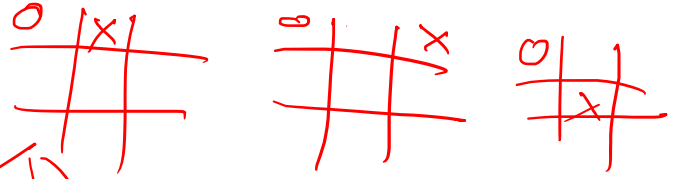
① Define the problem Precisely

State

All info about a particular env.



DFS



State space Repr / tree

(i) Initial State

(ii) final state

(iii) Define a state space that contains all the possible configuration of the relevant objects

Output — Path — Initial state or start state ⟹ Goal state

(2) Analyze the problem :- Various technique for solving the prob

(3) Isolate & Represent the task knowledge (Knowledge DB) that is necessary to solve the problem

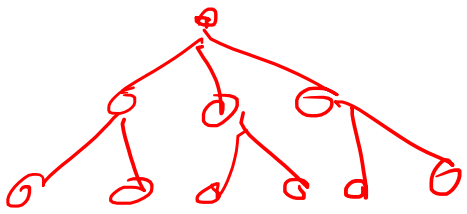
(4) Choose the best prob. solving technique & apply it to the particular

(5) Specify / Generate the set of rules / Production Rule that describes the actions.

Production System

A P.S. is a set of programs or mechanism typically used to provide solution to particular problem which are concerned with AI

① Production Rules



Search is General mechanism that can be used when no more direct method is known

$C_i \rightarrow A_i$

Conditional Part → Actions

- functions
- call to another fun
- call to another Production Rule

Eg:

② Knowledge DB → Info appropriate for particular task.

③ Control strategy

Specifies the order in which rules will be compared to the DB & way of resolving conflict.

→ BFS, Hill Climbing.
DFS
A*

Characteristic

Types of Production System

(1) Monotonic Production syste
(monotonic Reasoning)

Adding more knowledge does not reduce the set of propositions that can be derived.

A, B, C, D ⇒ ⊨

if it rains, I will not goto college

A →

A ∧ C → ~B

Anything that could be concluded,

Clause

If a conclusion is not invalidated by adding some more knowledge.

(2) Non Monotonic P.S. → If some conclusions can be invalidated by adding some more knowledge.

(3) Partially Commutative P.S. → A system in which the app^n of a particular sequence of rules transforms state X into state Y, then any permutation of these rules that is allowable also transforms state X into state Y.

Rule 1 — A
Rule 2 — B
Rule 3 — C
Conclusion — D

C
A
B
Conclusion — D

Ex in the op

~~A+B*C~~

X

2+3*5 = 5*5 = 25 ≠ 17

2*5+2 = 15+2 = 17

(4) Comm. Production System → Both monotonic & Partially Comm. (1 & 3)

App

1) Rich & Knight

Expert System

Automated Planning

Monotonic

Non-monotoni

Ignorable Problem

Partially comm.

Theorem Proving

Robot Navigation

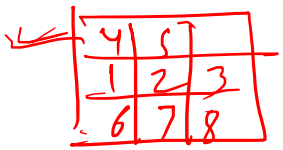
Non-Partially comm.

Chemical Synthesis

Bridge Games

Can use without Backtracking

Commutative → Ignorable Problem (Theorem Proving)



↓

Creating new thing rather than old things

can use without backtracking

NEN

Partially Com + Non-mono (Robot Navigation)

ENN

Order of the appⁿ is not critical

8 puzzle Problems

Non-Partially + Monotonic (Chemical Synthesis)

Features of Production Systems

1. Simplicity ⇒ if then structure
   - Knowledge Representation improves readability
2. Modularity ⇒ Addⁿ / Deletion is easy
   - discrete (knowledge)
3. Modifiability → Updations of new rules
4. Knowledge Intensive ⇒ stores pure knowledge

Issues | Challenges | Drawbacks.

1. Inefficiency ⇒

   A well devised [control strategy] to reduce this.

2. Opacity :→ 10 rules

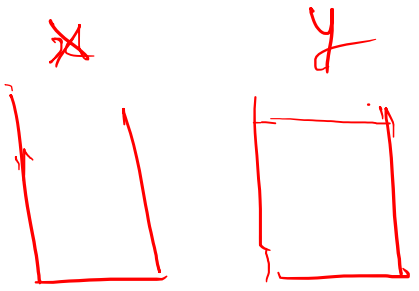
   Conflict Resolution — More priority to rules has less opacity

3. Absence of learning ⇒ Rule-based Productions.

   Rule 1, Rule 2 = Result (future use)

   does not use result for future use, hence it does not exhibit learning capabilities.

Water Jug Problem



Goal we need 2L of water in Jug X

Rule applied	x	y
—	0	0
2	0	3
9	3	0
2	3	3
7	4	2
5	0	2
9	2	0

30 rules

x	y	
0	0	—
4	0	1
1	3	8
0	3	6
1	0	10
4	1	1
2	3	8
2	0	

$x=2$

$4-2=2$

① $x<4$ $(x,y) \rightarrow (4,y)$

② $y<3$ $(x,y) \rightarrow (x,3)$

③ $x>0$ $(x,y) \rightarrow (x-p,y)$

④ $y>0$ $(x,y) \rightarrow (x,y-p)$

⑤ $x>0$ $(x,y) \rightarrow (0,y)$

⑥ $y>0$ $(x,y) \rightarrow (x,0)$

⑦ $y>0$, $x+y \geq 4$
$(x,y) \rightarrow (4, y-(4-x))$

⑧ $x>0$, $x+y \geq 3$
$(x,y) \rightarrow (x-(3-y), 3)$

⑨ $x+y \leq 4$, $y>0$
$(x,y) \rightarrow (x+y, 0)$

⑩ $x+y \leq 3$, $x>0$
$(x,y) \rightarrow (0, x+y)$

Control Strategies Requirements

To decide which rule to apply next during the process of searching for a soln to a problem.

1. It cause motion :-
2. It should be systematic

It should take less time

Search Strategies

⇒ Uninformed Search

1. BFS
2. DFS

Depth limited search
Uniform cost search
Beam Search

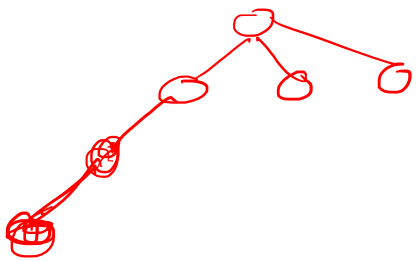
⇒ Informed Search

Hill Climbing
Best first search
A*
AO*

A Search strategy decides the order of the node expansion

✓ State Space Tree

- Completeness
- Time Complexity
- Space Complexity
- Optimality
- Systematicity



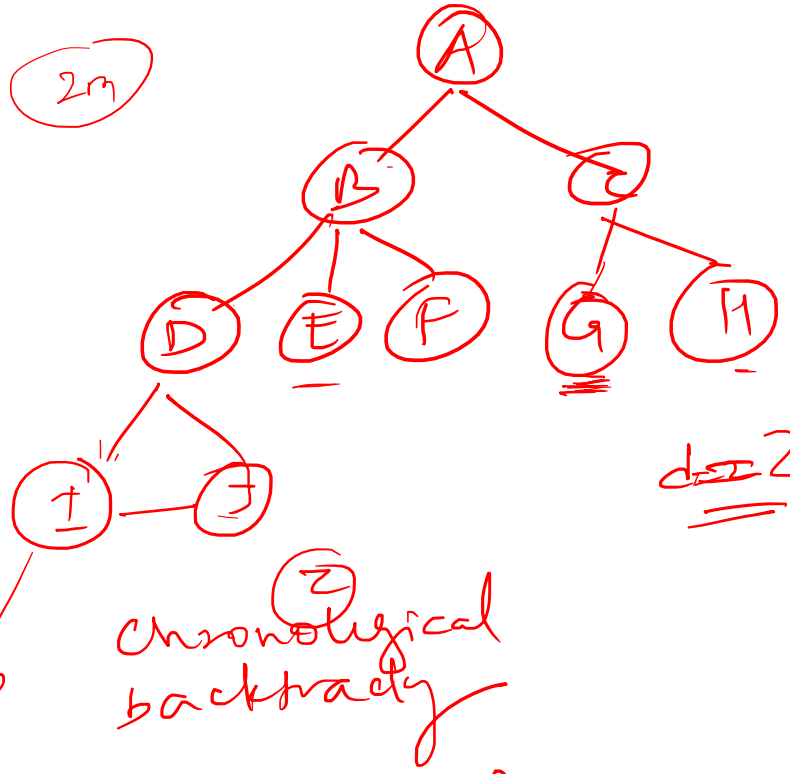
d = 10

b - max. branching factor

d - depth (least cost)

m - max. depth of state space (∞)

Depth first Search



maintain stack of nodes to visit

A B D I J E F C G H

A C G

depth finite

Complete ? No

Optimal ? No

Time Comp - $O(b^m)$

Space Comp - $O(bm)$

dep = 2

trap

chronological backtracking

Adv: Requires less memory since only nodes on the current path are stored

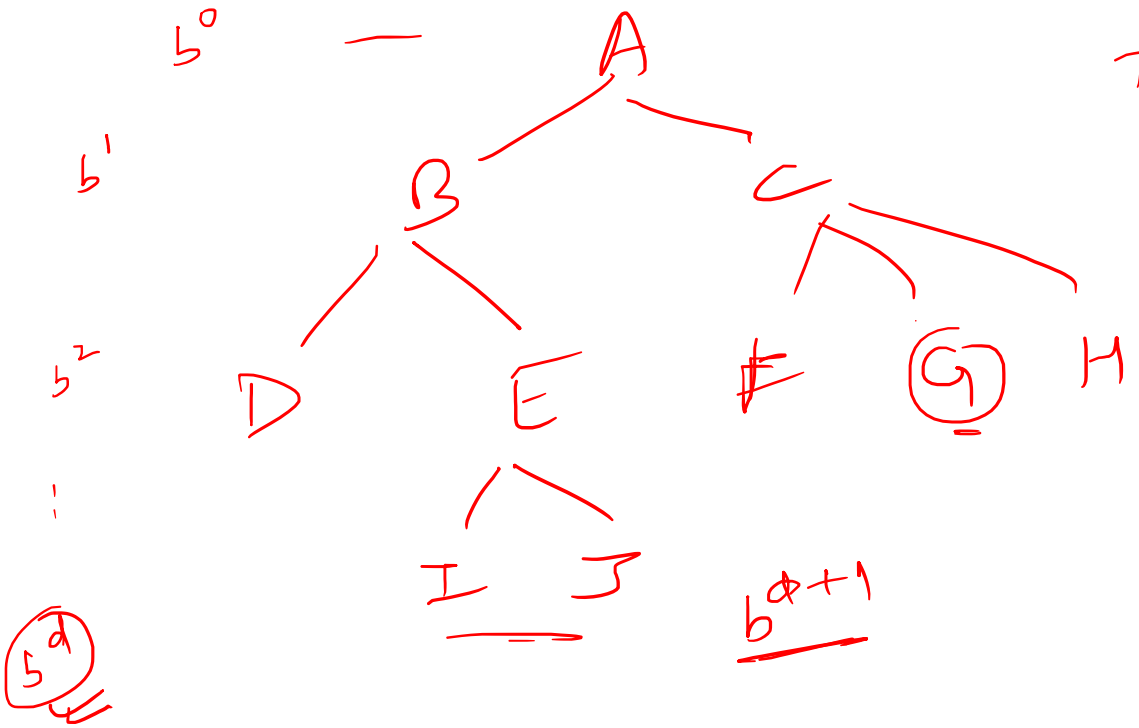
- Find the sol^n without exploring much of the state space

dis: It may get trapped in a dead-loop.

Breadth first search (BFS)

maintain Queue of nodes to visit

Layerwise



n nodes

traversal ABCDEFG

draw Storage Req. more

Adv ① It doesn't get trapped in a dead loop

② If always find the soln (complete)

Space $\rightarrow$ (b^d)

time $\rightarrow$ (b^d)

BFS

level by level

level order traversal

Queue

Slower

Complete

Complexity

← Greedy Algo (lowest node first)

Local optimum soln

App ⇒ Shortest Path
Prim | Kruskal

Web crawling

fb

DFS

deep

Preorder

Stack

fast

Not a complete

Complexity

Backtracking (explore one path until dead end)

App find strg connected components

Preorder

- Uniform Cost Search - edge wt = 1

Blind Search

- Beam search = storage n nodes
(BFS)

Complete – NO
Optimal – NO

2 10

Exponential

- Depth limited Search
(DFS)

d is fixed

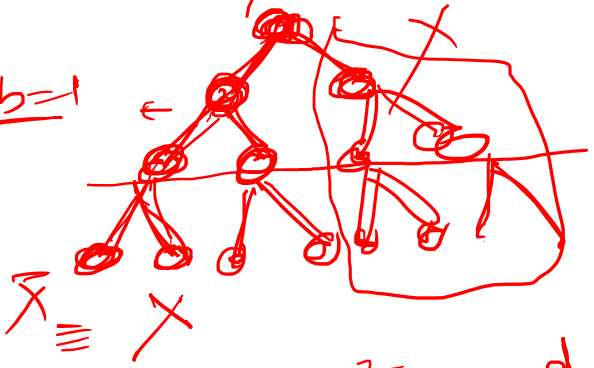
- Iterative Deepening Search

Complete ✓
Optimal ✓

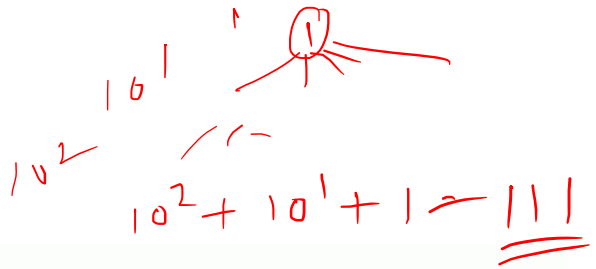
d=0 d=1 d=2

Space comp O(bd)

b=1



time comp = $(d+1)b^0 + d b^1 + (d-1) b^2 + \cdots b^d$



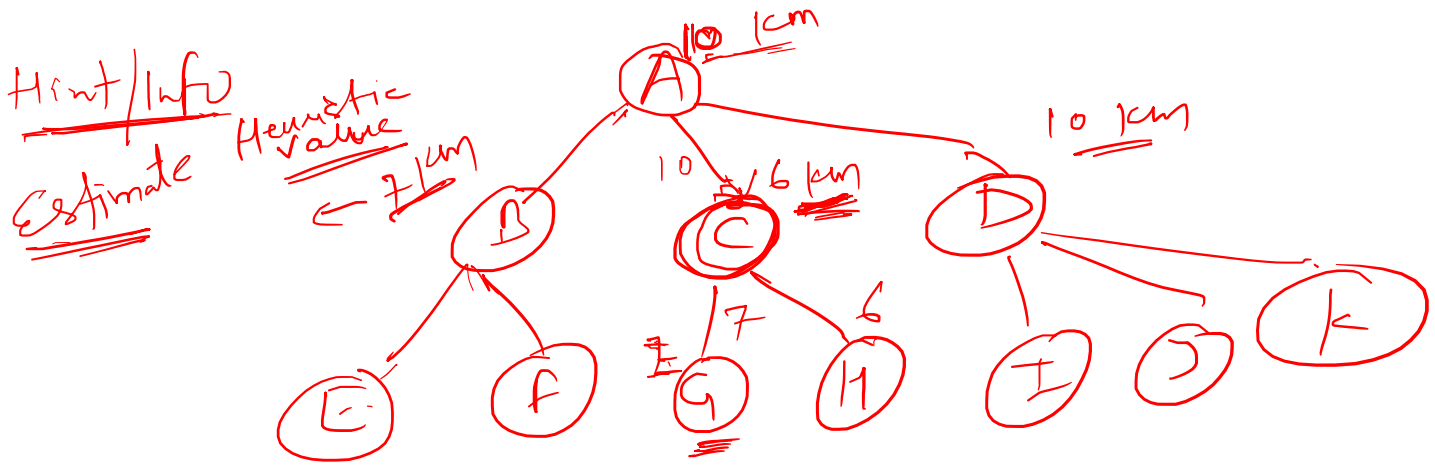
Time vs. Memory

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

Unit - II

Informed Search (Heuristic Search)



Node	h(n)
A	
B	
C	
D	
E	
F	

Evaluation function $f(n)$

Node is selected for expansion based on an eval. function that estimates cost to goal

$g(n)$ = Cost of current node from the initial node.

$h(n)$ = Heuristic fun. = estimated cost from current node to goal node

(Greedy) Best first Search

It combines the advantages of BFS & DFS

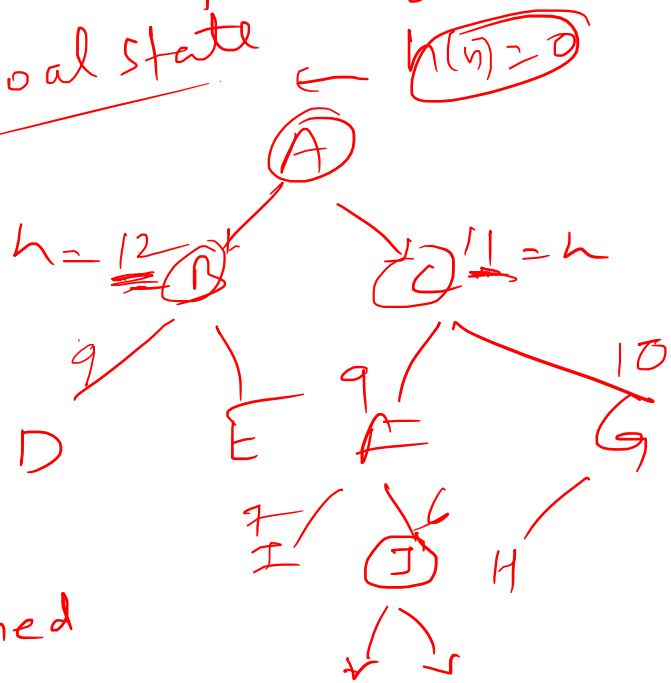
It follows single path at a time, but switches path whenever some competing path looks more promising than the current one.

$$f(n) = h(n)$$

DS Priority Queue

OPEN ⇒ Nodes that have been generated & have had heuristic function applied to them but which have not yet been examined

CLOSED ⇒ Nodes that have already been examined

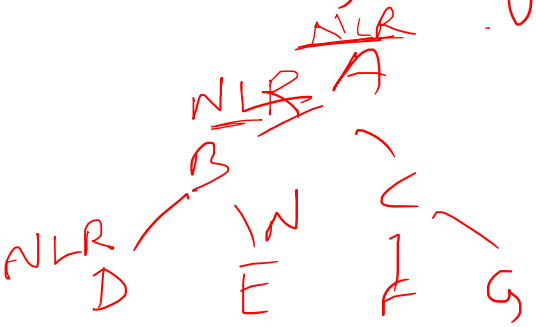


OPEN ~~A~~ ~~C~~ ~~B~~ ~~F~~ ~~G~~ ~~J~~ I G B (6 | 7 | 10 | 12)

CLOSED A C F J

DFS

Adjacency matrix



	0	1	2	3	4	5
	A	B	C	D	E	F
0	A	–	1	1	0	0	0
1	B	1	–	0	1	1	0
2	C						
3	D						
4	E						
5	F						

Adjacency list

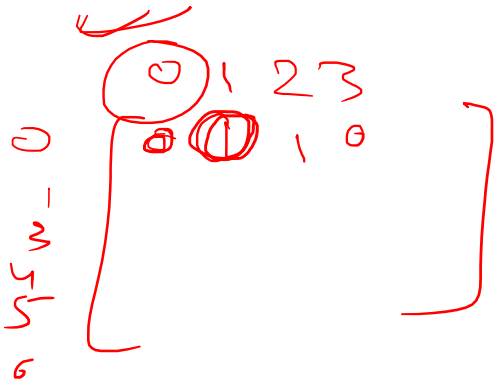
A → B → D
A → C → F

✓ Preorder traversal

① scanf (n)

for

for



② &G[i][j] for

③ visited[i] = 0

DFS(0)

DFS(i)

visited[i] = 1

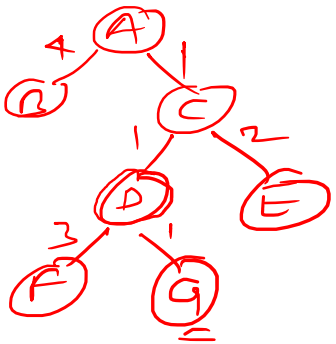
for (j)

if (!visited[j] && G[i][j] == 1)
DFS(j)

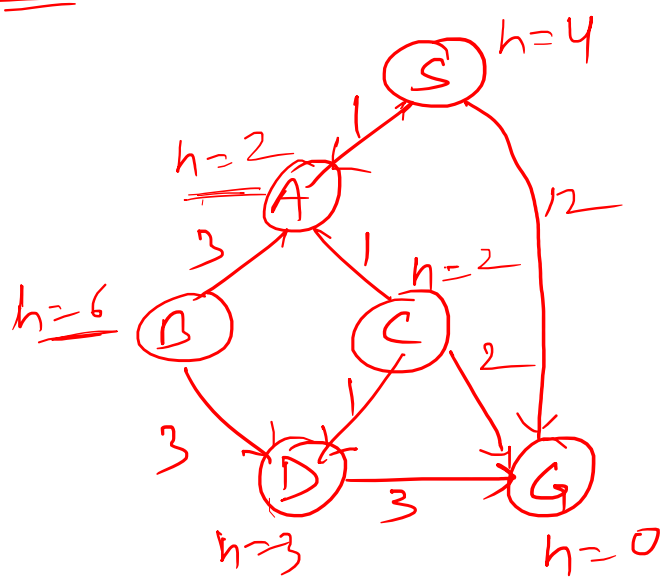
A* Algo.

consistent

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



Ex=



$S \to A \to C \to G) = 4$

A to G $g(n) = 1+1+1 = 3$

A to E $g(n) = 1+2 = 3$

A to D $g(n) = 1+1 = 2$

$h(n) = 1$

OPEN $\not{S}$ AG

CLOSE S

OPEN CBG

CLOSE SA

$S \to A$

$f(n) = 1+2 = 3$

$S \to G$

$f(G) = 12+0 = 12$

$S \to A \to B$ or $S \to A \to C$

$f(n) = 4+6 = 10$

$f(n) = 2+2 = 4$

$S \to A \to C \to D$

$f(n) = 3+3 = 6$

$S \to A \to C \to G$

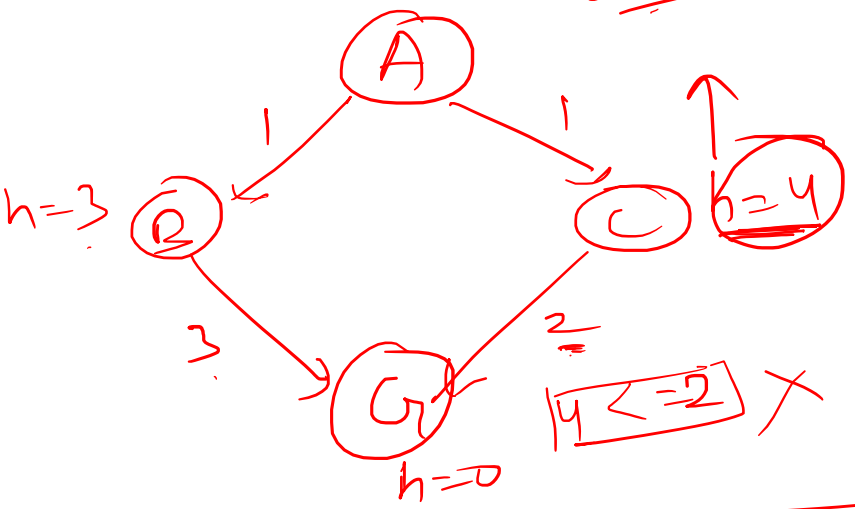
$f(n) = 4+0 = 4$

Stop.

Heuristic fun^n

Admissibility

overestimating



$A \to B$

$f(n) = 1 + 3 = 4$

$A \to C$

$f(n) = 1 + 4 = 5$

$A \to B \to G$

$f(n) = 4 + 0$

$= 4$

Stop

$4 <= 2$ ✗

$A \to C \to G$

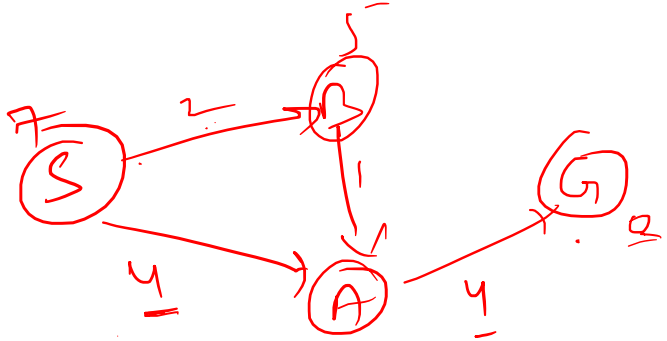
A H.F. is admissible if never overestimates the cost of reaching a goal from current node.

for every node n,

$h(n) <= h^*(n)$ → True cost/value to reach the goal node

If h(n) is Admissible, A^* is optimal (tree search)

Heuristic function should be consistent (Graph search)



$f(n) = 4+1 = 5$

$S \to A \to G$ ✓

$f(n) = 8+0 = 8$

$f(n) = 2+5 = 7$

$S \to B \to A$

exploded

✓ Admissible

✗ Consistent

h(n) is consistent if for every node n & for every successor n'

$h(n) <= c(n,n') + h(n')$

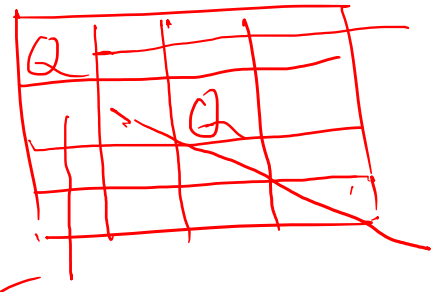
Every consistent H.f. is Admissible

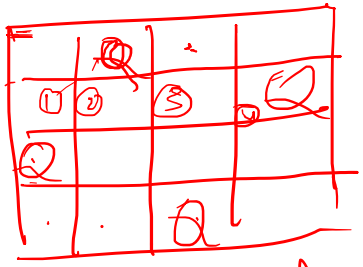
If h(n) is consistent, A* is optimal (minimz) (graph search)

---

n-Queens Problem

n=4





goal

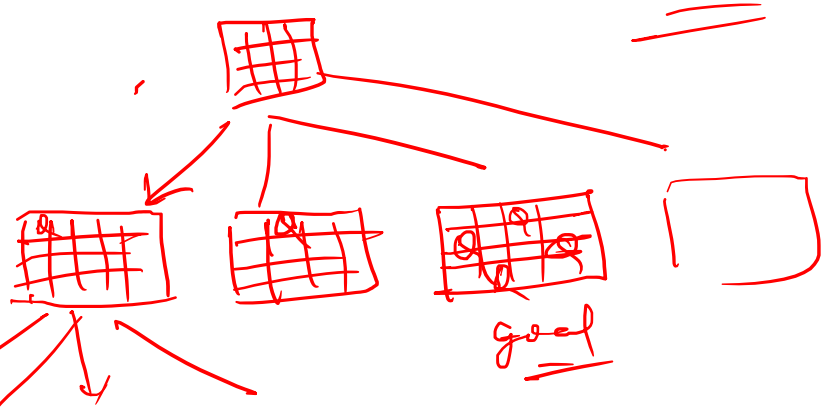
Sampling

Random Walk

Generate a state Randomly

Random walk

Randomly picking a neighboring state of the current state

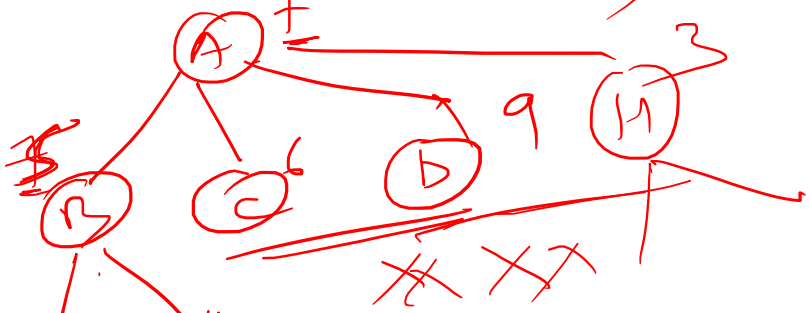


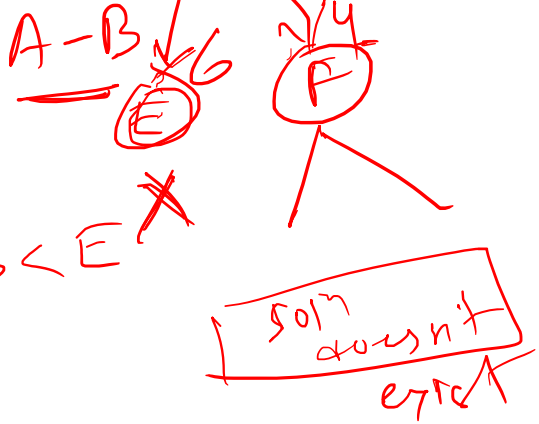
goal

① Generate & Test Algo. (h.f) X

(Random Walk)

② Hill Climbing (Greedy local search)





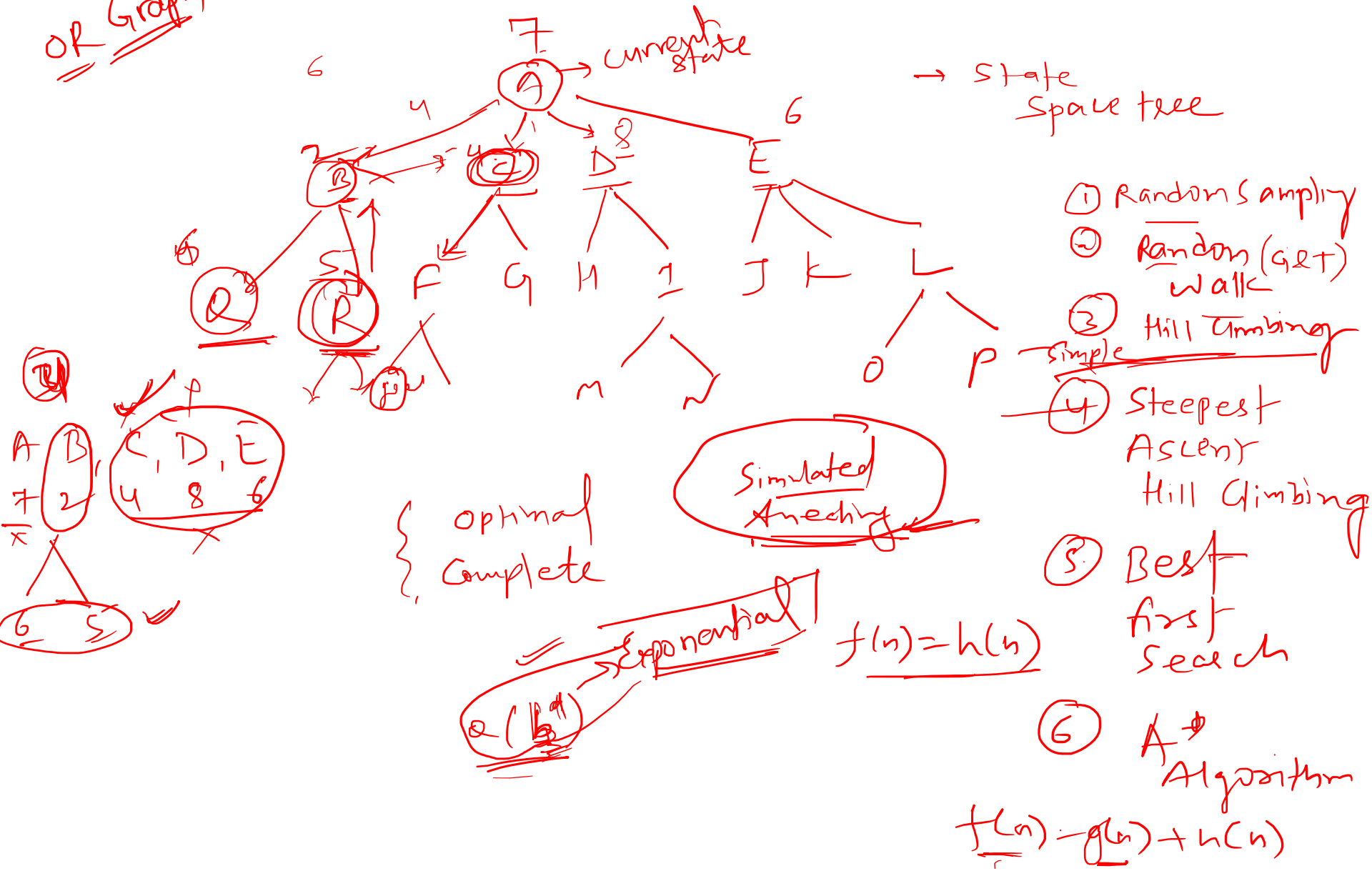
× A-C-G-J X

× A-D-H X

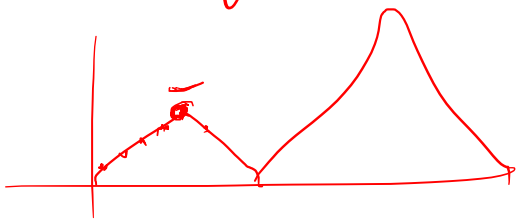
× A-C-G-I

Revision

OR Graph



Drawbacks of Hill Climbing



Local Maximum

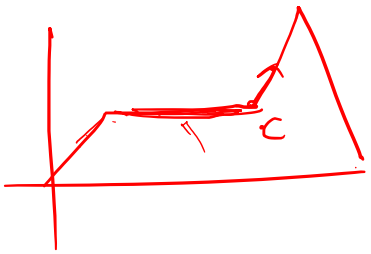
State that is better than all its neighbouring state but is not better than some other state further away

Terminate/stop

Goal state
or
No successor found of the current state

Plateau

flat area of search space in which all the neighbours state of current state contains the same value

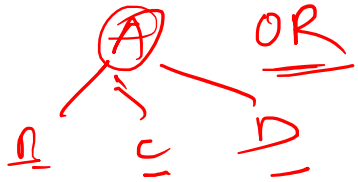


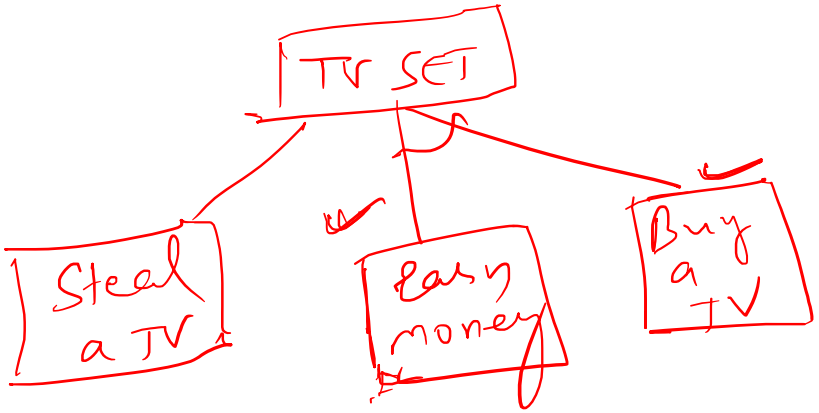
Ridges

Special form of local maxima. It has an area which is higher than surroundings but itself has a slope & cannot be reached in a single move

AO* Algo.

AND-OR GRAPH

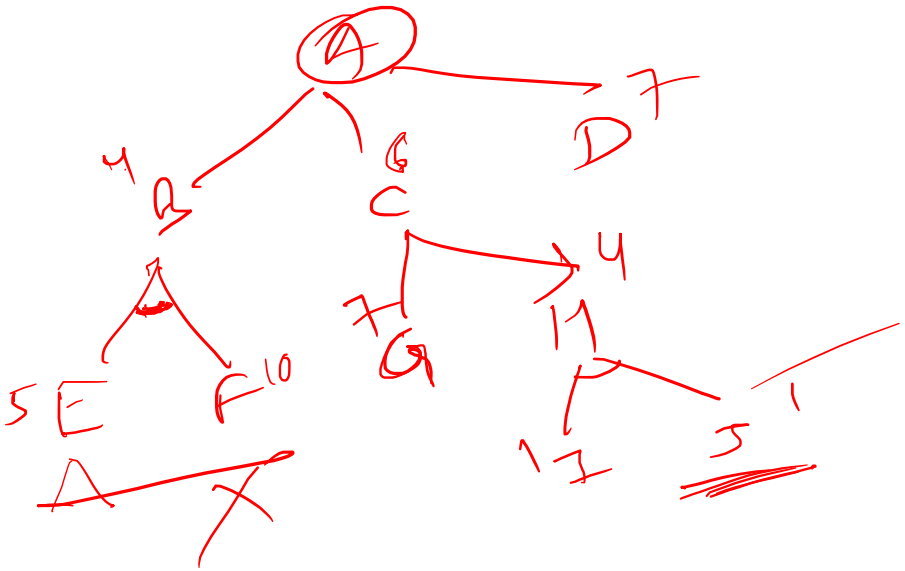




Stage 1

FUTILITY

cost < 20



Constraint Satisfaction Prob.

Cryptarithmetic Problem

A → 4

B → 6

1. Digits 0 to 9
2. variable → unique & distinct
3. Base is 10
4. Number must not begin with 0

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

M = 1

```
  9 E N D
  1 O R E
---------
1 O N E Y
```

E + 1 = N

N + R = E + 10

E + 1 + R = E + 10

R = 9 | 8

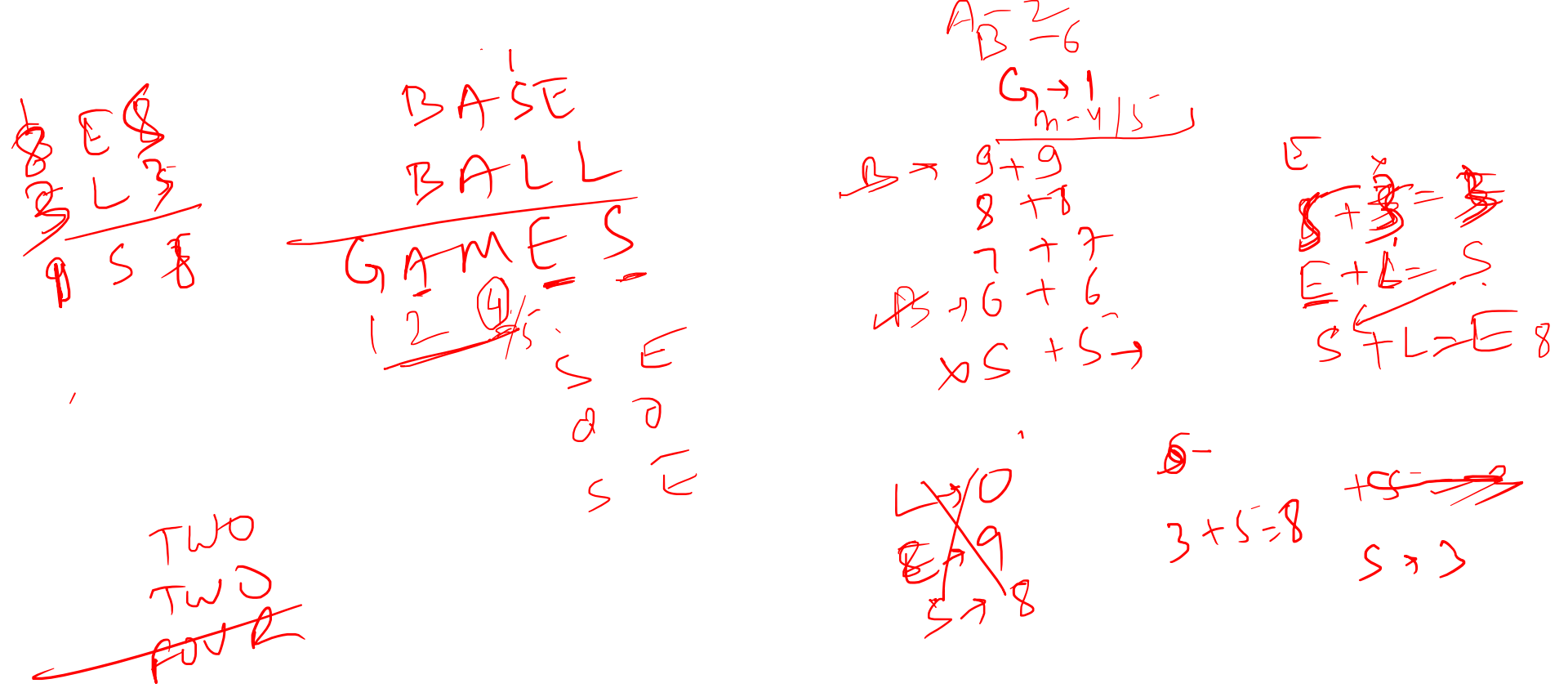
```
    7 + 4
  x 7 + 6
---------
    7 + 5
  x 6 + 5
```

× | 0 | 1 | 2 | 3

S	9
M	1
O	0
R	8
D	7
E	5
N	6

2, 3, 4, 5, 6

D + E = Y


BASE
BALL
GAMES
TWO
TWO
FOUR