~~Artificial Intelligence~~ Date

Artificial Intelligence

→ Applications of AI
- face recognization
- Recommendation systems
- Seach engines
- Speech recognization
- etc

→ AI is about acquiring knowledge, behaviour, emotions, signals, symbols to create AI that can replicate human intelligence completely.

→ Strong AI refers to machine which ~~can~~ supersedes human intelligence. (ChatGPT) whereas weak AI refers to limited part of mind implementation. It is intelligence that is limited to specific area (Siri)

★ Agents & Environments
- Agent: An agent is an entity or a program that interacts with it's environment by perceiving it's surrounding via sensors and acting through actuators.

Human agent: eyes, ears (sensors) hands, legs (actuators)

Agent = Architecture + Program

★ Rational Agents
- Rational agent is entity that aims for optimal actions on basis of what it perceived. (maximize its performance measure)

an agent is autonomous if it's behaviour is determined by it's experience.

PEAS is kept in mind while designing rational intelligence.

Date _____

✱ Environment types

① Fully vs ~~Obser~~ Partially Observable :

fully observable is when you have access or can sense complete state of an agent at point otherwise partially.
Chess (fully observable)
Driving (partially observable).

② Deterministic vs Stochastic :

if agent current state & action can completely determine next state then deterministic else if random in nature then stochastic.
Chess (deterministic)
Driving (stochastic)

③ Episodic vs Sequential :

agent's actions are divided into episodes (independent) (pick & drop robot). In sequential previous decision can affect all future decisions.
(dependent) (self driving cars).

④ Static vs Dynamic :

Environment is static when it's constantly changing itself when agent is in action. (rollercoast)
when environment is idle then it is dynamic. (empty house).

⑤ Discrete vs Continous :

when number of actions are finite then it is discrete (chess) to obtain output.
when actions cannot be numbered then it's continous (driving).

⑥ Single vs Multi agent :

environment consisting single agent (crossword) whereas environment consisting multi agent, more than one (chess, football).

Date

★ Agent Types

① Simple reflex agents:

agent selects an action based on current precept & conditions, ignoring rest of history. (limited intelligence, operates in partially observable environment)
e.g alarm clock [if x do y]

② Model-based reflex agents:

agent selects an action based on pre-defined condition (set of rules). uses internal memory to make decisions
e.g self driving cars.

③ Goal-based agents:

agent selects an action based on previous experiences, future, goal, user input.
It distinguishes itself from others thru ability of finding solution.
e.g GPS

④ Utility-based agents:

agent selects an action based not only goal but also the best way to reach goal. using preferences.
e.g GPS system finding shortest path.

⑤ Learning Agents:

agent selects its action based on feedback not only sensors. It learns to make improvements from past.
A critic gives learning element feedback and ~~how to~~ if it needs to be modified.
problem generator suggest new actions.
e.g human

Date _____

*Searching

- Searching falls under AI, human breaks down complicated problem into simpler steps so that computer can solve.

- Uninformed (Blind Search): does not contain domain knowledge instead it uses brute force to traverse whole tree until reaches goal node.

- Search algorithms helps in optimization, planning methods, problem solving by formulation of the problem.

- Problem formulation :

1. State
2. Initial State
3. Successor Function
4. Goal Test
5. Path Cost
6. Solution.

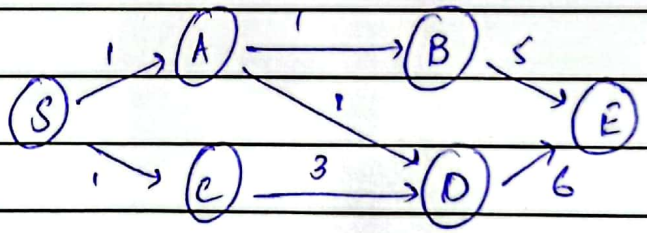
- Search Tree :

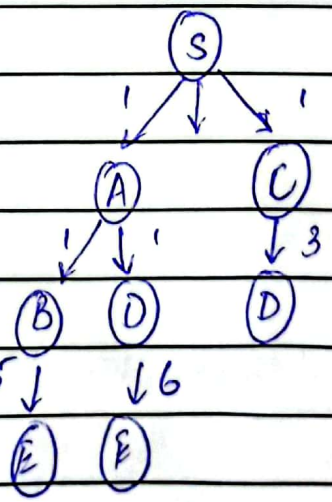
As search space is traversed search tree is generated by expansion of nodes by applying successor function.

fringe (set of nodes not visited)

A graph is traversed making initial node as it's root node.

arrangement of tree depends on traversal mode.



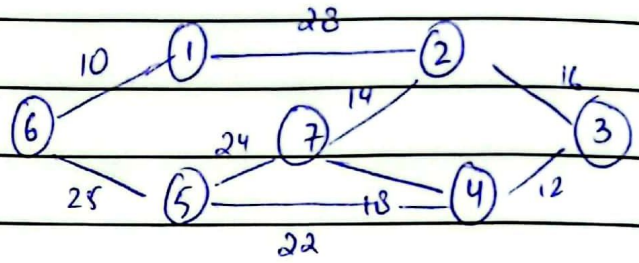


Date

- Kruskal Algorithm :

(1) List all edges (weight wise)
(2) Select smallest edge
(3) Select next smallest edge that does not make circuit
(4) Continue until all vertices explored and $(V_n - 1)$ edges.

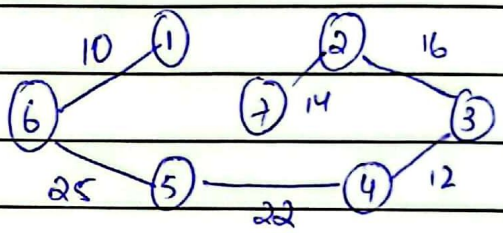
→ Prims Ques //



Sol //

- Prim's Algorithm :

(1) Draw nxn matrix with vertices labeled with weights
(2) Start from V_1, connect to nearest neighbour.
(3) Consider V_1 & V_i, connect without making circuit
(4) Continue until $(V_i - 1)$ edges.



Prim's	Kruskal
(1) $O(V^2)$ $O(E+\log V)$	$O(E+\log V)$
(2) faster in dense	faster in sparse

		1	2	3	4	5	6	7
(1)	1	-	28	-	-	-	10	-
(6)	2	28	-	16	-	-	-	14
(5)	3	-	16	-	12	-	-	-
(4)	4	-	-	12	-	22	-	18
(3)	5	-	-	-	22	-	25	24
(2)	6	10	-	-	-	25	-	-
	7	-	14	-	18	24	-	-

Date _____

☆ Searching Strategies

① Uninformed (blind search)
- number of steps, path cost unknown
- agent knows when on goal.

② Informed Search (heuristic search)
- agent has background info
- map, cost known.

- Search strategy ~~with be~~ is picked by evaluating following dimensions:
  ① Completeness (solution exists or not)
  ② Time complexity
  ③ Space complexity
  ④ Optimality (best solution)

b: max branching factor of tree
d: depth of least cost solution
m: maximum depth of space state

b (number of children at each node)

★ Uninformed Search

① Breadth-First Search

- Starts from root expands all successor nodes before moving to next level. (FIFO)

Algo:
① Enqueue root
② Dequeue node (if found return else examine unvisited successors) & enqueue them
③ if queue empty, return not found
④ Repeat ②.

Complete	Yes if b finite
Time	$O(b^{d+1})$
Space	$O(b^{d+1})$
Optimal	Yes

★★ not suitable for large graphs.

Date

(2) Uniform-Cost-First

- Visits next node with least total cost from root.
- uses lowest cost (cummulative) to find path from source to destination.
- priority queue.

Complete	Yes (if b finite)
Time	$O(b^{1+C/\varepsilon})$ ε (cost of each step)
Space	$O(b^{1+C/\varepsilon})$
Optimal	Yes

(3) Depth First Search

- It explores a path all the way to leaf before backtracking to another path.

Algo:

(1) Keep root vertex on top of stack
(2) Take top item of stack mark visited
(3) Add adjacent vertex unvisited to top of stack.
(4) Repeat 2 and 3 until stack empty.

Complete	No (in loops, infinite)
Time	$O(b^m)$
Space	$O(bm)$
Optimal	No

(4) Depth-Limited Search

- Similar to DFS but with limit
- solves issue of infinite paths.

Complete	NO
Time	$O(b^l)$
Space	$O(bl)$
Optimal	No if $(l<d)$

l : depth limit.

No complete : if goal is beyond limit.

if $(l<d)$

Date

(5) Iterative Deepening DFS

Complete	Yes (b finite)
Time	$O(b^{d/2})$
Space	$O(b^{d/2})$
Optimal	Yes (

- Similiar to DLS but with increasing limits.
- Seaches to Depth 0 if not found then depth 1 so on.
- if goal node found, it is nearest node and path is on stack.
- Stack → (Depth + 1) size
- One of the best uninformed seach methods.

Complete	Yes (if b is finite)
Time	$O(b^d)$
Space	$O(bl)$
Optimal	Yes (if step cost is 1)

(6) Bi-Directional Seach

- Seach foward from initial state and backwards from goal. two seaches meet in middle step.

Date

Informed Search

(1) Best - First Search

- expands the node with smallest straight line distance.
- better than depth first, worst-time complexities.

Complete	No (loops)
Time	$O(b^m)$
Space	$O(b^m)$
Optimal	No

(2) A* Search

- evaluation function = path cost + estimated cost.
- combines greedy & uss

Complete	Yes (finite)
Time	Exp // $O(b^d)$
Space	large
Optimal	Yes.

* Admissible Heuristics

$h(n) \leq h^*(n)$ - underestimation

$h(n) \geq h^*(n)$ - overestimation

estimated overestimation

acceptable.

* A* proof

Date _____

* Local Seach Algorithm

(1) Hill climbing :

- local search, greedy, no backtrack.
- evaluate initial state, loop until solution found, if goal then return else assign new state better than current.

(2) Beam Seach

(β = Beam Width)

- keep best β heuristics and remove back, takes care of complexity → (space)

problems :

(1) Local maximum

another state also present higher than the local maximum.

(2) Plateau

all neighbour states of current state are same, does not find best value.

(3) Ridges

higher are but has a stope within itself so cannot move.