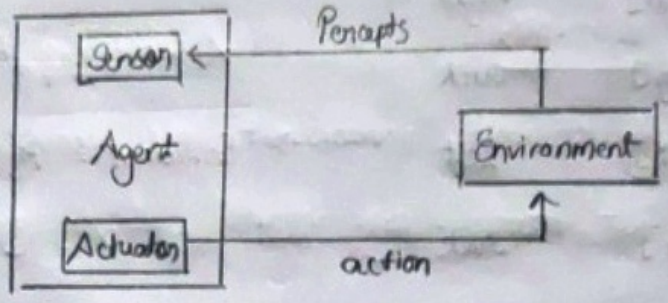
# LECTURE 2: Intelligent Agent

**Agent** → An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment using sensors.




**Rationality:**

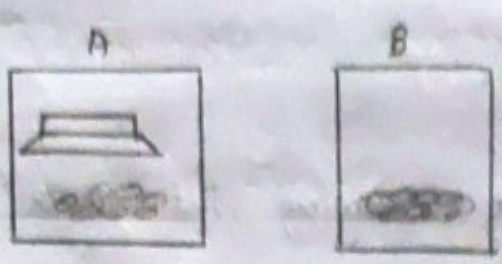
Rational at any given time depends on 4 things

1) Performance measure that defines the criterion of success.
2) Agent's prior knowledge of the environment
3) Actions the agent can perform
4) Agent percept sequence at the time

**Rational agent** → For each possible percept sequence, a rational agent should select an action that is expected to maximize performance measure given the evidence provided by the percept sequence, built in knowledge of the agent.

**Properties of rational agent:**

→ **Omniscience**: Expected vs Actual performance
→ **Learning capability**: Apriori knowledge
→ **Autonomous**: agent behavior is determined by its experience




| Percept sequence | Action |
|---|---|
| [A, clean] | Right |
| [A, dirty] | suck |
| [B, clean] | Left |
| [B, dirty] | suck |
| [A, clean] [A, clean] | Right |
| [A, clean] [A, dirty] | suck |

Action = Left, Right, Suck

→ Environment = A, B

## TASK ENVIRONMENT

### PEAS

P - Performance measure

E - Environment

A - Actuator

S - Sensor

Example: Automated taxi driver

| Agent type | Performance measure | Environment | Actuators | Sensors |
|---|---|---|---|---|
| Automated taxi driver | Safe, fast, legal, comfortable, maximize profit | Road, customer, Pedestrians, other traffic. | Steering, accelerator, brake, clutch, horn, signal. | Camera, sonar, speedometer, Engine sensor, accelerometer. |

Performance measure:

In an automated taxi driver our performance measure is to reach correct destination, minimize fuel consumption, wear & tear, minimize [cost, time, violations, disturbance], maximize [Profit, comfort, safety]. Some of these goal required trade-offs.

Environment:

A taxi driver deals with a variety of roads, ranging from rural, urban, 12-lane etc.

The road contain other traffic, pedestrians, animals, puddles and potholes. The taxi must interact with potential vs actual passenger. There is a probability of snow in some regions.

Actuators:

Actuator for automated taxi need to control the vehicle using [accelerator, brake, steering]. To communicate with the passenger using speaker or display. To communicate with other driver using horns & indicators.

Sensors:

Sensors include multiple camera to detect the environment, sonar to detect distance, GPS to navigate and speedometer to stay within legal speed range.

## PROPERTIES OF TASK ENVIRONMENT

### 1) Fully observable Vs Partially observable

If an agent sensor is capable to access the complete state of the environment at each point in time it is said to be fully observable.

Fully observable environment are convenient because the agent need not maintain any internal state/history.

If an agent sensor is noisy or inaccurate or part of the state is missing is said to be partially observable

An environment is unobservable if it has no sensors.

Eg: Chess - Fully observable - all board is visible and move

Driving - Partially observable - what is around the corner is unknown.

### 2) Single Vs Multi agent

An environment consist of only one agent is said to be single agent environment

Eg: Agent solving crossword puzzle.

An environment involving more than one agent is multi agent environment

Eg Agent playing chess (two agent environment)

3) Deterministic Vs Stochastic

If the next state of the environment is completely determined by the current state and action of the agent then it is deterministic

Eg: Chess - the board is fully observed and only a few possible moves for a chess piece at current state can be determined.

The stochastic environment is random in nature which cannot be determined by the agent

Eg: Self-driving cars - the action of the self driving are not unique and varies time : time.

4) Episodic Vs Sequential

In episodic task enviroment each of the agent actions is atomic. There is no dependancy b/w current and previous incidents.

In each incidents the agent receives the input from the environment and performs action.

Eg: Pick & Place robot in conveyor belts.

In a sequential environment previous decision can affect all future decision. The next action of the agent depends on what action he has taken previously and what action he is supposed to take in future

Eg: Chess

4) Static Vs dynamic

Dynamic : An environment that keeps constantly changing itself when the agent is up with some action

Eg: Self-driving cars

Static : An idle environment with no change in state

Eg: Cross-word puzzle

Semi-dynamic : If the environment does not change but the performance of the agent does

Eg: Chess with clock

5) Discrete Vs Continuous

Continuous : The environment in which the actions are performed cannot be numbered

Eg: Self driving cars

Discrete : If an environment has finite number of actions to obtain the output

Eg: Chess

Example

| Task environment | Observable | Agent | Deterministic | Dependency | Static | number of states |
|---|---|---|---|---|---|---|
| Crossword puzzle | Fully | Single | Deterministic | Sequential | Static | Discrete |
| chess | Fully | Multi | Deterministic | Sequential | Semi | Discrete |
| Self driving car | Partial | Multi | Stochastic | sequential | Dynamic | Continuous |
| Part picking robot | Partial | Single | Stochastic | Episodic | Dynamic | Continuous |

## LECTURE 3: Agent architecture

Agent = Program + architecture

The job of AI is to design an agent program that implements the agent function -- the mapping from percepts to action. The program runs on a computer device with sensor and actuators called architecture.

Eg: If the program recommends to walk, the architecture must have legs.

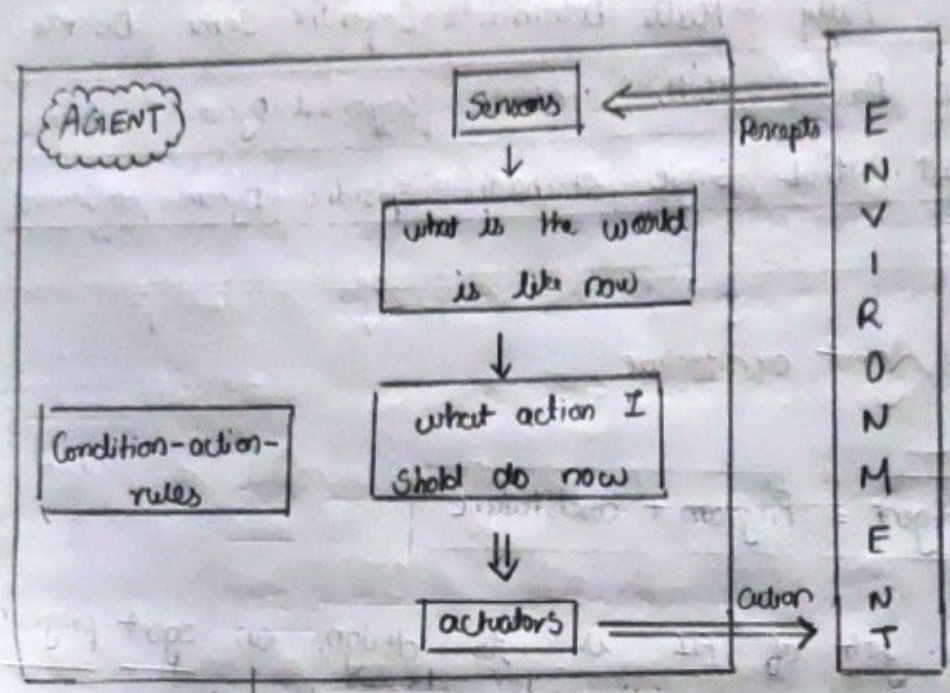
### ① Simple reflex agents

Simple reflex agent are the simplest agent. These agents take decisions on the basis of current percepts & ignore the rest of percept history.

These agent succeed only in fully observable environments.

Disadvantage:

* very limited intelligence
* not adaptive to changes in the environment




Function Simple-reflex agent (percept) returns an action

persistent : rules, a set of condition-action rules

state ← Interpret Input (Percept)

rule ← Rule match (state, rules)

action ← rule.action

returns action

Eg: function vacuum agent([location, status]) returns an action

if status = dirty then return Suck

else if location = A then return Right

else if location = B then return Left

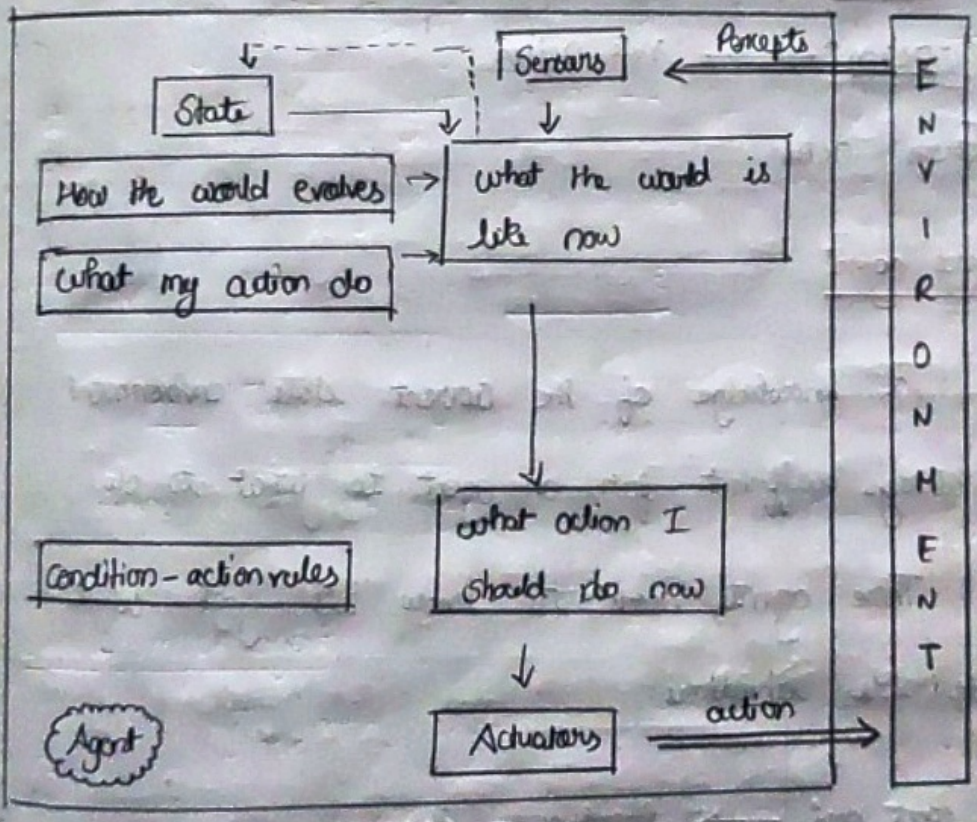
## ② Model based reflex agent

The model based reflex agent can work in a partially observable environment.

Important factors:

a) Model: It is the knowledge about "how things happen in the world" so it is called model based agent.

b) Internal state: It is the representation of current state based on percept history.



Function Model-based reflex agent (percepts) returns an action

persistent : state ; the agent current conception of the world

transition model ; how the next state depends on current state & action

Sensor model ; how the current world state is reflected in agent percept

rules ; a set of condition-action rules

action ; the most recent action (initially none)

state ← Update State (state, action, percept, model)

rule ← Rule match (State, rule)

action ← rule.action

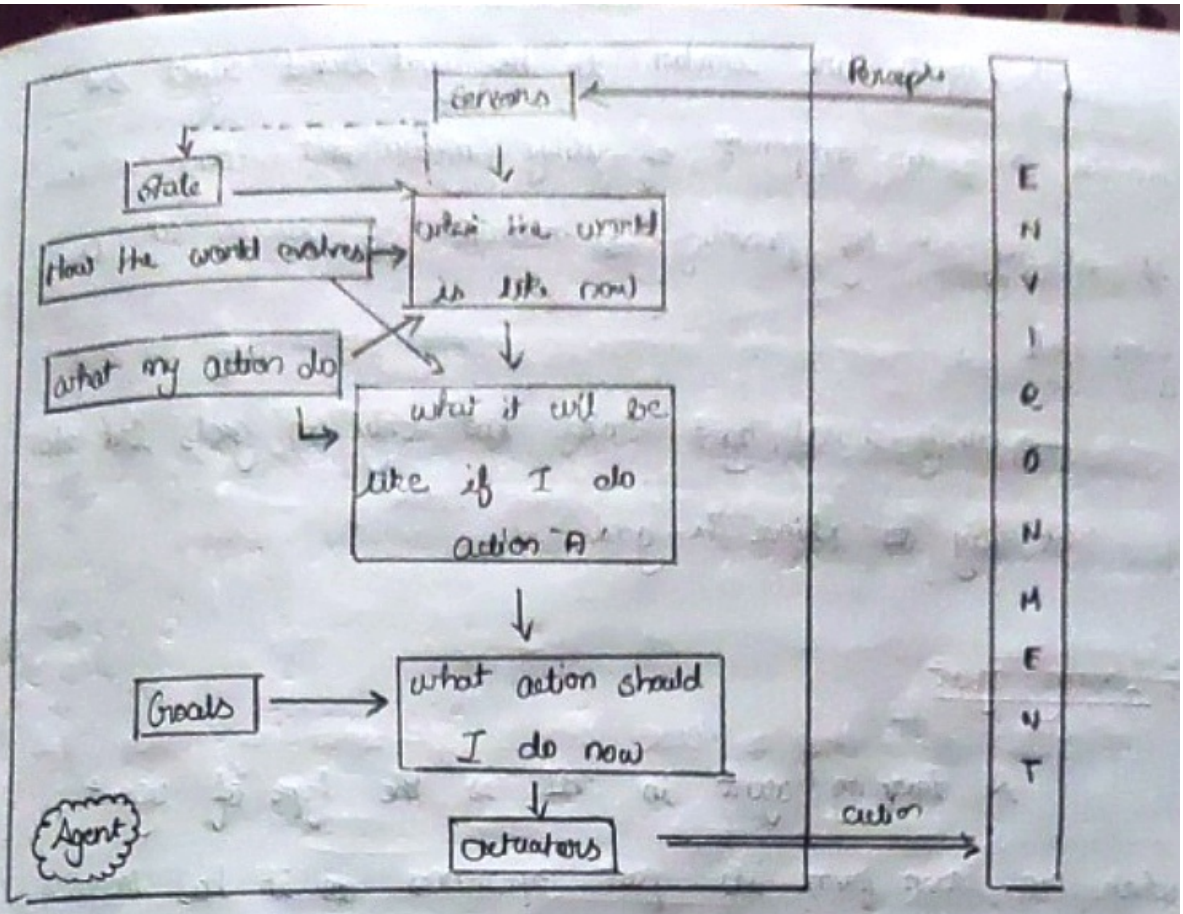
return action

## (3) Goal based agent

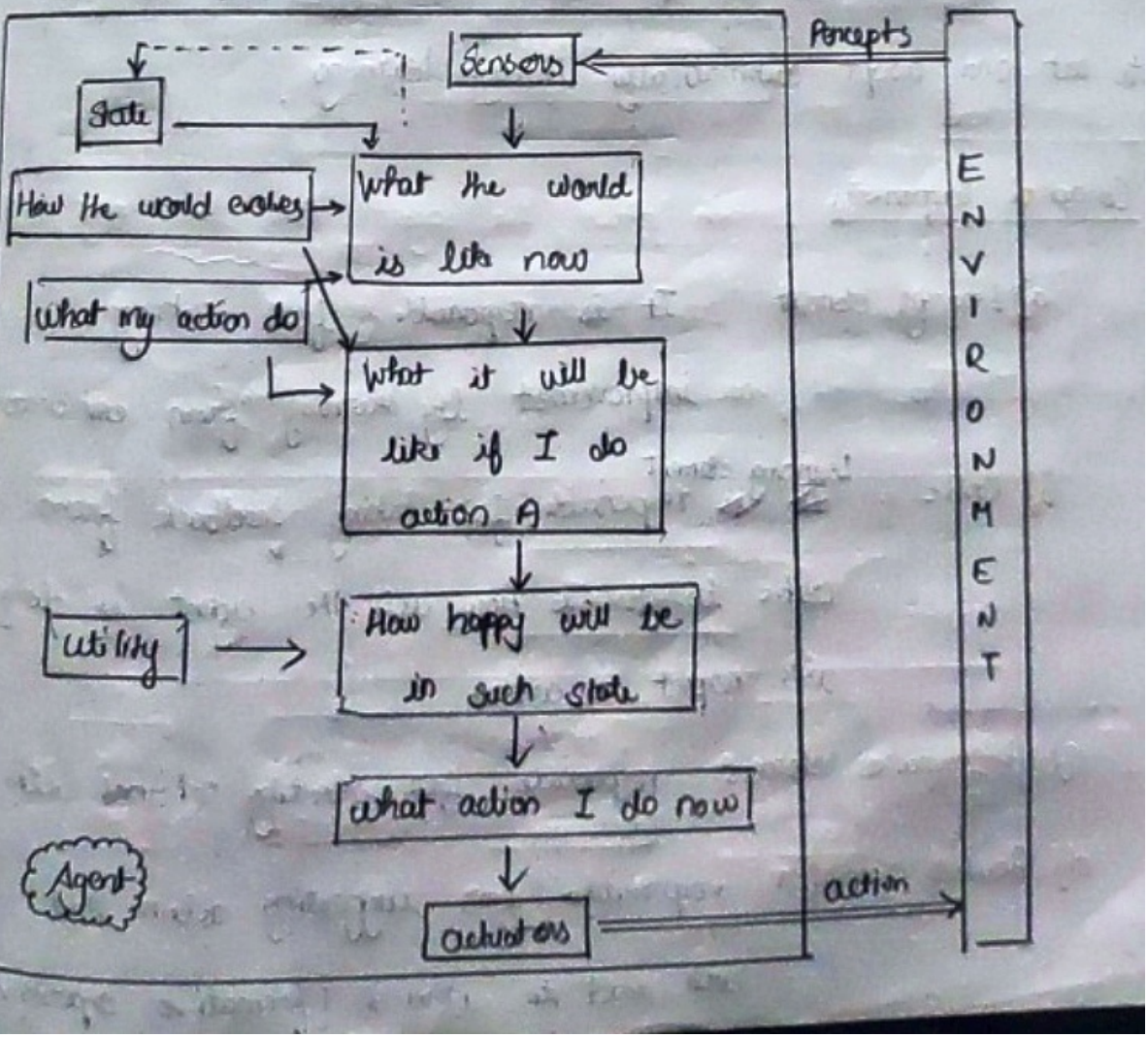
The knowledge of the current state environment is not always sufficient for an agent to what to do

These agents needs to know its goal which describes desirable situations

Agent goal → Searching + Planning




④ Utility based agent



The agents are similar to the goal based agent but provides an extra component of utility measurement which means it is different by providing a measure of success at a given state.

→ utility based agent based not only on goals but also the best way to achieve the goal.

③ Learning agent

A learning agent in AI is the type of agent which can learn from its past experiences or it has learning capabilities.

It starts to act on basic knowledge and then able to act and adapt automatically through learning.

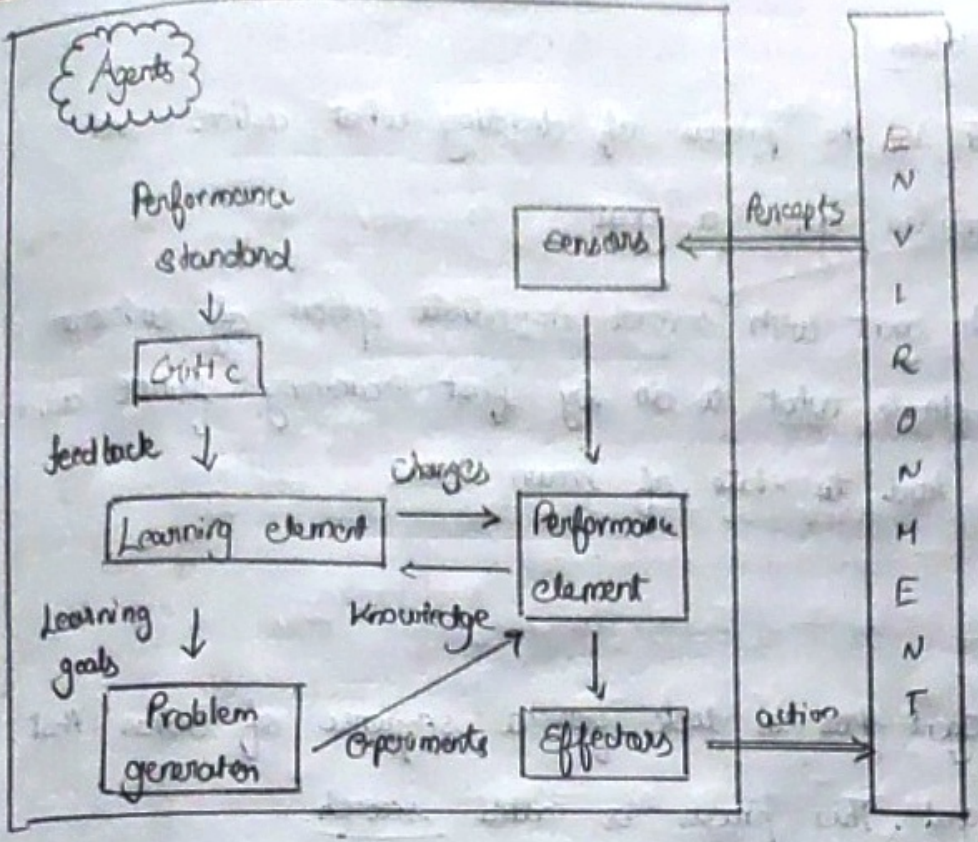
Conceptual components:

a) Learning element : It is responsible for making improvement by learning from environment

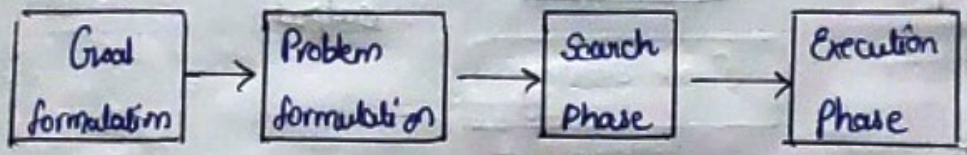
b) Critic : Learning element
It is responsible for taking feedback from critic to describe how well the agent is doing with respect to action.

c) Performance element : responsible for selecting external action

d) Problem generator : responsible for suggesting actions that will lead to new & informative experience




## PROBLEM SOLVING AGENT




### Goal formulation

Goal formulation based on the current situation and the agent performance measure is the first step in problem solving.

Goal = {set of world states}

The agent task is to find out how to act now, and in future so that it reaches its final goal.

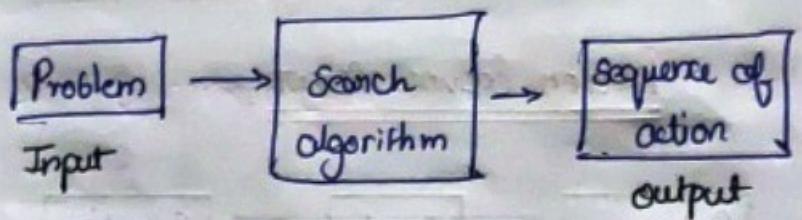
## Problem formulation:

It is the process of deciding what actions and states to consider given a goal

An agent with several immediate option of unknown value can decide what to do by first examining future actions that eventually lead to state of known value.

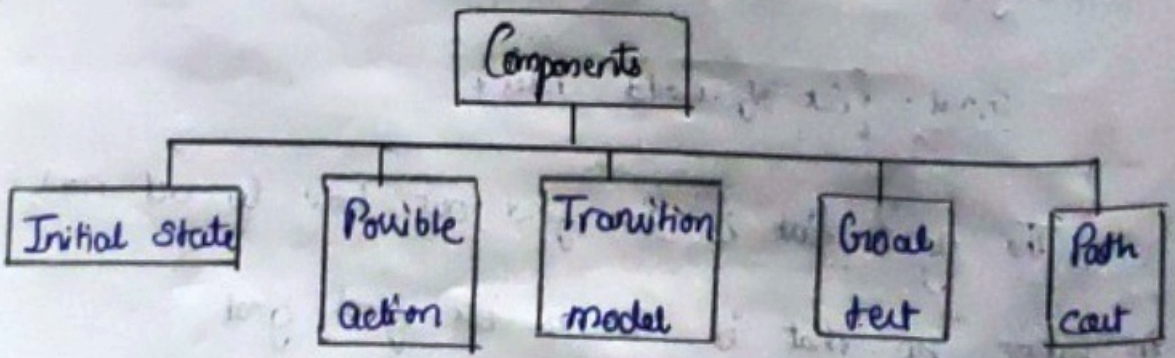
## Search phase:

The agent has to look for a sequence of action that reaches the goal. This process is called search.




## Execution phase:

After the search phase, the agent has to carry out the action that are recommended by the search algorithm. This final phase is called execution phase



## Initial state

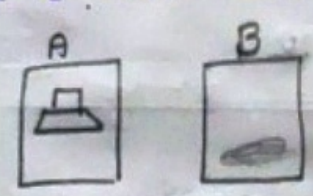
Initial state that the agent starts in. If the agent goes from A to B. Here 'A' is the Initial state

## Action

All the possible action available to the agent.

## Transition model

It returns the state the results from doing action 'a' in the state 's'. It's like execution phase.




In this case there are 3 action [Left, Right, Suck] But in our case left and suck does not have effect in this 's'.

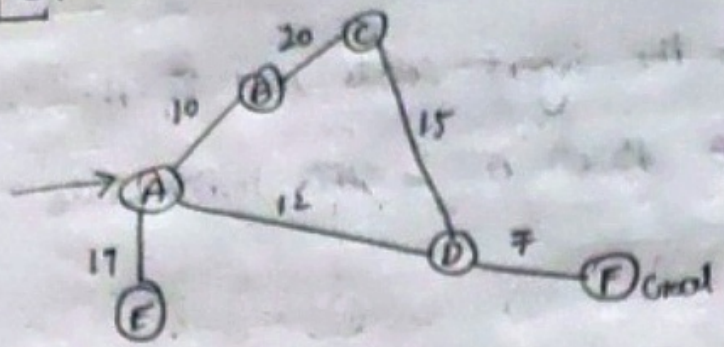
## Goal test

The goal test determines when a given state is a goal state or not

## Path cost

The last component of the problem is the path cost which is a function that assigns a numeric cost to each path.

Example :




Initial State : In (Arad / A)

Possible action : Actions (s) → {Go (B), Go (E), Go (B)}

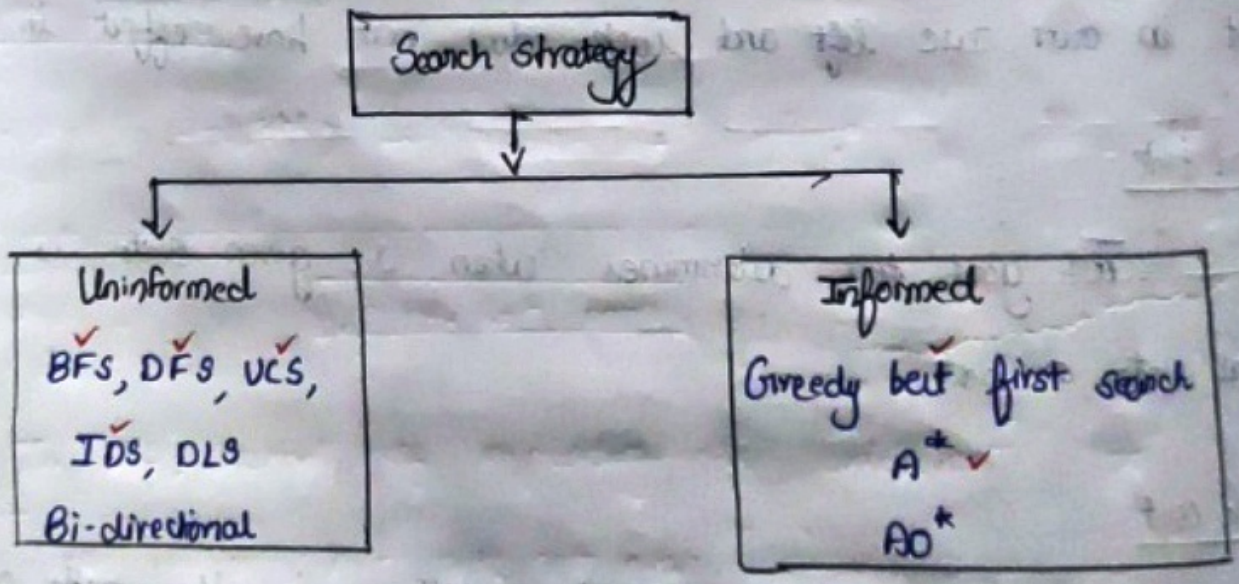
Transition model : Results (In (A), Go (D)) = In (D)

Goal state : Is goal (In (D)) = No

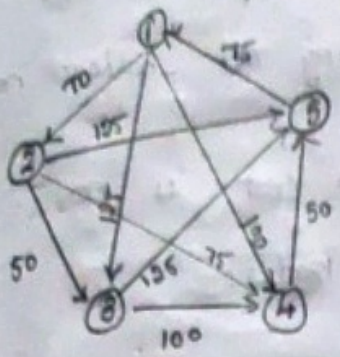
Path cost : cost (In (A), go (D)) = 12

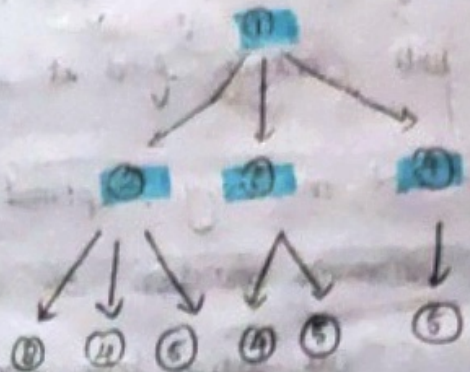
---

# LECTURE - 4 : UNIFORMED SEARCH



# Breadth first Search (BFS)



Start node : ①, Goal node : ⑤

→ Expanded

→ checked

Step 1 : ① → Test failed

Step 2 : (12) (13) (14) → Test failed

Step 3 : (13) (14) (123) (124) (125) → Test failed

Step 4 : (14) (123) (124) (125) (134) (135) → Test failed

Step 5 : (123) (124) (125) (134) (135) (145) → Test passed

Fail Fail Pass

c (1-2-5) = 70 + 125 = 195 //

Expanded = 4 [See tree]

Generated = 10 [number of nodes]

Max queue = 6 [Max length of the queue (step 6)]

## Evaluation

**Complete:** If the shallowest goal node is at depth 'd', BFS will eventually find it by generating all shallower nodes

**Optimal:** Not necessarily. Optimal if path cost is non decreasing function of depth of nodes.

Eg: all action have same cost

**Time Complexity:**

$$O(b^d)$$

b → breadth (branching factor), d - depth

nodes expanding at depth 1 = b, depth 2 = $b^2$ .....

**Space Complexity:**

$O(b^{d-1})$ → Explored set, $O(b^d)$ is frontier set

---

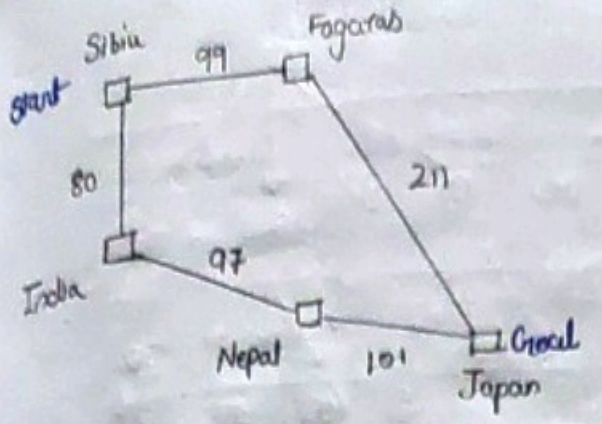
## Uniform Cost Search

→ It uses, instead of expanding the shallowest node, the node n with least path cost g(n) is expanded

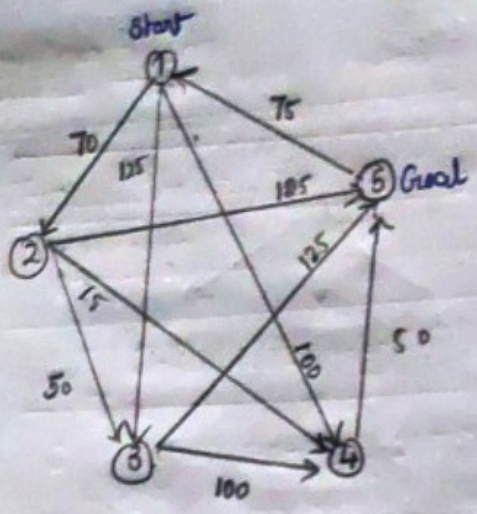
→ Sorting the frontier as priority queue by g(n)

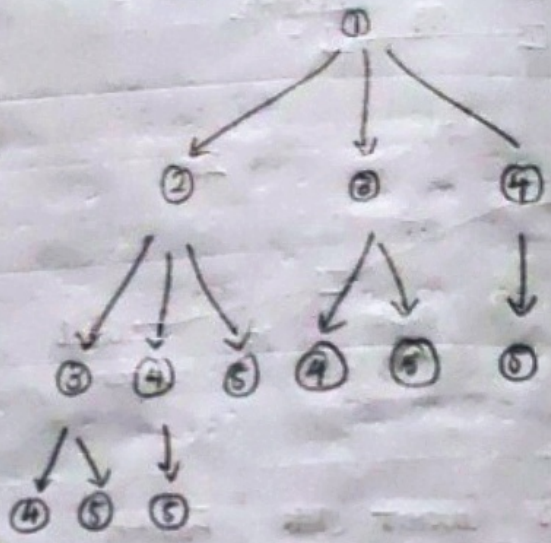
→ Goal test is applied during expansion

* If the goal node is found on generation may not be optimal




| Steps | Current state | Frontier | Expand step | Result |
|---|---|---|---|---|
| ① | Sibiu | [ ] | Expand Sibiu | Generates {(India, 80), (Fagaras, 99)} Add to frontier |
| ② | Sibiu | (India, 80) (Fagaras, 99) | Expand India (Least cost) | Generates {(Nepal, 177)} Add to frontier |
| ③ | India [Not a goal state] | (Fagaras, 99) (Nepal, 177) | Expand Fagaras (least cost) | Generates {(Japan, 310)} It is goal state but we don't test in generation |
| ④ | Fagaras (Not a goal state) | (Nepal 177) (Japan, 310) | Expand 'Nepal' Least cost | Generate {(Japan, 278)} # Don't test |
| ⑤ | Nepal (Not a goal state) | (Japan 278) | Expand 'Japan' | No further generation as goal test satisfied return solution |



(1)

Fail

(1-2) (1-4) (1-5)
70 100 185

Fail

(1-4 : 100) (1-2-3 : 120) (1-3 : 185) (1-2-4 : 145) (1-2-5 : 195)

Fail

(1-2-3 : 120) (1-3 : 185) (1-2-4 : 145) (1-4-5 : 150) (1-2-5 : 195)

Fail

(1-3 : 185) (1-2-4 : 145) (1-4-5 : 150) (1-2-3-4 : 170) (1-2-5 : 195)
(1-2-3-5 : 245)

Fail

(1-2-4 : 145) (1-4-5 : 150) (1-2-3-4 : 170) (1-2-5 : 195)
(1-3-4 : 225) (1-2-3-5 : 245) (1-3-5 : 250)

Fail

(1-4-5 : 150) (1-2-3-4 : 170) (1-2-4-5 : 195) (1-2-5 : 195)
(1-3-4 : 225) (1-2-3-5 : 245) (1-3-5 : 250)

Test - Pass

## Evaluation:

**Completeness:** It is complete if the cost of every step > Small +ve constant $\epsilon$

→ It will stuck in infinite loop if there is a path with infinite sequence of zero cost action

**Optimal:** It is optimal. Whenever it selects a node it is optimal path to the node.

**Time & Space complexity:** UCS guide by path not 'b' or 'd'

If $C^*$ is the cost of optimal solution and $\epsilon$ is the min action cost

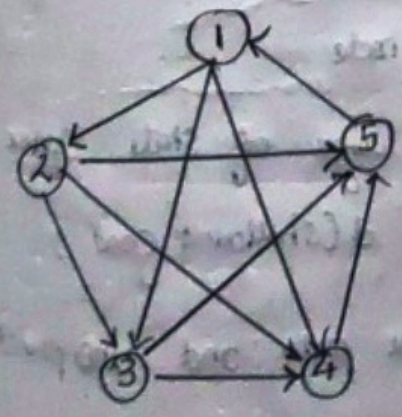
Worst case complexity = $O\left(b^{1+C^*/\epsilon}\right)$

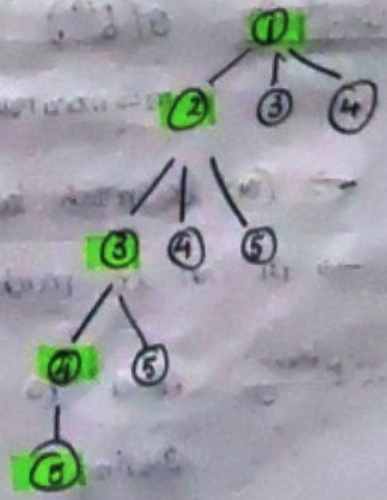
Action cost are equal [BFS performs better]

[Because goal test is done in expension in UCS, But in generation in BFS]

---

## DEPTH FIRST SEARCH



Algorithm tracing

| Iter | Open list / Frontier / Fringes | Closed list | Goal test |
|---|---|---|---|
| 1 | (1) | | Fail on (1) |
| 2 | (1-2)(1-3)(1-4) | (1) | Fail on (1-2) |
| 3 | (1-2-3)(1-2-4)(1-2-5)<br>(1-3)(1-4) | (1-2) | Fail on (1-2-3) |
| 4 | (1-2-3-4)(1-2-3-5)(1-2-4)<br>(1-2-5)(1-3)(1-4) | (1-2-3) | Fail on (1-2-3-4) |
| 5 | (1-2-3-4-5)(1-2-3-5)(1-2-4)<br>(1-2-5)(1-3)(1-4) | | Pass on (1-2-3-4-5) |

Evaluation:

Completeness: Complete in finite state spaces because it will eventually expand every node.

Optimal: Not optimal as it should stop when the goal node is reached without evaluating if there is a better path.

Time complexity: $O(6^m)$

m → maximum depth of node

→ Can be much larger than the size of state space

→ m can be much larger than d (shallowest goal)

Space complexity: Need to store only one path and unexpanded siblings.

→ Any node expanded with all its children can be removed

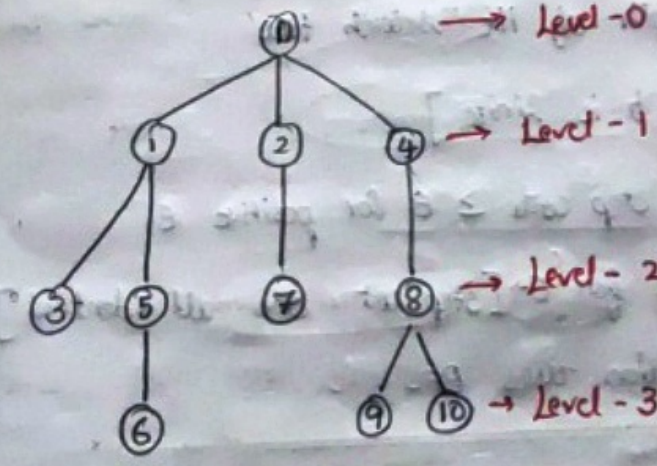
→ Require storge of $O(bm)$

## Iterative Deepening Depth First Search (IDS)

Runs DFS by gradually increasing the limit (L)

- First L=1, then L=2 and so on ... until goal is found.

**IDS = BFS + DFS**




| Depth | IDS |
|---|---|
| 0 | 0 |
| 1 | 0, 1, 2, 4 |
| 2 | 0, 1, 3, 5, 2, 7, 4, 8 |
| 3 | 0, 1, 3, 5, 6, 2, 7, 4, 8, 9, 10 |

## Comparing uninformed strategies:

| Criterion | BFS | UCS | DFS | IDS |
|---|---|---|---|---|
| Complete ? | Yes[a] | Yes[a,b] | No | Yes[a] |
| Time | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(b^m)$ | $O(b^d)$ |
| Space | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(bm)$ | $O(bd)$ |
| Optimal ? | Yes[c] | Yes | No | Yes[c] |

b → branching factor

d → depth of the shallowest solution

m → maximum depth of the search tree

Superscript:
- a → Complete [if b is finite]
- b → Complete if step costs $\geq \epsilon$ for positive $\epsilon$
- $C^*$ → Optimal value for {C → [step cost are all identical]}
- d → If both direction uses BFS

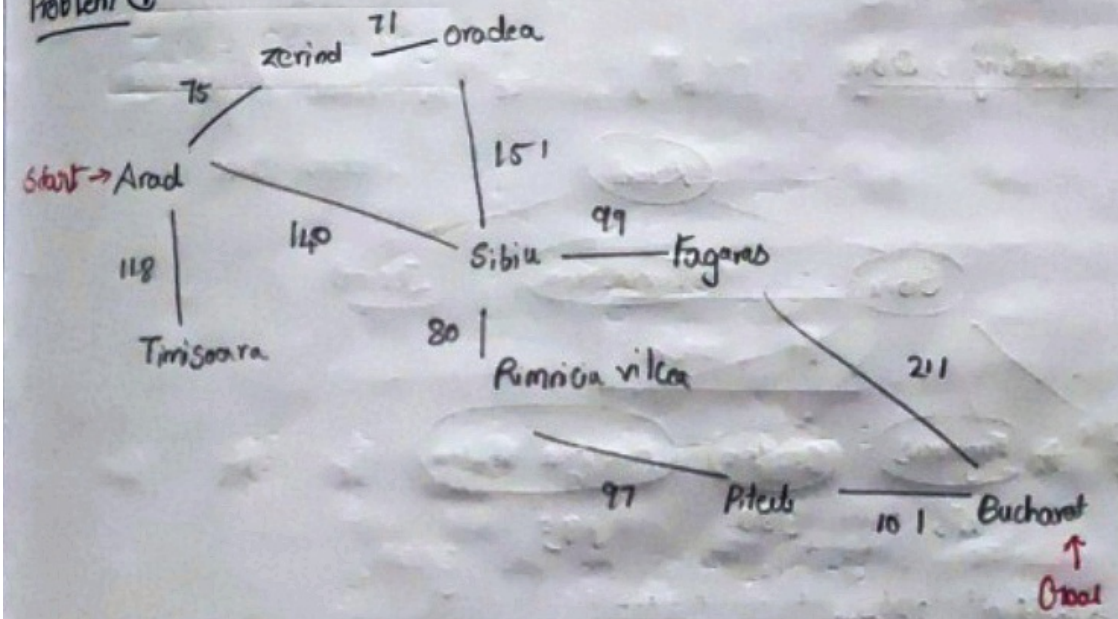
# Lecture 5 : Informed Search

## Greedy best first search

This approach expands the node that is closet to the goal by using heuristic function

$f(n) = h(n)$ $h(n)$ - heuristic function

Problem ①

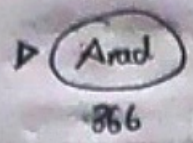



Heuristic values

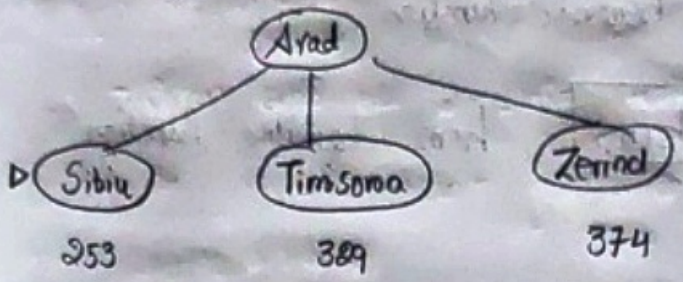
| City | h(n) |
|---|---|
| Arad | 366 |
| Zerind | 374 |
| Timisoara | 329 |
| Oradea | 380 |
| Sibiu | 253 |

| | |
|---|---|
| Fagaras | 176 |
| Rimnicu | 193 |
| Pitesti | 100 |
| Bucharest | 0 |

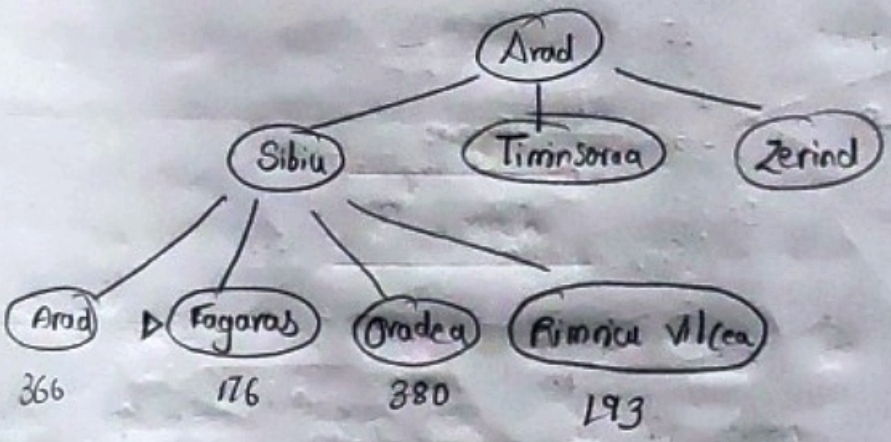
1) Initial state



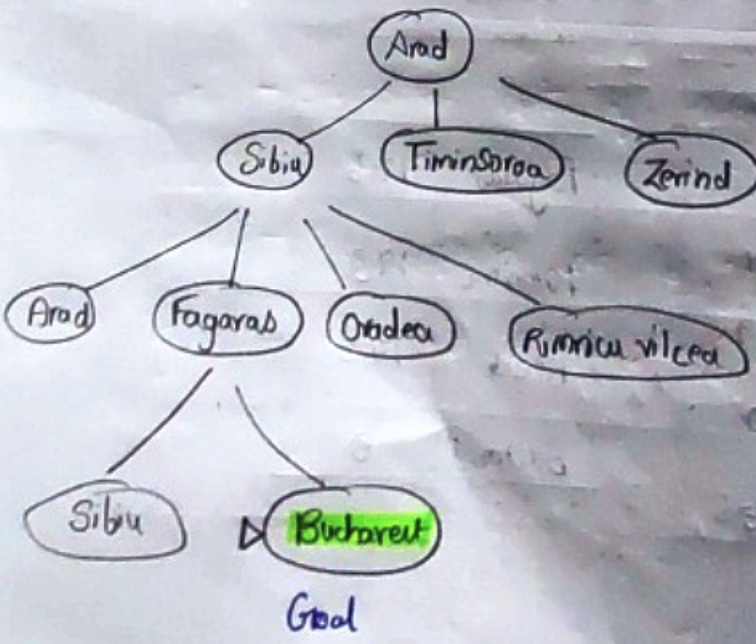

2) After expanding arad



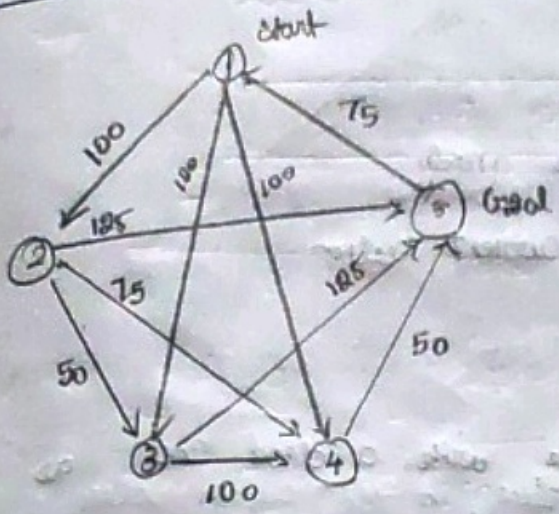

3) After expanding Sibiu

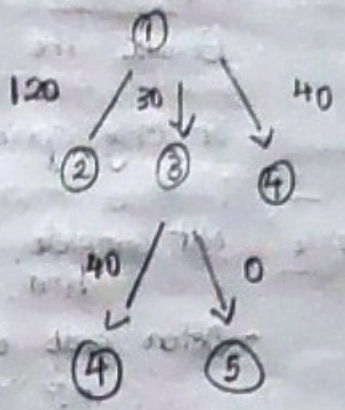



4) After expanding Fagaras



Problem ②




| n | hcm |
|---|---|
| 1 | 60 |
| 2 | 120 |
| 3 | 30 |
| 4 | 40 |
| 5 | 0 |

Initial : (1)

↓

(1,3) (1,4) (1,2) # closed list (1)

30 40 120

↓

(1,3,5) (1,3,4) [redacted] # closed list (1,3)

0 40

C(1-3-5) = 100+195 = 295

Expanded → 2

Generated → 6

Max queue length - 3

Evaluation:

Optimal : Not optimal

- Because the algorithm is greedy
- It only optimizes the current action

Complete : Not complete

- Often ends up in state with a dead end as the heuristic doesn't guarantee a path is only an approximately.

Time & space complexity : $O(6^m)$ where m - max depth of search tree

---

## A* Search

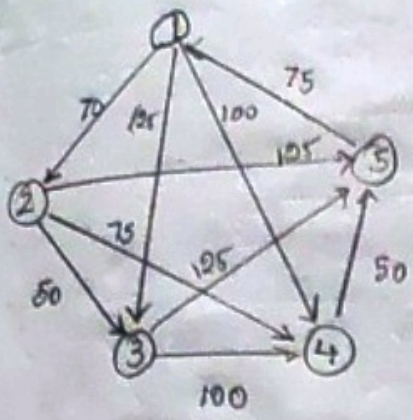
Expands the node which lies in the least path (estimated cheapest path) to the goal.

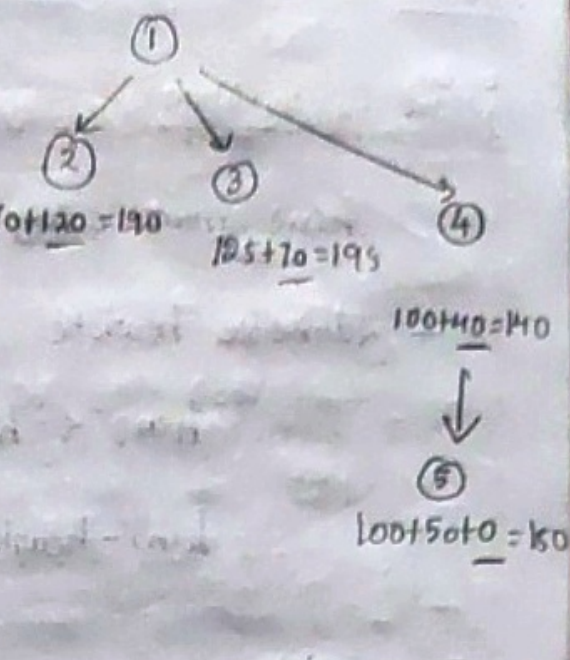
Evaluation function $f(n) = g(n) + h(n)$

$f(n)$ - estimated cost of cheapest path through node n

$h(n)$ - the expected cost to go from node to goal

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




| n | h(n) |
|---|---|
| 1 | 60 |
| 2 | 120 |
| 3 | 70 |
| 4 | 40 |
| 5 | 0 |

Initial : (1)

↓

(1-4) (1-2) (1-3)

140 190 195

↓

(1-4-5)

150

Cost (1-4-5) = 100+50 = 150

Expanded = 2

Generated = 5

Maximum queue length = 3

Evaluation :

Optimal on condition

h(n) must satisfy two conditions

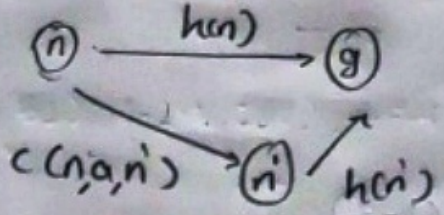
- Admissible heuristic

$h(n) <= h^*(n)$

h(n) - heuristic value, $h^*(n)$ - actual value

- Consistency

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




Complete :

- If the number of nodes with cost $<= C^*$ is finite
- If the branching factor is finite
- $A^*$ expands no node with $f(n) > C^*$ know as pruning

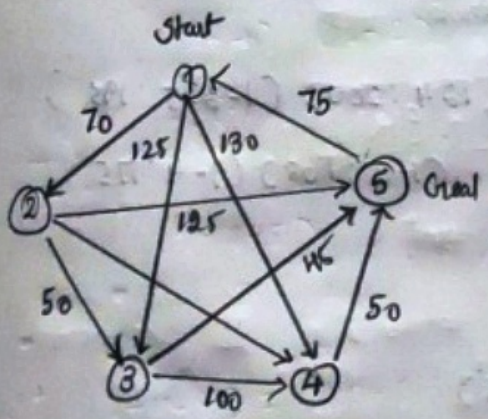
Time complexity :

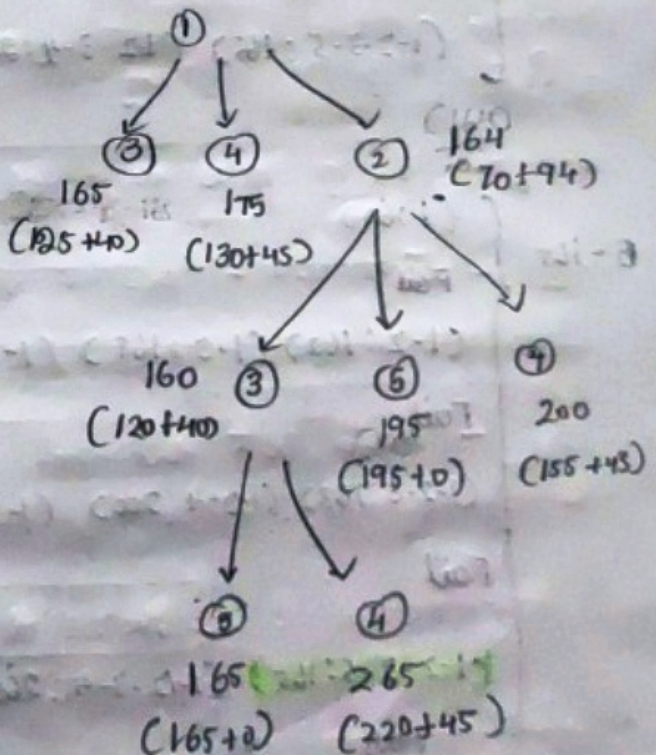
$O(b^\Delta)$ where the absolute error $\Delta = h^* - h$

#3 is goal

| n | h(n) | Is admissible? h(n) <= h*(n) | Is consistent? For every arc h(i) <= g(i,j)+h(j) |
|---|---|---|---|
| 1 | 80 | 80 <= (70+60) Y | (5→1) : 190 <= (75+80) N |
| 2 | 60 | 60 <= 50 N | (1→2) : 80 <= (70+60) Y |
| 3 | 0 | 0 <= 0 Y | |
| 4 | 200 | 200 <= (50+50+70+75) Y | (1→4) : 80 <= (100+200) Y<br>(2→4) : 60 <= (75+200) Y |
| 5 | 190 | 190 <= (75+70+50) Y | (2→5) : 60 <= (125+140) Y<br>(4→5) : 200 <= (50+190) Y |

## Iterative Deepening A*




| n | h(n) |
|---|---|
| 1 | 60 |
| 2 | 94 |
| 3 | 40 |
| 4 | 45 |
| 5 | 0 |

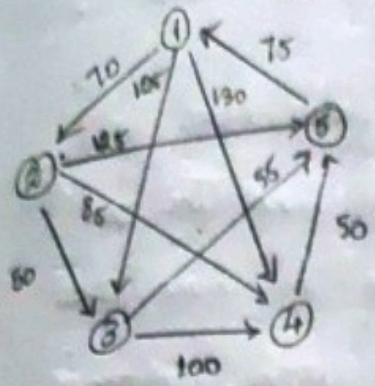
B=60
- (1:60)
- Fail
- (1-2:162) (1-3:168) (1-4:175)

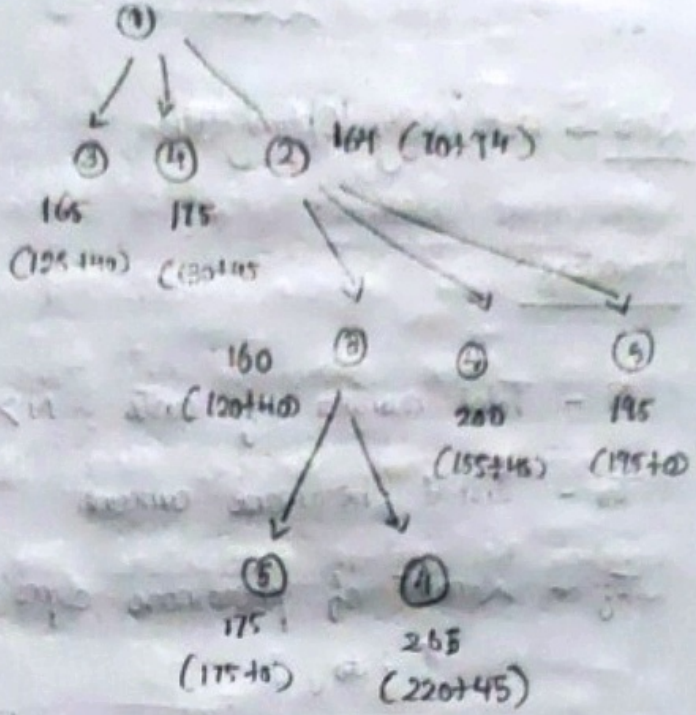
B=162
- (1:60)
- Fail
- (1-2:162) (1-3:168) (1-4:175)
- Fail
- (1-2-3:163) (1-2-4:200) (1-2-5:195) (1-3:168) (1-4:175)

B=163
- (1:60)
- Fail
- (1-2:162) (1-3:168) (1-4:175)
- Fail
- (1-2-3:163) (1-2-4:200) (1-2-5:195) (1-3:168) (1-4:175)
- Fail
- (1-2-3-5:165) (1-2-3-4:265) (1-2-4:200) (1-2-5:195) (1-3:168) (1-4:175)

B=165
- (1:60)
- Fail
- (1-2:162) (1-3:168) (1-4:175)
- Fail
- (1-2-3:163) (1-2-4:200) (1-2-5:195) (1-3:168) (1-4:175)
- Fail
- (1-2-3-5:165) (1-2-3-4:265) (1-2-4:200) (1-2-5:195) (1-3:168) (1-4:175)
- Pass

# Recursive Best first Search A*




| n | h(n) |
|---|---|
| 1 | 60 |
| 2 | 94 |
| 3 | 40 |
| 4 | 45 |
| 5 | 0 |

(1, 60)

(1-2, 164) (1-3, 165) (1-4, 175)

(1-2-3, 160) (1-3, 165) (1-4, 175) (1-2-5, 195) (1-2-4, 200)

(1-2-3-5, 175) (1-3, 165) (1-4, 175) (1-2-5, 195) (1-2-4-200)

(1-2-3-4, 265)

(1-3, 165) (1-2-175) (1-4, 175)

(1-3-5, 180) (1,2, 175) (1-4, 175) (1-3-4, 270)

(1-2, 175) (1-4, 175) (1-3, 180)

(1-2-3, 160) (1-4, 175) (1-3, 180) (12-5, 195) (1-2-4, 200)

(1-2-3-5, 175) (1-4, 175) (1-3, 180) (1-2-5, 195) (1-2-4, 200)

(1-2-3-4, 265)

Pass

# Lecture 7 : ACO

ACO → Ant Colony Optimization

Parameters :

N - Total number of ants ; $N > 1$

$\tau_0$ - Initial pheromone amount

$\tau_{ij}$ - Amount of pheromone deposited while traversing [ i to j ]

$\eta_{ij}$ - Cost of link $(i,j)$ → $\eta_{ij} = \frac{1}{f_k}$

$\alpha$ - Importance co-efficient of pheromone intensity

$\beta$ - Importance co-efficient of route cost

$\rho$ - Evaporation co-efficient ; $0 < \rho < 1$

$visit_k$ - Visited nodes table of $k^{th}$ ant

Q - Importance - Constant value pertaining to pheromone trail

$f_k$ - Route cost obtained by ant k

## Process of ACO

Initialization → Construction → Pheromone updation → Stopping criteria

Initialization:

→ Place pre-defined number of ants on starting point.

→ Set values for parameters $\alpha, \beta, \rho$.

→ Set $\tau_0$ to 0.

Construction:

Compute the next node transition probability

$$NTP_{ij} = \frac{(\tau_{ij})^{\alpha} (\eta_{ij})^{\beta}}{\sum (\tau_{ih})^{\alpha} (\eta_{ih})^{\beta}}$$

Pheromone updation:

$$\tau_{ij}^{new} = (\rho)\, \tau_{ij}^{old} + \Delta T_{ij}^{k}$$

$$\Delta T_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if } k^{th} \text{ ant paves } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

Stopping criteria

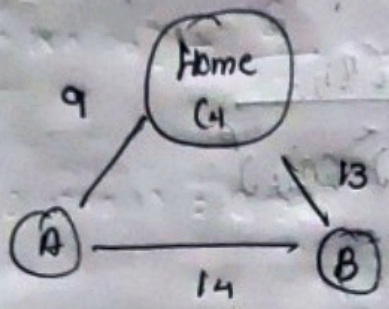
* Reaching predetermined number of iteration

(or)

* Reaching the goal.

## Problem:

Q1) Abhay is on a tour. He starts from home and visits two cities and has to reach back home. The cost of each route between cities and between his home and cities is given below in the form of graph. Determine the shortest path through ACO. Use the following information regarding the various parameters for ACO.




Pheromone matrix is given below

| | H | A | B |
|---|---|---|---|
| H | 0 | 0.15 | 0.26 |
| A | 0.15 | 0 | 0.48 |
| B | 0.26 | 0.48 | 0 |

Rate of evaporation = 0.1, Q=90, The relative importance of pheromone is 0.3, the relative importance of distance is 0.4.

Solution

Initialization

$\alpha = 0.3$

$\beta = 0.4$

$Q = 90$

$\rho = 0.1$

Starting point → Home

Next transition probability (1)

$$\eta_{HA} = \frac{1}{9} = 0.1111$$

$$\eta_{HB} = \frac{1}{13} = 0.0769$$

$$P_{HA} = \frac{(\tau_{HA})^{\alpha} (\eta_{HA})^{\beta}}{(\tau_{HA})^{\alpha} (\eta_{HA})^{\beta} + (\tau_{HB})^{\alpha} (\eta_{HB})^{\beta}}$$

$$= \frac{(0.15)^{0.3} (0.1111)^{0.4}}{(0.15)^{0.3} (0.1111)^{0.4} + (0.26)^{0.3} (0.0769)^{0.4}}$$

$$= \frac{0.2350}{0.4742}$$

$$\boxed{P_{HA} = 0.49}$$

$$P_{HB} = \frac{(\tau_{HB})^{\alpha} (\eta_{HB})^{\beta}}{0.4742} = \frac{(0.26)^{0.3} (0.0769)^{0.4}}{0.4742}$$

$$= 0.2392 / 0.4742 \Rightarrow \boxed{P_{HB} = 0.50}$$

Since $P_{HB}$ is more, Abhay move from Home to city B

Pheromone updation ($t=1$)

$$\tau_{ij}^{new} = \rho(\tau_{ij}^{old}) + \Delta\tau_{ij}^{K}$$

$\tau_{HA} = (0.1 \times 0.15) = 0.015$ $\left(\because \Delta\tau_{ij}^{K} = 0\right)$ for all except $\tau_{HB}$

$\tau_{HB} = (0.1 \times 0.26) + \frac{Q}{f_K} = 0.026 + \frac{90}{13} = 6.949$

$\tau_{AB} = (0.48 \times 0.1) = 0.048$

Since city B is visited the remaining city is A alone. It will be visited

(H) → (B) → (A)

Pheromone updation ($t=2$)

$\tau_{HA} = (0.1 \times 0.015) = 0.0015$

$\tau_{HB} = (0.1 \times 6.949) = 0.6949$

$\tau_{BA} = (0.1 \times 0.048) + \frac{90}{14} = 6.4333$

Since all cities are visited abhay will reach is home

(A) → (B) → (A) → (A)

Pheromone updation (t = 3)

$\tau_{HA} = (0.1 \times 0.0015) + \frac{90}{9} = 10.00015$

$\tau_{HB} = (0.1 \times 0.6949) = 0.06949$

$\tau_{AB} = (0.1 \times 6.4333) = 0.64333$

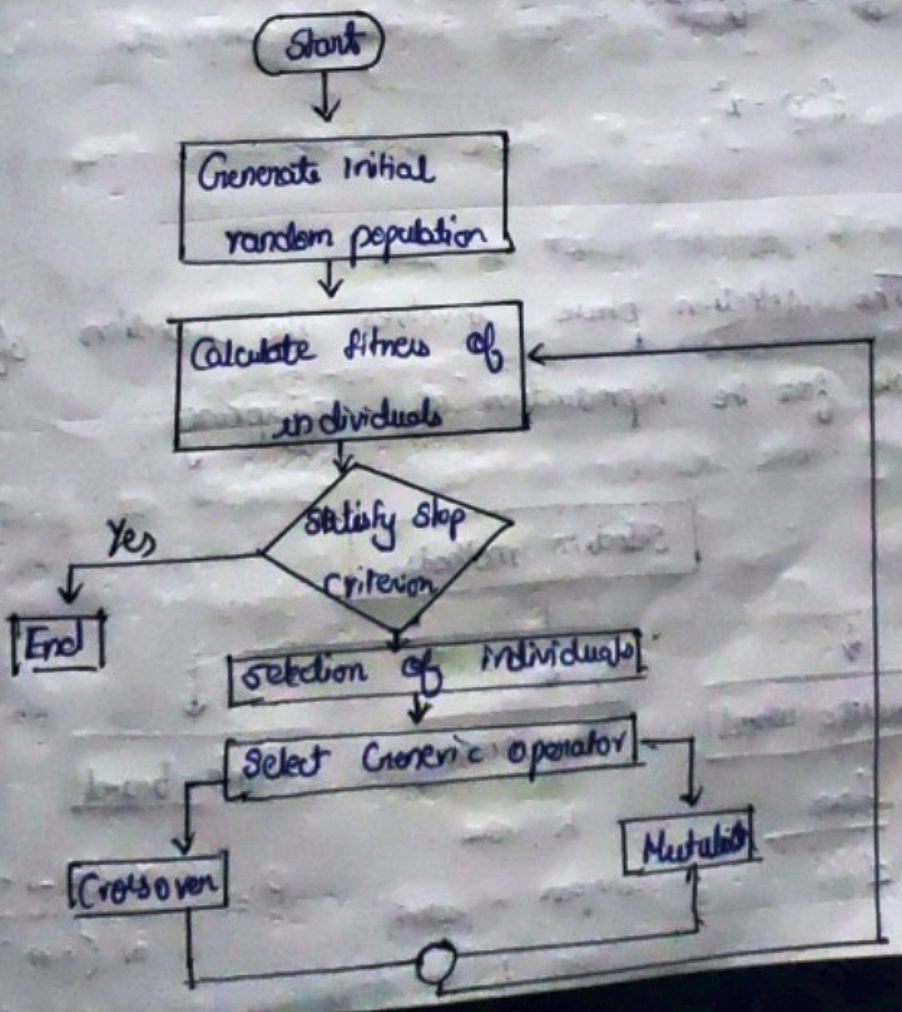
Final route

H - B - A - H

Lecture 6 :

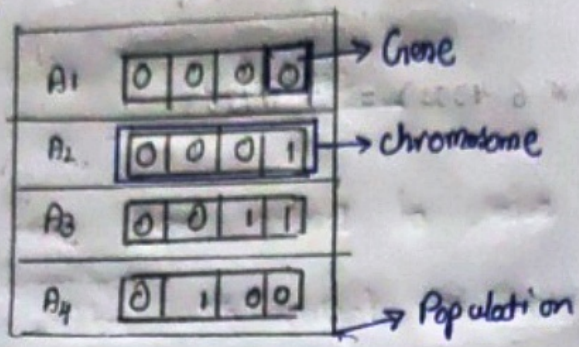
## Genetic Algorithm

Flowchart :



## Initialization:

The process of a genic algorithm starts by generating the set of individuals called population




## Fitness assignment:

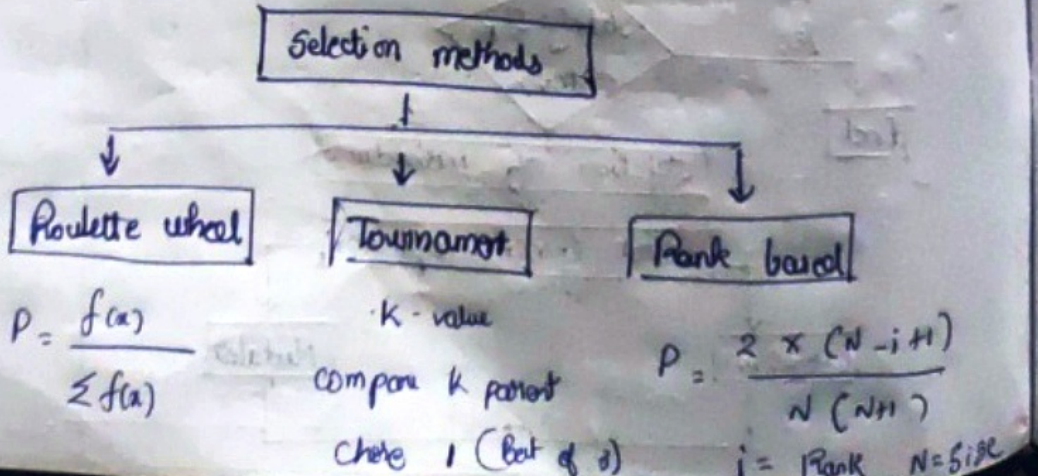
Fitness function is used to determine how fit an individual is?

In every iteration, individuals are evaluated based on the fitness function.

$$f(x) = x^2$$

## Selection:

The selection phase involves the selection of individuals for the reproduction of offspring.

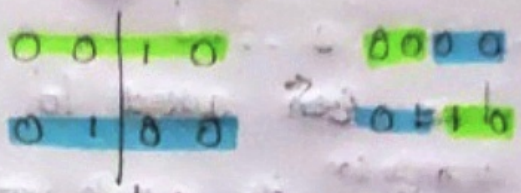


## Reproduction

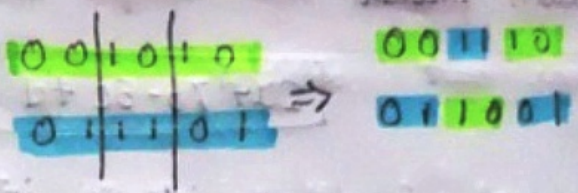
(i) Crossover :

The crossover plays a most significant role in the reproduction phase of genetic algorithm.

One point crossover :



Two point crossover :



(ii) Mutation :

Mutation helps in solving of premature convergence and enhances diversification.

The mutation inserts random gene in the offspring to maintain diversification in population

① Bit flip : 1 0 0 1 1 0 1 → 1 1 0 1 1 1 1

② Random reset : 1 2 3 3 5 6 → 1 2 4 3 5 1 {1:6}

③ Swap : 1 2 3 4 5 6 → 1 5 3 4 2 6

④ Scramble : 1 2 3 4 5 6 → 6 2 1 3 5 4

⑤ Inversion : 1 2 3 4 5 6 → 1 5 4 3 2 6

Termination :

After the reproduction phase a stopping criterion is applied as a base for termination

## Problem

Q) You have been asked to solve below linear equation problem with multiple variables

$$2a + 7b - 5c + d = 0$$

Where a, b, c, d are integers in range [-20, 20].

You have decided to solve this question using genetic algorithm. Show all the steps involved in solving this GA. Name each of your step and provide appropriate title.

Chromose [1, 2, 3, 4] where a=1, b=2, c=3, d=4

Generate population. (List of lists)

Fitness function : $f(x) = 2a + 7b - 5c + d$

Apply some selection criteria (roulette or best fit ch)

Apply crossover or mutation. (Your choice)

Solution :

Step ① :

Encoding technique -> Permutation encoding

Selection operator → Roulette wheel selection

Crossover operator → Two point crossover.

step ② :

Population size = 4

step ③ :

Initial population

| | | | | |
|---|---|---|---|---|
| ① | -1 | 2 | 4 | 6 |
| ② | 5 | 6 | -2 | 3 |
| ③ | -2 | -1 | 0 | -3 |
| ④ | 5 | 9 | 13 | 19 |

step ④ :

Fitness calculation

Higher fitness means higher probability

| Chromosome | Initial Population | F(x) | 1/F(x) | abs($\frac{1}{f(x)}$) | abs($\frac{1/f(x)}{\Sigma}$) | Selection |
|---|---|---|---|---|---|---|
| 1 | [-1, 2, 4, 6] | -2 | -0.5 | 0.5 | 0.659 | 3 |
| 2 | [5, 6, -2, 3] | 65 | 0.015 | 0.015 | 0.019 | 0 |
| 3 | [-2, -1, 0, -3] | -14 | -0.071 | 0.071 | 0.093 | 1 |
| 4 | [5, 9, 13, 19] | 27 | 0.037 | 0.037 | 0.048 | 0 |

$\Sigma = 0.758$

Parent combination

-1 2 4 6 ] ①
-1 2 4 6

-1 2 4 6 ] ②
-2 -1 0 -3

Step ⑤ Reproduction

Two point crossover

① -1 | 2 4 | 6 ⇒ -1 2 4 6
-1 | 2 4 | 6 ⇒ -1 2 4 6

② -1 | 2 4 | 6 ⇒ -1 -1 0 -6
-2 | -1 0 | -3 ⇒ -2 2 4 -3

Step ⑥ Fitness of offspring

| Chromosome | Offspring ① | F(x) |
|---|---|---|
| 1 | [-1, 2, 4, 6] | -2 |
| 2 | [-1, 2, 4, 6] | -2 |
| 3 | [-1, -1, 0, 6] | -3 |
| 4 | [-2, 2, 4, -3] | -13 |

The fitness value remains the same.

Step ⑦ :

Now we will take these four offsprings as parents and repeat the process until our fitness function is achieved.