CIT - 315

Artificial Intelligence

Book:

A modern approach - ~~Staun~~ Stuart J. Russell

Chapter → 1, 2, 7, 8, 16

Lab → python AI → logic → First order logic
→ Predicate logic

AI Project → AI application development

→ ① Hardware + Software ③ Dataset

② Chatbot

predictive analysis

Chapter - 1

Masud sir

05.05.2024

AI → Chapter 3:

Neural network

Traditional Algorithm vs AI algorithm

↓ ↓

train/estimation/ learn হয়। Input এবং process এবং output এবং

data pattern explore করে learn করে।

We train the algorithm and model

Solving Problem by searching —

Introduction —

Goal based agent/problem solving agent.

Uninformed search algorithm

Informed search algorithm

3.1 Problem solving agent:

1. Intelligent agent - maximise performance

Steps in problem solving

① Goal Formulation

② Problem

#observable
#known
#deterministic → output একটা হবে।

Fig-3.1

#Well defined problem and solutions – five component

1. Initial state
2. Action
3. Applicable
4. Transation model
5. State space * (1, 2, 3 collectively)
6. Path
7. Goal estate

#search
#Solution
#Execution

8. Goal Test
9. Path cost
10. Step cost
11. optimal Solution

3.4 Uninformed search strategies

blind search —

3.4.1 Breadth first search

1st expand → root node

→ All the succesor of root

→ their succesor

FIFO

b = branching fact

d = depth

complexity → $b + b^2 + b^3 + \cdots b^d$

→ $O(b^d)$

Completeness, optimal, space complexity, time complexity

3.4.2

Uniform cost search →

no care of steps,

care of cost

Total cost C^*

every cost ϵ

steps C^*/ϵ

~~Compt~~

3.4.3

depth-first-search →
not complete, not optimal.

m = maximum-depth

$O(b^m)$ → time complexity

$O(b \times m)$ → space complexity

space complexity কম, কিন্তু depth ও চলে যায়, এজন্য problem.

3.4.4 → depth limited search.

3.4.5 →

3.4.6 → Bidirectional

$b^{d/2} + b^{d/2} = 2b^{d/2}$ → complexity কম

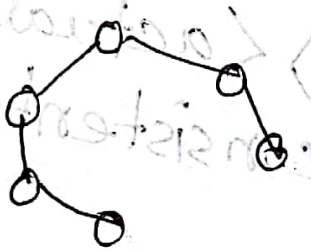
$2b^{d/2} < b^{d/2}$

A* search →

Heuristic function and its application.

heuristic function

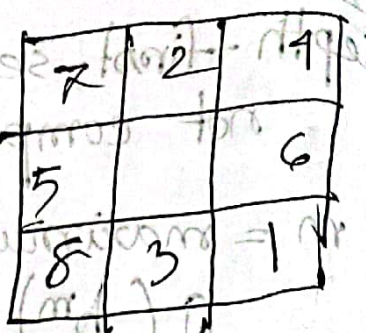
$h(n)$ = estimate cost of the cheapest path from node n to goal node.



$h(n) = g(n)$

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

$= h_1(n) + h_2(n)$



$h_1 = 8$

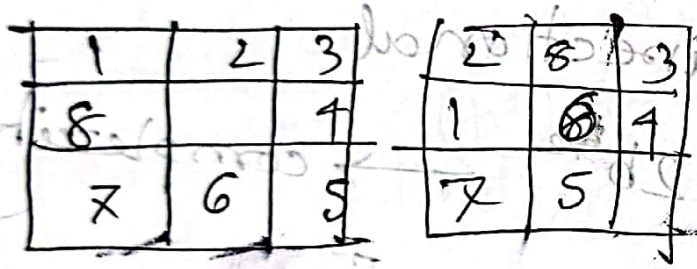
$h_2 = 3+1+2+2+2+3+3+2$

$= 18$

$h(n) = h_1 + h_2 = 26$

$g(n)$ = depth of the node

$h(n)$



heuristic may be

$h(n) >$ actual cost $\rightarrow$ optimal solution can be overlooked. O.S is not possible

$h(n)$ = actual cost $\rightarrow$ best case.

$h(n) <$ actual cost $\rightarrow$ admissible, consistent heuristic

consistent →

$h(n) \leq c(n, a, n) + h(n)$

If $h(n)$ is consistant then the value of $f(n)$ along any path are non decreasing.

Chapter - 4

Local search — Global minimum - lowest valley
Global maximum - highest valley

chap-6

6.1.3 definition - discrete domain, finite domain, infinite d, constraint language

revise function → remove inconsistent value

02.06.24

Chap - 7

Mehboob sir

Javatpoint → Knowledge represent
Knowledge - based agent AI

Knowledge base
Inference system
operations —
Approaches —

Generic knowledge based —
levels —

What is knowledge representation?
What to represent?
Type of knowledge
knowledge cycle
propositional logic
Artificial intelligence.
Syntax
Logical connectives
presedence
properties of operations.

Inference
Types

29-08 Chapte-13 Musad sir

13.2.1
Probability - Unconditional, Prior
conditional → ***
joint P(C∧T) / P(C T)

~~P(a|b) = P~~
$P(a \wedge b) = P(a|b)\, P(b)$
$P(b \wedge a) = P(b|a) \cdot P(a)$
$\Rightarrow P(a|b)\, P(b) = P(b|a) \cdot P(a)$
$\Rightarrow P(a|b) = \frac{b(b|a)\, P(a)}{P(b)}$ → bayes rules.

Fig 13.3 ****

31.08.24

Mahbub sir chap-7
Javatpoint
Propositional logic — totology, contradiction,
proposition — assertive sentence

Atomic proposition — with example
Compand n — n
Logical connectiveness — Negation, conjunction,
Truth tables
Precedence
Logical equivalence
properties of operation
Limitation
Types of inference rules —

Masud sir

13.1

Independence – Full joint distribution P(Toothache, Catch, Cavity, Weather) cloudy

Product rule : $P(A, B) = P(A|B)\, P(B)$

$P(\text{toothache, catch, cavity, cloudy}) = P(\text{cloudy} \mid \text{toothache, catch, cavity}) \cdot P(\text{toothache, catch, cavity})$

because independent relation

$= P(\text{cloudy}) \cdot P(\text{toothach, catch cavity})$

$P(a|b) = P(a)$, $P(b|a) = P(b)$ → যদি independent variable হয়,

$P(a \wedge b) = P(a) \cdot P(b)$

Bayes Rules – $P(b|a) = \frac{P(a|b)\, P(b)}{P(a)}$

$P(Y|X, e) = \frac{P(X|Y, e)\, P(Y|e)}{P(X|e)}$

Using Bayes Rule causing evidence –

$P(\text{cavity} \mid \text{toothache} \wedge \text{catch}) = \alpha\, P(\text{toothache} \wedge \text{catch} \mid \text{cavity})\, P(\text{cavity})$

$P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) = P(\text{toothache} \mid \text{cavity})\, P(\text{catch} \mid \text{cavity})$ → conditional

$P(\text{cavity} \mid \text{toothache} \wedge \text{catch}) = P(\text{toothache} \mid \text{catch}) \cdot P(\text{catch} \mid \text{cavity}) \cdot P(\text{cavity})$

$$P(y|x_1, x_2 \dots x_n) = \frac{P(x_1|y)\, P(x_2|y) \dots}{P(x)}$$

$$= P(y) \prod_{i=1}^{n} P(x_i|y)$$

$$P(y) = \text{argmax}_y \, P(y) \prod_{i=1}^{n} P(x_i|y)$$

Mahbub sir

01.09.24

Javatpoint →

Knowledge base Wumpus world.

Lab implement fn(n) → Wumpus moveable.

Stench, Breeze, Glitter, Bump, ~~Scratch~~ Scream

Mahbub sir

02.09.24

Wumpus World → Javatpoint

↓

Performance

Environment

actuators

sensors

properties

Agents step

Knowledge base wumpus world → Javatpoint

4×4 – 16 logic equation

2×4×4 – proposition variable

Table যেভাবে লিখতে হবে,

Theory তে থাকবে।

09.08 Mahbub sir

Microprocessor lab → report → assignment
AI " " → " "
AI Theory → AI related
Microprocesso Theory → scan assignm
Microprocessor → Text বই থেকে পড়তে হবে,
→ আগের বার থেকে Question
Marith → Flag
AI → 2 টা
→ 1st order logic + chapter 2
→ cha propositional logic + chapter 1

First order logic →
Modern Intelligence 2 ধরনের logic → 1st order
→ propositional

objects, properties, relations, functions.
individual identity ↓ distinguish between object

User provides — constant symbol, function symbo
Javatpoint → 1st order logic

Syntax
∀ → reverse A → All → universal quantifie
∃ → " E → Exsential "

Atomic sentence
Complex sentence

Subject, predicates

Quantifiers

Universal Q

Existential Q

Properties of Q

5 টি sentence এ কাজ, represent করা দেখাতে হবে।

Free and bound variable → কার আগে পর্যন্ত

~~18.4 Ev~~ 10.09.24 Masud sir

18.4 Evaluating and choosing best hypothesis

IID ? যদি independent হয় তবে assumption গুলো যদি সমান হয়।

Hold out cross-validation

K-fold cross-validation

LOOCV $k=n$

18.4.1

model selection : two task —

(i) model selection H{ }

(ii) Find the best Hypothesis — optimisation

Error Rate of a hypothesis as ~~a problem~~ the proportion of mistakes it makes ⊕ —

$h(x) \neq y$

fig 18.9

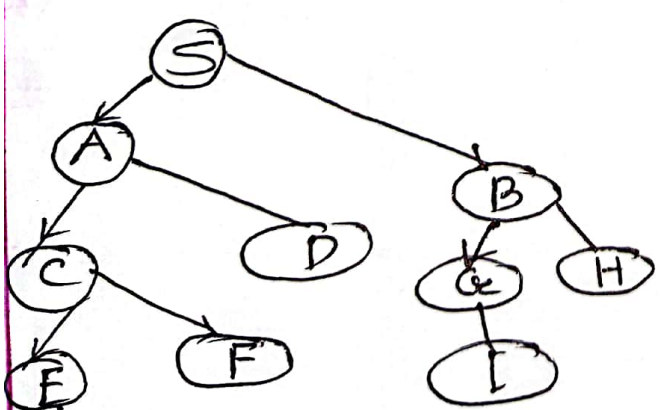
২ টি node নিয়ে কাজ করলে optimise

BFS –

- provide solution if any solution exists.
- helps to find the shortest path in goal state since it needs all nodes at same ~~hierarchichat~~ hierarchical ~~no~~ level before making a move to nodes at lower level.
- If more than one solution exists, then it will provide minimal solution which requires least number of steps

Disadvantages – Lots of memory

- Lots of time (if far away from root)
- inefficient for deeply layered spaces.



S → A → B → C → D → G → H →
E → F → I → K

Uniformed search strategies —

BFS

- All nodes are expanded in a given depth in search tree before any nodes are ~~expanded~~ at the next level expanded.
- This is achieved by FIFO queue for the frontier.
- Not good for time and space
- Total numbers of nodes generated —

$$b + b^2 + b^3 + \cdots - b^d = O(b^d)$$

- It expands shallowest node.
- optimal if path cost is non-decreasing function of the depth of the node

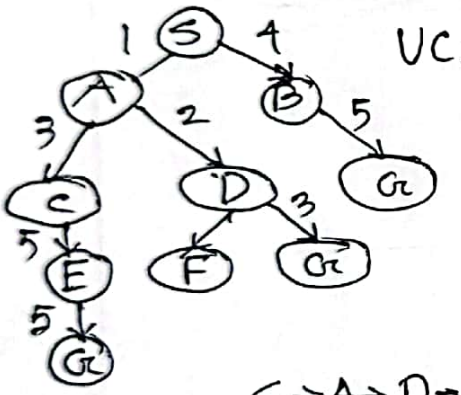
Uniform-cost search

- uniform ~~pat~~ cost search expands the node n with the lowest path cost $g(n)$
- store frontier as a priority queue ordered by g.
- Uniform cost search doesn't care about number of steps a path has, but only about the total cost. Therefore it will stuck infinite loop if there is a path with an infinite sequence of zero cost action.

- Optimal because at every state the path with least cost is chosen

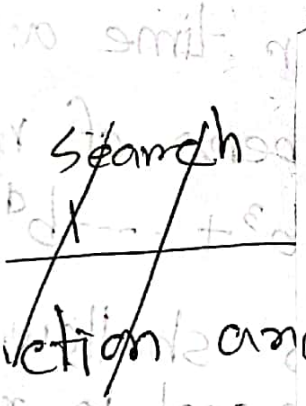
- C^* = cost of optimal solution
  worst case time and space complexity is $O(b^{1+[C^*/\epsilon]}) \gg b^d$
- If all steps are equal $O(b^{1+[C^*/\epsilon]}) = b^{d+1}$

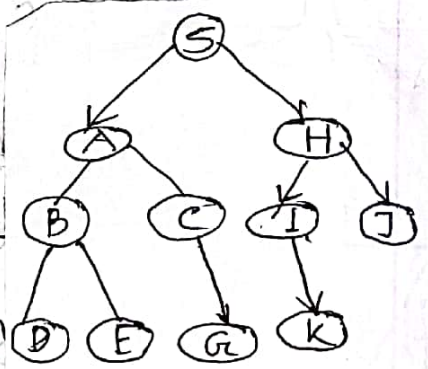
- The goal test is applied to a node when it is selected for expansion rather than when it is first generated.



UCS

S→A→D→G





S→A→B→D→E→C→G

DFS

~~Depth-limited search~~

Depth-first-search

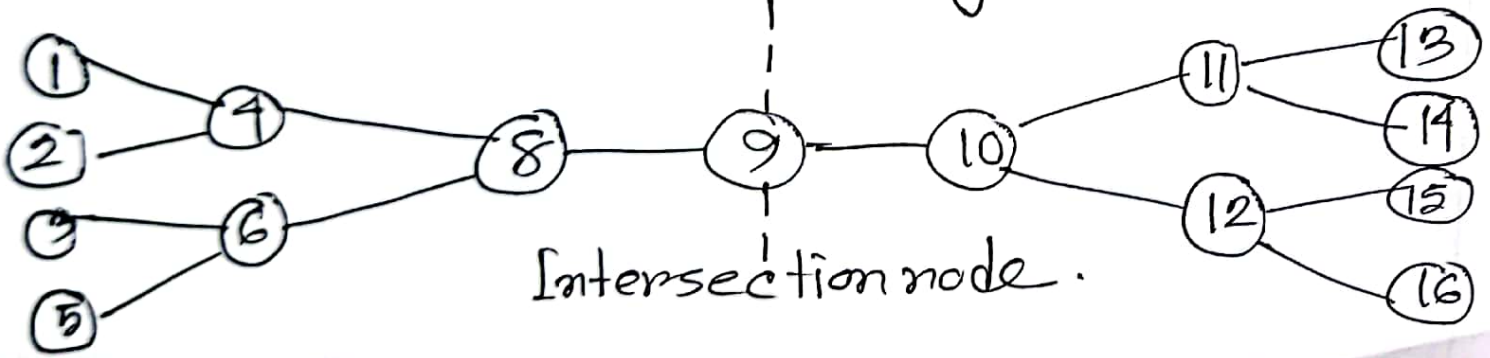
- expand ~~depe~~ deepest node
- Use LIFO queue
- variant of depth-search called backtracking search.

- Once node has been expanded, it can be removed from memory as soon as all its descendants have been fully explored. So it uses less memory

- Combines DFS and BFS benifits in terms of first search and memory efficiency.
- guarantee to find optimal solution as long as the cost of each edge in the search space is the same
- Repeats all works of previous steps - drawback

Bidirectional search

- Replace one single search graph with two small subgraph in one start from initial vertex and another start from goal vertex and stops when two intersects.
- It is fast, less memory, helpful for large graph that has no way to make smaller.
- We should know the goal state in advance



- There is the possibility that many steps states keep re-occuring and there is no guarantee of finding the solution.

Depth-limited search - It solves infinite Path problem:

- Time complexity $O(b^l)$
- space complexity $O(bl)$
- Two type failure - ①standard failure - no solution
  ②cut off failure - no solution in depth limit
- It also not optimal even if $l > d$

Iterative depending depth-first search -

- It gradually increase the depth limit until the goal is found.
- Worst case $N(IDS) = (d)b + (d-1)b^2 + \dots (1)b^d$
- is preffered when the search space is large and depth is unknown.

Bidirectional search -

- One forward from inital state and other backward from the goal.
- Time complexity $= O(b^{d/2})$ = space complexity

---

* The graph search version, which avoids repeated state and redunt path

Criterion	Breadth First	Uniform Cost	Depth First	Depth Limited	Iterative Deepening	Bidirectional
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,b]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Optimal	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

b = branching factor.
d = depth of shallowest solution
m = maximum depth of search tree
l = depth limit

[a] = complete if b is finite
[b] = complete if step costs $\geq \epsilon$ for positive ϵ
[c] = optimal if step cost are all identical
[d] = if both direction use breadth first search.

A* search

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

- $f(n)$: evaluation function
- $g(n)$: the cost to reach the node from start node
- $h(n)$: Heuristic function: estimated cost to get from the node to the goal.

Heuristic Function: A heuristic function (algorithm) or simply a heuristic is a shortcut to solving a problem when there are no exact solutions for it or the time to obtain the solution is too long.

Heuristic properties –

1. Admissible condition: If an algorithm produces an optimal result it is considered admissible.
2. Completeness – An algorithm ends with a solution
3. Dominance Property – If A_1 and A_2 are two heuristic algorithm and have h_1 and h_2 n function respectively, then A_1 will dominate A_2 if h_1 is superior to h_2 for all possible value of node n
4. Optimality Property: An algorithm will thoroughly, allowable, and dominates the other algorithms

Heuristic Function h(n):

h(n) = Estimate cost of the cheapest path from node n to goal node

A* search →

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

g(n) = depth of node

h(n) = number of misplaced tiles

How to make heuristic Function—

7	2	4
5		6
8	3	1

Start node

–	1	2
3	4	5
6	7	8

Goal node

$h_1 = 8$

$h_2 = 3+2+2+2+3+3+2+1 = 18$

$h(n) = h_1 + h_2 = 8 + 18 = 26$

A*

Final

1	2	3
8		4
7	6	5

Initial

2	8	3
1	6	4
7		5

g = 0
h = 4

2	8	3	g=1
1	6	4	h=5
	7	5	f=1+5 =6

2	8	3	g=1
1		4	h=3
7	6	5	f=3+1 =4

2	8	3	g=1
1	6	4	h=5
7	5		f=5+1 =6

2	8	3	g=2
	1	4	h=3
7	6	5	f=3+2 =5

2		3	g=2
1	8	4	h=3
7	6	5	f=5

2	8	3	g=2
1	4		h=4
7	6	5	f=6

	8	3	g=3
2	1	4	h=3
7	6	5	f=6

2	8	3	g=3
7	1	4	h=4
	6	5	f=7

	2	3	g=3
1	8	4	h=2
7	6	5	f=5

2	3		g=3
1	8	4	h=4
7	6	5	f=7

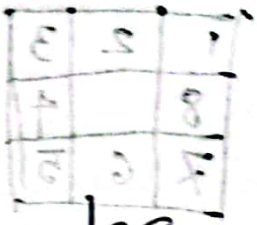
1	2	3	g=4
	8	4	h=1
7	6	5	f=5

1	2	3	g=4
8		4	h=0
7	6	5	f=4

Final state

1	2	3	g=5
7	8	4	h=2
	6	5	f=7

Optimality of A* search →



(1) $h(n) >$ actual cost

→ ~~Optimality~~ Optimum solution can be overlooked, optimum solution is not possible

(2) $h(n) =$ actual cost

→ Best case scenario. If $h(n)$ approximates actual cost, searching uses minimum of node of the goal.

(3) $h(n) <$ actual cost

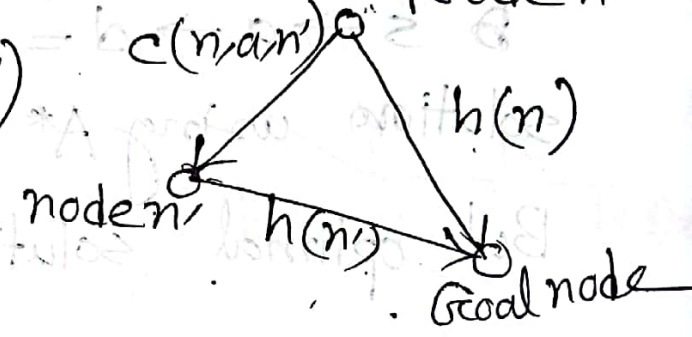
→ Admissible, consistent heuristics.

admissible হলে consistent হবে, কিন্তু consistent হলে admissible নাও হতে পারে।

Consistent heuristic —

A heuristic $h(n)$ is consistent (or monotonic) if, for every node n and every successor n' of n generated by any action, the estimated cost of reaching the goal from n is no longer greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'.

$$h(n) \leq c(n, a, n') + h(n')$$



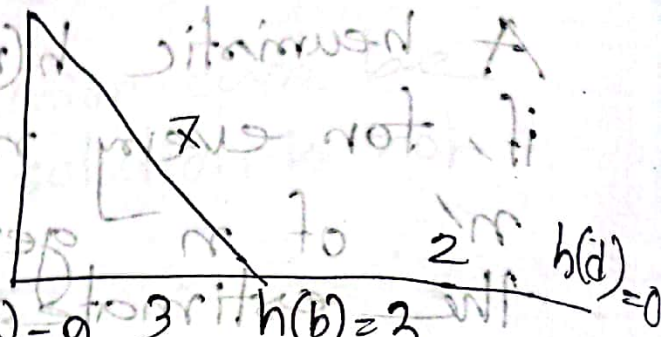
admissible হলে consistent হবে, কিন্তু consistent হলে admissible নাও হতে পারে।

Admissible heuristic → An admissible heuristic is that never overestimates the cost to reach the goal.

$h(n) \leq$ actual cost.

h(n) > actual cost

From start



$S \rightarrow a = 2+9 = 11$

$S \rightarrow b = 7+3 = 10$

So, we pop b, then explore neighbours a and d.

But a is visited, we pop d

$S \Rightarrow a \rightarrow d = 7+2+0 = 9$ This is the solution using A*.

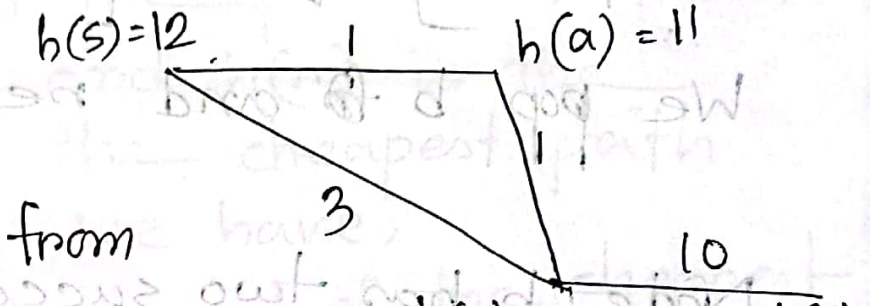
But optimal solution is $S \rightarrow a \rightarrow b \rightarrow d =$

$2+3+2+0 = 7$

A* doesn't provide optimal solution because h(n) > actual cost.

Heuristic must be admissible and consistent for the optimal solution in the A* search algorithm →

We will see an example for that the heuristic is admissible but not consistent.



For admissible,

shortest distance from s to dest →

$s \to a \to b \to d = 1+1+10 = 12$

For node s, $h(s) = 12 \leq 1+1+10$ or, $12 \leq 3+10$

For node b, $h(b) = 0 \leq 10$

For node a, $h(a) = 11 \leq 1+10 = 11$

Therefore this is addmissible heuristic

For consistent,

We will see

	$h(n)$		$c(n,n') + h(n')$
s to a	12	$\leq$	1 + 11
s to b	12	$\geq$	3 + 0 (not satisfied)

Therefor heuristic is admissible but not consistent.

If heuristic is admissible but not consistent it ~~won't~~ it may not provide optimum solution.

If we apply A* algorithm →

Starting node S has two successors a and b. S has push two value in priority queue [a = 1+11 = 12, b = 3+0 = 3]

We pop b and remained [a = 12].

Node b has two successor, a and d. but a has visited already, so we will push the value d. [a = 1+11 = 12, d = 3+10 = 13]

Then pop a and remained [d = 13].

d = 13 is the solution.

The solution is not optimal. We know here optimal solution is 1+1+10 = 12

If heuristic is admissible but not consistent, it may not be provided optimum solution. For optimum solution, Heuristic must be consistent. If it is consistent, it must be admisible.

Most widely used best first search is A* search. It evaluates nodes by combining $g(n)$, the cost to reach the node and $h(n)$, the cost to get from the node to the goal.

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

since $g(n)$ gives the path cost from start node to n, and $h(n)$ is the estimated cost of the cheapest path from n to goal, we have,

$f(n)$ = estimated cost of the cheapest solution through n.

11.09.24

Maxid sir

18.4.2

From error rates to loss

Error 1% (spam → nonspam) better Error 5%. (nonspam → spam)

Expected utility. is means of loss function in ML

Loss function :. The loss function $L(x, y, \hat{y})$ is defined as the amount of utility lost by ~~Ped~~ predicting $h(x) = \hat{y}$ when the connect answer is $f(x) = y$

Lo/1 → 1 loss

Theory of learning → less important

18.6

18.6.3 (fig 2 টি চলবে না।

18.6.4 (

Neutral network কী? কীভাবে কাজ করে? Architecture function

back prop কী? algorithm,

mutilayer, feed forward কী কীভাবে হয়?

3 → 1

4 → 1

6, 13 → 1

18 → 1

back propagation ***