

CS249 (Artificial Intelligence - 3)

→ Book: ROSELL & NORVIG (read from 1st chap → get motivated from this chap.)

→ 54 students : 11 groups } Assignment - 1

1 - 5

:

36 - 40

:

51 - 54

11 weeks - 11 groups

will prepare Basic Slides of that week,

↓

Class Notes } Blue / Black Topics } white Background

Deadline: every week Saturday

(Include citations & contribution of every person)

Marks are given by Students }

avg. to 10

→ Quiz / Assignment

MSE

ESE

Min Project

Real life

App. of AI
(group of n=3)

20 - 30%

30%

80% - 40%

AI :

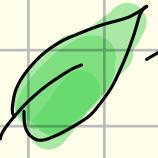
→ Thinking / Perception / Action

1st week of April Deadline LAB MSE / ESE

Cognitive Perceptive

→ Model Targeted at thinking Perception and action to explain

Past, Predict future, understand the subject, control the "World".

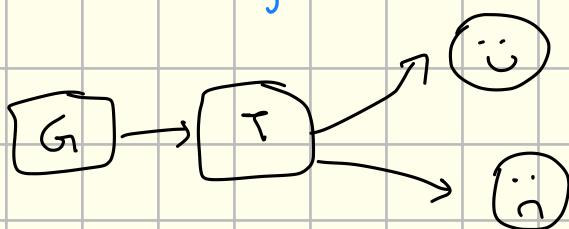
→  which tree does this leaf belong to?

to answer this question we try to recall our Past experiences & try to answer the Q.

→ we will try to Perform some Representation that

Supports our Q. of leaf matching to tree

→ Generate of Test:



→ generator shld'nt generate redundant

→ shld be informative
so that we have

→ Representation right

* SIMPLE ≠ TRIVIAL

→ SIMPLE: Can be Powerful

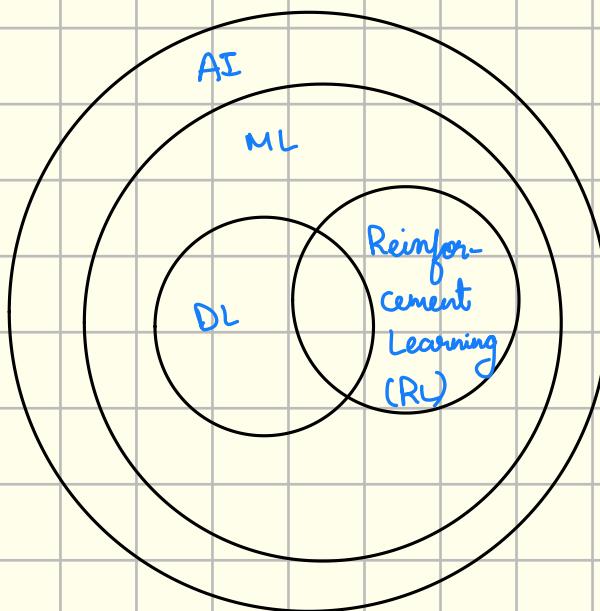
→ TRIVIAL: makes it sound its not only simple but has a

little *worth*

→ SIMPLE IDEAS may be Powerful

→ Don't always go for the Complicated one

→ Big Picture :



→ Solve the AI problems using REASONING

→ Algorithms Enabled by
Constraints Enforced AT
Representation that supports

Model targeted at Thinking / Perception / Action

→ Engineer → "Building tool kits"

ITAN

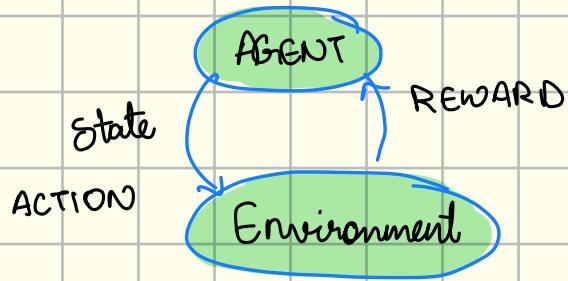
Computer Scientists

→ Computational Account of Intelligence

(understanding the Background that how they work)

→ actions are based on your Cognitive Perception of Observation

Eg: Autonomous Cars



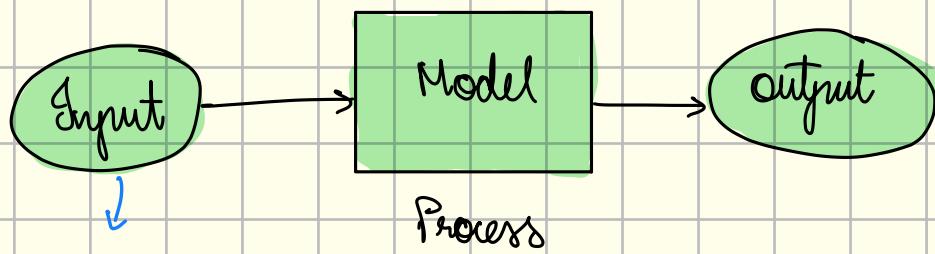
" KNOWLEDGE IS POWER "

→ LAB : Python → w3SCHOOLS
↓
Frameworks (LIBRARIES)

4-5 memb. group
Own Laptop

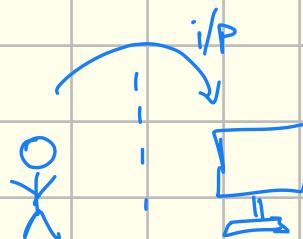
→ Prerequisite : ALGORITHMS

→ AI : Computational Modelling



if the input is
anything other than nos \Rightarrow 1st convert them to nos

Turing Test (1950)



- The Computer is interrogated by a human in a teletype
 - IT PASSES, if the human can't tell if there is a comp. or a human at the other end.
- (↑)
O/p
if the human identifies that this o/p is given by human \Rightarrow comp.
Passes turing test

→ The ability to solve Problem (computational Problem)

- Search: Efficient TRIAL & ERROR

SPACE / TIME Complexity trade off

"a student is poor in Algo"

- Use of Domain knowledge (Heuristic)

Block file : insertion > deletion / updation \Rightarrow

which type of DS is used?

\rightarrow linked list

insertion $O(1)$

- Popular Techniques : Linear Programming

Integer Programming

Dynamic Programming

Heuristic Search

Evolutionary Algorithm

AGENT :

- It may be a Computer Program but it should be capable of doing "more"
- AI models created should be a Responsible AI model

KNOWLEDGE AND DEDUCTION:

- How to store & retrieve knowledge?
- How to Interpret facts & rules and able to deduce?
- The GAP between knowledge and realisation.
- Logics of knowledge
 - (knowledge based systems : KBS
 - Expert systems : ES
- Automated Theory Problems FORMAL verification)

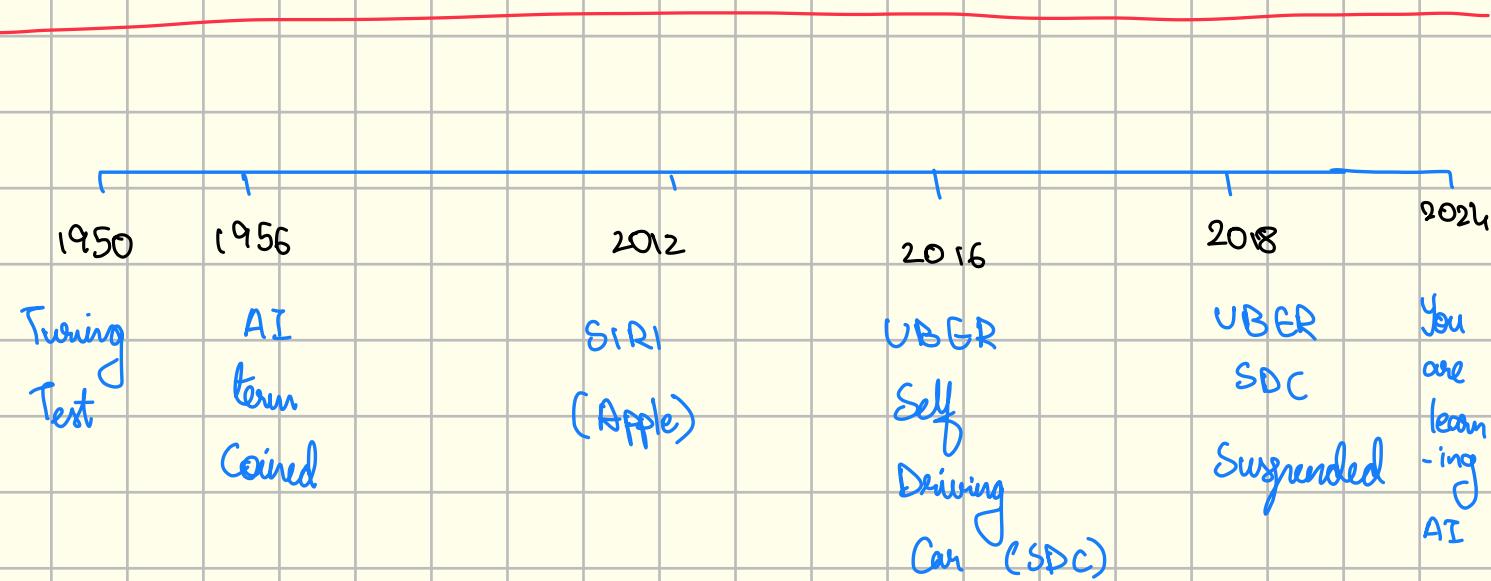
The Ability to learn:

- Can we learn to solve a problem better?
 - learning the answer.
 - learning the rule of GAME

→ learning to PLAN

→ Belief Network

→ Perception of Network N/w



What is AI?

- Automated Problem Solving
- ML
- Logic and Deduction

Human - Comp. Interaction

- Comp. Vision
- NLP
- Robotics

FUNDAMENTALS:

- ① The notion of expressing computation as an ALGORITHM
- ② GODEL's INCOMPLETENESS Theorem (1931)
- ③ CHURCH - TURING THESIS (1936)

→ Turing machine is capable of computing any

Computable Functions

→ This is accepted definition of Computing

④ 2 Notions of Interactability

- NP Completeness
- Reduction

⑤ Problem Solving by Search

- State space search
- Prob. reduction ,
- Game Playing

⑥ logic and deduction

(1st order logic temporal logic)

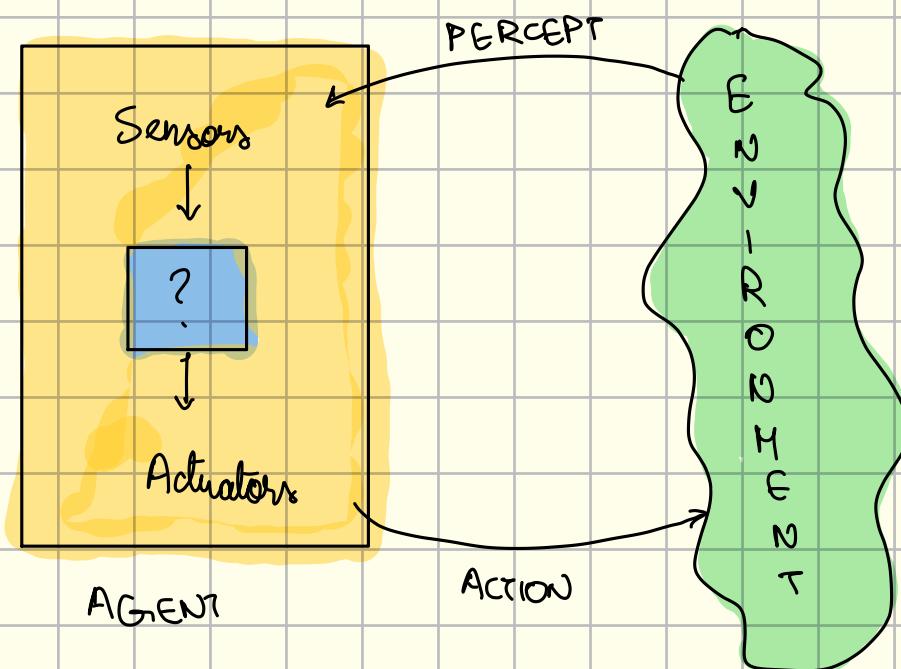
- Planning
- Reasoning under uncertainty
- Learning
- additional topics

AGENT :

- * An agent is just something that ACTS.

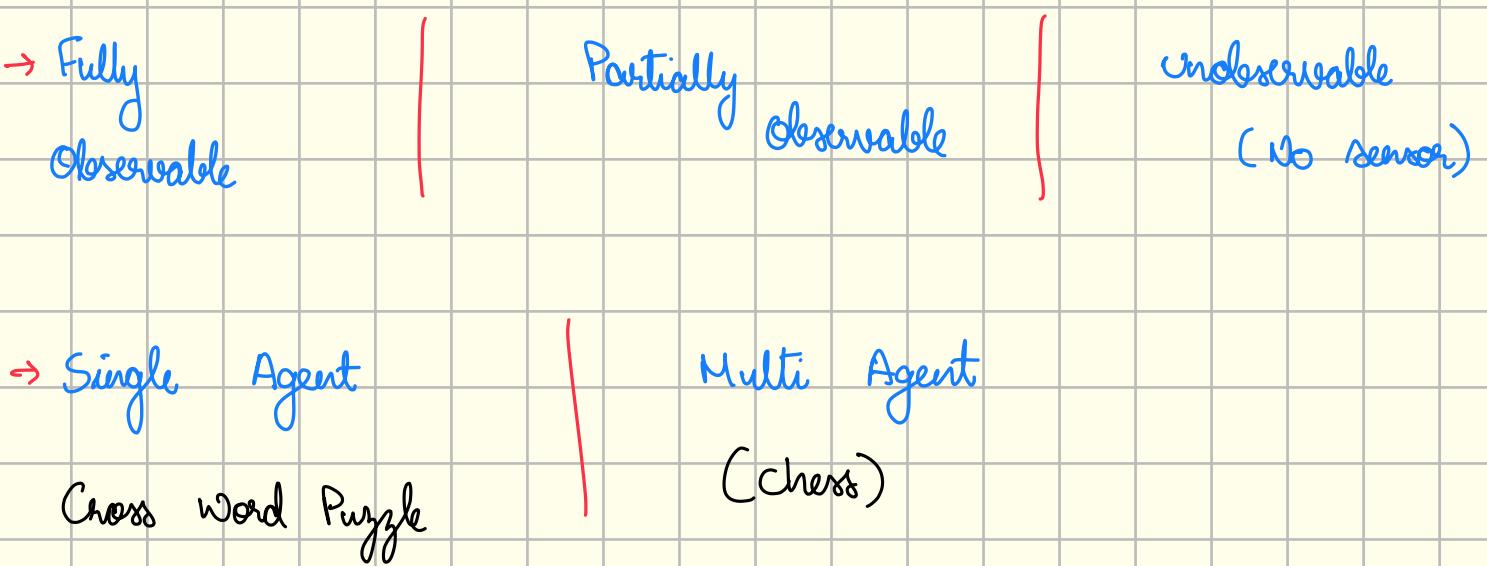
- All comp. programs do something but AGENT does something more.
 - Operate autonomously
 - Perceive their Environment
 - Persist over a prolonged time-period
 - Adopt to change, create or pursue goals.

- A Rational agent is one that acts so as to achieve the best outcome or where there is uncertainty
- the best expected outcome.



Agent type	Performance Measure	Environment	Activator	Sensor
------------	---------------------	-------------	-----------	--------

TAXI DRIVER	SAFE, LUXURY, COMFORT	Road Pedestrian	A, B, C, Steering	Speedometer, GPS, Comp. Vision, rotative eye, camera
MEDICAL diagnostic system	health of the Patient	Hospital, Staff of Patient	Tests, Treatments	Questionnaires
Interactive English Tutor	Students Score	Set of Students	Exercise	Smart Phone/ key Board



→ **Deterministic** v/s **Stochastic Environment**

↓

Next action is determined by the current state

↓

Next action can't be determined
Uncertain / Non-deterministic

→ Episodic v/s Sequential

)

agents experience is divided into atomic episodes.

→ many states together make an environment

^{1st} Episode

Eg: In a game ^{1st} play : go through states → know the ↑ environment

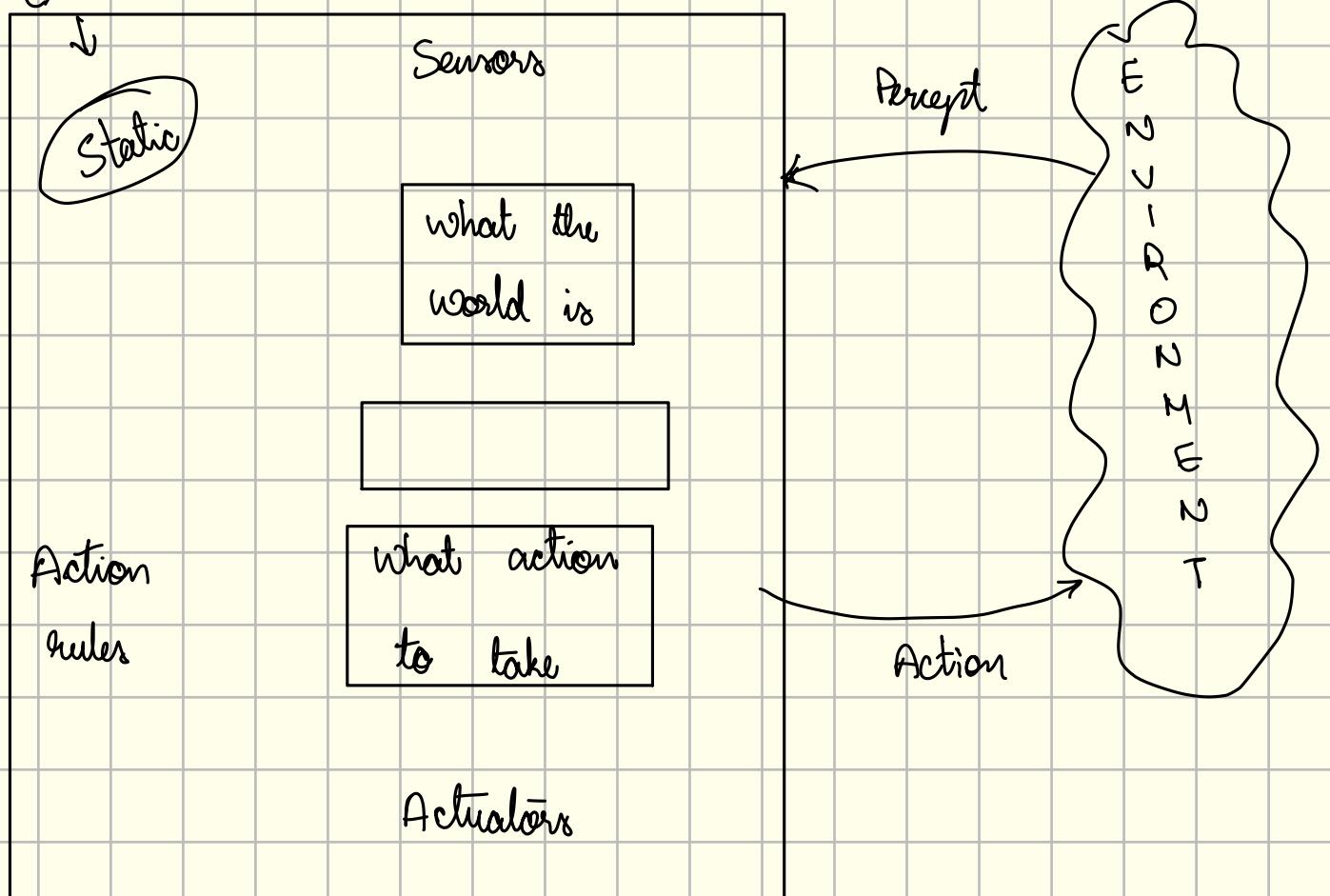
2nd Play : " " → " " ↓
2nd Episode

→ Discrete | Continuous

→ known v/s Unknown

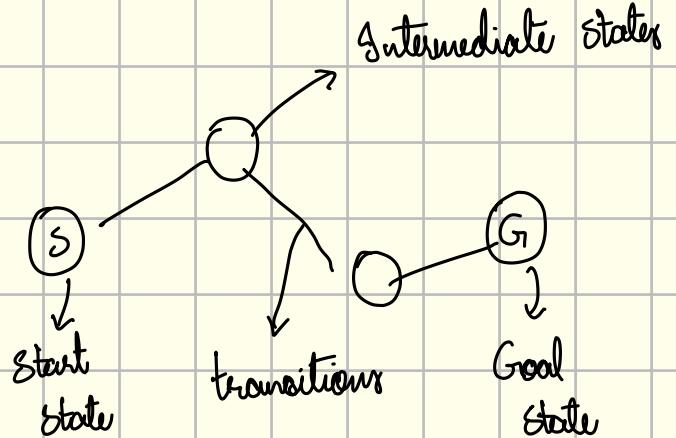
Discrete	Total Environment	Observation	Agent	Deterministic	Episode	static
Discrete	Crossword	full	Single	Deterministic	Sequential	static
"	Chess with	"	Multi	"	"	Semi
Continuous	Test driving	Partially	Multi	Stochastic	"	Dynamic

Critic



SEARCH FRAMEWORKS

- Uninformed / BLIND
- INFORMED / HEURISTIC
- Problem reduction
- Game Tree Search
- Others



BASIC SEARCH PROBLEMS:

→ Given $[S, s_0, O, G]$ where

S is the (implicitly specified) set of states (diff. b/w implicit & explicit)

$s \in S$ is the start state

O is the state transition operators

$G \subseteq S$ is the set of goals

→ To find a sequence of state transition leading from s to G

→ 15 Puzzle game.

Eg: 8 Puzzle

1	2	3
4	5	6
7	8	

G

6	7	2
5	1	4
8		3

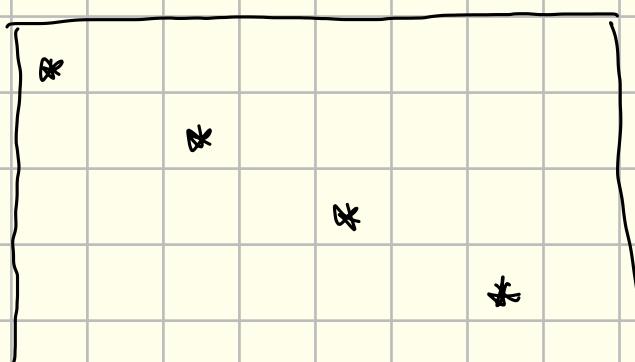
s

4 1 2
5 3
7 8 6

4 2
5 3
7 8 6

O : movement of blank tile in LRUD

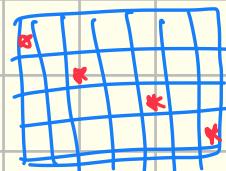
→ 8 Queen Problem:



8 Queen Problem :

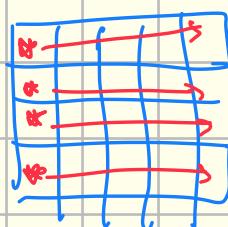
Placing 8 queens so that none attack each other

- (i) A state is arrangement of 0-8 queens on board
 Operators add a queen to any sequence.
- (ii) A state is any arrangement of 0-8 queens with
 none - attacked operators place a queen in the left
most empty Column.



$$8 \rightarrow 8^2 \rightarrow 8^3 \dots$$

- (iii) A state is any arrangement of 8 queen operators
 move an attacked queen to another square in the
 same row.



Each queen has
 7 blank space to
 move

$$7 \times 8 \rightarrow (7 \times 8)^2 \rightarrow (7 \times 8)^3 \dots$$

8 queen
 available moves
 (7x8) x (7x8) = ...
 1st move 2nd move

Missionaries & Cannibals Problem

M

C

→ 3 M & 3 C are one side of a river, along with

a BOAT that can hold 1 or 2 people.

→ Find a way to get out to the other side without leaving a group of M outnumbered by c.

$$S = \{ \langle x, y, z \rangle, x \in \{0, 1, 2, 3\}, y \in \{0, 1, 2, 3\}, z \in \{0, 1\} \}$$

$x \rightarrow$ no. of M on left

$y \rightarrow$ " " c on right

$z \rightarrow 0$: boat on left

1: boat on right

$$S = \{3, 3, 0\}$$

$$G = \{\langle 0, 0, 1 \rangle\}$$

$\langle 0, 0, 0 \rangle$

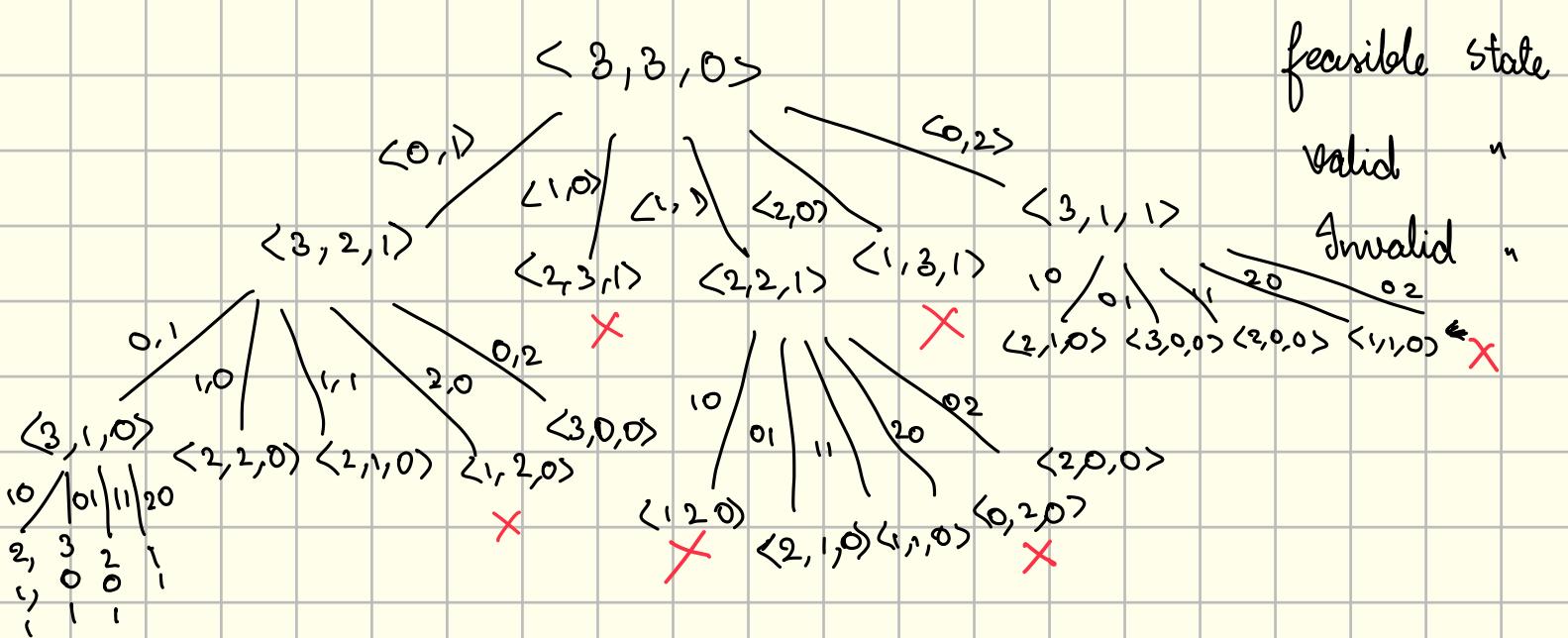
Infeasible State

$O : \{ \langle O_1, O_2 \rangle$	$\langle 0, 0 \rangle \times$
\downarrow	$\langle 0, 1 \rangle$
$\#M$	$\langle 1, 0 \rangle$
\downarrow	$\langle 1, 1 \rangle$
O_1 on boat	$\langle 2, 0 \rangle$
O_2 on boat	$\langle 0, 2 \rangle$

$$\left. \begin{array}{l} O_1 \in \{0, 1, 2\} \\ O_2 \in \{0, 1, 2\} \\ 1 \leq O_1 + O_2 \leq 2 \end{array} \right\}$$

$$O : \{ \langle O_1, O_2 \rangle : O_1 \in \{0, 1, 2\}, O_2 \in \{0, 1, 2\}, 1 \leq O_1 + O_2 \leq 2 \}$$

→ draw the entire chart of how many min state transitions are required to reach goal state.



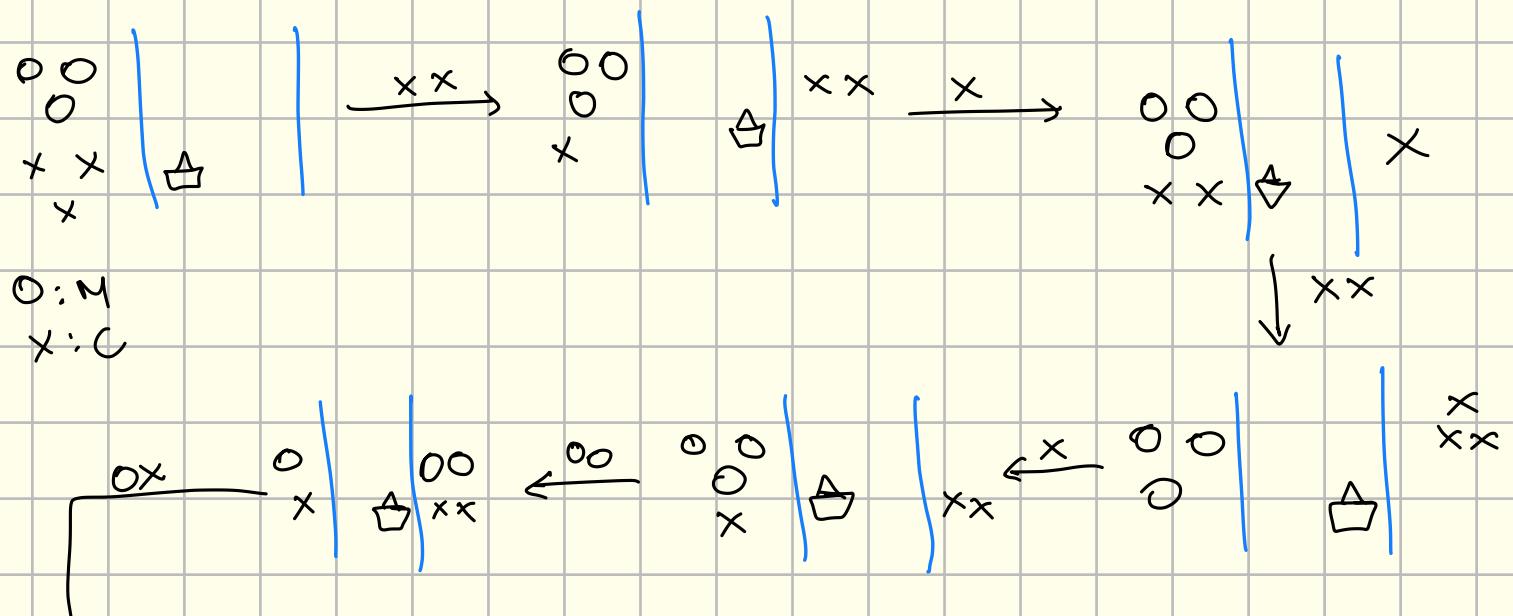
Valid

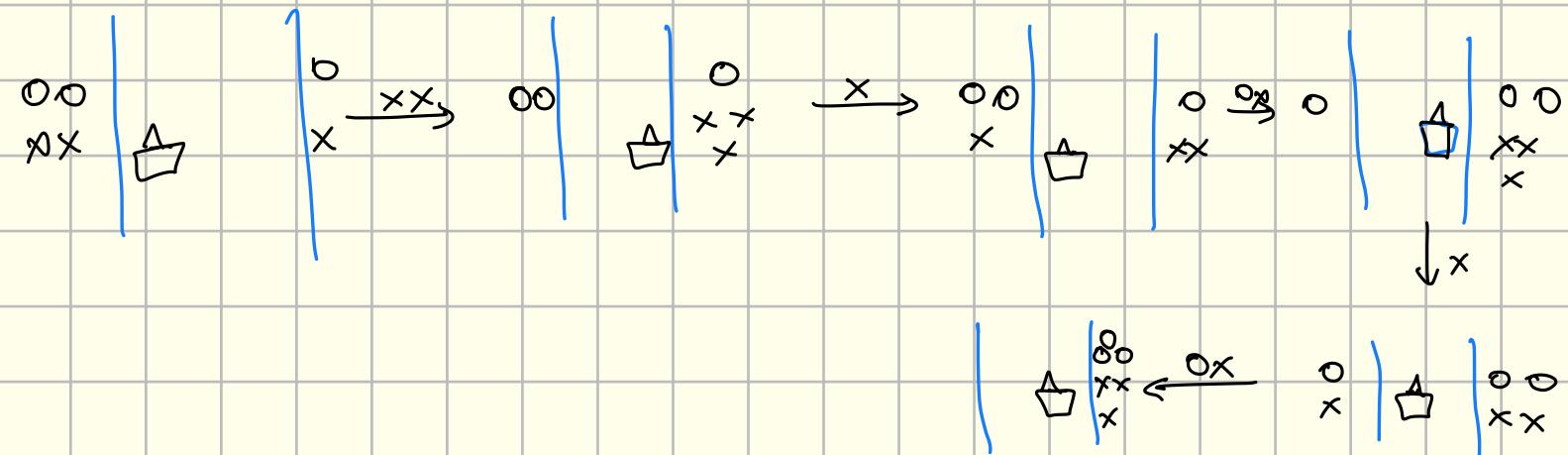
R Feasible State

Invalid (based on some constraint)

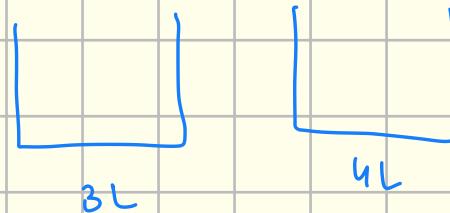
R Infeasible State (not reachable from s)

* Undefined State (Not matching as per the def.ⁿ of the state.)



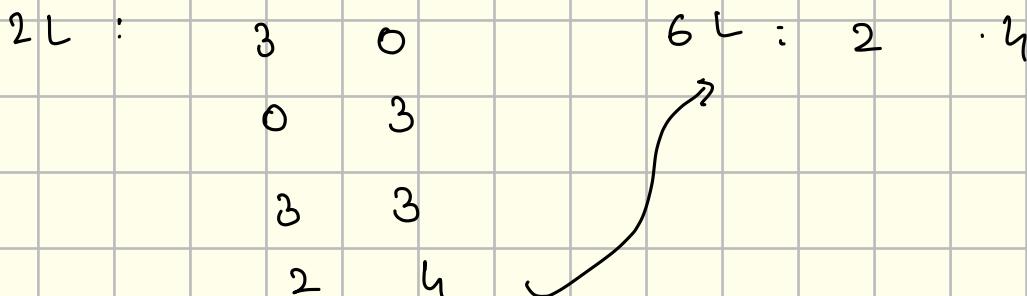
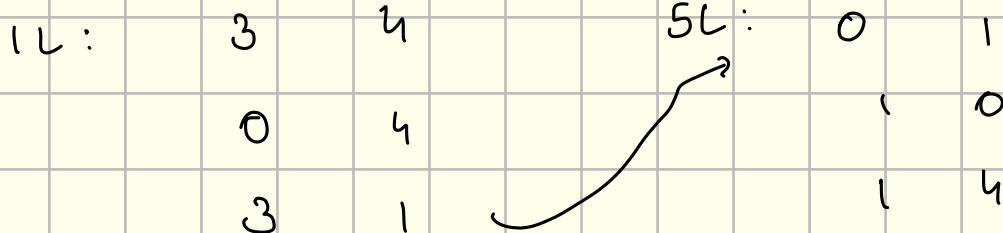


JUG / JAR PROBLEM:



Problem: measure $0, 1, 2, \dots, 7$ L using 3L & 4L jugs

0 3 4 7



→ Similarly we can do it for 4L & 5L vessels.

Formulate this into an AI Problem:

→ S, O, G

$S: \{ \langle x, y \rangle : \begin{array}{l} x: \text{amount of water in } 1^{\text{st}} \text{ vessel}, 0 \leq x \leq 3 \\ y: \text{amount of water in } 2^{\text{nd}} \text{ vessel}, 0 \leq y \leq 4 \\ x, y \in \mathbb{Z} \end{array} \}$

$S: \{ \langle 0, 0 \rangle \}$

$O: (0, y) : \text{making the } 1^{\text{st}} \text{ jar empty}$

$(x, 0) : \text{making the } 2^{\text{nd}} \text{ jar empty}$

$(3, y) : \text{fill } 1^{\text{st}} \text{ jar}$

$(x, 4) : \text{fill } 2^{\text{nd}} \text{ jar}$

$(x+k, y-k) : \text{transfer } k \text{ lt. of water from } 2^{\text{nd}} \text{ jar to } 1^{\text{st}} \text{ jar}$

$$k = \min(3-x, y)$$

$(x-k, y+k) : \text{transfer } k \text{ lt. of water from } 1^{\text{st}} \text{ to } 2^{\text{nd}} \text{ jar}$

$$k = \min(x, 4-y)$$

$G: \{ (x, y) : \begin{array}{l} 0 \leq x \leq 3 \\ 0 \leq y \leq 4 \\ x, y \in \mathbb{Z} \\ x+y = T \\ T \in \mathbb{Z} \end{array} \}$

* * *

TRAVELLING SALESMAN PROBLEM:

→ Problem: there is a salesman he has to travel to diff. cities to sell his products.

- 1) Visit every city except source one of only once

2) Start from source & come back to source.

3) Minimize Cost (Cost can be time / Money)

→ AI Problem Statement : S, S, O, G, C
 └→ Cost.

S: Source

exactly once

G: find a path with min. Path that visits every city & comes back to source

S: Multiple Paths available

O: the different transition available to go from one city to another

P D → H C B P

P → D B H C P

→ $S! \Rightarrow$ diff. ways are possible.

C: Cost : can be money / time.

SEARCH ALGORITHMS : (BASIC)

① Initialize : Set $OPEN = \{S\}$

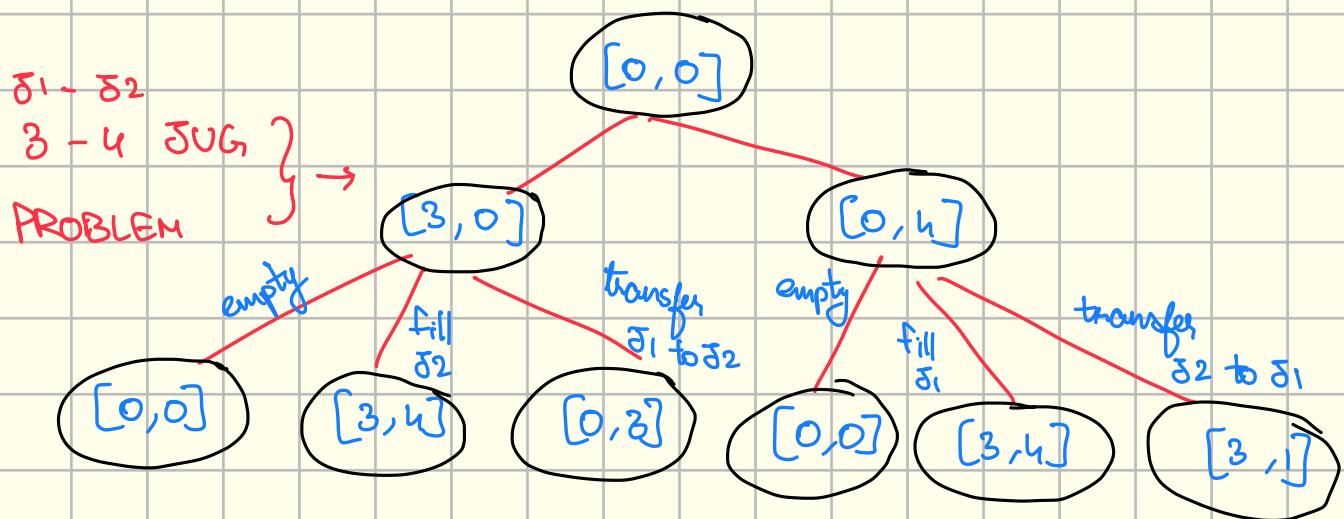
② Fail : If $OPEN = \{\}$, terminate with failure

③ Select : Select a state n from $OPEN$

④ Terminate : If $n \in G$, terminate with success

(5) Generate the successor / neighbours of n units δ_1 and δ_2 and insert them in OPEN.

(6) LOOP : Go to step (2)



BFS \rightarrow Queue
DFS \rightarrow Stack

SEARCH ALGO FOR STATE SPACE SEARCH :

→ we will maintain 2 lists, OPEN, CLOSED, for searching

$$P_n = [\delta, \text{OPEN}, \delta, \text{CLOSED}]$$

Initialization : Insert δ to OPEN, CLOSED = []

Termination with : Remove node n from OPEN
Success
if $n \in G$ then terminate with success
add n in CLOSED

loop

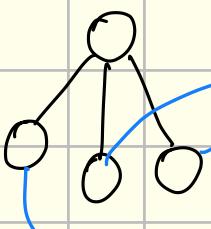
Selection : Apply all $o \in O$ on 'n' to generate
all neighbours n of n [$n \subseteq S$]

+ $m \in N$ check if $m \notin (\text{OPEN} \cup \text{closed})$

Add m in OPEN

Termination with Failure : If OPEN is empty, terminate with failure
otherwise go for looping / reset iteration

→ Algorithm : Time Complexity | Space Complexity | Completeness | Optimality



b: branching factors

d: depth

Acyclic graphs

↳
trees

$$\begin{array}{l} b=3 \\ d=1 \end{array}$$

→ The Algo is generic and "blind" because
it is not using any domain knowledge.

→ The Characteristics of the algo depends on what data structure
we are using for OPEN

→ A graph is created however the graph G_i is an **Implicit graph**. because all the vertices and edges are not known beforehand.

(1st we decide the operation of based on that we get the vertices else we don't know ver. beforehand.)

→ we begin with start state, keep on building G_i till we reach goal state

BFS | DFS : time Complexity $O(b^d)$

b : branching factor
d : depth

Space Complexity :

DFS : $O(b^d)$ d : The max. depth of the graph from start state

BFS : $O(b^d)$ d : min. level at which goal state can be formed

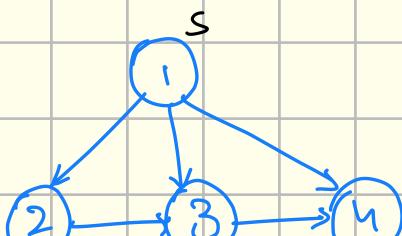
Completeness & Optimality :

DFS : Incomplete & Not Optimal

BFS : Optimal & Complete

work :

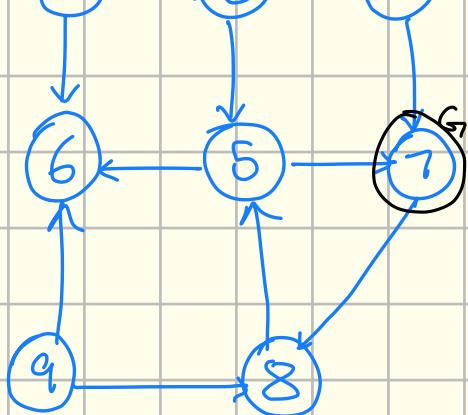
AI RELATED RIDDLES



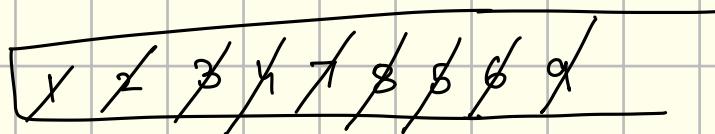
PERFORM DFS :

Stack :

1, 2, 3, 4, 7, 8, 5, 6, 9

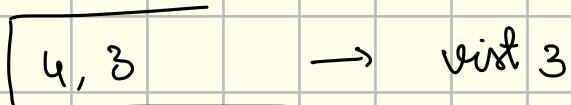
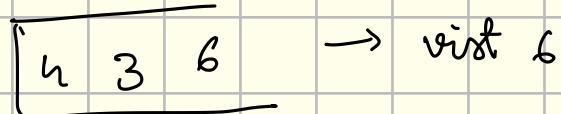
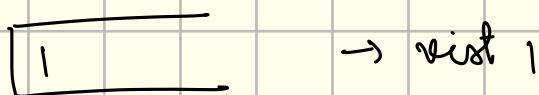


OPEN [stack]:



$\Rightarrow 1 \ 4 \ 2 \ 8 \ 5 \ 6 \ 3 \ 2 \ 9$

OPEN [stack]:



PERFORM QUEUE:

1

→ visit 1

2 3 4

→ visit 2

3 4 6

→ visit 3

4 6 5

→ visit 4

6 5 7

→ visit 6

5 7

→ visit 5

7

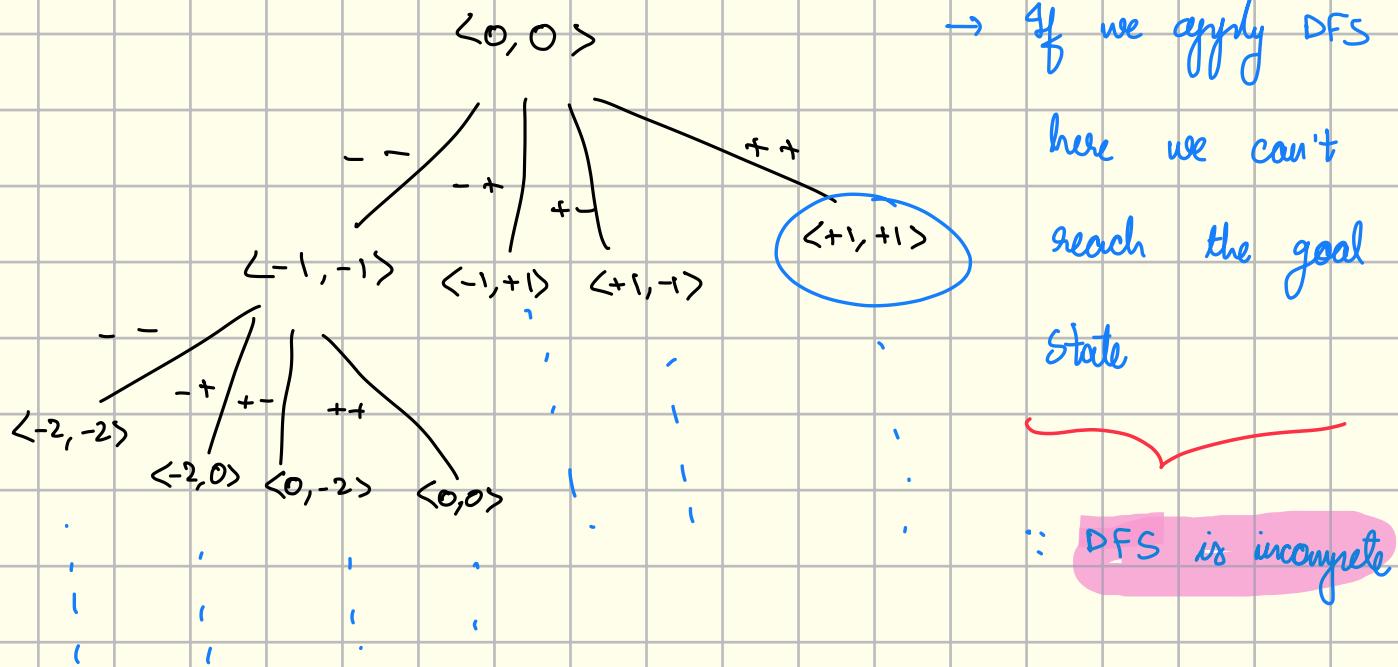
→ visit 7

→ STATE SPACE may be infinite

$s_0 : \langle 0, 0 \rangle$

$G_i : \langle 1, 1 \rangle$

$O : \langle +, + \rangle, \langle -, - \rangle, \langle +, - \rangle, \langle -, + \rangle$



COMPLETENESS: If start from s if there exist G then after completing the search we should be able to reach the goal state

→ diff. b/w. A Goal of the Goal

BFS:

→ Time: $b^d \rightarrow 1 + b + b^2 + \dots + b^d = O(b^d) \rightarrow d: \text{depth of graph}$

→ Space: b^d

$d: \text{depth of goal state}$

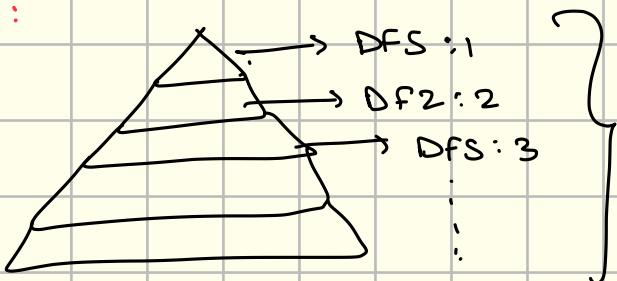
DFS:

→ Time: b^m

→ Space: $b \cdot m \rightarrow m \text{ depth} \times b \text{ brothers} \rightarrow m: \text{depth of the state space tree}$

	TIME	SPACE	COMPLETENESS	OPTIMALITY
BFS	$O(b^d)$	$O(b^d)$	✓	✓
DFS	$O(b^m)$	$O(b \cdot m)$	✗	✗
Depth limited DFS (DDFS)	$O(b^l)$ l: depth limit	$O(b \cdot l)$	✗	✗
Iterative Deepening DFS (IDFS)	$O(b^d)$	$O(b \cdot d)$	✓	✓
Bidirectional BFS (if Possible) (BBFS)	$O(b^{dr_2})$	$O(b^{dr_2})$	✓	✓

IDFS:



we keep changing the depth and apply the DFS.

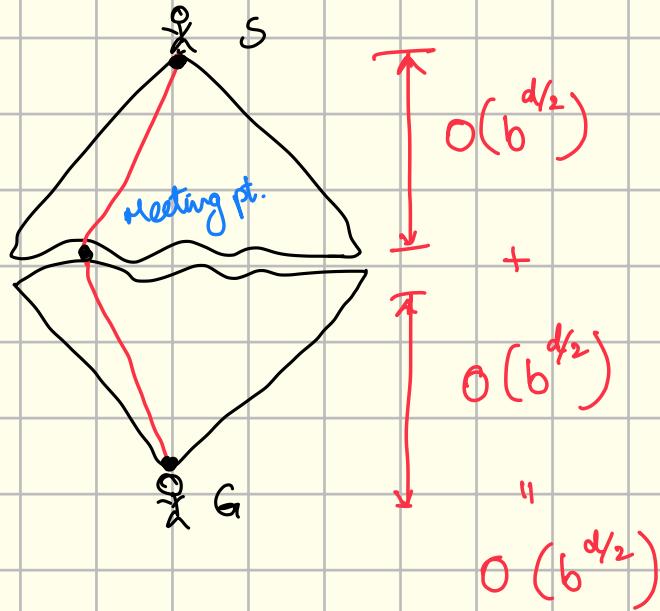
Space : $O(b \cdot m)$

Time : $O(b^m)$

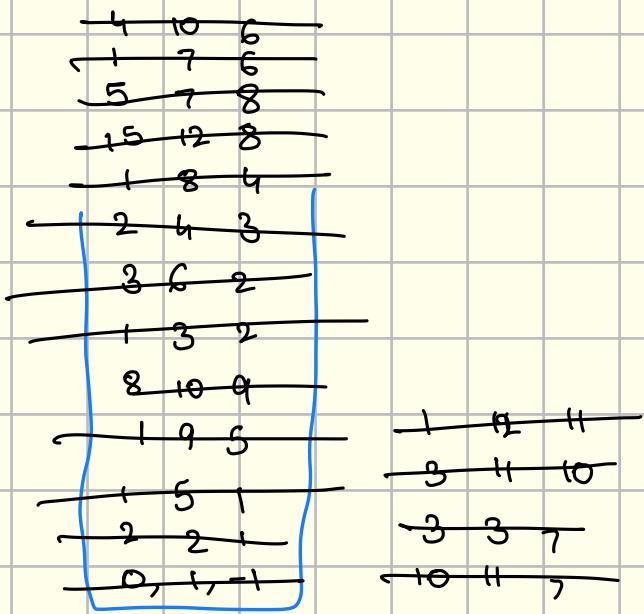
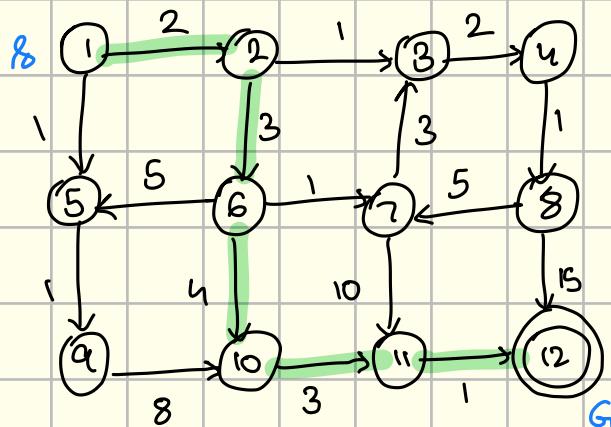
→ It is Complete & Optimal

BIDIRECTIONAL :

→ It may not always be possible to go from goal to s



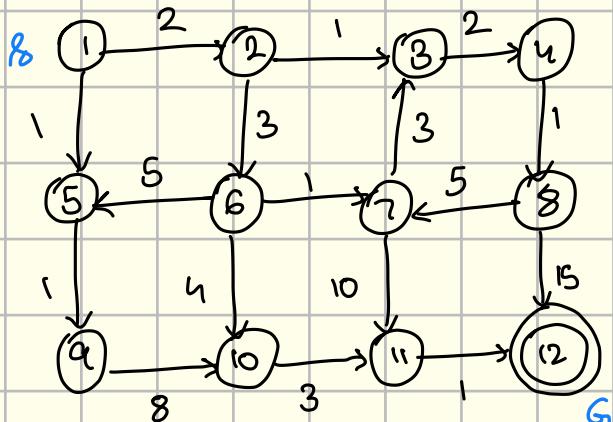
→ An Algo is called **COMPLETE**, if it **always** returns a solution / Goal state, if there exists a goal state which is reachable from the start state.



	1	2	3	4	5	6	7	8	9	10	11	12
vis	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅
dist	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

dist	0	2	3	5	1	5	6	6	2	9	12	13
par	-1	1	2	3	1	2	6	4	5	6	10	11

par	-1	1	2	3	1	2	6	4	5	6	10	11
par	-1	1	2	3	1	2	6	4	5	6	10	11



OPEN

$1^{(0)}$
$2^{(1)} \quad 5^{(1)}$
$2^{(2)} \quad 9^{(1)}$
$9^{(1)} \quad 3^{(3)} \quad 6^{(5)}$
$3^{(3)} \quad 6^{(5)} \quad 10^{(6)}$
$6^{(6)} \quad 10^{(9)} \quad 4^{(5)}$
$10^{(9)} \quad 4^{(5)} \quad 7^{(6)} \quad 10^{(9)}$
$7^{(6)} \quad 10^{(9)} \quad 8^{(6)}$
$10^{(9)} \quad 8^{(6)} \quad 11^{(6)}$
$10^{(9)} \quad 11^{(16)} \quad 12^{(11)}$
$10^{(16)} \quad 12^{(21)} \quad 11^{(12)}$
$12^{(21)} \quad \boxed{12^{(15)}}$

CLOSED :

$1^{(0)} \quad 5^{(1)} \quad 9^{(1)} \quad 3^{(3)} \quad 6^{(5)} \quad 4^{(5)} \quad 7^{(6)} \quad 8^{(6)} \quad 10^{(9)} \quad 11^{(12)} \quad 12^{(13)}$

Homework : UNIFORM COST SEARCH

FIRST SEARCH ALGORITHM :

- 1) Initialize : Set OPEN - {s}
- 2) FAIL : If OPEN = {} . Terminate with failure.
- 3) SELECT : Select a start, n , from OPEN
- 4) TERMINATE : If $n \in G$, terminate with success
- 5) EXPAND : Generate the successors of n using O and insert in OPEN

6) LOOP : Goto Step 2

SAVING THE EXPLICIT SPACE:

- 1) $\text{CLOSED} = \{\}$
- 2) FAIL: \times
- 3) SELECT: Select a state n from OPEN and save n in closed
- 4) \times
- 5) Expand: for each successor m in n , insert m in OPEN only if $m \notin [\text{OPEN} \cup \text{CLOSED}]$

SEARCH AND OPTIMIZATION:

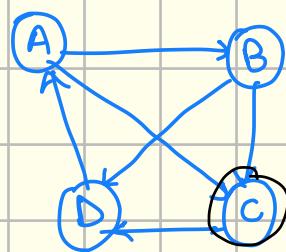
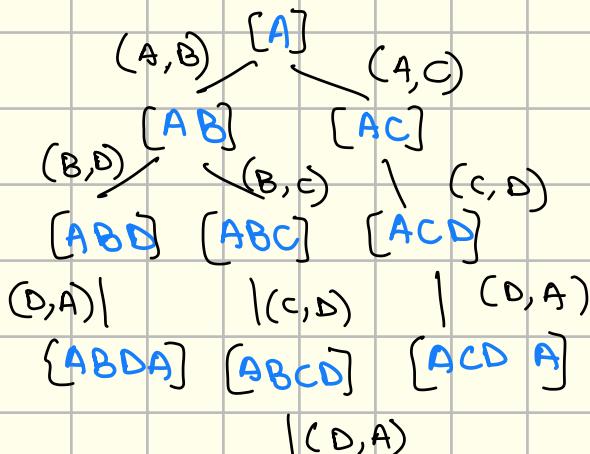
→ Given : $[S, S_0, O, G, C]$

→ To Find :

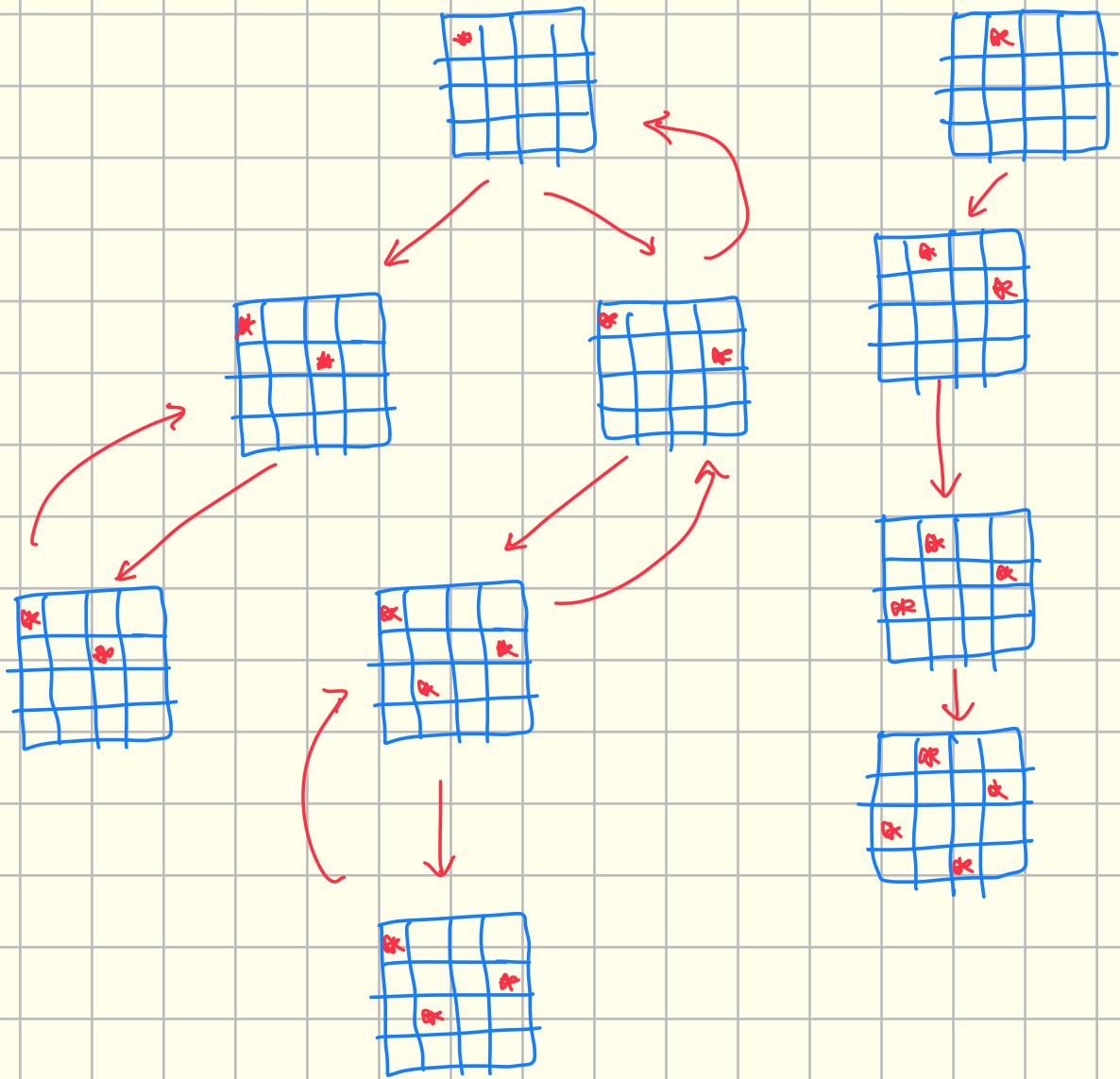
(i) → A min^m cost sequence of transition to a goal state

(ii) Queen Problem → A sequence of transitions to the min^m cost goal

(iii) → A min^m cost sequence of transition to a min^m cost



[A B C D A]



Homework: Think about (iii)

→ UCS (Uniform Cost Search) assumes all operations have a cost

① Initialize : set $OPEN = \{s_0\}$
 $CLOSED = \emptyset$, set $C(s_0) = 0$

② FAIL : if $OPEN = \emptyset$, terminate w/ FAIL

③ Select : select the min "cost" state 'n'

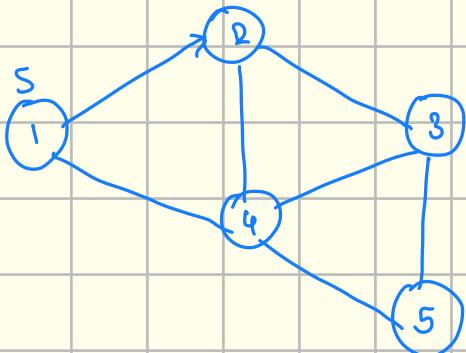
from OPEN and solve 'n' in closed

- ④ Terminate : If NEG , terminate with success
- ⑤ Expand : Generate the successors of n using \circ for each successor, m .
- If $m \notin [\text{OPEN} \cup \text{CLOSED}]$
- set $c(m) = c(n) + c(n, m)$
- If $m \in [\text{OPEN} \cup \text{CLOSED}]$
- set $c(m) = \min \{c(m), c(n) + c(n, m)\}$
- If $c(m)$ has decreased and $m \in \text{CLOSED}$, move it to OPEN

⑥ LOOP

HomeWork: find diff. b/w UCS & Dijkstra \rightarrow why the diff.?

UCS (diff. with Dijkstra)



$$G: \{2, 3, 4, 5\}$$

UCS :



$u.\text{Cost} = 0$ (cost to reach v from s)

$v.\text{Cost} = 5$ (cost to reach v from s)
↳ Cost(u, v)

Dijkstra \rightarrow no -ve edges
UCS \rightarrow allow -ve edges

$n.\text{cost}$ - The cost to reach n from s

Initialization : Open : [s] Closed : [] $s.\text{cost} = 0$

Iteration : Remove the lowest cost state n from OPEN.

CLOSED.append(n)

(use Heap / priority Queue) \rightarrow ($O(1)$ to find min elem.)

Termination : if $n \in G$ Then terminate with success.
with Success

(This termination cost is used only when all operator cost are +ve)

Selection : Apply all $o \in O$ on ' n ' to obtain weight of n (say)

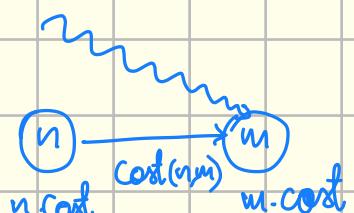
$m \notin [\text{OPEN} \cup \text{CLOSED}]$

$$m.\text{cost} = n.\text{cost} + \underbrace{\text{cost}(n, m)}_{B}$$

If $m \in \text{OPEN}$ and $m.\text{cost} > B$

Revise the cost of m

$$m.\text{cost} = B$$



This cond. " n "
will be
encountered
only when
there is
-ve weight

If $m \in \text{CLOSED}$ and $m.\text{cost} > B$

Revise cost of m

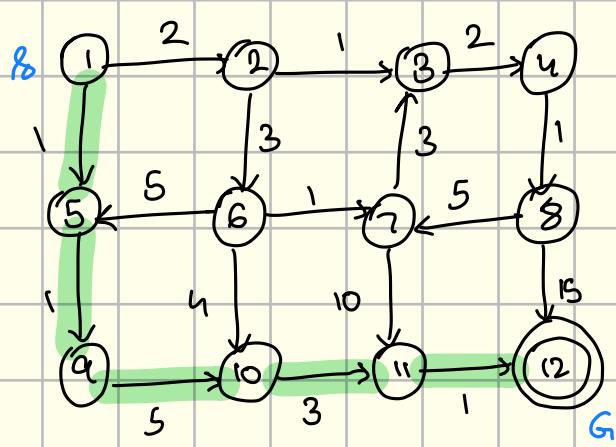
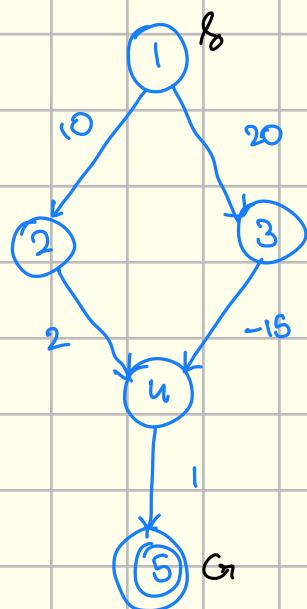
$$m.\text{cost} = B$$

move m from CLOSED to OPEN

Terminate with Failure : If $\text{OPEN} = []$, terminate with failure

Exhaustive Search : Terminate cond " with graph having -ve edge

Example :

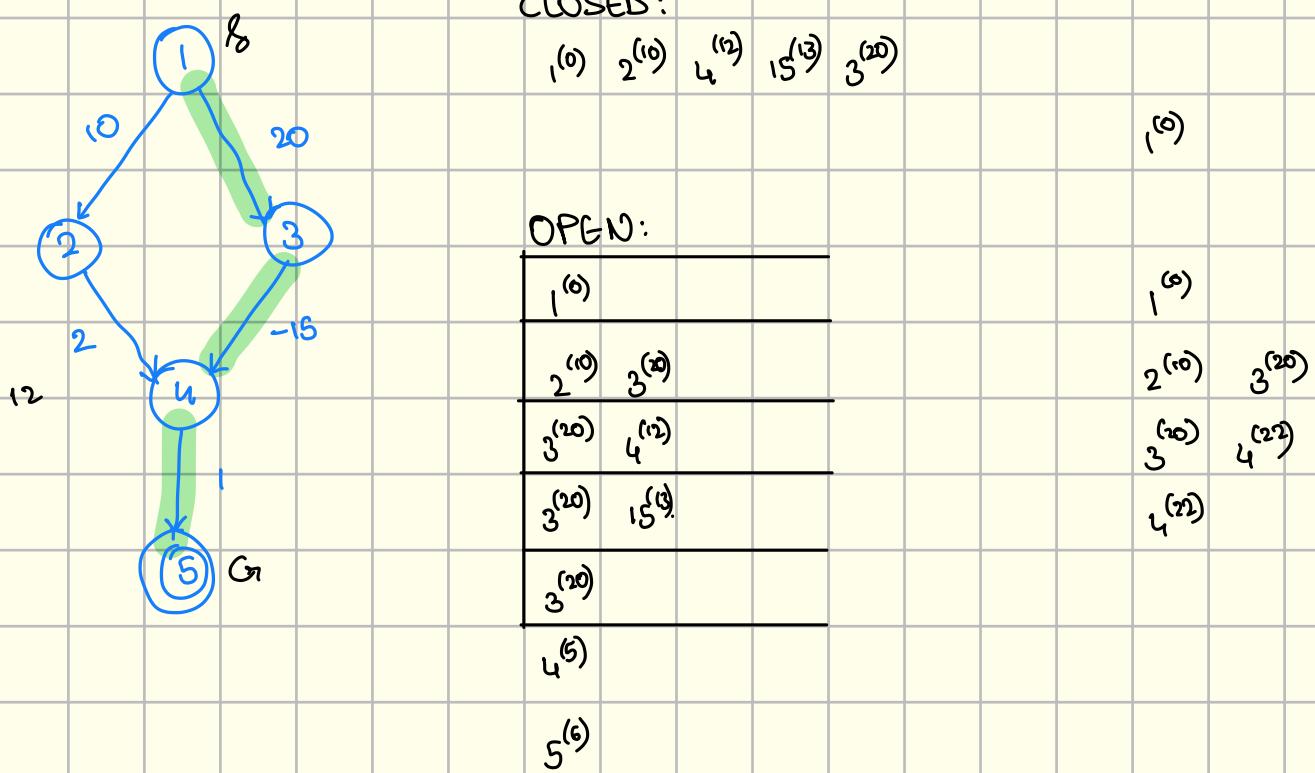


CLOSED :

$1^{(0)}$	$5^{(1)}$	$2^{(2)}$	$4^{(2)}$	$3^{(3)}$	$6^{(5)}$	$4^{(6)}$	$7^{(6)}$
$8^{(6)}$	$10^{(7)}$	$1^{(10)}$	$12^{(11)}$				

OPEN :

$1^{(0)}$
$5^{(1)}$ $2^{(2)}$
$2^{(2)}$ $9^{(2)}$
$4^{(2)}$ $3^{(3)}$ $6^{(5)}$
$3^{(3)}$ $6^{(5)}$ $10^{(7)}$
$6^{(5)}$ $10^{(7)}$ $4^{(6)}$
$10^{(7)}$ $4^{(6)}$ $7^{(6)}$ $10^{(7)}$
$10^{(7)}$ $7^{(6)}$ $8^{(6)}$
$8^{(6)}$ $11^{(6)}$
$10^{(7)}$ $11^{(10)}$ $7^{(6)}$ $12^{(21)}$
$11^{(10)}$ $12^{(21)}$ $11^{(10)}$
$12^{(21)}$ $12^{(11)}$



Theorem: The UCS algo applied on a problem having +ve operator cost will return optimal sol" (Prove this)

UCS

Terminate with Success : If $n \in G$ then terminate with success

(This termination condition is used ONLY when all operators costs are +ve)

if $m \in \text{CLOSED}$

Terminate with Failure : If $\text{OPEN} = []$, then terminate with failure

ONLY Termination condition for graph

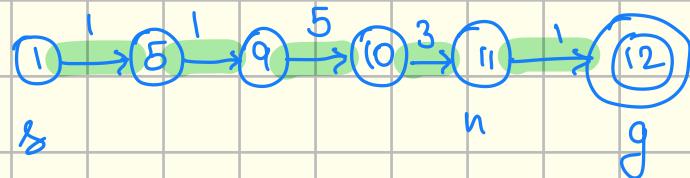
with -ve edge.

(This is being used when graph has -ve edge)

Theorem : (only for +ve operators cost)

(A) VCS applied on a prob. having +ve operator cost will return optimal solution

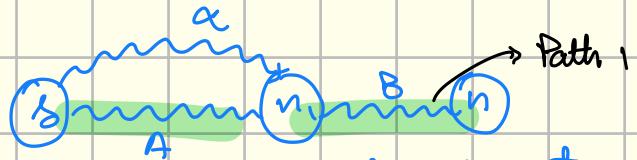
(B) PROPOSITION : At least one node in the path from s to the nes (having optimal cost) should be in OPEN before n is being removed from OPEN



(C) There does not exist any other path from s to n

having lesser cost than n . cost.

→ (i) At least one node belonging to the path having "OPTIMAL COST" from s to n should be in OPEN always before retrieving n .



Proof: optimal cost from

$$s \text{ to } n = C^*(n) = A+B$$

$$\text{In Path-1 : } A+B = C^*(n) \quad \text{--- (1)}$$

Assume: $\alpha < A$

$$\alpha + B < c^*(n) \quad (2)$$

Statement (2) contradicts statement (1)

(D) $g \in G$: closest optimal goal from s . (Corollary +ve Operator's cost)

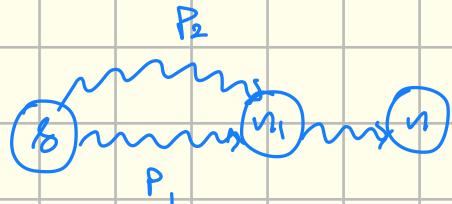
Optimal cost $c^*(n) \leq c^*(g)$; $+ n \in S$ will be expanded before g .

$$c^*(10) = 7$$

$$c^*(9) = 2$$

$$c^*(12) = 11$$

(2) If s to $n \rightarrow$ optimal



then s to $n_1 \rightarrow$ optimal

P is the optimal path from s to n through n_1 with cost $c^*(n)$

P_1 is the subpath of P from s to n_1

we need to proof P_1 is optimal subpath from s to n_1

Assume P_1 is not optimal path from s to n_1 with cost $c(n_1)$

let's $\exists P_2$ which is optimal path from s to n_1

$$\Rightarrow c^*(n_1) < c(n_1)$$

Path from s to n = optimal path from s to n_1 + path from n_1 to n

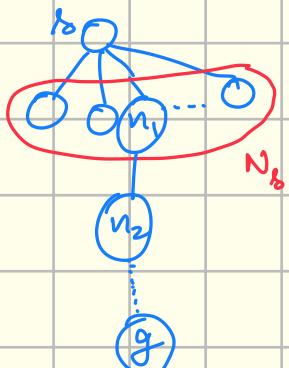
$$= c^*(n_1) + P(n_1, n) < c^*(n)$$

PROOF OF B:

Base Case : N_p : all the nodes reachable directly from s are in OPEN in the 1st step of Algo

→ Let, P be the optimal path from s to a node $n \in S$

consists of atleast one node in N_s



→ Let's assume, s to n consists of n_1 and n_2

n_1 is directly reachable from s , is in OPEN

→ Until n_1 is not being removed, (B) is true.

where n_1 is expanded, n_2 will be visited

(i) n_2 is not in $(OPEN \cup CLOSED)$ $\Rightarrow n_2$ is brought to OPEN

(ii) $n_2 \in [OPEN \cup CLOSED] \Rightarrow$ cost of n_2 will be revised through n_1 because of (C)

further (B) can be proved using induction

Proof (A): Can be proved by (B) of



Note: If the graph has unit weight /

same weight we BFS instead of UCS

UCS :

- If all operator costs are +ve, then the algo finds the min Cost sequence of transition to a goal.
- No state comes back to OPEN from CLOSE (+ve)
- If operator has unit / same cost then UCS is same as BFS
- for -ve operator cost, it may come back to OPEN from CLOSED

BRANCH AND BOUND : (+ve Edge Cost) (Basic Version)

- ① Iterative : Set $OPEN = [s]$, $CLOSED = []$ Set $c(s) = 0$
 $c^* = \infty$
- ② Terminate : If $OPEN = []$, then return c^*
- ③ Select : Select a state n from OPEN and save in CLOSED
- ④ New Bound Assignment : If $n \in G$ and $c(n) < c^*$, then
set $c^* = c(n)$ and goto step 2
- ⑤ Expand : If $c(n) < c^*$ generate the successor of n
for each successor m :

if $m \notin [\text{OPEN} \cup \text{CLOSED}]$:

set $c(m) + c(n, m)$ and insert m in OPEN
B

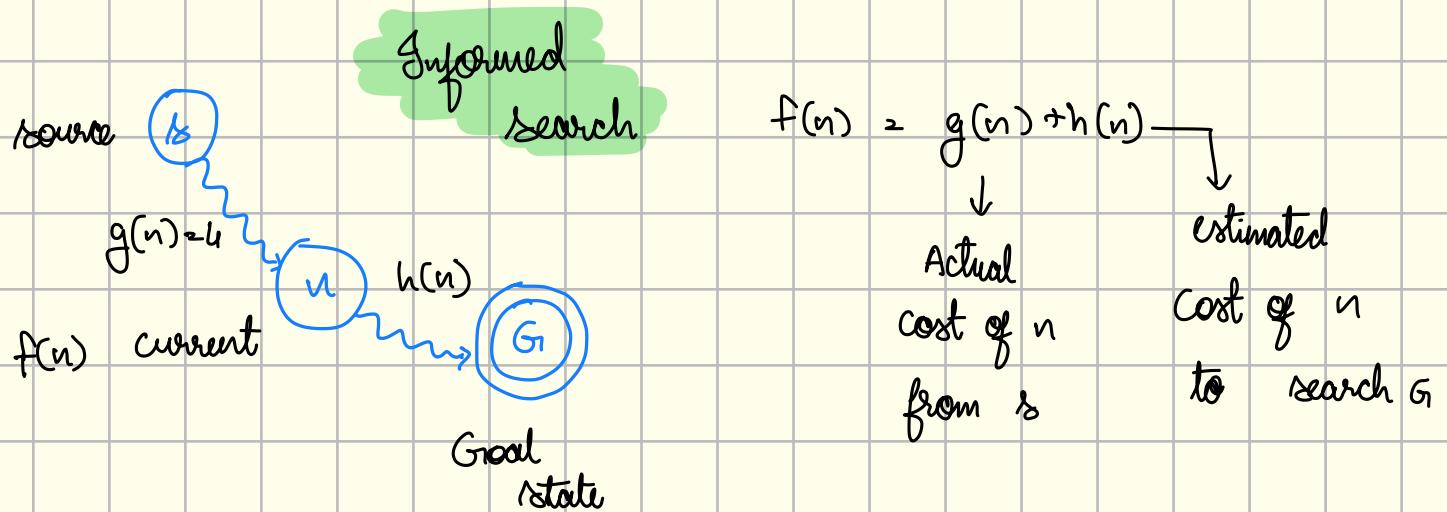
if $m \in [\text{OPEN} \cup \text{CLOSED}]$:

set $c(m) = \min(c(m), B)$

if $c(m)$ has decreased of $m \in \text{CLOSED}$:

move to OPEN

⑥ Go to step 2.



→ Underestimate

→ Overestimate

$h^*(n) = \text{Optimal Cost to reach } G_1 \text{ from } n$

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

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

start
 $g(s) = 0$

$h(G_1) = 0$
Goal

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



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

G

'n'

$$h(n) = 12$$

at least
12 tiles need
to be moved

$$h(n)$$

(# tiles not in
correct position)

for 8 Queen

Problem what
will be the
heuristic?

Read Russel Norvig

→ Heuristic

Q		
	Q	
Q		
		Q

G: all Q's are not in attacking position

heuristic = cost of MST
(underestimate)

'n'

$$2 \times \text{cost(MST)}$$

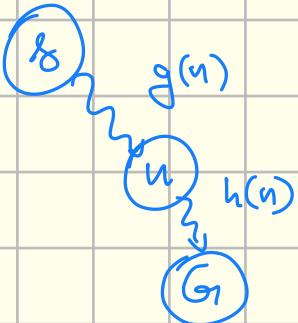
$$h(n) = 2$$

$h > 2$ (overestimate)

pair of Q's in attacking position : $h(n)$

$g(n) = \text{func}^n$ of state and operator

$h(n) = \text{func}^n$ of only state (will not change)



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

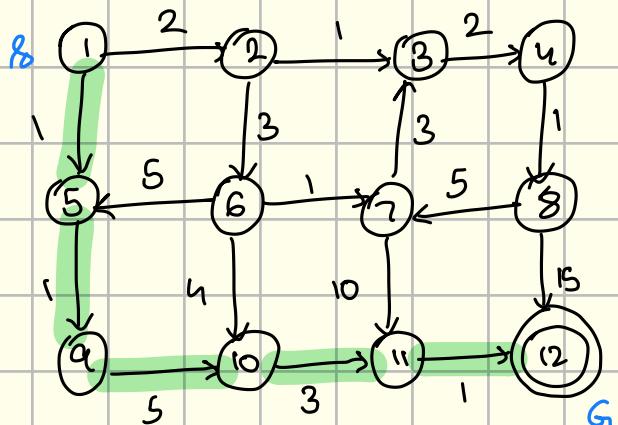
$f^*(G)$ } optimal value to reach G
 $g^*(G)$

INFORMEDNESS :

→ h_1 & h_2 are 2 UNDERESTIMATED HEURISTICS

→ h_1 is more informed than h_2 , if & n es, $h_1(n) \geq h_2(n)$

$$\rightarrow g(1) = 0 (\because g(\text{start}) = 0)$$



$$\begin{aligned} \rightarrow h(1) &= 11 & h(a) &= 9 \\ h(2) &= 11 & h(10) &= 4 \\ h(3) &= 18 & h(11) &= 1 \\ h(4) &= 16 & h(12) &= 0 \\ h(5) &= 10 \end{aligned}$$

$h(n)$: Optimal path from n to G

→ If & n $h(n) = h^*(n)$ then only the nodes in the optimal path

will be expanded

↳ Retrieved from OPEN

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

$$f(1) = g(1) + h(1) = 0 + 11 = 11$$

$$f(2) = g(2) + h(2) = 2 + 11 = 13$$

$$f(3) = g(3) + h(3) = 3 + 18 = 21$$

$$f(5) = g(5) + h(6) = 1 + 10 = 11$$

$$f(9) = g(9) + h(9) = 2 + 9 = 11$$

→ when the heuristic is **UNDERESTIMATE** ie $f(n) \leq h^*(n)$

Eg: $h_1(n_1) = 5$ $h_1(n_2) = 6$

$h_2(n_1) = 7$ $h_2(n_2) = 4$

→ given K UNDERESTIMATED heuristic derive the best heuristic

$$h_1, h_2, \dots, h_k \Rightarrow K \text{ underestimate heuristic}$$

the best heuristic will be $\max(h_1, h_2, \dots, h_k)$ as it would

be closest to the optimal heuristic value.

$$\therefore \text{BEST: } f(n) = \max(h_1, h_2, \dots, h_k)$$

→ If $h(n) = 0$ then $f(n) = g(n) \Rightarrow$ algo. to be applied is UCS

A* Algorithm :

→ Initialization : $\text{OPEN} = [s]$, $\text{CLOSED} = []$, $s.\text{cost} = f(s) = h(s)$

→ Iteration : Remove the lowest cost (based on $f(n)$) state n from OPEN

→ Selection : $(m \cdot cost = n \cdot cost + cost(n, m)) \times$

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

$$g(m) = g(n) + cost(n, m)$$

$$(m \cdot cost > n \cdot cost + cost(n, m)) \times$$

$$\cancel{\{ g(m) > g(n) + cost(n, m) \}}$$

→ VCS : UnInformed

A* : Informed



If $m \notin [OPEN \cup CLOSED]$

$$g(m) = g(n) + cost(n, m)$$

If $m \in [OPEN \cup CLOSED]$

$$g(m) = \min(g(n), g(n) + cost(n, m))$$

If $g(m)$ is ↓ if $m \in CLOSED$

put m in OPEN

→ $h(n) = 0 \Rightarrow$ VCS \Rightarrow Many Nodes will be Expanded

$h(n) = h^*(n) \Rightarrow$ min^m nodes will be expanded

→ A* uses heuristic search with underestimated heuristic

It always gives optimal solution

Q) which nodes are being expanded in VCS but not in A*?

In VCS :

$$\forall n, g(n) \leq g^*(G_i)$$

$$c(n) \leq c^*(n)$$

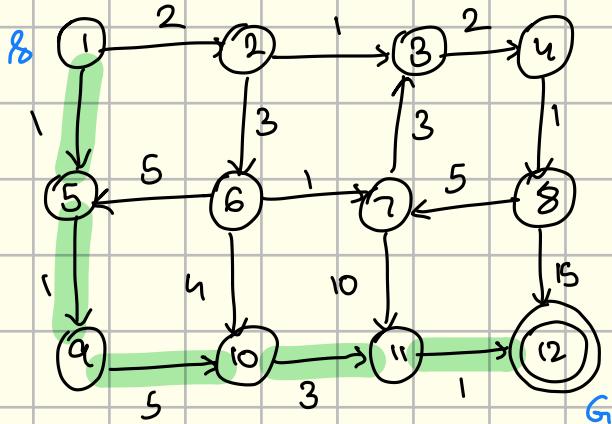
In A* : $\forall n, f(n) \leq \left(g^*(G_i) - f^*(G_i) \right)$

$$g(n) + h(n)$$

$$g(n) < g^*(G_i) < f(n)$$

$$g(n) + h(n)$$

these n 's will be expanded in VCS not in A^*



→ for node 2

$$g(n) = g(2) = 2$$

$$f(2) = 13$$

$$g^*(n_2) = 11$$

$$g(n) < g^*(G_i) < f(n)$$

$$2 < 11 < 13$$

→ which nodes are being expanded in VCS but not in A^*

$$\text{In } g(n) < \left[\begin{array}{l} g^*(G_i) \\ = f^*(G_i) \end{array} \right] < f(n)$$

→ In A^* ; take priority Q , based on $f(n)$

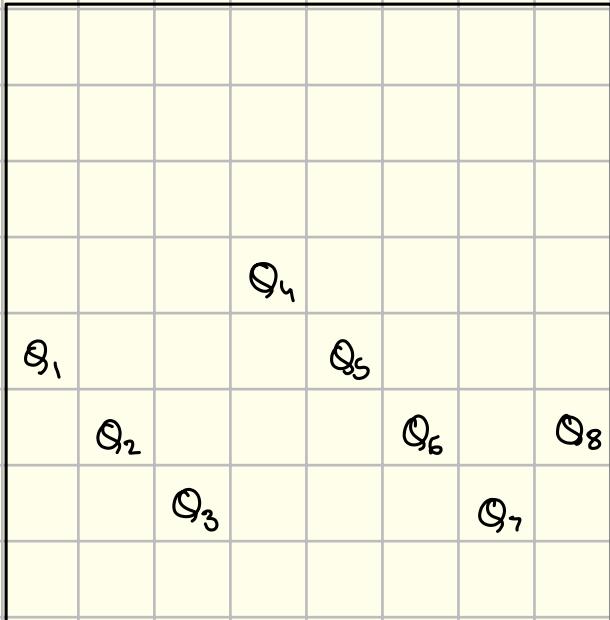
In VCS; $\sim \sim \sim \sim \sim \sim g(n)$

LOCAL SEARCH:

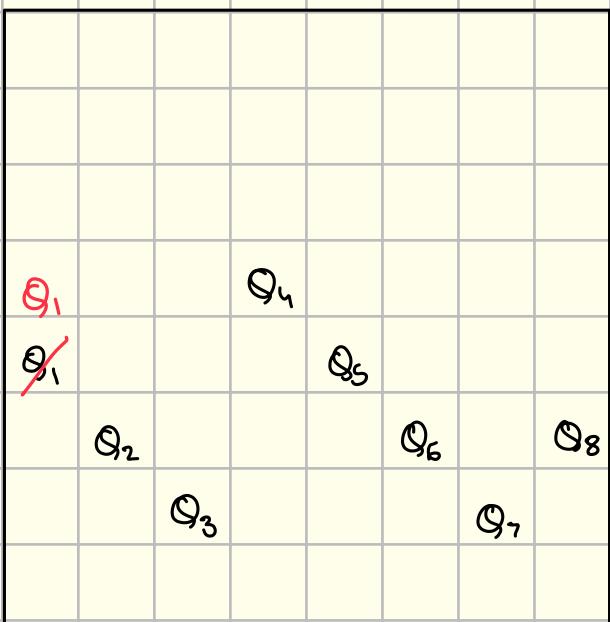
HILL CLIMBING:

→ for 8 Queen Problem:

$h(n)$: # pair of Q's in attacking position



$Q_1 \rightarrow (Q_2, Q_3, Q_5)$	= 3
$Q_2 \rightarrow (Q_3, Q_4, Q_6, Q_8)$	= 2
$Q_3 \rightarrow (Q_5, Q_7)$	= 2
$Q_4 \rightarrow (Q_5, Q_6, Q_7)$	= 3
$Q_5 \rightarrow (Q_6, Q_7)$	= 2
$Q_6 \rightarrow (Q_7, Q_8)$	= 2
$Q_7 \rightarrow (Q_8)$	= 1
$Q_8 \rightarrow ()$	= 0
<hr/>	
$h(n) = 17$	



$Q_1 \rightarrow (Q_4)$	= 1
$Q_2 \rightarrow (Q_3, Q_4, Q_6, Q_8)$	= 2
$Q_3 \rightarrow (Q_5, Q_7)$	= 2
$Q_4 \rightarrow (Q_5, Q_6, Q_7)$	= 3
$Q_5 \rightarrow (Q_6, Q_7)$	= 2
$Q_6 \rightarrow (Q_7, Q_8)$	= 2
$Q_7 \rightarrow (Q_8)$	= 1
$Q_8 \rightarrow ()$	= 0
<hr/>	
$h(n) = 15$	

→ HC is a local search technique

→ 1st Choice Hill Climbing:

from the successor, choose 1st better i.e in

→ Slopest Ascent Hill Climbing:

look all 56 neighbours, from these neighbours choose the

best i.e 12

→ Stochastic Hill Climbing:

look the better neighbour than (17),

from them Probabilistically choose better uphill move

(if minimization)

→ Random Restart Hill Climbing: (RRHC)

Multiple time, Run H.C (hill climbing) with random initial

state each time

→ limitation of HC: may get stuck in local maximum / minimum

→ In H.C we are not seeing local solution.

→ H.C is also called as a greedy local search algorithm as

it always grabs a good neighbour without thinking about the next step.

→ Geometric distribution : $P_x(x) = (1-p)^{x-1} p$

$$\text{Mean} = \frac{1}{p} \quad \Rightarrow \quad p = \frac{1}{\text{Mean}}$$

SIMULATED ANNEALING:

→ If we get a bad solution: choose that with some probability

→ $\Delta E = -\alpha$: -ve

As, $T \rightarrow 0$ prob: $e^{\frac{\Delta E}{T}} = e^{\frac{-\alpha}{T}} = \frac{1}{e^{\frac{\alpha}{T}}} \approx 0$

Hill Climbing
Boils down to

⇒ Probability = 0 ⇒ Not taking bad solution

As, $T \rightarrow \infty$ prob: $e^{\frac{\Delta E}{T}} = e^{\frac{-\alpha}{T}} = \frac{1}{e^{-\frac{\alpha}{T}}} \approx 1$

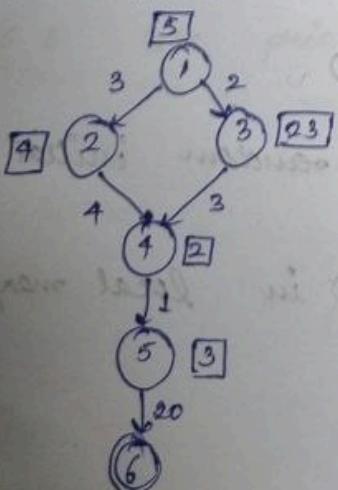
whatever solution we get we consider it

If $\Delta E < 0$. → Always take worst state
If $\Delta E > 0$, → better solution

(RANDOM WALK)

Few points on A*

1. A heuristic is called admissible if it always underestimates, i.e. we always have $h(n) \leq f^*(n)$, where $f^*(n)$ denotes the minimum distance to a goal state from state n .
 - * For finite state spaces, A* always terminates.
 - * At any time before A* terminates,
 2. There exists in OPEN, a state n such that is an optimal path from s to a goal state with $f(n) \leq f^*(s)$.
- * If there is a path from s to a goal state, A* terminates (even when the state space is infinite).
- ③ Algo A* is admissible, i.e. if there is a path from s to a goal state, A* terminates by finding an optimal path.
- * If A_1 and A_2 are 2 versions of A*, such that A_2 is more informed than A_1 , then A_1 expands at least as many states as A_2 .
- * If we give 2 or more admissible heuristic we can take their MAX to a stronger admissible heuristic.



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

$h(n)$ always underestimate heuristic
before A* steps

A* need but consider for
if the the
sp val
path con
e ①

OPEN	
$f(n)$	
2 ⁷ , 3 ²⁵	
3 ²⁵ , 4 ⁹	
3 ²⁵ , 5 ¹¹	
3 ²⁵ , 6 ²⁸	
6 ²⁰ , 4 ⁷	
6 ²⁸ , 5 ⁹	
6 ²⁶	

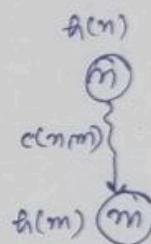
CLOSED	
15, 2 ⁷ , 4 ⁹ , 5 ¹¹	
3 ²⁵ , 4 ⁷ , 5 ⁹ , 6 ²⁶	

▷ A* heuristic . if monotone and admissible , then we don't need to transfer any NODE from CLOSED to OPEN , but if non-monotone then we may need to transfer.

consistent / Monotone

for every node n . $h(n) \leq c(n,m) + h(m)$

$$h(n) - c(m,n) \leq h(m)$$



* if the monotone restriction is satisfied .

then A* has already found an optimal path to the state it selects for expansion .

* if the monotone restriction is not satisfied , the f-values of states expanded by A* , is non-decreasing .

Path Max

converts a non-monotone heuristic to monotone heuristic

① During generation of successor m of n

set $h'(m) = \max [h(m), h(n) - c(m,n)]$
and use $h'(m)$ at m

admissible underestimate \rightarrow optimal
 $\overbrace{\text{overestimate}}^{\text{can still reach}}$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	10010	10011	10012	10013	10014	10015	10016	10017	10018	10019	10020	10021	10022	10023	10024	10025	10026	10027	10028	10029	10030	10031	10032	10033	10034	10035	10036	10037	10038	10039	10040	10041	10042	10043	10044	10045	10046	10047	10048	10049	10050	10051	10052	10053	10054	10055	10056	10057	10058	10059	10060	10061	10062	10063	10064	10065	10066	10067	10068	10069	10070	10071	10072	10073	10074	10075	10076	10077	10078	10079	10080	10081	10082	10083	10084	10085	10086	10087	10088	10089	10090	10091	10092	10093	10094	10095	10096	10097	10098	10099	100100	100101	100102	100103	100104	100105	100106	100107	100108	100109	100110	100111	100112	100113	100114	100115	100116	100117	100118	100119	100120	100121	100122	100123	100124	100125	100126	100127	100128	100129	100130	100131	100132	100133	100134	100135	100136	100137	100138	100139	100140	100141	100142	100143	100144	100145	100146	100147	100148	100149	100150	100151	100152	100153	100154	100155	100156	100157	100158	100159	100160	100161	100162	100163	100164	100165	100166	100167	100168	100169	100170	100171	100172	100173	100174	100175	100176	100177	100178	100179	100180	100181	100182	100183	100184	100185	100186	100187	100188	100189	100190	100191	100192	100193	100194	100195	100196	100197	100198	100199	100200	100201	100202	100203	100204	100205	100206	100207	100208	100209	100210	100211	100212	100213	100214	100215	100216	100217	100218	100219	100220	100221	100222	100223	100224	100225	100226	100227	100228	100229	100230	100231	100232	100233	100234	100235	100236	100237	100238	100239	100240	100241	100242	100243	100244	100245	100246	100247	100248	100249	100250	100251	100252	100253	100254	100255	100256	100257	100258	100259	100260	100261	100262	100263	100264	100265	100266	100267	100268	100269	100270	100271	100272	100273	100274	100275	100276	100277	100278	100279	100280	100281	100282	100283	100284	100285	100286	100287	100288	100289	100290	100291	100292	100293	100294	100295	100296	100297	100298	100299	100300	100301	100302	100303	100304	100305	100306	100307	100308	100309	100310	100311	100312	100313	100314	100315	100316	100317	100318	100319	100320	10