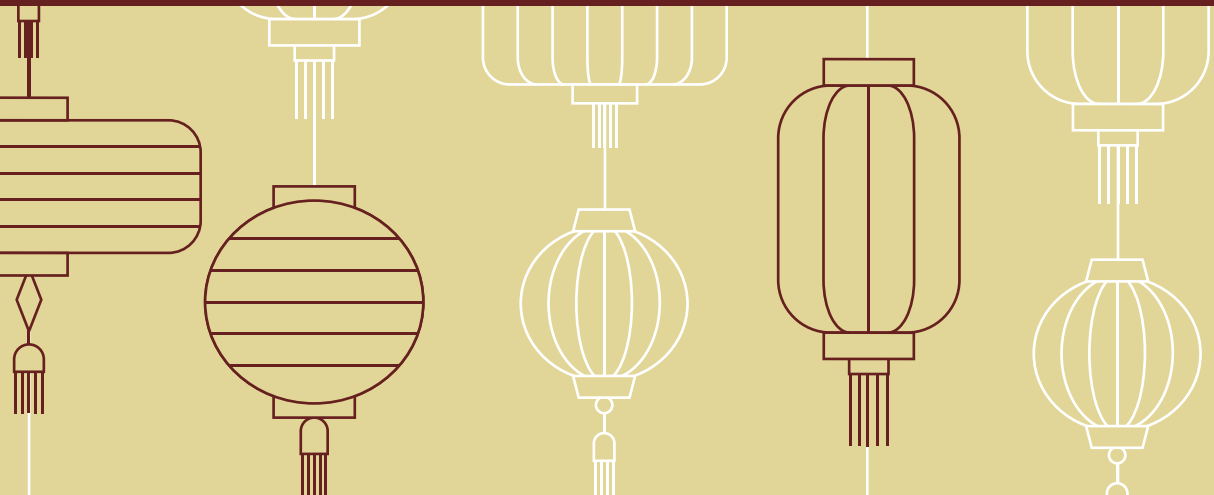
Shivambu Dev Pandey

Artificial Intelligence



Aug 4

Aug 5

Agent

Human
Robot
Device

$f : P^* \rightarrow A$

P^* → percept history

A → Action

f → Agent function

Intelligent Agent

Percept sequence

Performance Measure : Time, Power, Amount of Dust
Environment : Tiles
Actuators : Left, Right, Suck, Noop
Sensors : camera, lidar

Taxi Driver

P	Time, speed, Safety, Comfort
E	Road, Pedestrians, Vehicles, Customers
A	Forward, Backward, Left, Right, Stop
S	Camera, Lidar, IR, GPS, speedometer Sonar, odometer, accelerometer engine sensors, keyboard

Robot

P	Speed, Accuracy
E	Real World Scenarios
A	Display
S	Camera, LiDAR

	Object Rec Robot	Taxi Driver	Vacuum cleaner
Observability	✓	X	✓
Static / Dynamic	D	D	S
# of agents	1	1	1
Discrete / Contin	D	C	D
Episodic / Seq.	E	S	E
Deterministic / ND	D	D	D

Episodic / sequential

Past will affect future

	Chess	Solitaire	Mine Sweeper
Obs.	Yes		
Sta	Dynamic		
#	Multi agent		
Dis/cont	D		
Episodic/Seq.	Seq.		
Det / ND	D		

	Solitaire	Mine Sweeper
P		
E		
A		
S		

Aug 8

Fully observable / Partially

Single agent / multiagent - partially cooperative
partially competitive

Episodic / Sequential

Discrete / Continuous -

Deterministic / ND

Internet Shopping Agent

P · Price, offers, delivery time, quality, effectiveness / time tokens

E · Internet / websites, shippers / vendors.

A · display product, link

S · Webcrawlers (text, images, videos)

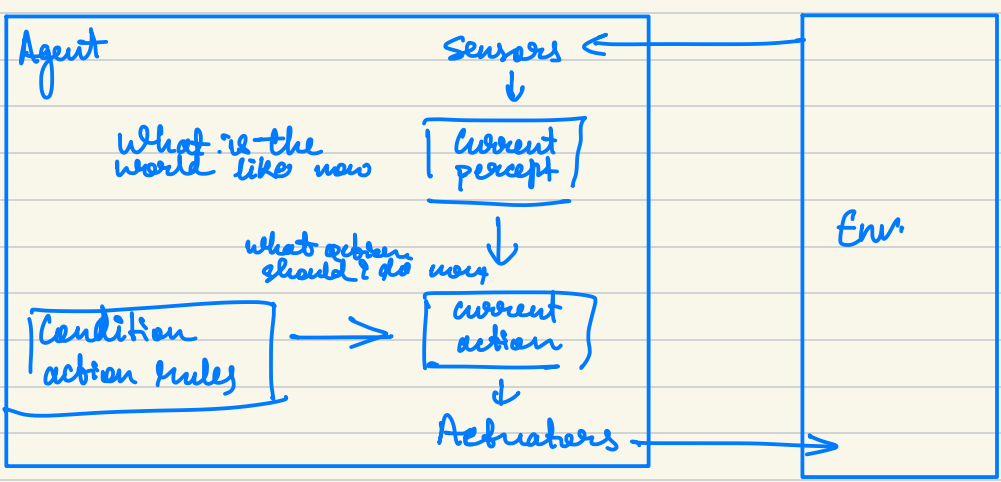
→ single
→ dynamic
→ continuous
→ seq.
→ non-deter

Types of Agents

sensors
actuators
memory
thinking capacity

- Simple Reflex Agent

- only current percept
- ignores percept history



if – then mechanism

- Model-based Reflex Agents
  - efficient in partially observable env
  - has memory
  - maintains an internal state

State
How the world evolve
Implication of actions

Sensors
what is the world like now

- Goal Based Agents
  - we need to keep track of goal to ensure effectiveness
- Utility Based Agents
  - has a utility fn to measure its preferences

Simple Reflex : Vacuum Cleaner
Model Based : Taxi Driving
Goal Based : Nav, puzzle solver, sudoku
Utility Based : chess, scheduler, game playing

End of Module 1

Problem Solving and Search

→ Problem Solving Agents

Solution (Goal state)

Initial state ⟶ sequence of action ⟶ Goal state

searching

Problem Solving has 4 phase

- Goal formulation
- Problem form
- Search
- Execution

Romania Problem

→ set of possible states
→ set of initial states
→ one or more goal states
→ actions available
→ transition model
→ action cost function ACTION-COST(s, a, s')

source action destin

Path : Sequence of actions.

Solution : path that leads to goal state from initial state

Optimal : solution with lowest cost.

State space can be represented as graph

- Model - an abstract mathem. desc.
- abstraction.

Problem types

- Deterministic, fully observable — single state
- Non-observable — conformant
- ND / partially obs. — contingency, policy, interleave
- Unknown state spa — exploration problem (online

Single - State problem formulation

problem is defined by 4 items

initial state

successor fn

goal test

path cost (sum of all cost)

e.g $c(x, a, y)$

x - source

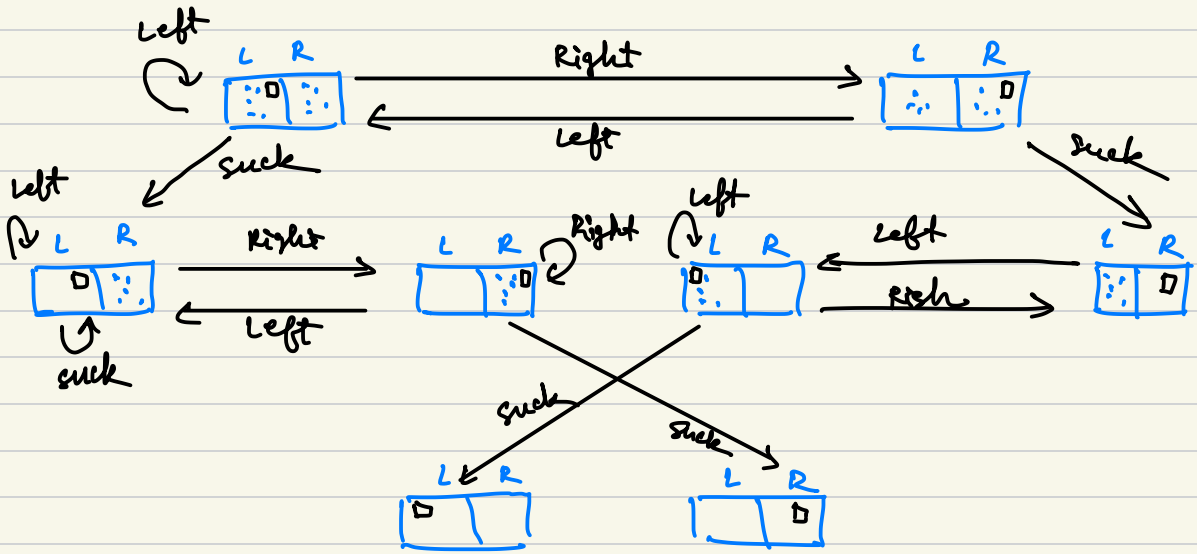
a - action

y - dest.

Selecting a state space

- Abstraction

Vacuum World



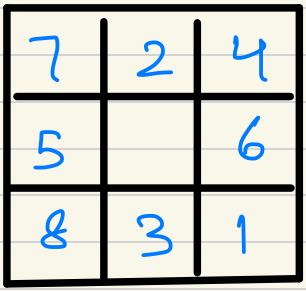
state : integer dirt & robot location

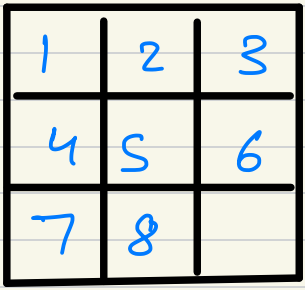
actions – Left, Right, Suck, NoOp

goal – no dirt

path cost – 1 per action (0 for no op)

The 8 puzzle





states : integer

action : L, R, U, D

cost : 1 per move

goal test: goal state

search algo
search tree
node
edge

expand
ACTIONS, RESULT
frontier

interior / exterior

fringe (unvisited nodes)

Search strategies

- completeness (if sol exist, found/not)
- time complexity (time to expand the nodes)
- space complexity (mem to expand the nodes)
- optimality (if optimal sol exist found/not)

Time & Space

b = maximum branching factor
d = depth of the least cost soln
m = max depth of state space

Uninformed Search Strategies

Informed Search Strategies

USS —
- BFS
- UCS — Uniform cost
- DFS
- DLS — Depth Limited
- IDS — Iterative Deepening

only uses info in the problem statement.

BFS

optimal if path cost = 1

else non optimal

time $O(b^{d+1})$

span $O(b^{d+1})$

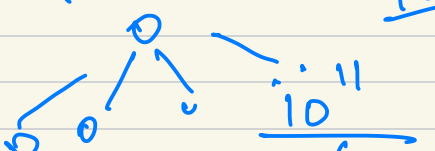
completeness Yes (if b is finite)

BfS: consider a search tree with $b = 10$ & $d = 10$

processing speed of a computer is 1 million nodes/sec, each node requires 1KB of memory

Compute total time & space required

for



10^2

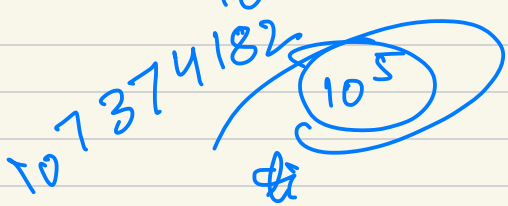
10^{11}

10^6

1,000,000

$\frac{10^{11}}{2^{10}}$

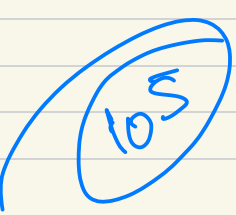
10^3



9765625 0

10

10^1 1



Aug 12

Uniform Cost Search

Expand least cost unexpanded node

fringe = queue ordered by path cost, lowest first

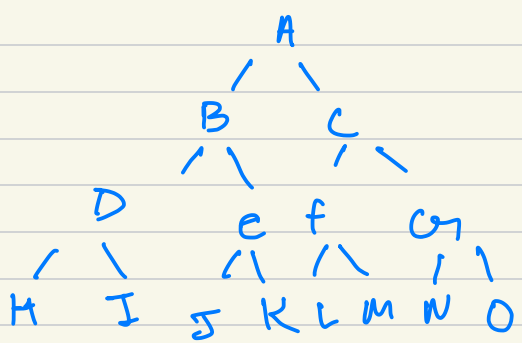
bfs with priority queue
(if all wt = 1 = bfs)

Complete: Yes if stepcost $\geq \varepsilon$

Time: $O(b^{\lceil C^*/\varepsilon \rceil})$

DFS

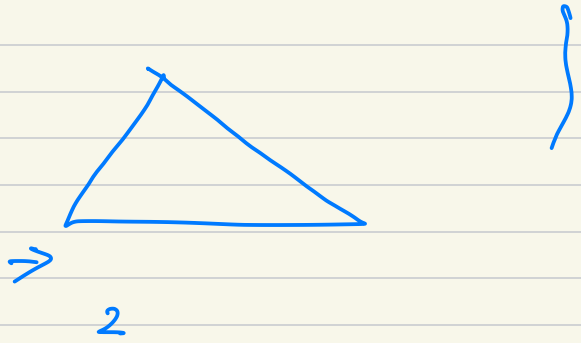
expands deepest node
LIFO / stack



b — branching factor
d — depth
m — max depth

time: $O(b^m)$
space: $O(b \cdot m)$
Complete: if finite search tree
Optimal: not optimal

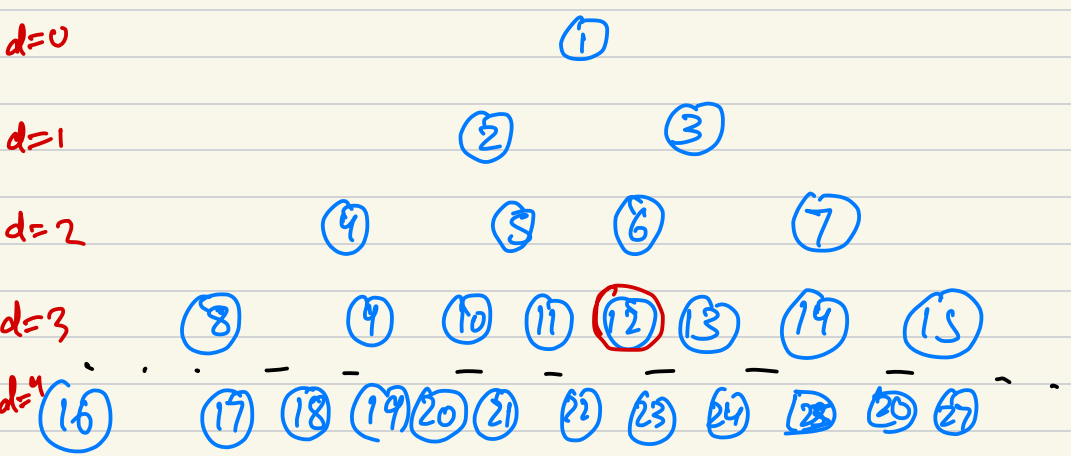
	BFS	DFS
C	3	9
L	12	11



2

Depth Limited Search

Dfs with depth limit l



Dest. is 12
limit = 2

T $O(b^l)$
S $O(bl)$

dfs = dls + (l = ∞)

Iterative Deepening Search

for $l=0$ to ∞

recursively execute DLS

Goal 23

$l = 0$

$l = 1$

$l = 2$

$l = 3$

$l = 4$

T: $O(b^d)$

S: $O(b \cdot d)$

Complete · Yes

Optim - Yes if cost = 1 for all edges

BFS, DFS, DLS, IDS } exhaustive search till end of search tree / solution

Aug 18

Informed Search

Uninformed

exhaustive

only uses info in the problem statement

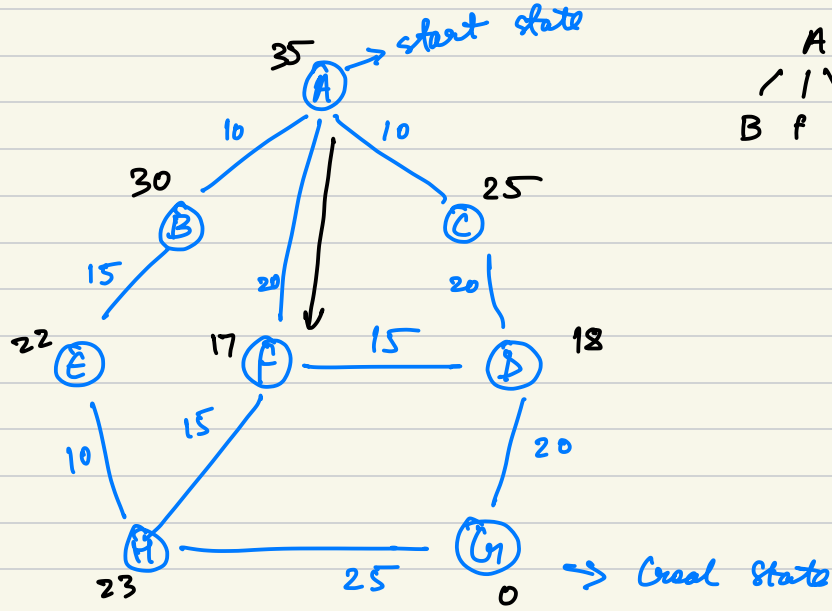
blind

unreliable / intractable when problem is complex

Informed (Heuristic Search)

- heuristic fn ($h(n)$)
- uses knowledge beyond the problem def
- evaluation f^n to help decide which nodes to explore first
- HSA explores nodes in ascending order of eval f^n
- Greedy Best first search (GBFS)
- A^* search

Greedy BFS



Explore the least $h(n)$ node first

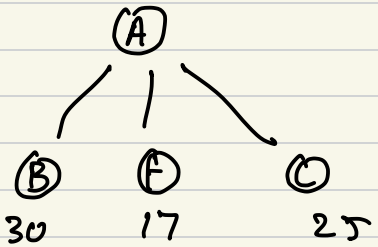
$G(n)$ ⇒ original cost

$h(n)$ ⇒ straight line distance to Goal Node G

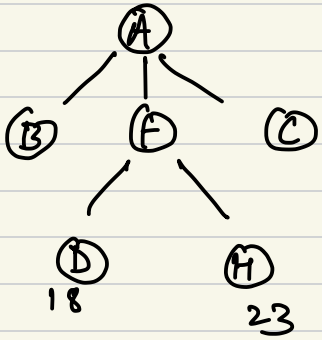
h_{SLD} ⇒

A	35
B	30
C	25
D	18
E	22
F	17
H	23
G	0

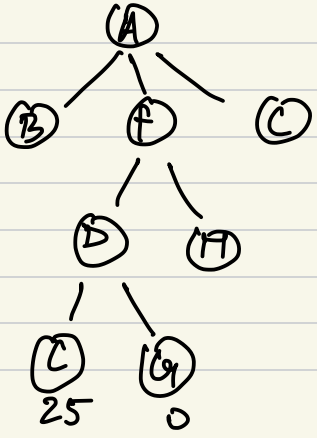
Step 1:



Step 2 Explore F



Step 3 Explore D



Step 4 Explore G_1

Goal node $\Rightarrow G_1$

$A \rightarrow F \rightarrow D \rightarrow G_2$

$G_1(n) = 55$ $(20 + 15 + 20)$

T: $O(b^m)$ S: $O(b^m)$

selection of heuristic fn matters

Complete: Not complete, might stuck in loops maintain visited list

Time: $O(b^m)$

Space: $O(b^m)$

Optimal: No

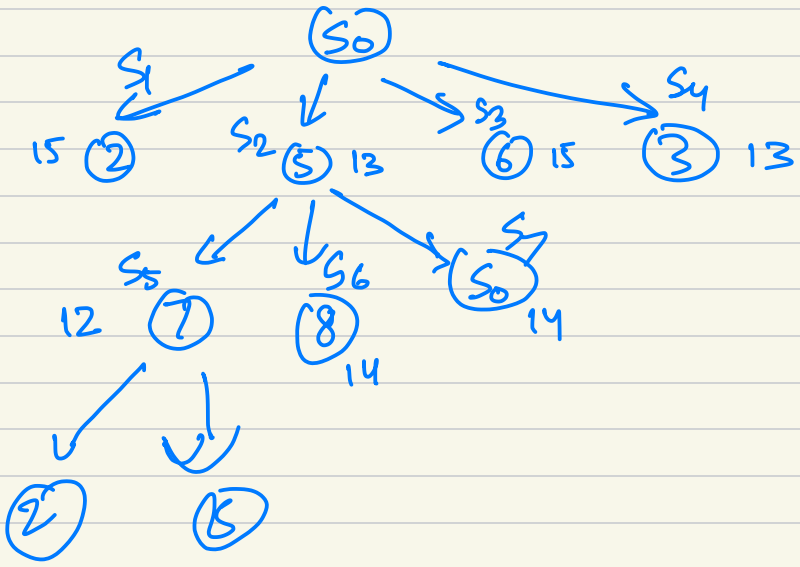
$h(n) \leq g(n)$ Not always valid.

7	2	4
5		6
8	3	1

1	2	3
4	5	6
7	8	

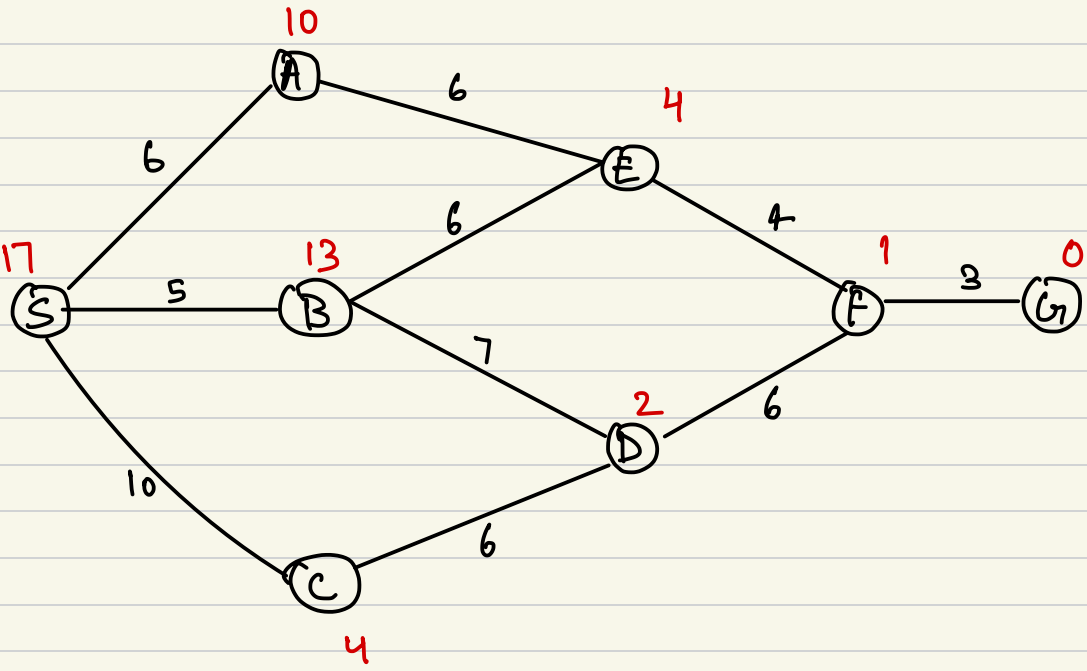
$h(n)$

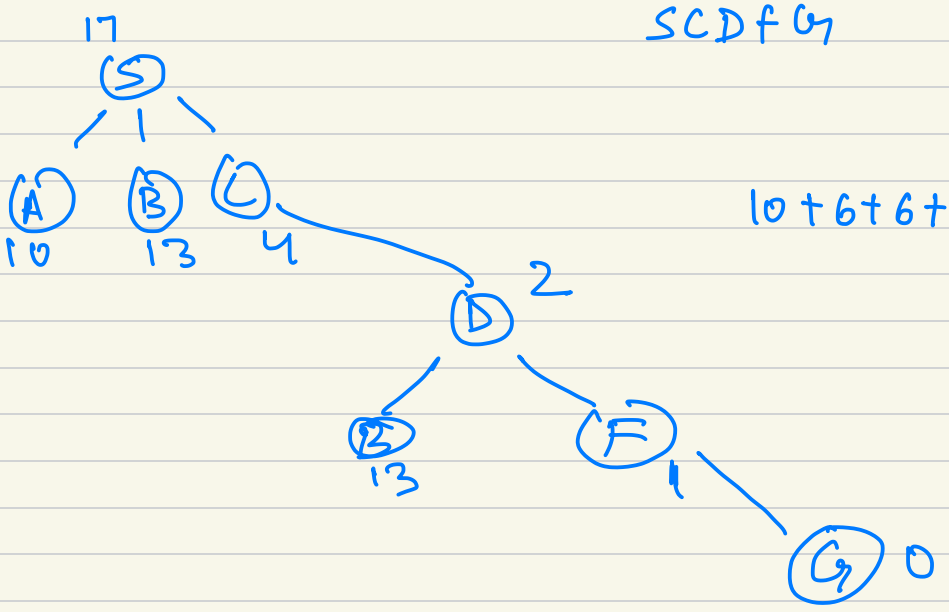
1	4
2	0
3	3
4	3
5	1
6	0
7	2
8	1
	14



Implement solution in python.

Aug 19





SCDfG

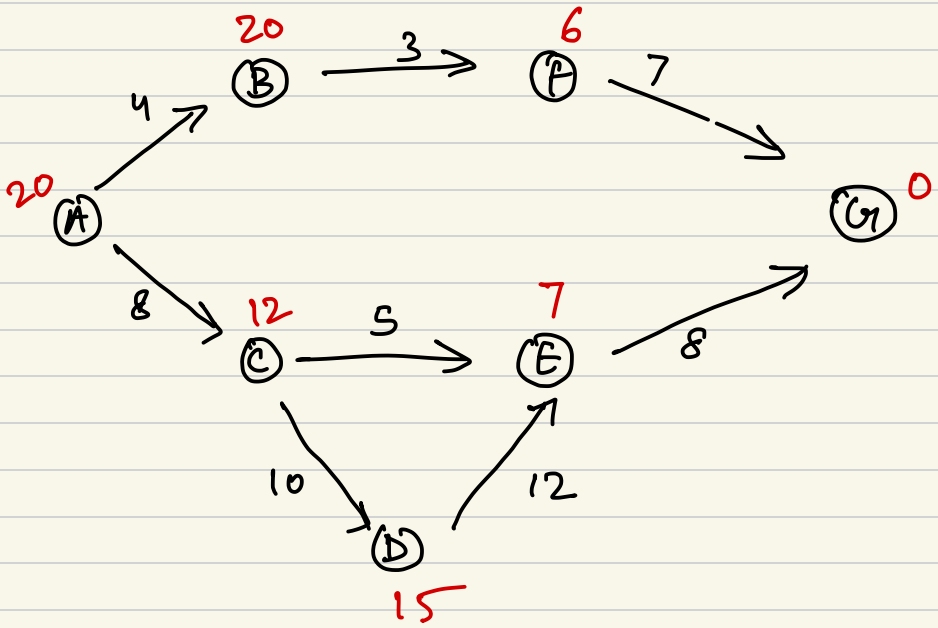
10+6+6+3=25

A* Search

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

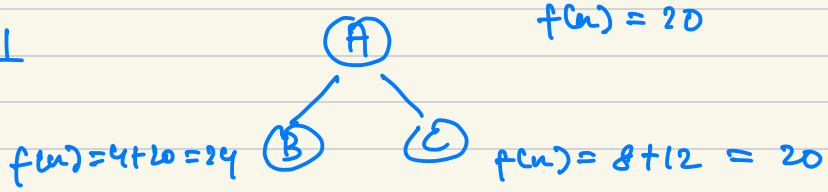
$g(n)$: actual path cost till node 'n' from source
$h(n)$: heuristic cost from node 'n' to goal node
$f(n)$: estimated cost of best path

- best first search

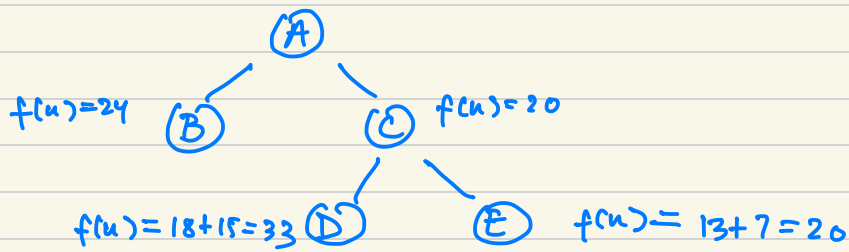


Step 0 (A) $f(n) = 0 + 20 = 20$

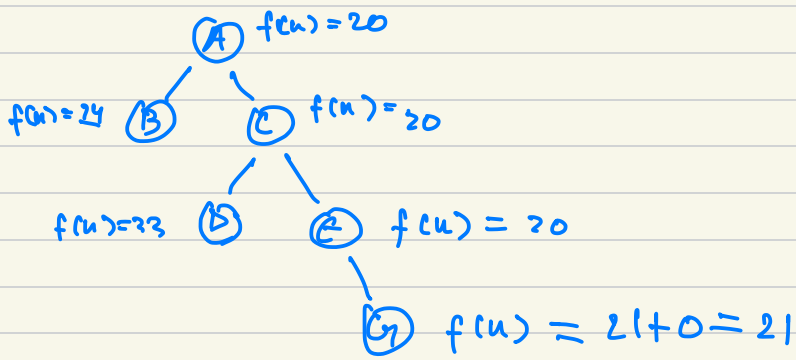
Step 1



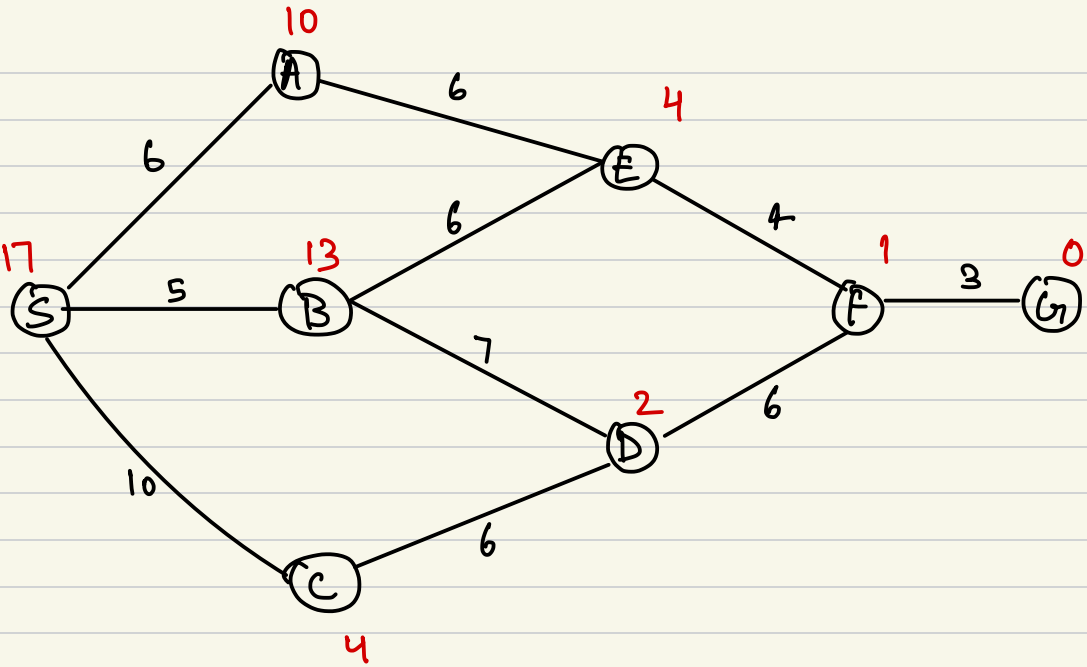
Step 2



Step 3



A → C → E → G = 21



S f(n) = 0+17 = 17

f(n)=16 A B C f(n)= 14

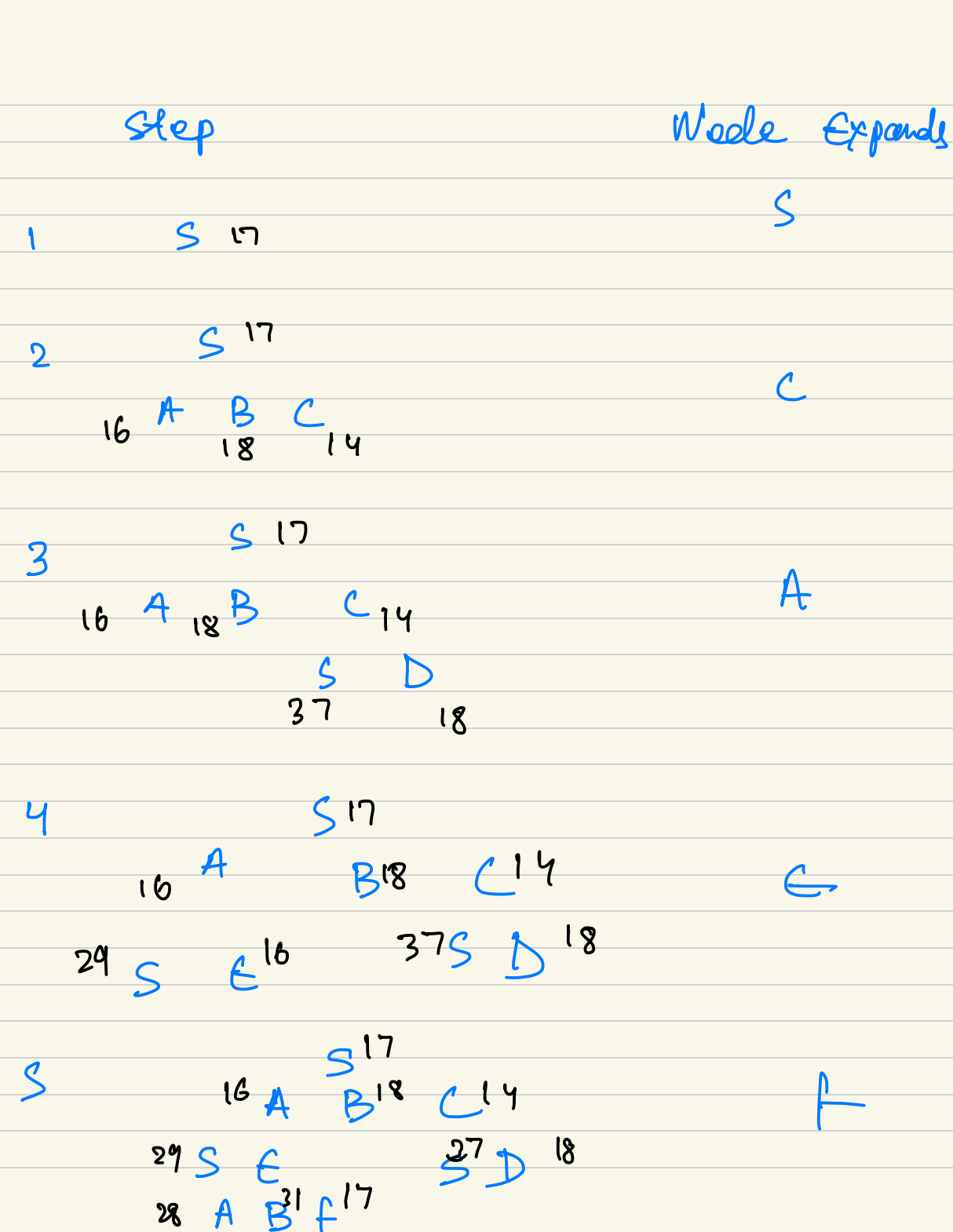
f(n)=18

f(n)=16 E D f(n) = 18

B f

f(n) 31 f(n)=17

G f(n) = 19


Step
Node Expands
1
S 17
S
2
S 17
16 A B C
18 14
C
3
S 17
16 A 18 B C 14
S D
37 18
A
4
S 17
16 A B 18 C 14
29 S E 16
37 S D 18
C
5
S 17
16 A B 18 C 14
29 S E
27 S D 18
28 A B 31 F 17
F

6

S

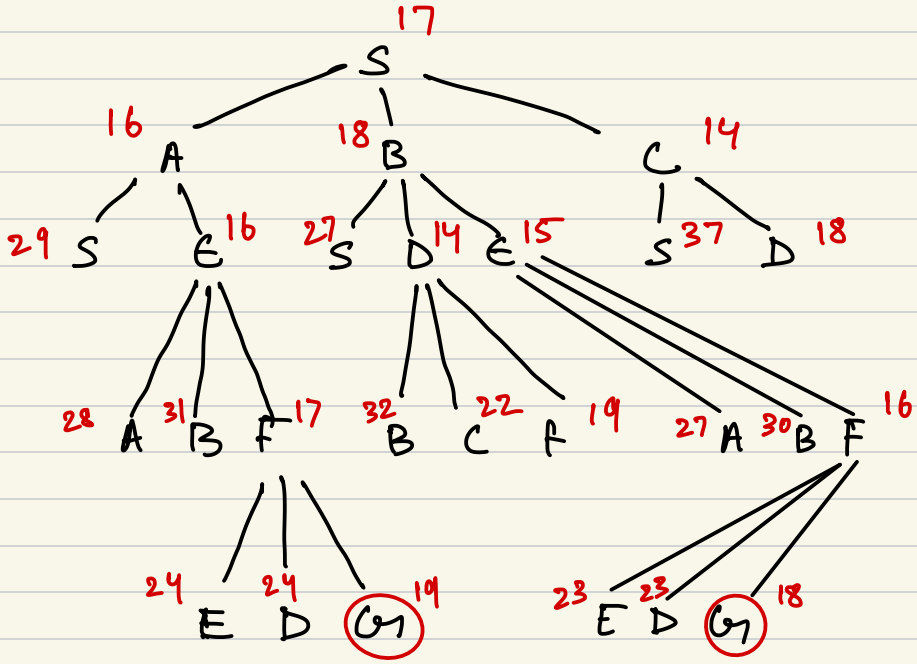
A B 19 C

B

S E S D 18

29 A 31 B F 17

24 E 20 D C 19



Solution: $S \rightarrow B \rightarrow E \rightarrow f \rightarrow G_1$

Cost = 18

Aug 25

A* Search

– A* Admissibility

- $h(n)$ → admissible → optimal solution

↦ admissible heuristic never overestimates the cost to reach the goal.

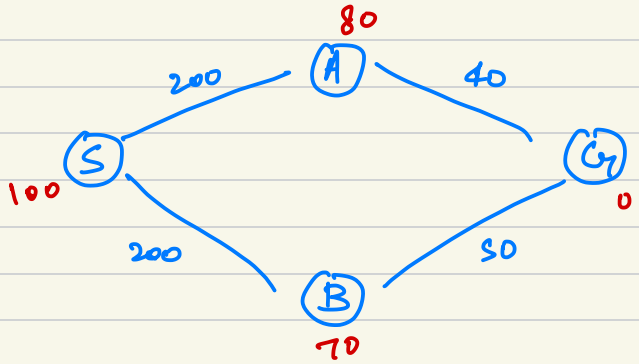
Ex. SLD

Actual cost from 'n' to goal — $h^*(n)$

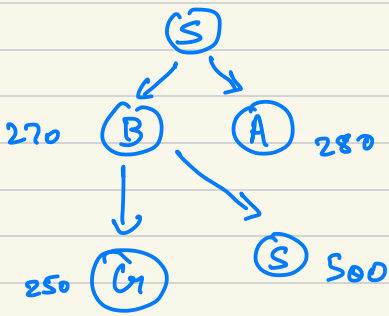
Heuristic cost from 'n' to goal — $h(n)$

$h(n) \leq h^*(n)$ Underestimate

$h(n) \geq h^*(n)$ Overestimation



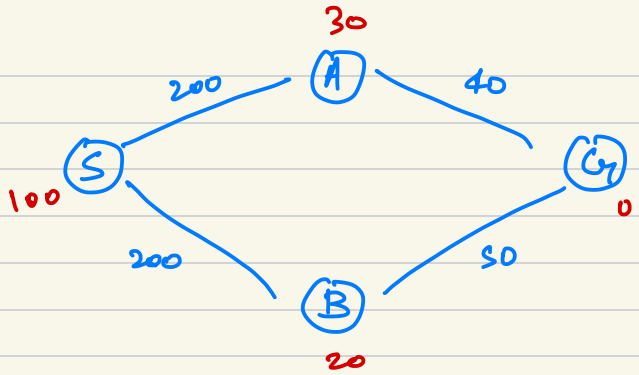
$h(S) = 100$
$h(A) = 80$
$h(B) = 70$



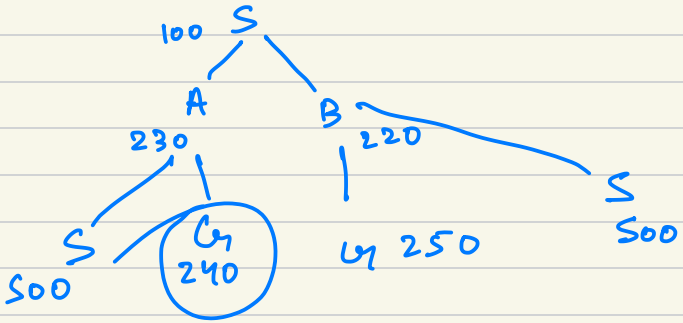
$f(n) = g(n) + h(n)$ $S \rightarrow B \rightarrow Gr$: 250 $h(n)$

$f^*(n) = g^*(n) + h^*(n)$ $S \rightarrow A \rightarrow Gr$: 240 $h^*(n)$

$h(n) > h^*(n)$



$h(S) = 100$
$h(A) = 30$
$h(B) = 20$
$h(G_n) = 0$



$S \to A \to G_n$: 240 $f(n) = g(n) + h(n)$

$S \to A \to G_n$: 240 $f^*(n) = g^*(n) + h^*(n)$

- A^* is complete optimal when $h(n) \leq h^*(n)$

1) $f(n) > C^*$ (node 'n' on optimal path is not expanded)

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

2) If node 'n' on optimal path is expanded

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

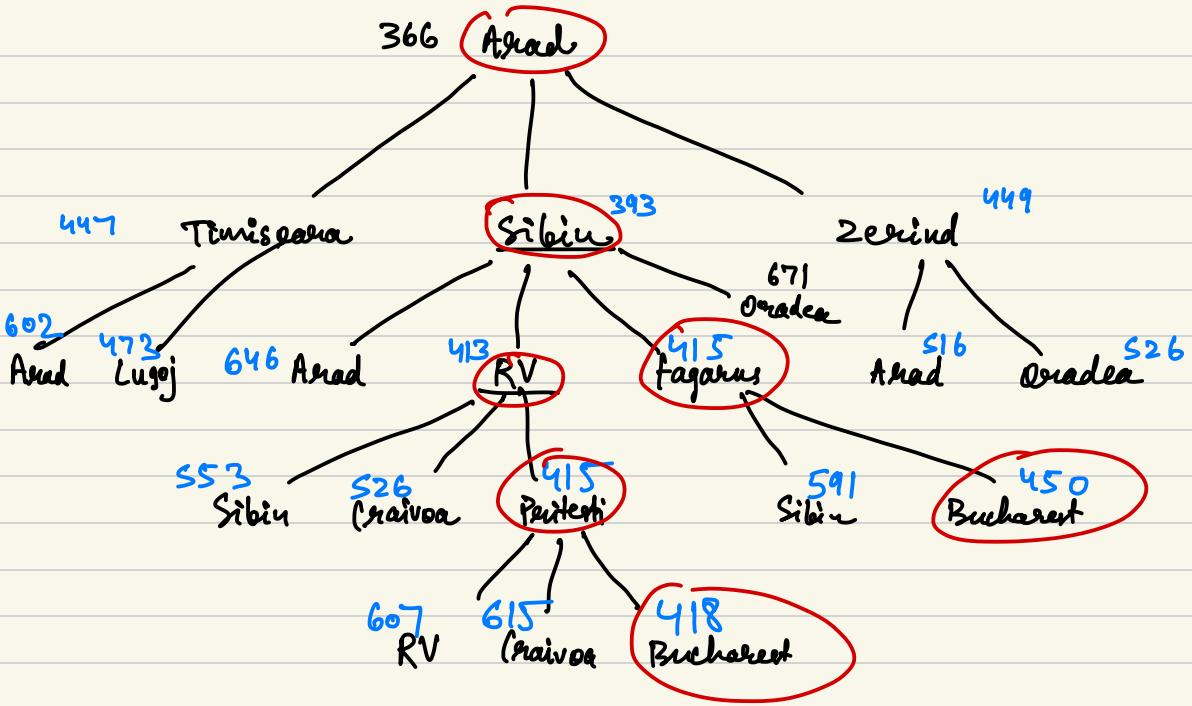
3) If $h(n) \leq h^*(n)$ as per admissible property

$$f(n) \leq g^*(n) + h^*(n)$$

$$g(n) + h(n) \leq g^*(n) + h^*(n)$$

4) $f(n) \leq C^*$

By 1 & 4, proof of contradiction suboptimal path 1 is incorrect and A^* always gives optimal solution for admissible heuristic.



Arad → Sibiu → RV → Pitesti → Bucharest

$f(n) = 418$

Search contours

$f_i < f_{i+1}$

Optimality of A^*

A^* { all, some, no } $f_n < C^*$, $f_n = C^*$, $f_n > C^*$

Aug 29

Local Search Algorithms

- start to neighbouring, do not keep track of path and visited state.

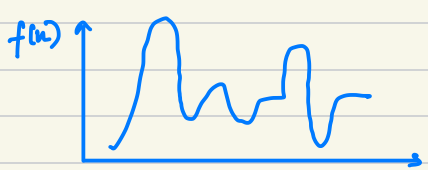
Objective fn: Best value for fn from neighbour state

Hill climbing search

- Global maximum
- Global minimum
- Local maximum
- Local minimum
- ridge / plateau

→ Explore neighbour node which is having greater value of obj fun compared to current node.

→ terminates when it reaches a peak



→ might get stuck at local maxima
→ ridge / plateau

Hill Descent Algorithm

→ Explore neighbour node which is having lowest value of obj fun compared to current node.

global maximum — hill climbing

global minimum — gradient descent

Hill descent : Minimizing h

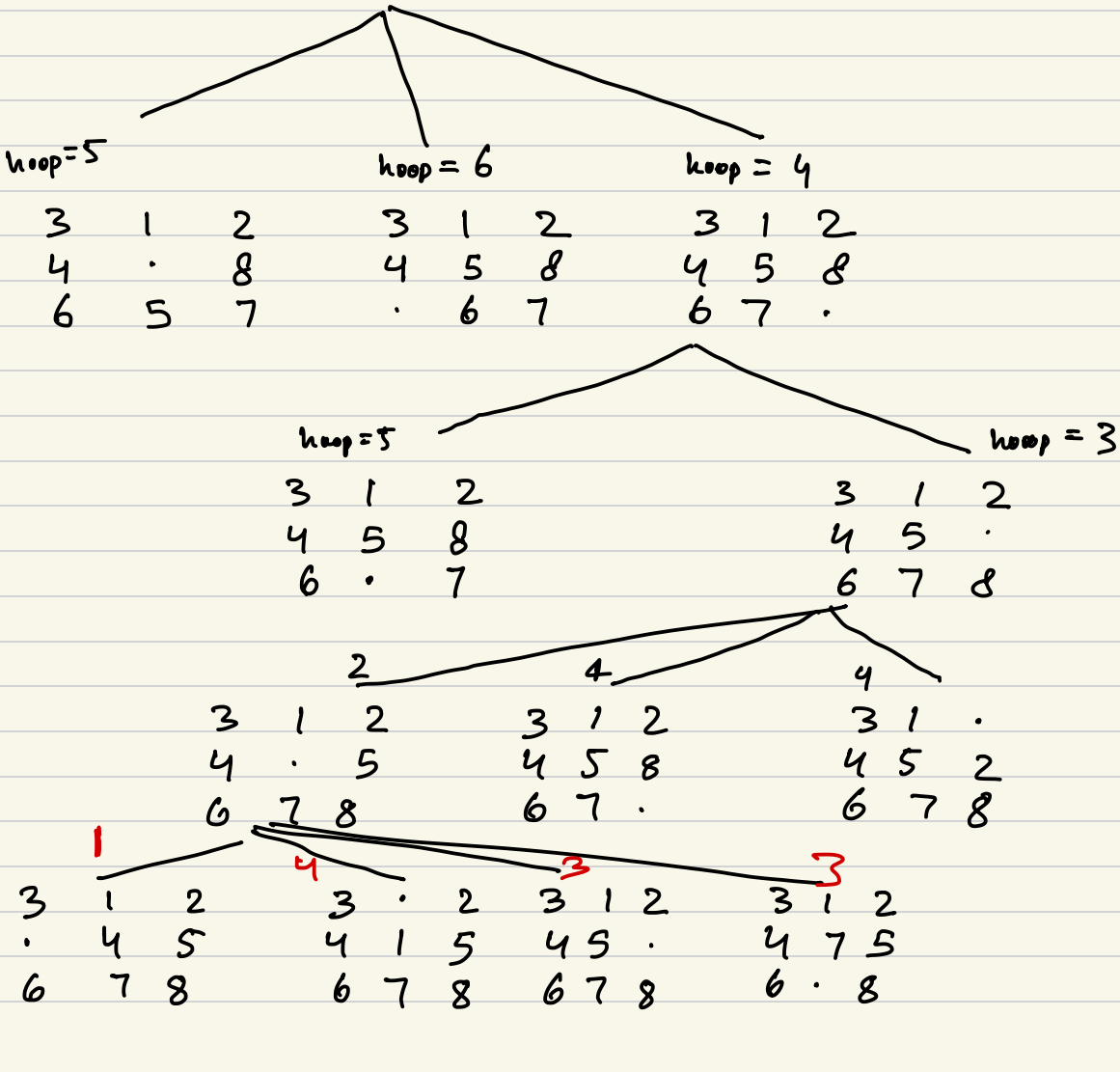
start

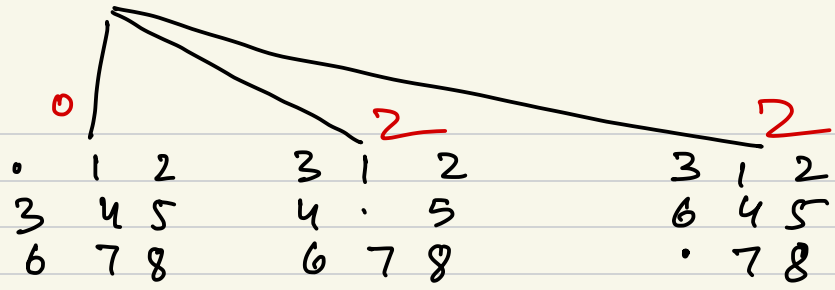
3	1	2
4	5	8
6	·	7

h = 5

goal

·	1	2
3	4	5
6	7	8





N- Queens Problem : minimize conflicts (h.f)

	A	B	C	D
1	4	2	4	3
2	4	Q_2	3	Q_3
3	Q_1	3	Q_4	4
4	3	4	2	4

h=5

Q_1 – Q_2 2

Q_2 – Q_3

$Q_1 - Q_2 - Q_3 - Q_4$ 3

$Q_2 \to B_1$ $\quad$ $Q_4 \to C_4$

3	Q_2	3	1
3	5	2	Q_3
Q_1	3	Q_4	2
1	4	0	2

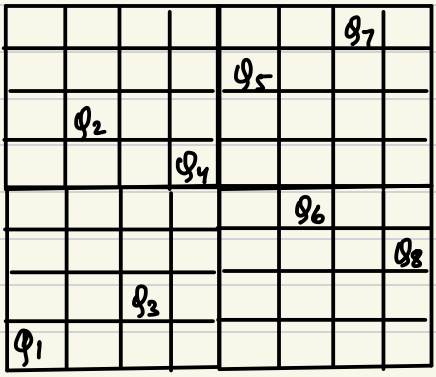
h=2

$Q_4 \to C_4$

	Q_2		
			Q_3
Q_1			
		Q_4	

Types of Hill climbing

- Stochastic
- First choice
- Random - restart



Sept 1

- Stochastic

Initial

- FC
- RR

- Simulated Annealing

(move it out of local min.)

- shake hard initially (high temp) (more bad moves)
- gradually reduce the intensity (lower the temp)

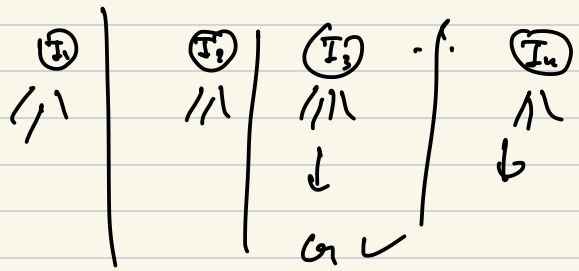
- Allow bad moves dependent temp.

Schedule (t)

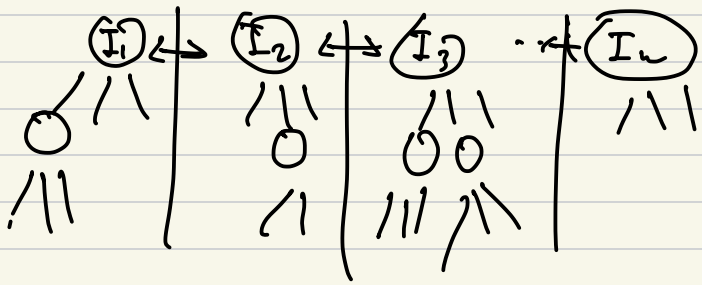
ΔE

Local Beam Search / vs. k random restart

KRR executed in isolation



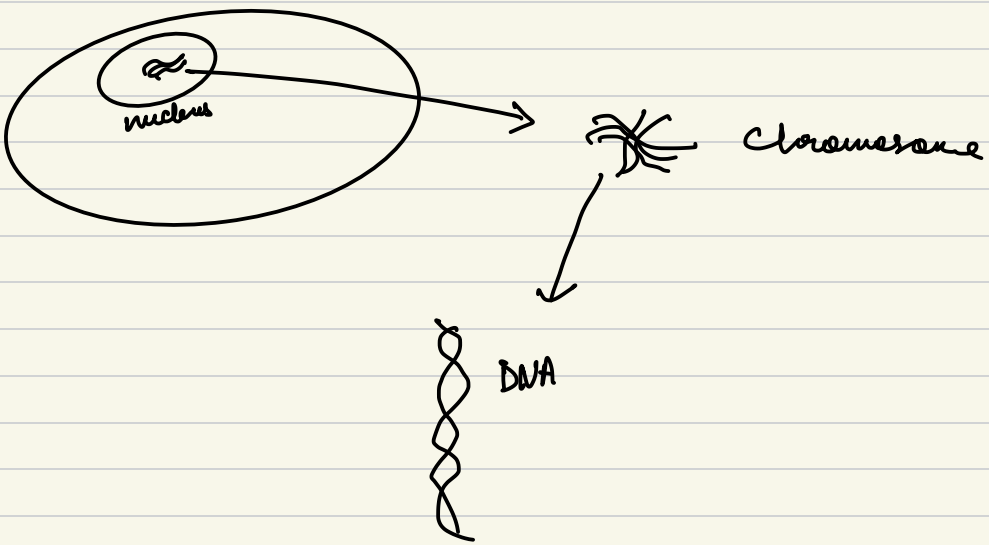
LBS communicates together



Like Survival of the fittest

Evolutionary Algorithm

- → Genetic Algorithm ✓
- → Ant Colony Optimization
- → Partial Swarm Optimisat -
- → Blind Driven Optimization



mRNA → □□□□□□

Fitness Selection Pair Genes-Over Mutation

Richard Frankel Stanford

Amas

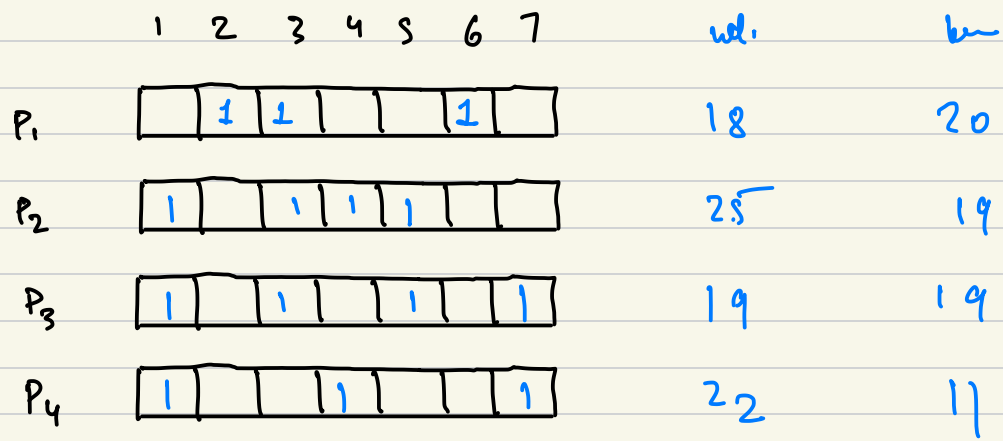
Alternating string

Succession function
fitness function
Solution Test

New Population generation

Item :	1	2	3	4	5	6	7
Benefit:	5	8	3	2	7	9	4
Weight:	7	8	4	10	4	6	4

max wt. = 22



	wt	Ben
011 | 0010	18	20
101 | 0101	19	19
100 | 1001	21	11

Crossover one pt, 3rd bit after MSB

P1 & P2 : C1 101 0010

C2 011 0101

P_1 & P_4 C_3 011 1001
C_4 100 0010

P_2 & P_4 C_5 100 0101
C_6 101 1001

Mutation : flip bit (LSB) in C_2, C_4, C_6
flip 3rd bit from MSB in C_1, C_3, C_5

		wt	Bcn
C_1'	0100101	16	19
C_2'	1010011	21	21
C_3'	0101001	22	14
C_4'	1000011	17	18
C_5'	1010101	19	19
C_6'	1011000	21	10

Adversarial Search

Sept 2

AI
Stuart Russel
UC Berkley

Standard

det., obs., 2p, turn taking, zero-sum

Initial

Two player zero sum games

max - p1
min - p2

- Always max makes the first move
- end of the game utility value is assigned for each player

So
To-Move (s)
Actions (s)
Result (s, a)
Is-Terminal (s)
Utility (s, p)

Game Tree

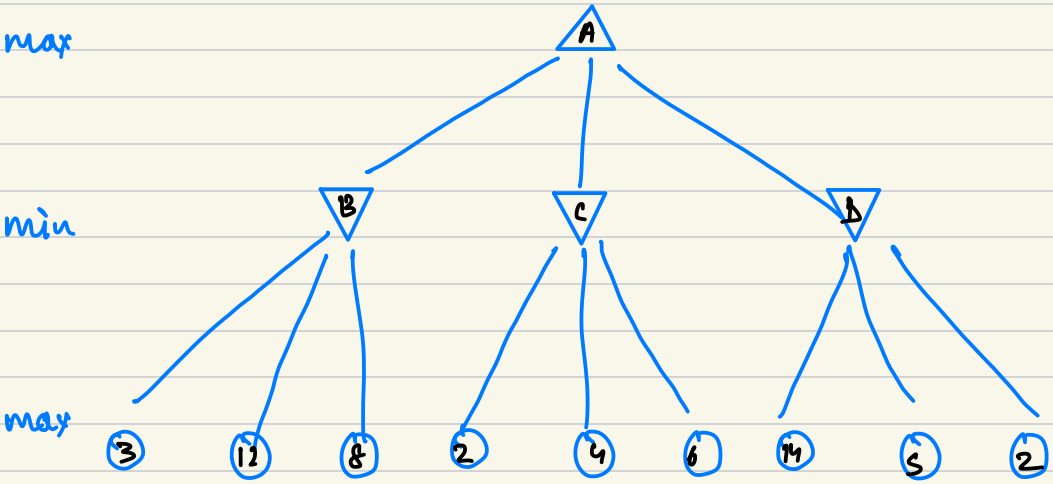
Tic Tac Toe

Max (x)

+1 win 0 draw -1 loss

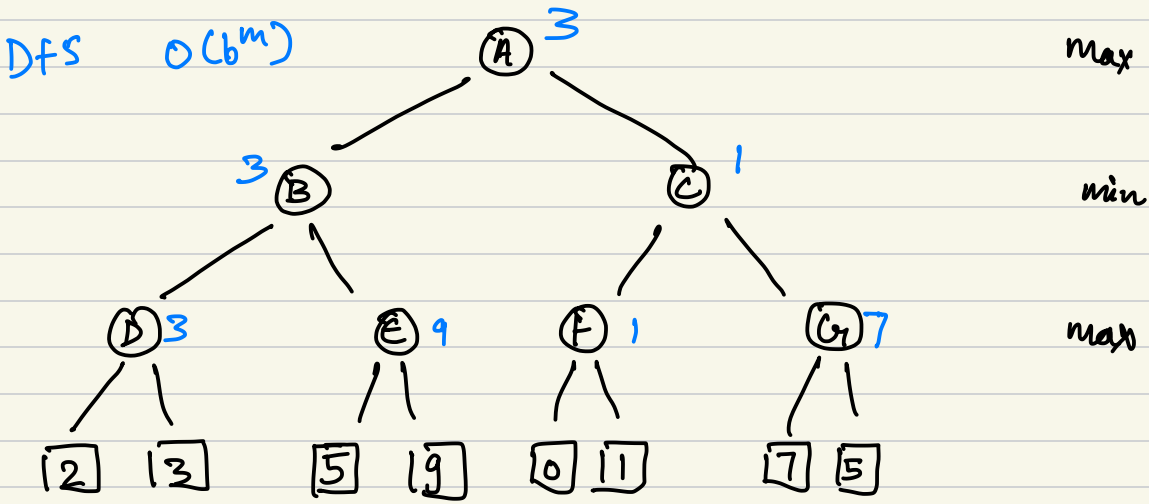
Max Strategy - contingent plan

Minimax

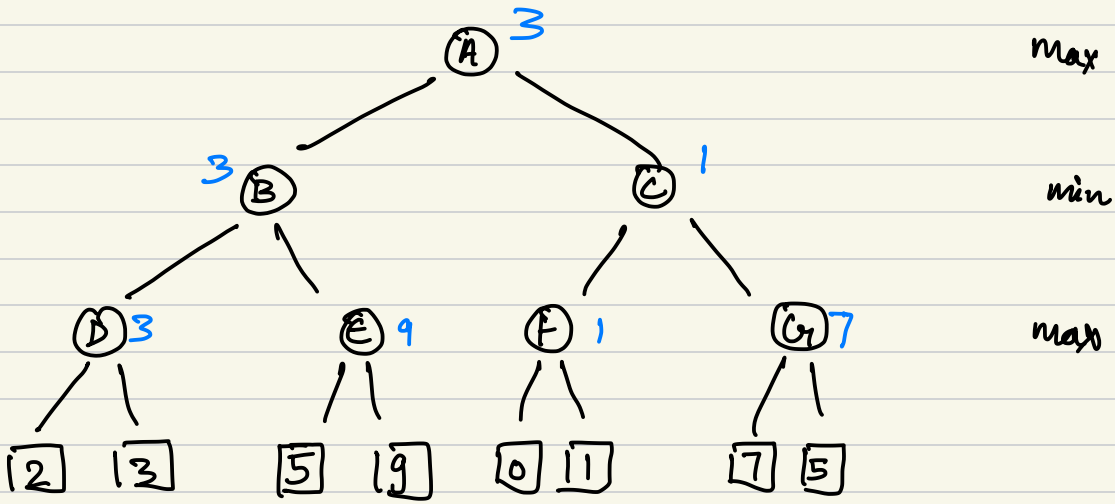


Sep 12

— Minimax Algorithm



Alpha beta pruning



Initially $\alpha = -\infty$

$\beta = \infty$

Update α = max node

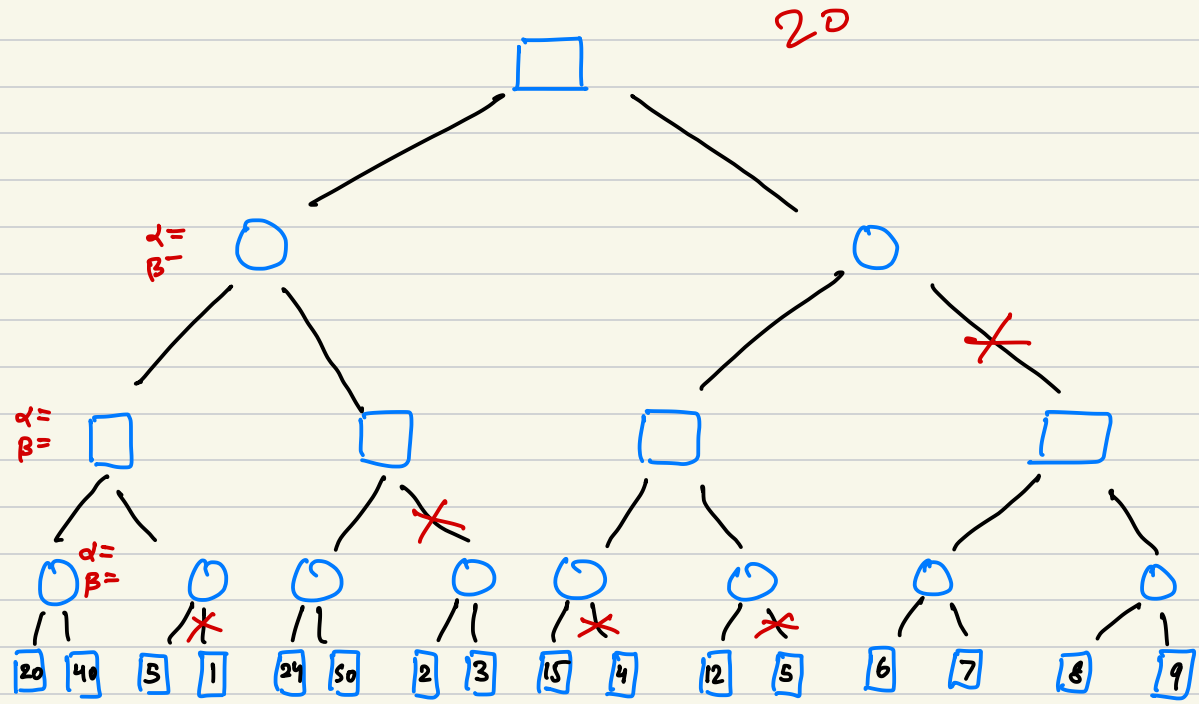
β = min node

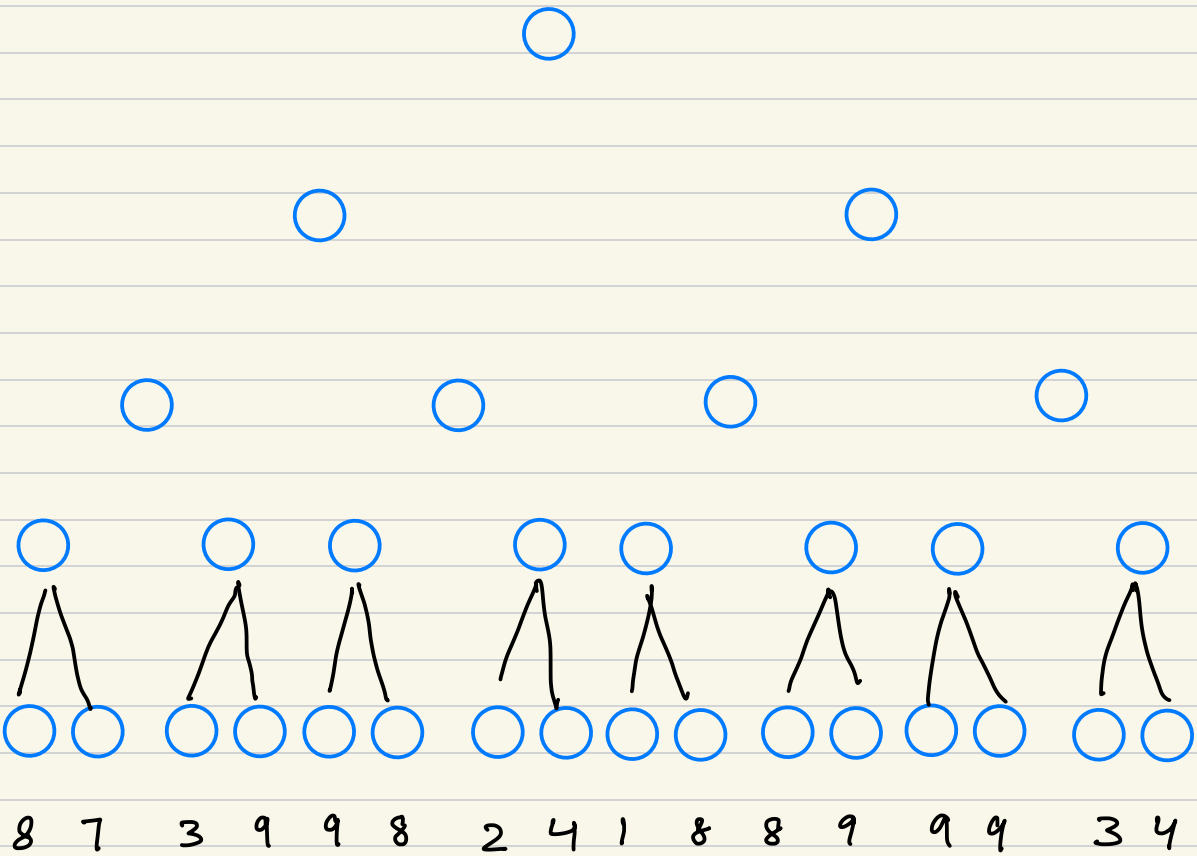
$\alpha \geq \beta$ prune branch

break

break

Node		α	β	Node value
max	α-β(A)	3	∞	5
min	α-β(B)	$-\infty$	3	3
max	α-β(D)	3	∞	3
max	α-β(E)	5	3	5
min	α-β(C)	3	∞	
max	α-β(F)	3	∞	1
max	α-β(G)			


20
α=
β=
α=
β=
α=
β=
20
40
5
1
24
50
2
3
15
4
12
5
6
7
8
9


8 7 3 9 9 8 2 4 1 8 8 9 9 9 3 4

Sep 15

CSP

- variables $X = \{ X_1, X_2, \ldots X_n \}$
- values $\Rightarrow$ domain $D = \{ D_1, D_2 \ldots D_n \}$
- constraints $C = \{ (\text{scope}, \text{rel}) \}$

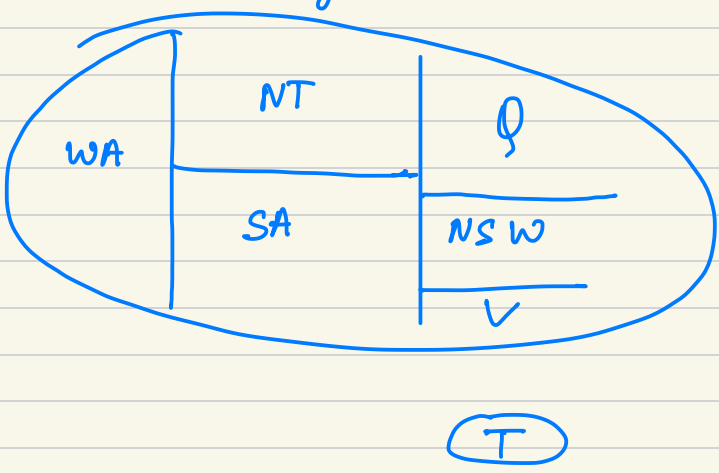
$$C = \{ ((X_1, X_2), X_1 > X_2) \}$$

$$D_i = \{ V_1, \ldots V_n \}$$

- Consistent / legal assignment
- complete assignment
- partial assignment

Map coloring : Color = {r, g, b}

no neighbours should have same colour



Variables, $X = \{WA, NT, SA, Q, NSW, T\}$

$D_i = \{r, g, b\}$

$C = \{ SA \neq WA,\ NT \neq WA,\ WA \neq NT,\ Q \neq NT,\ Q \neq NSW,\ NSW \neq V,\ V \neq SA,\ NSW \neq SA,\ Q \neq SA \}$

$X = (SA \neq WA) = C((SA, WA), (SA \neq WA))$

are abbreviations (Scope, rel)

b r

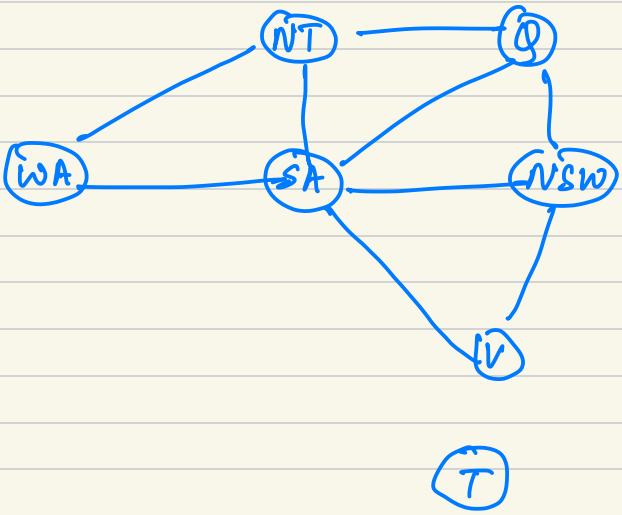
r b

g r

r, g, b

Binary csp

Constraint graph



Cryptarithmetic

```
  BASE
+ BALL
-------
 GAMES
```

Variables = X = { A, B, E, L, G, M, S }

Domain = D_i = {0, ... 9}

Constraint = Alldiff = (A, B, E, L, G, M, S)

C4 C3 C2 C1

```
  B 7 A 4
  S 8 E 3
  B 7 A 4
  L 5 L S
G 1 A 4 M 9
  E 3 S 8
```

$+L \quad S+L = E+L \qquad A+L+E+L = 5+L$

$S+L = G \qquad E+L = S$

$E \qquad 2L = L \qquad L = 1 \text{ or } 2 \qquad AL+L = E$

C_4 C_3 C_2 C_1

B 8 A 6 S 9 E 4

B 8 A 6 L 5 I 5

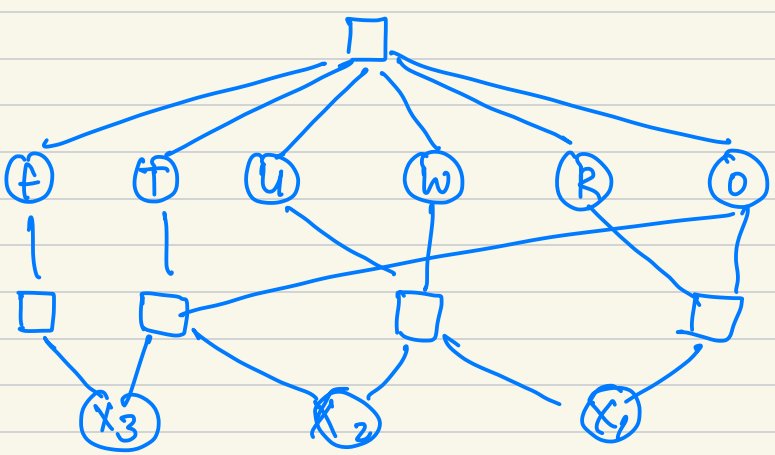
C_4 1 A 7 M 3 E 4 S 9

C_4 C_3 C_2 C_1

T 7 W 3 O 4

T 7 W 3 O 4

F 1 O 4 U 6 R 8



Variables : F T U W R O X_1 X_2 X_3

Domains : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Constraints all diff (F, T, U, W, R, O)

$$O + O = R + 10 \cdot X_1$$

Standard Search formulation (Incremental)

Backtracking

Improving Backtracking

Heu 1 MRV (Min. Rem. Val)

Heu 2 Degree heuristic (Most constraining)

3 Least Constraining Val

Choosing & Assigning values

- Forward Checking
- Constraint Propagation