Priyank Lohariwal Mob:- 9775958301
Roll- 00171050105 5 Email:- lohariwalpriyank@gmail.com,

Artificial Intelligence ~~BE~~ BCSE - Final year

1. a) In the turing test, the behaviorist approach is used to classify an agent as intelligent.

Turing test

3 rooms contains a person, a computer & an interrogator. The interrogator can communicate with the other 2 by tele type. The interrogator tries to distinguish between the person and the machine. If the machine is successful in fooling the interrogator into believing that it is the human, then the machine passes the test and is considered intelligent.

The turing test is a one-sided test. Even if the machines are not actually "intelligent", they can still use some kind of trickery to ~~make the inter~~ pass the test and to be classified as intelligent.

Most programs stresses on simple syntactic analysis and generation of sentences using pattern matching with known sentences, vocabulary and key words.

Ex:- 'Eugene Goostman'. He portrayed as being a 13 year old Ukrainian boy whose 1st language is not English. He ~~uses simple reverse analogy~~ & he used simple ~~grammatical Syntat~~ syntactic analysis to generate sentences to ask their interrogators to 'forgive' his minor grammatical errors and asks a counter question unrelated to the one being asked.

Hence, having these points in mind, we can definitely deduce that

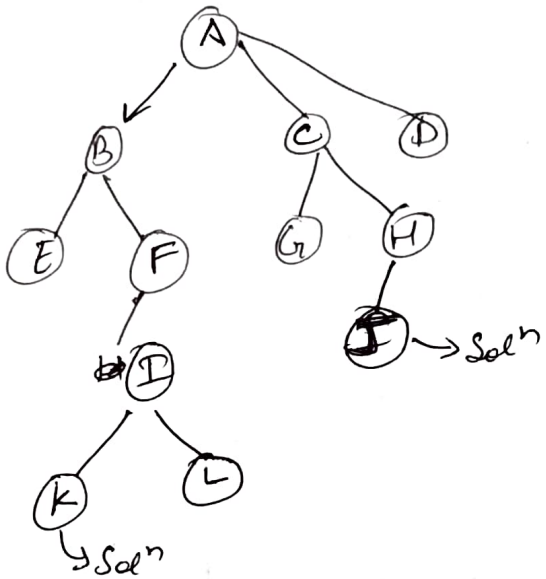
"Turing test is not a good way to judge a Rational Agent."

---

2. a) Iterative Deepening Search: An optimal Strategy:-

Iterative deepening ~~uses~~ uses depth Limited Search(DLS) and iteratively increases the depth until a solution is found.

Depth Limited DFS is not optimal in itself.



DFS Search result :- A B F I K

A valid Soln but not optimal.

Hence, we can see that Depth Limited DFS is not optimal if we set depth as 4.

But in case of IDS →

A

A B C

A B E F C G H

A B E F I C G H (J)

optimal Soln

Since, IDS increases the depth ~~limit~~ limit by 1 at each iteration, it is guaranteed to find an optimal solution if it exists. (given all paths have same cost).

---

b) ~~For~~

~~Let us~~

For a search space, let us assume

b = average branching factor of a node

d = maximum depth ~~upto which algorithms may reach~~ of the graph.

Generally, the time complexities of BFS, DFS & IDS are $O(b^d)$: However, we can quantify the cost values for various search techniques are:—

BFS: $$\frac{b^{d+1} + b^d + b - 3}{2(b-1)}$$

DFS: $$\frac{b^{d+1} + bd + b - d + 2}{2(b-1)}$$

IDS: $$\frac{b^{d+2} + b^{d+1} + b^2 - 4bd - 5b + 3d + 2}{2(b-1)^2}$$

As, $b^d > bd \Rightarrow$ cost(BFS) > cost(DFS)

Also, cost(IDS) $\approx \left(\frac{b+1}{b-1}\right)$ cost(BFS)

$\Rightarrow$ cost(IDS) > cost(BFS)

Hence, the time complexities of BFS, DFS & IDS are related as

cost(IDS) > cost(BFS) > cost(DFS)

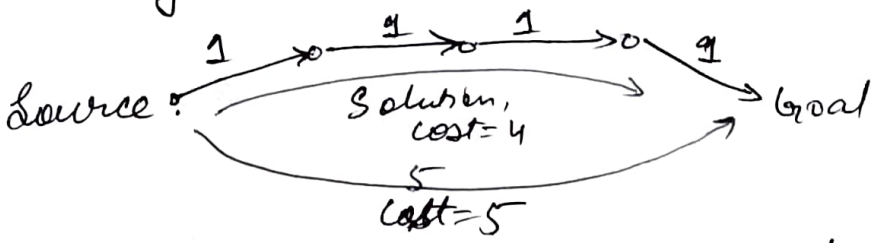
c) When an average node of a tree has a larger number of child nodes, i.e, branching factor (b) is greater than the depth of the tree (d), then we prefer iterative broadening over Iterative deepening.

On the contrary

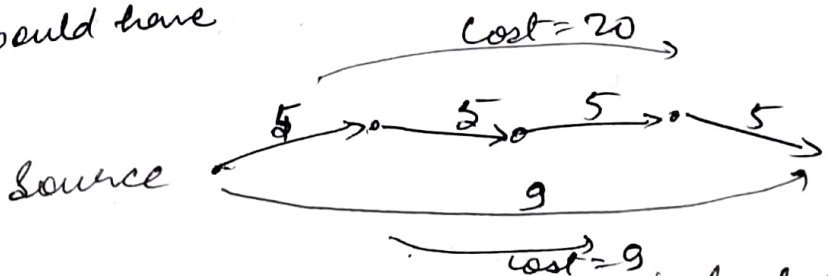
If $d >> b$, we prefer Iterative deepening.

d) Yes, it is highly likely that the Solution path will change. This is even more likely on paths with multiple hops.

Ex:- Initially :-



If we increase all weights by 4, we would have



we can see that the cost of a path gets increased by a factor of k x no. of edges in the path.

However, another path which is a valid Solution set gives a more optimal Solution

Hence, if we have a path with less no. of nodes such that the modified cost gets decreased is less than the previous optimal soln, the result changes.

3. a) ~~Ques~~ True.

General Graph Search Algorithm is applicable for various strategies.

What happens in this technique is, at each iteration, we reorder the ~~unv~~ non-visited or OPEN nodes. The scheme of this reordering is not fixed and the programmer has the freedom to choose any arbitrary scheme or some scheme based on a heuristic metric.

We can choose any data structure and any sorting technique we want.

If we use a stack or a queue, DFS and BFS would get implemented respectively.

If we use a priority queue for OPEN nodes storage with some heuristic cost, we will get informed searches like Best first search, A* search etc.

---

b) False.

If the time complexity of a search algorithm is higher, it ~~will not~~ may provide optimal solution.

Ex:-

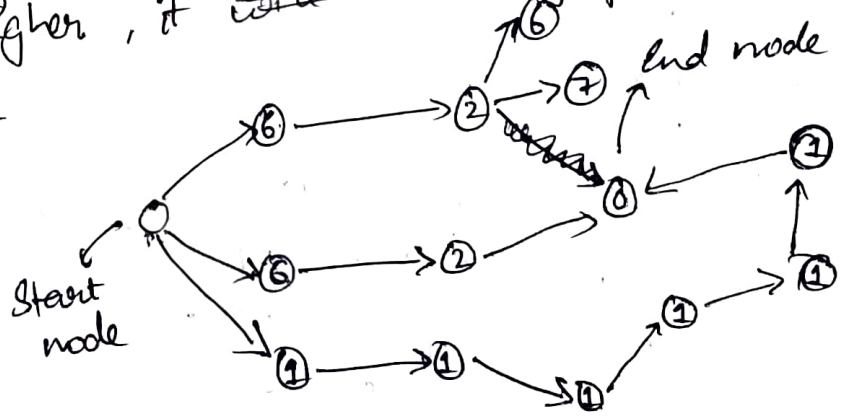


Fig 3.1

0017105010S5

Consider the previous graph in fig. 3.1 depicting a ~~Search~~ state space graph. The values in each node depicts ~~it~~ the particular ~~state's~~ state's heuristic value.

Applying ~~BFS~~ Breadth First Search (BFS)

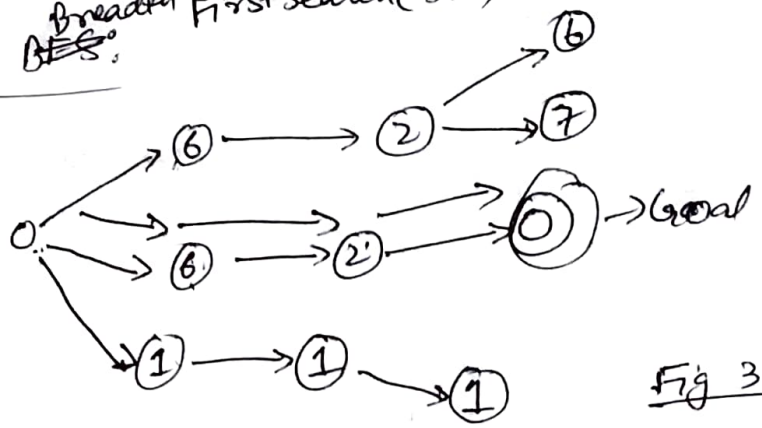


Fig 3.2

Fig 3.2 shows the nodes travelled by BFS to reach ~~find~~ the ~~root~~ goal state.

No. of nodes travelled = ~~9~~ 11

No. of nodes in the resultant path = 4 (including start & end state)

Applying ~~heuristic~~ Best first search:

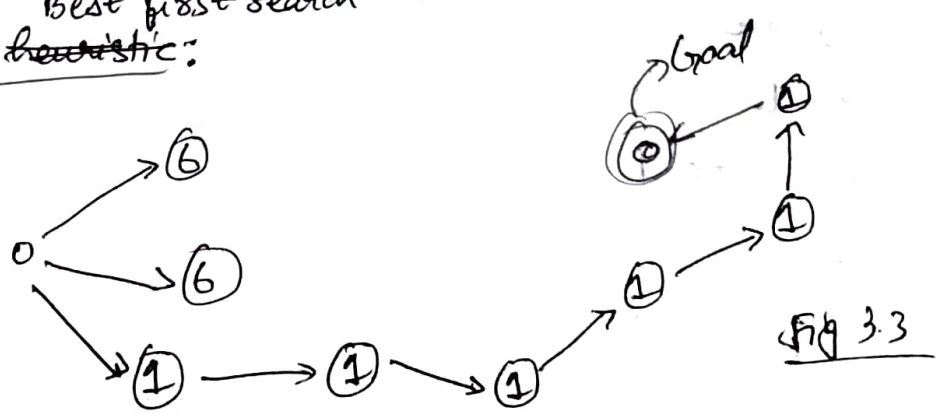


Fig 3.3

Fig 3.3 shows the nodes travelled by Best first search to reach the goal state.

No. of nodes travelled = 10

No. of nodes in the resultant path = 8

~~In the~~

Priyank Lohariwal

In the given example, we can see that the time complexity of BFS is greater than that of Best first Search but it still gives an optimal solution.

---

c) True.
Heuristic search processes are better than blind search techniques.

Blind Search:-
→ Brute in nature → lengthy process
→ Large memory used → Search process remembers all the unwanted nodes which are of no use at all.

Heuristic Search:-
→ Uses some kind of information of the problem to reduce search space
→ No large memory used
→ Heuristic functions are used for searching.
→ Quicker process

Ex:- 15-puzzle problem

If we apply blind search technique, total no. of states to be explored $\simeq 10^{13}$. This becomes a non-polynomial algorithm.

Applying a heuristic function, such as manhattan distance or the number of misplaced tiles, the search space reduces drastically (approx. 1000)
Hence, heuristic approaches are better.

Priyank Loharwal.

5.a) Normal hill climbing suffers from problems like getting stuck at plateaus, ridges or getting stuck at some local optimas. If the method is conducted only once, then the chances of facing these issues are considerable. If the algorithm is run multiple times, the chances of reaching local optimas, ridges or plateau regions decreases considerably. Thus, the chance of attaining a solution becomes high. Hence, random restart hill climbing is better than simple hill climbing process.

b) Simulated Annealing algorithm:-

```
begin
    t=max
    Select a random state sc and compute its
    functional value f(sc)
        while (t>=tmin)
        for l=1 to n
            begin
            pick an adjacent state sn from sc at
            random and compute f(sn)
            If f(sc) >=f(sn) set sc=sn
            else set sc=sn with probability
             {exp(-(f(sn)-f(sc))/t}
            end
        decay t
        end
end.
```

If we look at the probability function,

$$P(t) = \exp\left(-\frac{(f(s_c) - f(s_n))}{t}\right)$$

when, temperature t is high, the value of the probability is high.

∴ The chances that s_n is set as s_c is also high.

Hence, At high temperatures, the algorithm behaves like a random search algorithm.

---

c) In linear normalization, we normalize the fitness values into a range of max to min (say 10 to 100).

In roulette wheel selection, however, we select the new generation probabilistically with greater preference to higher fitness values.

As such, it is possible to lose variety via this probabilistic selection process.

In linear norm, we can control the degree of variety in the population as the selection is on us.

9. Perceptron :-

An artificial neurone is a system that tries to mimic a biological neurone. It takes inputs from its surroundings ~~or~~ processes it in its body and spits the output to many other neurones that may require it.

A single neurone connected by weights to a set of inputs producing a single output is known as a perceptron.

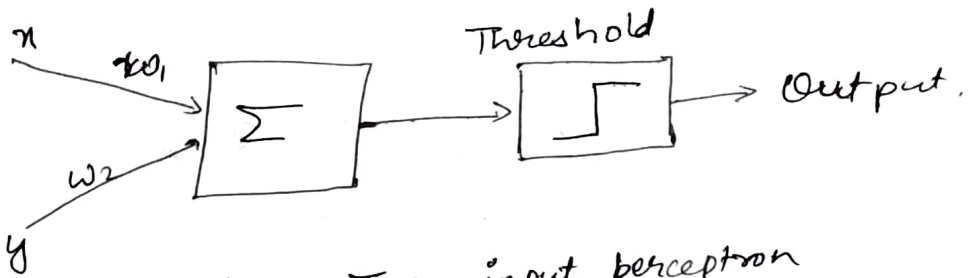


Fig 9.1 - Two input perceptron

→ Let x & y be two inputs and w_1, w_2 be the weights.

→ If $w_1 x + w_2 y > \theta$, then the output is 1 else 0, where

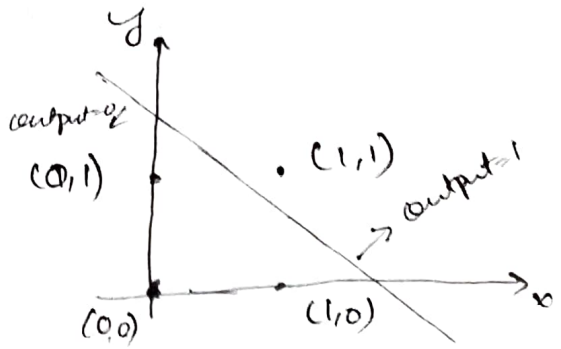
θ = threshold

→ $w_1 x + w_2 y = \theta$ is the separating line.

For an AND function, we have

x	y	output
0	1	0
1	0	0
1	1	1
0	0	0

In order to use a perceptron as an AND function, we must select θ such that the separating line comes as shown in the figure.



Considering, $w_1 = 1$ & $w_2 = 1$

We have,

$$w_1 x + w_2 y = \theta$$

$$x + y = \theta$$

For AND function, we must have.

$1+1 > \theta$ (truth values)

$0+1 < \theta$
$1+0 < \theta$ {false values}
$0+0 < \theta$

Hence, selecting $1 < \theta < 2$ and $w_1 = 1$, $w_2 = 1$, AND function is modelled.

One possible solution,

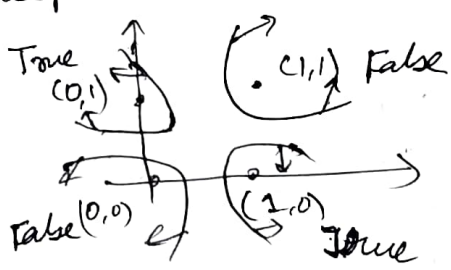
$\theta = 1.5$

The separating line, $x + y = 1.5$

Limitations of a single layer perceptron

A single layer perceptron can only be used in cases where the resultant classes are linearly separable by a hyperplane.

Hence, functions like XOR are not implementable by a single layer perceptron.



(XOR Function)

7. a) The predicates are :-

$S_0 = \{ g(x,y), h(x,y) \}, \{ g(z,u), h(w,u) \}, \{ g(t,t), h(v, f(v)) \}$

Disagreement set : $\{x, z, t\}$

$\sigma = \{ z/x \}$

$S_1 = \{ g(z,y), h(z,y) \}, \{ g(z,u), h(w,u) \}, \{ g(t,t), h(v, f(v)) \}$

Disagreement set : $\{z, t\}$

$\sigma = \{ z/t \}$

$S_2 = \{ g(z,y), h(z,y) \}, \{ g(z,u), h(w,u) \}, \{ g(z,z), h(v, f(v)) \}$

Disagreement set : $\{y, u, z\}$

$\sigma = \{ y/u \}$

$S_3 = \{ g(z,y), h(z,y) \}, \{ g(z,y), h(w,y) \}, \{ g(z,z), h(v, f(v)) \}$

Disagreement set : $\{y, z\}$

$\sigma = \{ y/z \}$

$S_3 = \{ g(y,y), h(y,y) \}, \{ g(y,y), h(w,y) \}, \{ g(y,y), h(v, f(v)) \}$

Disagreement set : $\{y, w, v\}$

$\sigma = \{ y/w \}$

$S_4 = \{ g(y,y), h(y,y) \}, \{ g(y,y), h(y,y) \}, \{ g(y,y), h(v, f(v)) \}$

Disagreement set : $\{y, v\}$

$\sigma = \{ y/v \}$

E

$S_5 = \{g(y,y), h(y,y)\}, \{g(y,y), h(y,y)\}, \{g(y,y), h(y, f(y))\}$

Disagreement set = $\{y, f(y)\}$

Since, the Literals in the disagreement set consists of Similar variables or term, $f(y)$ consists y,

Hence, unification failed.

---

b) WFF

$\sim(\forall x)\{P(x) \to \{(\forall y)[P(y) \to P(f(x,y))] \wedge \sim(\forall y)[Q(x,y) \to P(y)]\}\}$

Eliminating ($\to$) implies :-

$\sim(\forall x)\{\sim P(x) \vee \{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge \sim(\forall y)[\sim Q(x,y) \vee P(y)]\}\}$

Negating quantifiers :-

$\exists x\{\sim\{\sim P(x) \vee \{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge \exists y \sim[\sim Q(x,y) \vee P(y)]\}\}$

$\equiv \exists x\{P(x) \wedge \sim\{(\forall y)[\sim P(y) \vee P(f(x,y))] \wedge \exists y[Q(x,y) \wedge \sim P(y)]\}\}$

$\equiv \exists x\{P(x) \wedge \{\sim(\forall y)[\sim P(y) \vee P(f(x,y))] \vee \sim(\exists y)[Q(x,y) \wedge \sim P(y)]\}\}$

$\equiv \exists x\{P(x) \wedge \{\exists y \sim[\sim P(y) \vee P(f(x,y))] \vee \forall y \sim[Q(x,y) \wedge \sim P(y)]\}\}$

$\equiv \exists x\{P(x) \wedge \{\exists y[P(y) \wedge \sim P(f(x,y))] \vee \forall z[\sim Q(x,z) \vee P(z)]\}\}$ (Renaming variables)

Removing quantifiers, Using skolem constant A in place of x & y & using $g(x)$ skolem function in place of y

$\equiv P(A) \wedge \{[P(g(x)) \wedge \sim P(f(A, g(x))] \vee [\sim Q(A,z) \vee P(z)]\}$

$\equiv P(A) \wedge (P(g(x)) \vee \sim Q(A,z) \vee P(z)) \wedge (\sim P(f(A,g(x))) \vee \sim Q(A,z) \vee P(z))$

(Using distribution laws)

∴ Clauses :-

1. $P(A)$
2. $P(g(x_2)) \vee \sim Q(A, z_2) \vee P(z_2)$
3. $\sim P(f(A, g(x_3))) \vee \sim Q(A, z_3) \vee P(z_3)$

---

c) ~~Prove:~~ Premises :-

i) $\forall x\ (p(x) \rightarrow q(x))$

$\equiv \forall x\ (\sim p(x) \vee q(x))$

$\equiv \sim p(x) \vee q(x)$

ii) $\forall x\ (q(x) \rightarrow r(x))$

$\equiv \forall x\ (\sim q(x) \vee r(x))$

$\equiv \sim q(x) \vee r(x)$

To prove : $(\forall x)\ (p(x) \Rightarrow r(x))$ ✱

~~negate~~ Negate the conclusion :- $\sim (\forall x)\ (p(x) \Rightarrow r(x))$

$\equiv \exists x \sim [\sim p(x) \vee r(x)]$

$\equiv p(x) \wedge \sim r(x)$

Clauses :

c_1 : $\sim p(x) \vee q(x)$

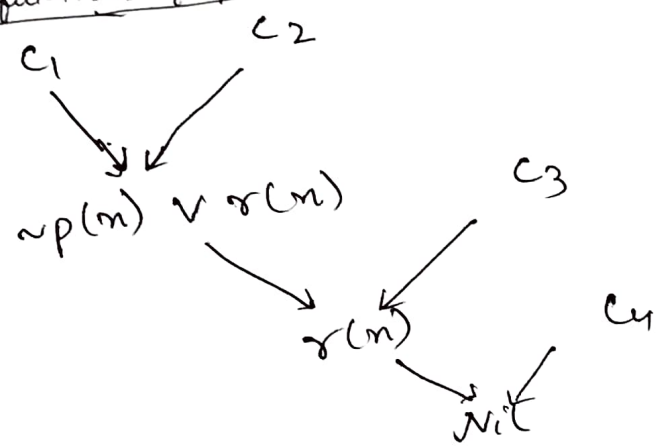
c_2 : $\sim q(x) \vee r(x)$

c_3 : $p(x)$

c_4 : $\sim r(x)$

~~c_5 (negate the conclusion)~~ :

Resolution Refutation Graph



Hence, proved.

d) Resolution refutation is sound and complete.

i) Hence, if a contradiction exists in the initial clause set, we can definitely reach a null state via refutation

ii) If no such contradiction exists, we will get all possible information and the algorithm will ultimately terminate as the set of resolvants is finite.