Ch : 3 Heuristic Search Techniques

DELUXE
PAGE NO.:
DATE:

→ "Heuristic is method that might not always find the best solution but is certain to find good solution in a reasonable time."

→ Heuristic Search is an AI search technique that employs heuristic for its moves. Heuristic is a rule of thumb that probably leads to a solution.

example: The travelling salesman problem is a famous touring problem in which each city in a network of connected cities must be visited exactly once. The goal is to find the shortest tour.

Heuristic Search Methods:

The main idea behind most search techniques is to preserve and expand a set of half done solution paths.

ex: 8 puzzle

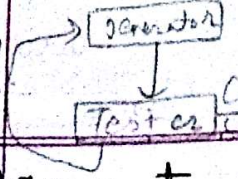
→ We have already discussed two very basic search strategies:

1) Depth first search
2) Breadth first search.

Some other Techniques:

1) generate and Test
2) Hill climbing
3) Best-first search
4) problem reduction
5) Constraint Satisfaction
6) Means-ends analysis

Teacher's Sign _____



1) (*) generate and Test : (No feedback)

Algorithm : Generate-and-Test

1. Generate a possible solution. For some problems this means generating a particular point in the problem space. For other, it means generating a path from a start state.

2. Test to see if this is actually a solution by comparing the chosen point or the endpoint of the chosen path to the set of acceptable goal states.

3. If a solution has been found, quit. Otherwise, return to step 1.

→ The generate and test algorithm is a depth first search procedure since complete ~~procedures~~ solutions must be generated before they can be tested.

(DFS with backtracking)

→ generate and test generating solutions randomly, but then there is no guarantee that a solution will ever be found.

Three forms of generate and test :

1) In its most systematic form, it is only an exhaustive search of problem space. (Systematic form)

2) Solution can also be generated Randomly but Solution is not guarnteed. also known as british Museum algorithm (pure random)

DATE:

Between these two, practical Approach in which the search process systematically but some paths are not consider because they seem unlikely to lead to a solution. This evaluation is performed by heuristic function

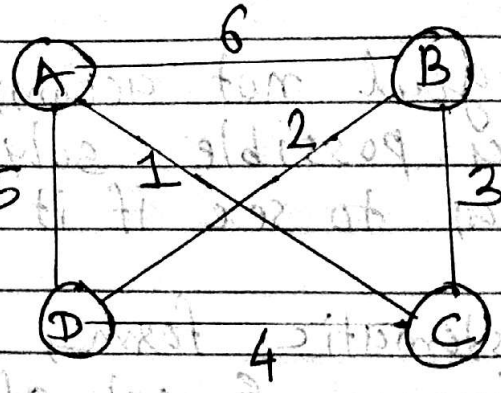
This algorithm works in following manner.

1) generator module : It is used to create the possible solution.

2) Tester Module : It tests each of the proposed solution either accepting or rejecting solution

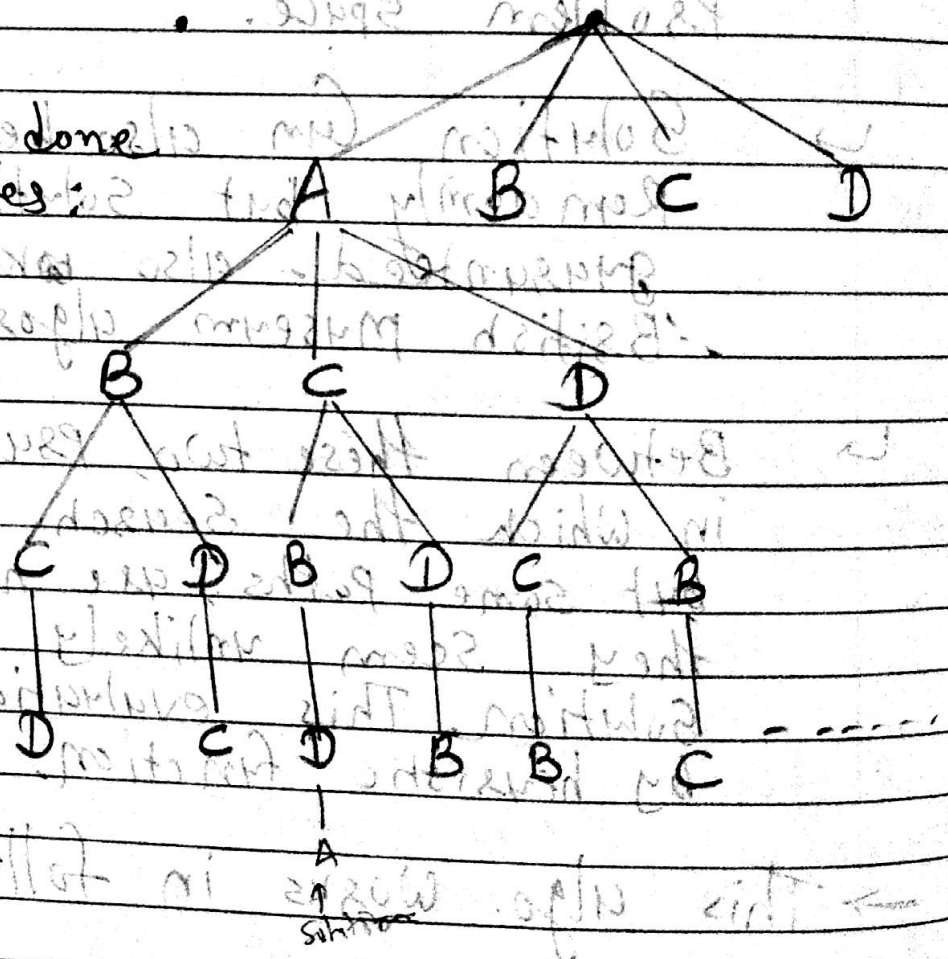
→ Example :- Traveling Salesman problem (TSP)

↳ Traveler want to know the shortest route that visits all the cities once.



→ TSP Possible is done in order of cities :

1. A-B-C-D
2. A-B-D-C
3. A-C-B-D
4. A-C-D-B



Disadvantage

- Expensive
- Solution may not be very optimal

DATE:

example:

Consider the puzzle that consists of four six-sided cubes, with each side of each cube painted one of four colours (R, B, G, W)

→ A solution to puzzle: Consists of an arrangement of the cubes in a row such that on all four sides of row one block face of each colour is showing.

→ This problem can be solved by a person in several minutes by systematically and exhaustively trying all possibilities.

→ It can be solved even more quickly using a heuristic generate and Test procedure

DELUXE
PAGE NO.:
DATE:

2) (*) Hill climbing :

Hill climbing is a variant of generate-and-test in which feedback from the test procedure is used to help the generator decide which direction to move in the search space.

→ In a pure generate and test procedure, the test function responds with only yes or no.

→ But if the test function is augmented with a heuristic function that provides an estimate of how close a given state is to a goal state.

→ Hill climbing can be used to solve problems that have many solutions.

→ Hill climbing is often used when a good heuristic function is available for evaluating states but when no other useful knowledge is available.

for example: Suppose you are in an unfamiliar city without a map and you want to get downtown you simply aim for tall buildings

The Heuristic function is just distance between the current location and the location of the tall buildings and the desirable states are those in which this distance is minimized.

Recall the question "Is a good solution absolute or relative?"

Absolute solutions exist whenever it is possible to recognize a goal state by examining it. Getting downtown is an example of such a problem. For these problems, Hill climbing can terminate whenever a goal state is reached.

Only relative solutions exist, however, for maximization or minimization problems, such as traveling salesman problem. In these problems there is no apriori goal state. For problem of this sort, it makes sense to terminate hill climbing when there is no reasonable alternative state to move to.

(*) Simple Hill climbing:

1) **Algorithm: Simple Hill climbing:**

1) Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise continue with the initial state as the current state.

2) Loop until a solution is found or until there are no new operators left to be applied in the current state. and

a) Select an operator that has not get been applied to current state and apply it to produce a new state.

b) Evaluate the new state.

i) If it is goal state, then return it and quit.

ii) If it is not a goal state but it is better than the current state, then make it the current state.

iii) If it is not better than the current state, then continue in the loop.

The key difference between this algorithm and the one we gave for generate and test is the use of an evaluation function as a way to inject task-specific knowledge into the control process.

In this algorithm, we have asked the question "Is one state better than another?"

For the algorithm to work, a precise definition of better must be provided. In some cases it means a higher value of the heuristic function. In others, it means a lower value.

ex: the puzzle of the four colored blocks: To solve the problem, we first need to define a heuristic function that describe how closed particular configuration is to being a solution.

→ one such function is simply the sum of the number of different colors on each of the four sides.

→ A solution to the puzzle will have a value of 16.

PAGE NO.:
DATE:

Rule: Simply pick a block and rotate it 90 degrees in any direction.

Now hill climbing can begin. We generate a new state by selecting a block and rotating it. If the resulting state is better, then we keep it. If not, we return to the previous state and try a different perturbation.

2) Steepest - Ascent Hill Climbing : Or

Gradient Search :

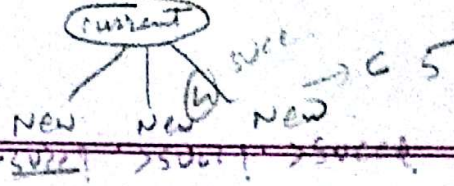
→ A useful variation on simple hill climbing considers all the moves from the current state and selects the best one as the next state.

Algorithm : Steepest - Ascent

(1) Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.

(2) loop until a solution is found or until a complete iteration produces no change to current state:

a) let SUCC be a state such that any possible successor(new) of the current state will be better than SUCC.



b) for each operator that applies to the current state do:

i) Apply the operator and generate a new state.

ii) Evaluate the new state. If it is a goal state, then return it and quit. If not, Compare it to succ. If it is better, then set succ to this state. If it is not better, leave succ alone.

c) If the succ is better than current state, then set current state to succ.

ex: the puzzle of four colored blocks: we must Consider all Perturbations of the initial state and choose the best. for this problem, this is difficult since there are so many possible moves.

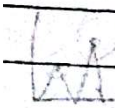
There is a tradoff between the time required to select a move (usually longer for steepest-ascent hill climbing) and the number of moves required to get to a solution (usually longer for basic hill climbing) that must be Considered when deciding Which method will Work better for a particular problem.

Both Simple and steepest-ascent Hill climbing may fail to find a Solution. Either algorithm may terminate not by finding a goal state but by getting to a state from which no better states can be generated. This

DELUXE
PAGE NO.:
DATE:

will happen if the program has reached either a local maximum, a Plateau, or a Ridge.

A local maximum: is a state that is better than all its neighbors but is not better than some other states farther away. At a local maximum, all moves appear to make things worse.



A plateau: is a flat area of the search space in which a whole set of neighboring states have the same value. On a plateau, it is not possible to determine the best direction in which to move by making local comparisions.

A ridge: is a Special kind of local maximum. It is an area of the search space that is higher than surrounding areas and that itself has a slope.

but it can not be reached in a single move.

→ where there are steep slopes and the search direction is not towards the top but towards the side.

Solution:

There are some ways of dealing with these problems.

Solution of local maximum:

Backtrack to some earlier node and try going in a different direction.

→ This is particularly reasonable if at that node there was another direction that looked as promising as almost as promising as the one that was chosen earlier.

→ To implement this strategy, maintain a list of paths almost taken and go back to one of them if the path that was taken leads to a dead end.

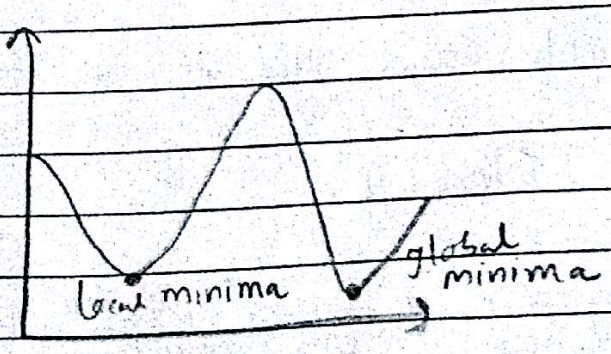
2) Solution of Plateau:

Make a big jump in some direction to try to get to a new section of the search space. This is a particularly good way of dealing with plateaus. If the only rules available describe single small steps, apply them several times in the same direction.

3) Solution of Ridges:

Apply two or more rules before doing the test. This corresponds to moving in several directions at once. This is particularly good strategy for dealing with ridges.

→ Hill climbing is a greedy algorithm.



Hill climbing is a local method. by which we mean that it decides what to do next by looking only at the "immediate" Consequences of its choice rather than by exhaustively exploring all the Consequences.

Example : Block World problem using hill climbing Technique.

A	H
H	G
G	F
F	E
E	D
D	C
C	B
B	A

Initial State Goal State

[fig : 1 Hill climbing problem]

→ Consider the ~~B~~ Blocks World problem shown in fig : 1

Operators :

1) Pick up one block and put it on the table.

2) Pick up one block and put it on another one.

Heuristic function : (local)

Add one point for every block that is resting on the thing it is supposed to be resting on.

DATE:

Substract one point for every block that is sitting on the Wrong thing.

A	-1
H	+1
G	+1
F	+1
E	+1
D	+1
C	+1
B	-1

Initial state

H	+1
G	+1
F	+1
E	+1
D	+1
C	+1
B	+1
A	+1

goal State

→ Using this function, the goal state has a Score of 8.

$h_1(G) = 8$

The Initial state has a Score of 4.

$h_1(I) = 6 - 2 = 4$

(One Point added for blocks **C, D, E, F, G and H** and one point Subtracted for blocks **A and B.**)

→ There is only one move from the initial state, namely to move block A to the table.

Let this State be "A"

$\therefore h_1(A) = 7 - 1 = 6$

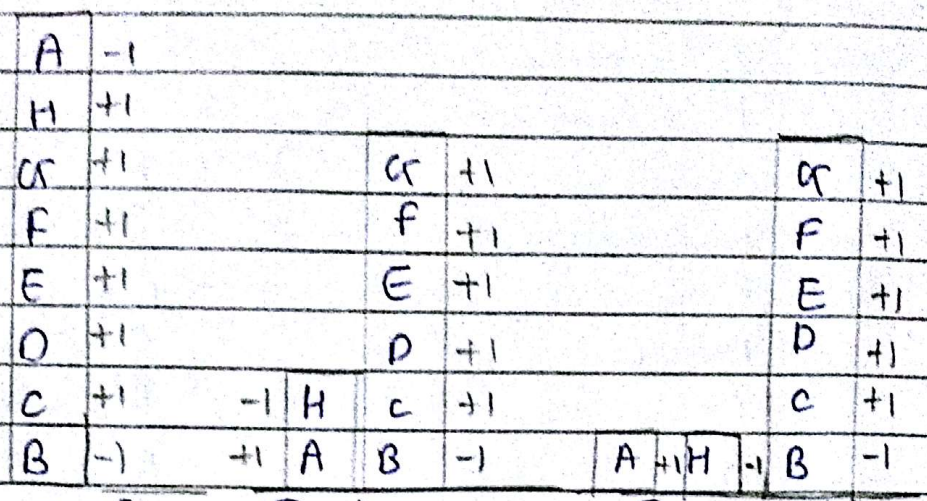
H	+1
G	+1
F	+1
E	+1
D	+1
C	+1
B	-1
A	+1

fig: 3

Teacher's Sign

DELUXE
PAGE NO.:
DATE:

→ Now from the new state, there are three possible moves, which is shown in fig: 4



[fig: 4 "Three possible moves"]

Three possible moves:

1. ("B") Pick up block A and put it back on H

2. ("c") Pick up block H and Put it over A.

3. ("D") Pick up block H and put it on Table

local maximum

$H_1(B) = 6 - 2 = 4$

$H_1(C) = 6 - 2 = 4$

$H_1(D) = 6 - 2 = 4$

→ Hill climbing will halt because all these states have lower scores than the current state

→ The process has reached a local maximum that is not the global maximum.

DELUXE
PAGE NO.:
DATE:

→ Modify Heuristic function and try it. Suppose we try the following heuristic function in place of the first one.

⇒ New Heuristic function: (global)
for each block that has the correct support structure (i.e. the complete structure underneath it is exactly as it should be), add 1 point for every block in the structure

→ for each block that has an incorrect support structure, subtract one point for every block in the existing support structure.

	A	-7		H	7
	H	-6		G	6
	G	-5		F	5
	F	-4		E	4
—	E	-3		D	3
—	D	-2		C	2
—	C	-1		B	1
—	B	0		A	0

Initial state: [fig: 5]

Goal state (G)

$H_2(I) = 0 + (-1) + (-2) + (-3) + (-4) + (-5) + (-6) + (-7)$

$H_2(I) = -28$

$H_2(G) = 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7$

$H_2(G) = 28$

There is only one move, from the initial state, putting block 'A' on the table.

Teacher's Sign

$H_2(A) = 0 + (-1) + (-2) + (-3) + (-4) + (-5) + (-1) + 0$

$= -21$

		H	-6
		G	-5
		F	-4
		E	-3
		D	-2
		C	-1
A	0	B	0

[fig: 6 first move from initial state called "A"]

Now, There are three possible moves. which is shown in fig: 7.

	A	-7													
	H	-6													
	G	-5					G	-5					G	-5	
	F	-4					F	-4					F	-4	
	E	-3					E	-3					E	-3	
	D	-2					D	-2					D	-2	
	C	-1	-1	H			C	-1					C	-1	
	B	0	0	A			B	0	A	0	H	0	B	0	

① "B" ② "C" ③ "O"

$H_2(B) = 0 + (-1) + (-2) + (-3) + (-4) + (-5) + (-6) + (-7) = -28$

$H_2(C) = 0 + (-1) + (-2) + (-3) + (-4) + (-5) + 0 + (-1) = -16$ →

$H_2(D) = 0 + (-1) + (-2) + (-3) + (-4) + (-5) + 0 + 0 = -15$

PAGE NO.:
DATE:

→ 3rd move (D) is chosen, which is correct one.

→ Here the global information is stored in the heuristic function, so Hill climbing is still effective.

→ It is hard to construct perfect heuristic function like previous one.

→ Computation cost may be high.

DELUXE
PAGE NO.:
DATE:

3) (*) Simulated Annealing:

Simulated Annealing is a variation of hill climbing in which at the beginning of the process, some downhill moves may be made.

The idea is to do enough exploration of the whole space early on so that the final solution is relatively insensitive to the starting state.

This should lower the chances of getting at a local maximum, a plateau or a ridge.

There are two notational changes in Simulated Annealing:

1) Use the term "objective function" in place of the term "Heuristic function"

2) We attempt to minimize rather than maximize the value of the objective function.

→ Thus we actually describe the process of valley descending rather than hill climbing.

In physical Annealing, physical substances such as metals are melted and then gradually cooled until some solid state is reached.

The goal of this process is to produce a minimal energy final state. Thus this process is one of the valley descending in which the objective function is the energy level.

Teacher's Sign

DELUXE
PAGE NO.:
DATE:

→ Physical Substances usually move from higher energy configurations to lower ones. So the valley descending occurs naturally.

→ But there is some probability that a transition to a higher energy state will occur. The probability is given by the function.

Probability of Accepting $\Rightarrow P = e^{-\Delta E/kT}$

Worst case

Where,

ΔE = change in energy ($E_2 - E_1$)

T = Temperature

k = Boltzmann's Constant

→ The Temperature increases the probability of selecting worst case increase.

$T \uparrow \rightarrow P \uparrow$

→ The rate at which the system is cooled is called the Annealing Schedule.

→ These properties of physical Annealing can be used to define an analogous process of Simulated annealing.

analogous process

→ In this analogous process. ΔE is generalized so that it represent not specifically the change in energy but more generally the change in the value of the objective function.

(ΔE = Change in value of objective function)

DELUXE
PAGE NO.:
DATE:

In the physical process, temperature is a well defined notion, measured in standard units.

The variable k describes the correspondence between the units of temperature and the units of energy.

Since in the analogous process, the units for both E and T are artificial, it makes sense to incorporate k into T.

Thus we use revised probability formula

probability $p' = e^{-\Delta E/T}$

of Accepting

worst case.

Algorithm : Simulated Annealing :

Evaluate the initial state. If it is a goal state then return it and quit. otherwise, Continue with initial state as current state.

Initialize BEST-SO-FAR to the current state

Initialize T according to the annealing Schedule.

loop until a Solution is found or until there are no new operators left to be applied in the current state.

Teacher's Sign

a) Select an operator that has not yet been applied to the current state and apply it to produce a new state.

b) Evaluate the new state. Compute

ΔE = (value of current) − (value of new state)

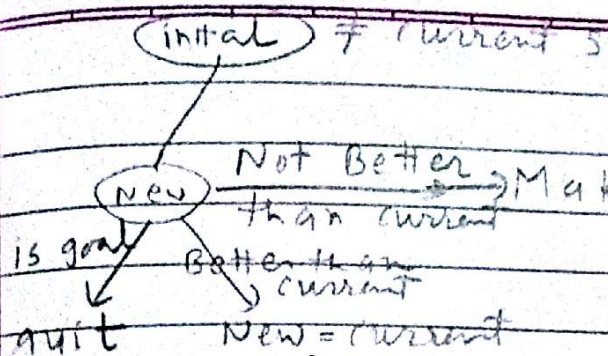
i) If the new state is a goal state, then return it and quit.

ii) If it is not a goal state but is better than current state, then make it the current state. Also set BEST-SO-FAR to this state.

iii) If it is not better than current state, then make it the current state with probability p' as defined above. This step is usually implemented by invoking a random number generator to produce a number in the range [0, 1]. If that number is less than p', then the move is accepted, otherwise do nothing.

c) Revise T as necessary according to the annealing Schedule.

5 Return BEST-SO-FAR, as the answer.



final value of p' and compare with random number between [0,1]

[0,1] < p' → Move Accepted 0.5 < 0.7 ✓

[0,1] > p' → Do Nothing 0.5 > 0.2 ✗

To implement this revised algorithm, it is necessary to select an annealing schedule, which has three components,

1) Initial value to be used for temperature.

2) The criteria that will be used to decide when the temperature of the system should be reduced.

3) The amount by which the temperature will be reduced each time it is changed

4) When to quit.

Simulated Annealing is often used to solve problems in which the number of moves from a given state is very large.

4) (*) BEST-FIRST SEARCH:

Best first search, which is a way of combining the advantages of both depth first search into a single method.

→ OR Graphs:

→ DFS is good because: it allows a solution to be found without all competing branches having to be expanded.

→ BFS is good because: it does not get trapped on dead-end paths.

→ one way of combining the two is to follow a single path at a time, but switch paths whenever some competing path looks more promising than the current one does.

→ At each step of best first search process, we select most promising of the nodes we have generated so far. This is done by applying an appropriate heuristic function to each of them.

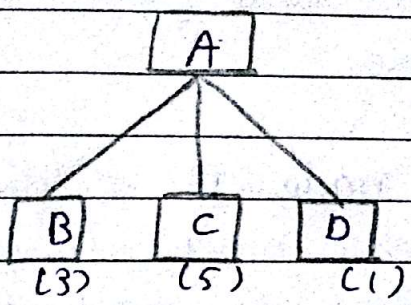
→ We then expand the chosen node by using the rules to generate its successors. If one of them is a solution, we can quit. If not, all those new nodes are added to the set of nodes generated so far. Again the most promising node is selected and the process continues.

Step : 1

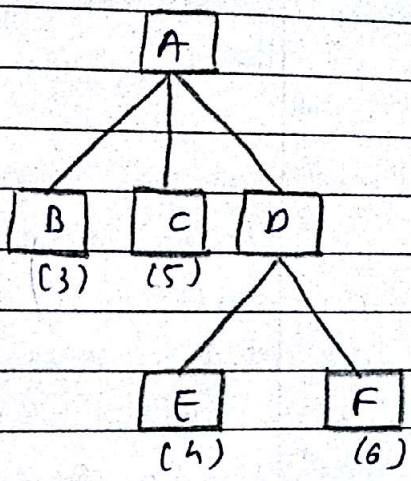
A

	Open	Closed
Step 1	A	-
Step 2	B, C, D	A
Step 3	B, C, E, F	A, D
Step 4	C, E, F, G, H	A, B, D
Step 5	C, E, G, H, I, J	A, B, D, E

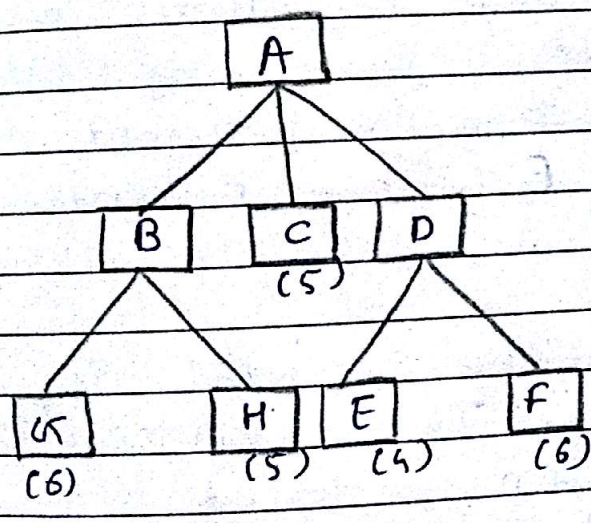
Step : 2



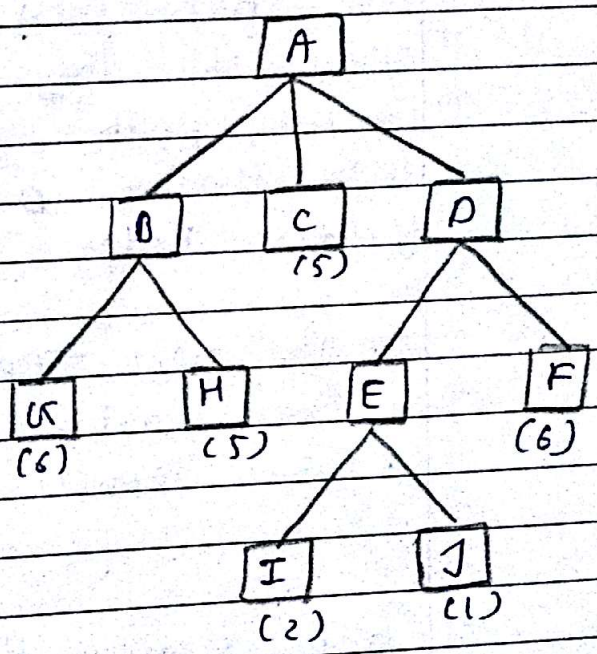
Step : 3



Step : 4



Step : 5



[fig 8 : Best first Search]

fig 8 :

Step: 1 Initially, there is only one node. So it will be expanded

Step: 2 Doing So generates three new nodes. The heuristic function in this example, is

an estimate of the cost getting to a solution from a given node, is applied to each of these new nodes.

Step: 3 Since node D is the most promising, it is expanded next, producing two successor nodes E and F. But then the heuristic function is applied to them.

Step: 4 another path, that going through node B, looks more promising. So it is expanded, generating nodes G and H.

Step 5 But again when these new nodes are evaluated they look less promising than another path. So attention is returned to the path through D to E. E is then expanded, yielding nodes I and J.

At the next step, J will be expanded, since it is the most promising. This process can continue until a solution is found.

Comparision of Best first Search with Hill climbing:

1) In Hill climbing the best state is selected and the remaining are forgotten (rejected)

→ Here All states are recorded (revisited) later if the selected path becomes less promising.

2) Hill climbing is greedy algorithm. It always

looks for a state better than the current state. If such state is not found, it terminates.

→ In Best first search, A worser than current state can also be visited.

OR graph: The graph in which each branches represents an alternative problem solving path is called OR-graph.

To implement graph search procedure, we will need to use two list of nodes.

1) OPEN : (list of open):
→ It contains nodes that have been generated and have had the heuristic function applied to them but which have not yet been examined. expanded.

→ OPEN is actually a priority queue in which the elements with the highest priority are those with the most promising value of the heuristic function.

2) CLOSED : (list of closed)
→ It contains nodes that have been expanded. It is required for searching a graph rather than a tree.

→ Since whenever a new node is generated, we need to check whether it has been generated before.

DELUXE
PAGE NO.:
DATE:

Algorithm: Best first Search.

1. Start with OPEN containing just the initial state.

2. Until a goal is found or there are no nodes left on OPEN do:

a) Pick them best node on OPEN.
b) generate its successors.
c) for each successor do:
   i) If it has not been generated before, evaluate it, add it to OPEN, and record its parent.

   ii) If it has been generated before, change the parent if this new path is better than the previous one. In that case, update the cost of getting to this node and to any successors that this node may already have.

DATE:

A Heuristic function $f' = g + h'$

$f' = g + h'$

Where f' = estimated cost from Initial state to goal state along the path that generated the current node.

g : Cost of getting from the Initial state to current node.

h' : estimate of additional cost of getting from current node to a goal state.

The actual operation of the algorithm proceeds in steps, expanding one node at each step, until it generates a node that corresponds to a goal state.

At each step, it picks the most promising of the nodes that have so far been generated but not expanded.

It generates the successors of the chosen node, applies the heuristic function to them and adds them to the list of nodes after checking to see if any of them have been generated before.

By doing this check, we can guarantee that each node only appears once in the graph.

(*) The A* Algorithm:

Best first search is a simplification of an A* algorithm. This algorithm uses the same f', g, and h' functions.

Algorithm: A*

(1) Start with OPEN containing only the initial node. Set that node's g value to 0, its h' value to whatever it is, and its f' value to h' + 0, or h'. Set CLOSED to the empty list.

(2) Until a goal node is found, repeat the following procedure:

If there are no nodes on OPEN, report failure

Otherwise, pick the node on OPEN with the lowest f' value. Call it BESTNODE. Remove it from OPEN.

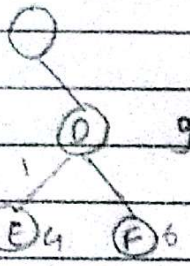
If BESTNODE is a goal node, exit and report a solution

otherwise generate the successors of BESTNODE but do not set BESTNODE to point to them yet.

for each such SUCCESSOR do the following:

a) Set SUCCESSOR to point back to BESTNODE These backwards links will make it impossible to recover the path once a solution is found

b) Compute g(SUCCESSOR) = g(BESTNODE) + the cost of getting from BESTNODE to SUCCESSOR.



c) See if SUCCESSOR is same as any node on OPEN. If so, call that node OLD. Since this node already exists in the graph, we can throw SUCCESSOR away and add OLD to the list of BESTNODE's successors.

Teacher's Sign

DELUXE
PAGE NO.:
DATE:

Now we must decide whether OLD's parent link should be reset to point to BEST NODE. It should be if the path we have just found to SUCCESSOR is cheaper than the current best path to OLD.

c) If OLD is cheaper, then we need do nothing.

If SUCCESSOR is cheaper, then reset OLD's parent link to point to BESTNODE, record the now cheaper path in g(OLD) and update f'(OLD).

d) If SUCCESSOR was not ON OPEN, see if it is on CLOSED. If so, call the node on CLOSED OLD and add OLD to the list of BESTNODE'S Successors. Check to see if new path or the OLD path is better and set the parent link and g and f' values appropriately.

e) If SUCCESSOR was not already on either OPEN OR CLOSED, then put it on OPEN and add it to the list of BESTNODE'S Successors. Compute f'(SUCCESSOR) = g(SUCCESSOR) + h'(SUCCESSOR).

Teacher's Sign _____

Example : 8 - puzzle problem :

$\hat{h_1}$ = number of tiles out of place with respect to goal state.

Initial state

2	8	3
1	6	4
7		5

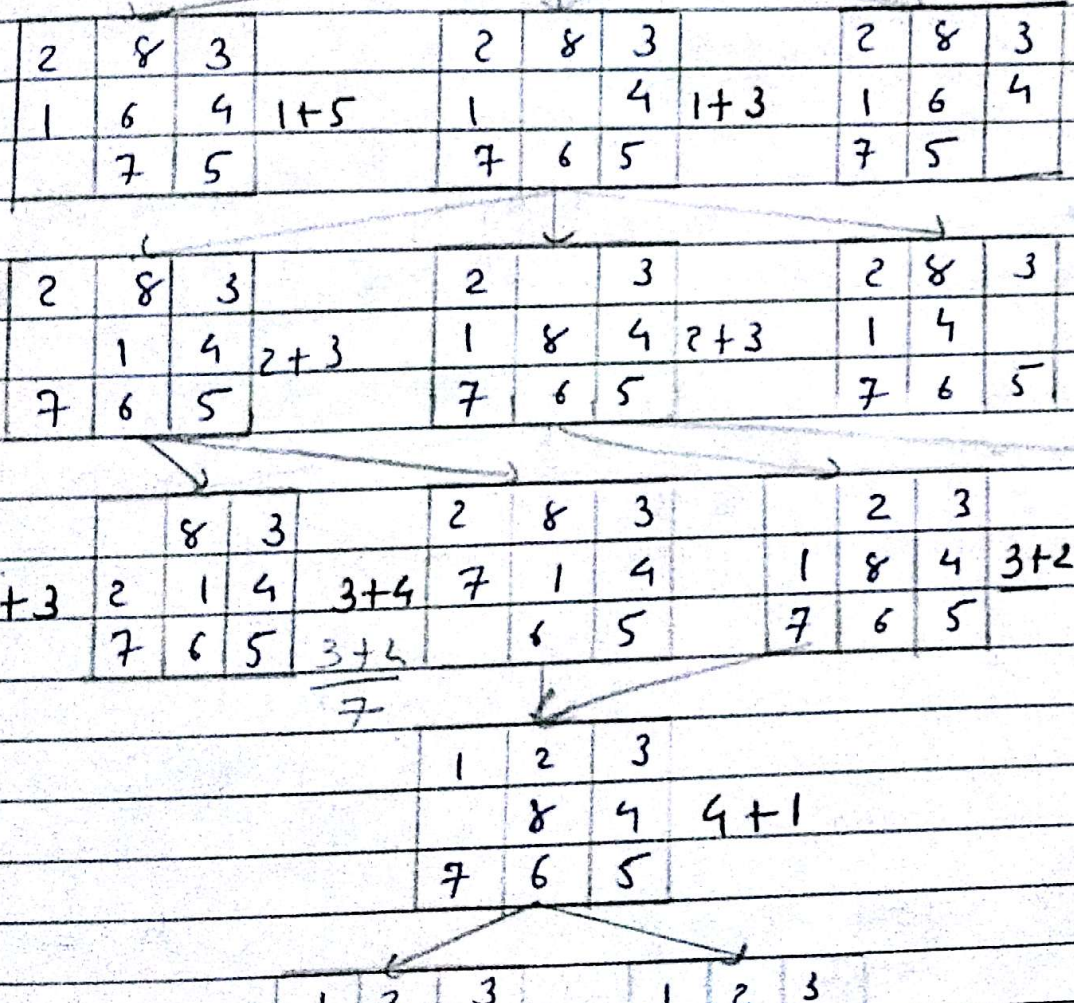
goal state

1	2	3
8		4
7	6	5

$f^1 = g + h^1$
$= 0 + 4$

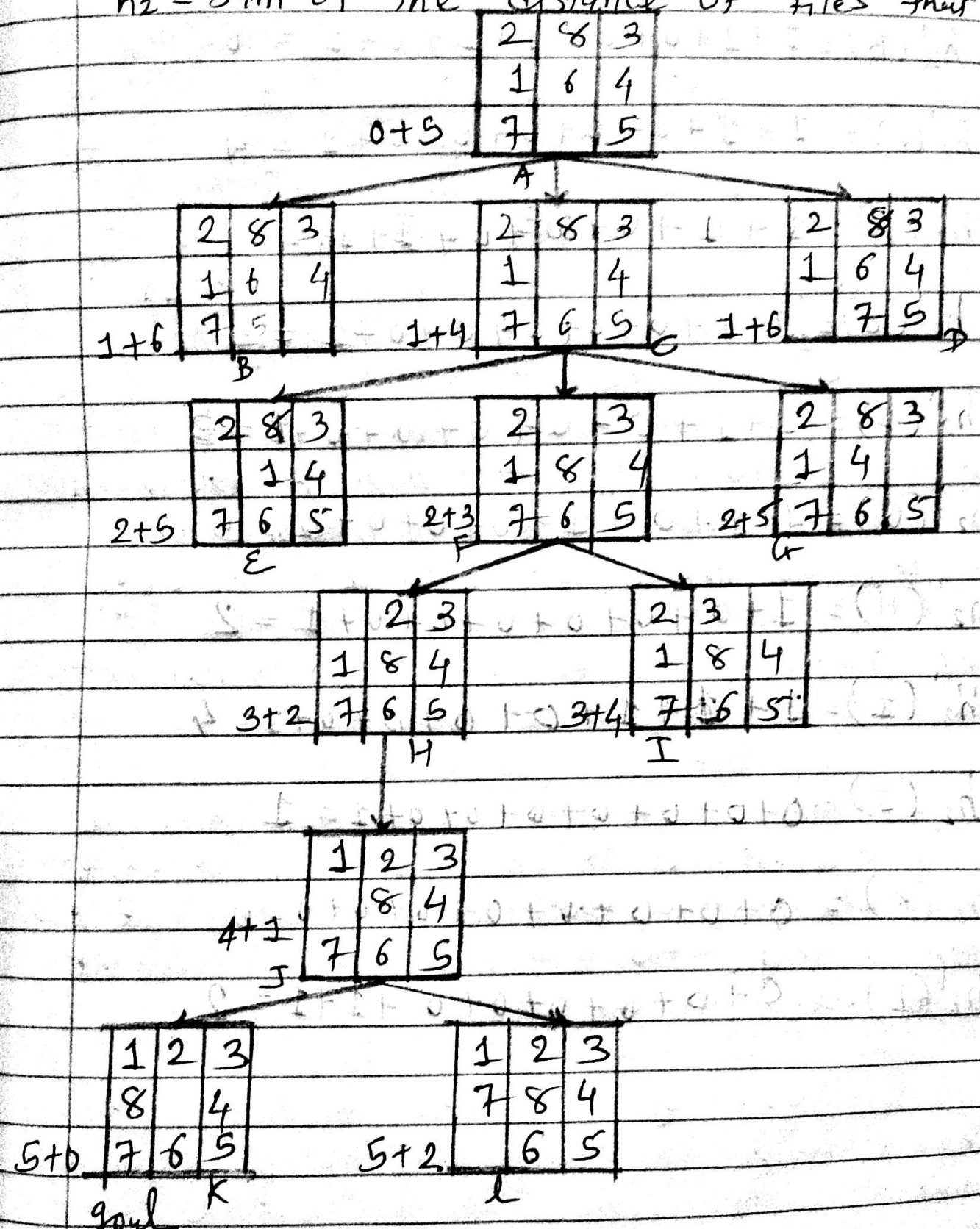
2	8	3
1	6	4
7		5

1+1+1+2+1+1 = 6



↳ Now Second Heyristic function

$\hat{h_2}$ = Sum of the distance of tiles that are out of place.



↳ $h_2'(A) = 1+1+0+0+0+1+0+2 = 5$

$h_2'(B) = 1+1+0+0+1+1+0+2 = 6$

$h_2'(C) = 1+1+0+0+0+0+0+2 = 4$

$h_2'(D) = 1+1+0+0+0+1+1+2 = 6$

$h_2'(E) = 2+1+0+0+0+0+0+2 = 5$

$h_2'(F) = 1+1+0+0+0+0+0+1 = 3$

$h_2'(G) = 1+1+0+1+0+0+0+2 = 5$

$h_2'(H) = 1+0+0+0+0+0+0+1 = 2$

$h_2'(I) = 1+1+1+0+0+0+0+1 = 4$

$h_2'(J) = 0+0+0+0+0+0+0+1 = 1$

$h_2'(K) = 0+0+0+0+0+0+0+0 = 0$

$h_2'(L) = 0+0+0+0+0+0+1+1 = 2$

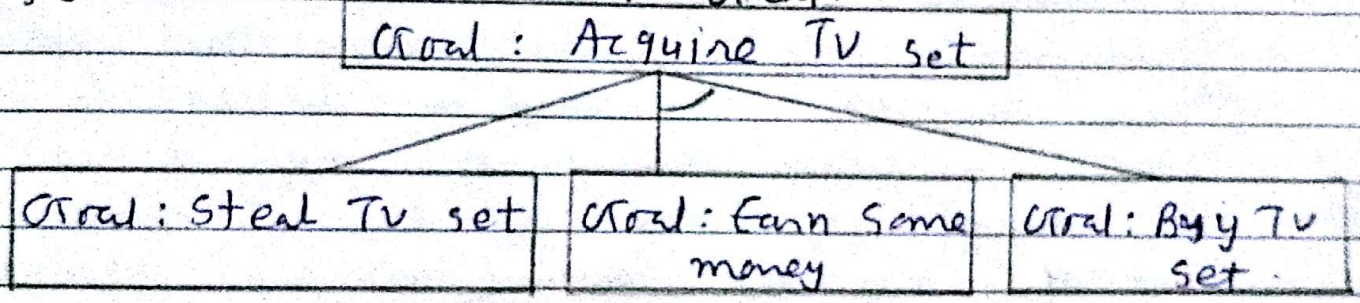
Problem Reduction:

AND - OR Graphs:

The AND-OR Graph (or tree) is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems, all of which must then be solved.

This decomposition, or reduction, generates arcs that we call AND arcs. One AND arc may point to any number of successor nodes.

fig: 11 Simple AND-OR Graph.



This algorithm should find a path from the starting node of the graph to a set of nodes representing solution states.

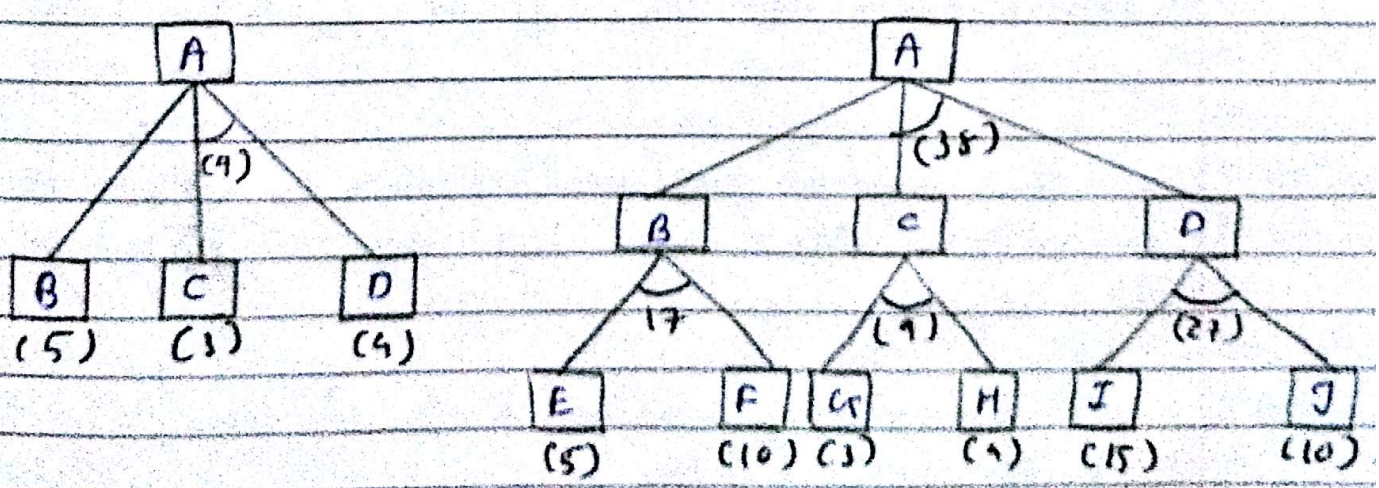


fig: 12 AND-OR graph.

Teacher's Sign

PAGE NO.:
DATE:

In fig 12 (a), the top node A has been expanded producing two arcs, one leading to B and one leading to C and D.

The numbers at each node represent the value of f' at that node.

We assume that every operation has a uniform cost, so each arc with a single successor has a cost of 1 and each AND arc with multiple successors has a cost of 1 for each of its components.

If we look at the nodes and choose for expansion the one with the lowest f' value, we must select C. But using the information now available it would be better to explore the path going through B since to use C we must also use D for a total cost of $9\ (\overset{3+}{C}+\overset{4+2}{D}+c)$ Compare to the cost of $6\ (\underset{5+1}{B}+1)$ that we get by going through B

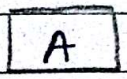
fig 12(b) The most promising single node is G with the f' value of 3. It is even part of the most promising arc G-H with a total cost of 9.

But that arc is not best path because to use it we must also use the arc I-J with a cost of 27.

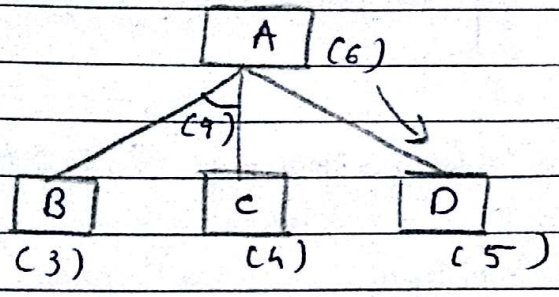
path from A, through B to E and F is better with a total cost of 18.

The Problem Reduction Process is illustrated in fig-13

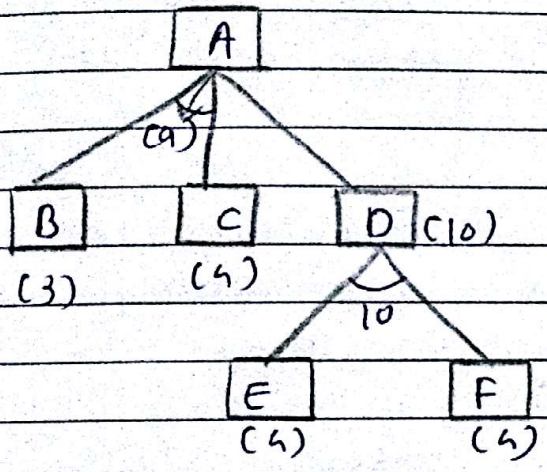
Step : 1



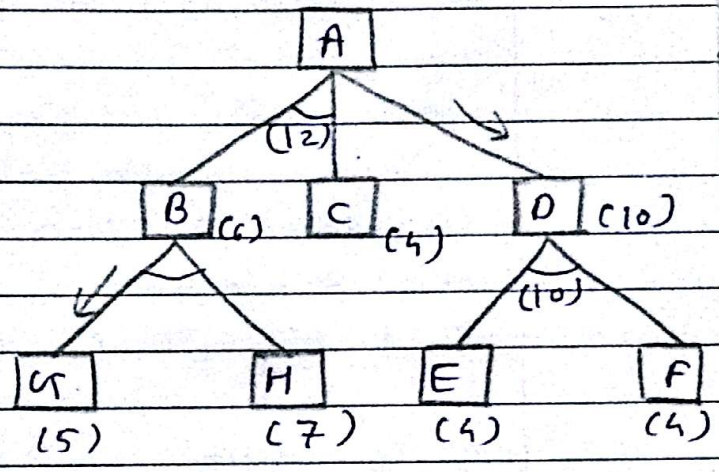
Step : 2



Step : 3



Step : 4



In fig 13 :

→ At step 1, A is the only node, so it is at the end of the current best path. It is expanded, yielding nodes B, C and D.

→ The arc to D is labeled as the most promising one emerging from A, Since it costs 6 Compared to B and C which costs 9.

→ In step 2, Node D is chosen for expansion. This process Produces one new Arc. the AND arc to E and F with a Combined Cost estimate of 10.

→ So, we updates the f' value of D to 10.
going back one more level, we see that
this makes the AND arc B - C better
than the arc to D. So it is labeled as
current best path.

↳ At Step 4, We will have to expand Both B and C eventually, so let's choose to explore B first.

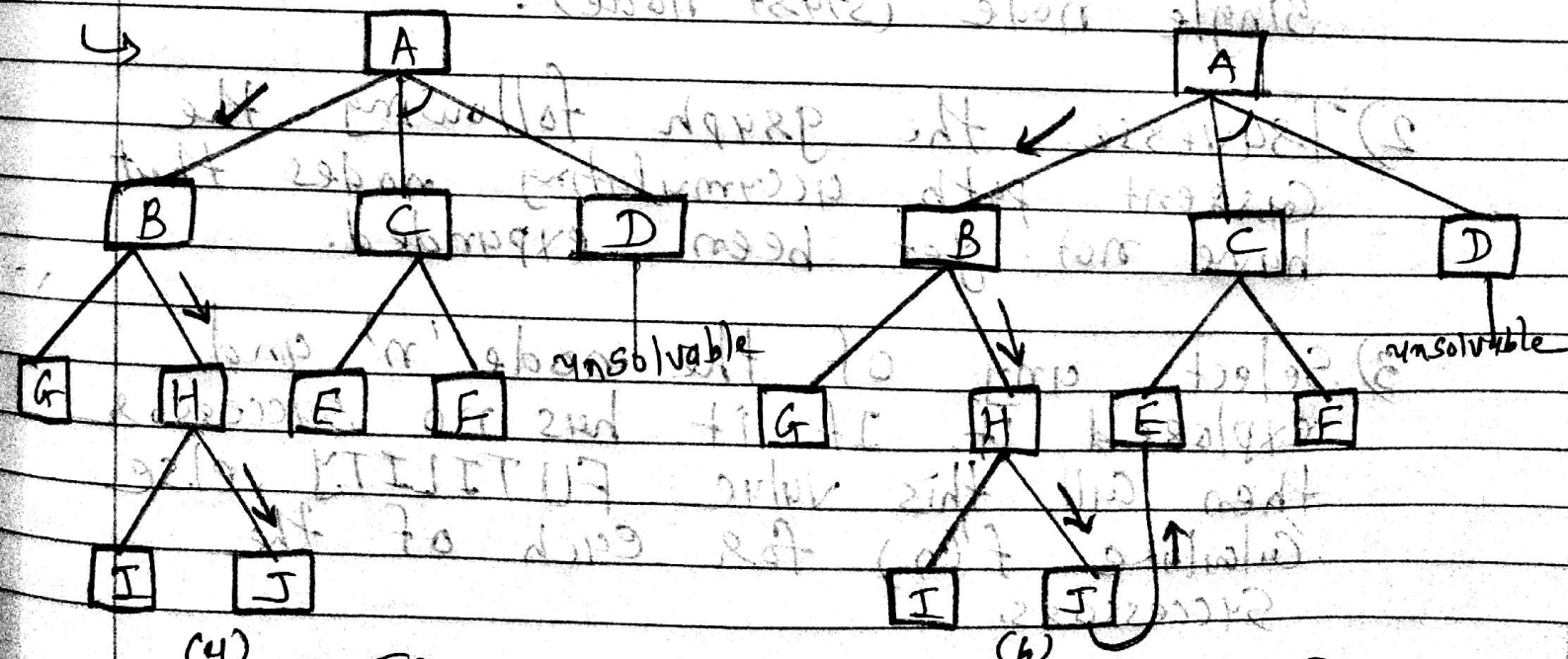
↳ This generates two new nodes G and H. Propagating their f' value backward, we updates f' of B to 6. This requires updating the Cost of the AND arc B-C to 12 (6+4+2).

↳ After doing that, the arc to D is again the better path from A.

↳ In next step we will expand node E or Node F. this process continues until either a solution is found or all path have led to dead ends

⇒ A longer Path may Be Better ?

↳



[fig.14, A longer Path May be Better]

Date

↳ Consider the example shown in fig-14(a) the nodes were generated in alphabetical order. Now suppose that node J is expanded at the next step and that one of its successor is Node E, producing the graph shown in fig (b).

↳ This new path to E is longer than the previous path to E going through C. But since the path through C will only lead a solution if there is also a solution to D, which we know there is not, the path through J is better.

↳ Algorithm: AO*

1) Place the initial graph on open with single node (start node).

2) Traverse the graph following the current path accumulating nodes that have not yet been expanded.

3) Select any of the node 'n' and explored it. If it has no successor then call this value FUTILITY, else calculate f'(n) for each of the successors.

4) If $f'(n)=0$ then mark the node as SOLVED.

5) Change the value of $f'(n)$ for the newly created node to reflect its successors by back propagation.

6) Whenever possible use the most promising routes.

7) If the starting node is SOLVED or value is greater than FUTILITY then stop, else repeat from step 2.

↳ In order to describe an algorithm for searching an AND-OR graph, we need to exploit a value that we call FUTILITY.

↳ FUTILITY should be chosen to correspond to a threshold.

↳ If the estimated cost of a solution becomes greater than the value of FUTILITY, then we abandon the search.

5) (*) Constraint Satisfaction:

→ Many problems in AI can be viewed as problems of Constraint satisfaction in which the goal is to discover some problem state that satisfies a given set of Constraint

→ example of this sort of problem include crypt arithmetic puzzles and many real world perceptual labeling problems.

→ In Constraint Satisfaction problems design must be created within fixed limit on time, cost and materials.

Crypt arithmetic:

In crypt arithmetic Assign a decimal digit to each of the letters in such a way. that the answer to the problem is correct. If same letter occur more than once, it must be assigned the same digit each time.

Teacher's Sign

Algorithm : Constraint Satisfaction

1) Propagate available Constraints. To do this, first set OPEN to the set of all objects that must have values assigned to them in a Complete Solution. Then do until an inconsistency is detected or until OPEN is empty.
   (a) Select an object OB from OPEN. Strengthen as much as possible the set of Constraints that apply to OB.
   (b) If this set is different from the set that was assigned the last time OB was examined or if this is the first time OB has been examined, then add to OPEN all objects that share any Constraints with OB.
   (c) Remove OB from OPEN.
2) If the union of the Constraints discovered above defined a solution, then quit & report the solution.
3) If the union of the Constraints discovered above defines a Contradiction, then return failure.
4) If neither of the above occurs, then it is necessary to make a guess at & Something in order to proceed To do this,

loop until a solution is found or all possible solutions have been eliminated:

(a) Select an object whose value is not yet determined and select a way of strengthening the constraints on that object.

(b) Recursively invoke constraint satisfaction with the current set of constraints ~~set~~ augmented by the strengthening constraint just selected.

PAGE NO.:
DATE:

Cryptarithmetic examples:

$C_4\ C_3\ C_2\ C_1$

S E N D

\+

M O R E

M O N E Y

→ Initial state : No two letter have the same value

→ The sum of the digits must be as shown in the problem.

1) $\boxed{M = 1}$ Since two single digit numbers plus a carry can not total more than 19.

So $\boxed{M = 1}$ and $\boxed{C_4 = 1}$

2) $C_3 + S + M > 9$

$C_3 + S + 1 > 9$

$C_3 = 0$ $\qquad$ $C_3 = 1$

$S + 1 > 9$

$S > 8$

So $S = 9$

$S + 1 + 1 > 9$

$S + 2 > 9$

$S > 7$

So $S = 8$ or 9

So $\boxed{S = 8 \text{ or } 9}$

Teacher's Sign

PAGE NO.:
DATE:

3) If S = 8 then $C_3 = 1$

$C_3 + S + M = O + C_4$

$1 + 8 + 1 = 10$

C_4 9 4 O

or

$1 + 8 + 1 = O + 10$

$O = \emptyset$

So O must be $\emptyset$ $\boxed{O = \emptyset}$

If S = 9 then

$C_3 = 0$ / $C_3 = 1$

$C_3 + S + M = O + C_4$ $C_3 + S + M = O + C_4$

$0 + 9 + 1$ $1 + 9 + 1 = O + C_4$

$= 10$ $= 11$

C_4 O C_4 O

O = 1 Not possible because M = 1

4) $C_2 + E + O = N + C_3$

$C_3 = 0$ / $C_3 = 1$

$C_2 + E + \emptyset = N + O$ $C_2 + E + \emptyset = N + 10$

$C_2 = 0$ / $C_2 = 1$ $C_2 = 0$ / $C_2 = 1$

$E = N$ $C_2 + E = N$ $E = N + 10$ $E = N + 10 - 1$

Not possible $1 + E = N$ $E = N + 9$.

True.

C_3	C_2	1	0
	E	4	
	O	$\emptyset$	
	N	9	

C_3	C_2	1	0
	E		E
	O		$\emptyset$
	N		$\emptyset$

$N = \emptyset$

Not possible.

If E = 9 then Carry will be generated but if E = 9 then N = $\emptyset$ is not possible because O = $\emptyset$.

So $\boxed{C_2 = 1}$ and $\boxed{C_3 = 0}$

Teacher's Sign

5) $C_3 = 0$ $S_0 = S = 9$ According to step 2.

			1	C_4
S	E	N	D	
		9	E	N	D
M	O	R	E	
		1	0	R	E
M	O	N	E	Y
		1	0	N	E	Y

6) $N + R + C_1 = E + 10$

$E + 0 + 1 = N = E + 1 = N$

$N + R + C_1 = E + 10$ put $N = E + 1$

$E + 1 + R + C_1 = E + 10$

$R = 10 - 1 - C_1$

$R = 9 - C_1$

$C_1 = 0$ → $R = 9$ Not possible because $S = 9$

$C_1 = 1$ → $R = 8$ ✓

7) Now we have to proceed by guessing

→) $E = 2$ then $N = 3$ ✗

① C_1 Not possible

$D \rightarrow 8$ or 9 then total is > 10 because $R = 8$, $S = 9$.

	E	2
y	0	1 → x

Teacher's Sign _____

DELUXE
PAGE NO.:
DATE:

E = 3 then N = 4 ✗

D [7], 8, 9 not possible values.
E [3]
R S
y ∅

If put D = 7 then y = ∅ not possible because O = ∅

E = 4 then N = 5 ✗

D 6, 7, 8, 9
E 4, 4, R, S
y ∅, 1, X

If put D = 6 then y = ∅ not possible (O = ∅)
If put D = 7 then y = 1 not possible (M = 1)

E = 5 then N = 6 ✓

D 5 6, 7, 8, 9
E 5 5, 5, R, S
y ∅ 1, 2, X

If D = 5 then y = O not possible (O = ∅)
IF D = 6 then y = 1 not possible (M = 1)
If D = 7 then y = 2 possible ✓

So [D = 7], [E = 5], [y = 2]

N = E + 1
= 5 + 1 = 6
[N = 6]

DELUXE
PAGE NO.:
DATE:

All Constraints are satisfied:

$$\begin{array}{r} 1\,0\,1\,1 \\ 9\,5\,6\,7 \\ 1\,0\,8\,5 \\ \hline 1\,0\,6\,5\,2 \end{array}$$

$C_4 C_3 C_2 C_1$

2) E A T

T H A T

A P P L E

Answer

$$\begin{array}{r} 1\,1\,0\,1 \\ 8\,1\,9 \\ 9\,2\,1\,9 \\ \hline 1\,0\,0\,3\,8 \end{array}$$

1) $A = 1$

2) $C_3 + T = P + 10$

$C_3 + T > 9$

So if $T = 9$ and $C_3 = 1$

$1 + 9 = 10$

C_4 P

$C_4 = 1$, $P = 0$

OR

$1 + 9 = P + 10$

$P = 0$

3) $E + H + C_2 = P + C_3$

$E + H + C_2 = P + 10$

4) $A + A + C_1 = L$

$2A + C_1 = L$

$C_1 = 0$ $C_1 = 1$

$2A = L$ $L = 2A + 1$

$L = 2$ $L = 3$

5) $T + T = E + C_1$

$9 + 9 = 18$

C1 E

So $E = 8$ and $C_1 = 1$

So from step: 4

If $C_1 = 1$ then $L = 3$

from Step: 3

$8 + H + C_2 = P0 + 10$

$C_2 = 0$ $C_2 = 1$

$H = 2$ $H = 1$

not possible because $A = 1$

Teacher's Sign

DATE:

3) C_3 C_2 C_1

1 O D D

2 O D D

3 ———

4 E V E N

5

6 1) $C_3 = 1$

7 $E = 1$

8

9 2) $D + D + C_1 = E + C_2$

10 $2D + C_1 = 1 + C_2$

11 $C_2 = 0$ $\qquad$ $C_2 = 1$

12

13

14 $2D + C_1 = 1$ $\qquad$ $2D + C_1 = 1 + 10$

E 1 15 $2D + \phi + 1 = 1$ $\qquad$ $2D + C_1 = 11$ $(C_1 = 1)$

16 $D = \phi$ $\qquad$ $D = 5$

17 if we take $D = 0$

18 $D + D = N$

19 $\phi + \phi = \phi$

20 if $D = \phi$ then $N = \phi$

21 not possible.

22

23 if we take $D = 5$

24

25 $D + D = N$

26 $5 + 5 = 10$

27 C_1 $\quad$ N

28

29 $\boxed{D = 5}$ and $\boxed{N = \phi}$ $\therefore$ $D = 5$ then $\boxed{C_2 = 1}$ from (1)

30

31 3) $0 + 0 + C_2 = V + 10$

32 $2 0 + 1 = V + 10$

33 $20 = V + 9$ $\qquad$ 10, 11, 12, 13, 14, 15, 16

34

35 $\boxed{0 = 6}$ or $\boxed{8}$

Teacher's Sign:

$C_2 C_1 C_0$

TWO
TWO
FOUR

7 6 5
7 6 5
1 5 3 0

1) $F = 1$

2) $T + T + C_2 = O + 10$

$2T + C_2 > 9$

$C_2 = 0$ ↙ ↘ $C_2 = 1$

$2T = 10$
$T = 5$
$O = \phi$
$R = \phi$
} not possible

$2T + 1 = 10$
$T = 5$
$O = 1$ → not possible

$C_2 = 0$ ↙ ↘ $C_2 = 1$

$2T = 12$
$T = 6$
$O = 2$
$R = 4$ → not possible

$2T + 1 = 12$
$T = 6$
$O = 3$
$R = 6$ → not possible

But $R > 9$
became $C_1 + W + W = U$
where $C_1 = 1$

$R > 9$

$C_2 = 0$ ↙ ↘ $C_2 = 1$

$2T = 14$
$T = 7$
$O = 4$
$R > 9$ ($4 + 4 = 8$) not possible

$2T + 1 = 14$
$T = 7$
$O = 5$ ∴ $R = \phi$
$5 + 5 = 10$ ($R = 10$)

C_1 R

$C_1 = 1$ | $R = 0$ | $C_2 = 1$

Teacher's Sign

3) $W + W + 1 = U + 10$

$2W + 1 = U + 10 \quad (> 9)$

If we take $W = 5$ then $U = 1$ not possible $(F = 1)$

$\boxed{W = 6}$ then $\boxed{U = 3}$ ✓

5) $C_5\ C_4\ C_3\ C_2\ C_1$

D O N A L D

G E R A L D

R O B E R T.

<9.

→

1) $D + D = T + 10$

$D = 5$

$T = 0$

$C_1 = 1$.

2) $C_5 + D + G < 9$

$C_5 + G < 4$

$C_5 = 1$ ↙ ↘ $C_5 = 0$

$G < 3$ $G < 4$

$G = 1, 2$ $G = 1, 2, 3$.

3) $O + E + C_4 = O + 10$

$E + C_4 \neq 0$ and $E \neq 0$ as $T = 0$

we assume $\boxed{E = 9}$ $\boxed{C_4 = 1}$ and $C_5 = 1$

4) ~~$L + L + C_4 = O + 10$~~

$L + L + C_1 = R$

$2L + 1 = R$

$2L$ is even So R is odd.

($R = 1, 3$ or 7)

$A + A + C_2 = E$

$2A + C_2 = 9$

$2A + 1 = 9$

$A = 4$ and $C_2 = 1$

Since $C_2 = 1$

$2L + 1 = R + 10$

5) $D + C_4 + C_5 = R$

$5 + C_4 + 1 = R$

$C_4 + 6 = R$ ($CR = 1$, 3 or 7).

$C_4 = 1$

$R = 7$.

$2L + 1 = R + 10$

$2L + 1 = 7 + 10$

$2L + 1 = 17$

$2L = 16$

$L = 8$

6) $N + R + C_3 = B + 10$ as $C_4 = 1$

$N + 7 + C_3 = B + 10$

If $C_3 = 0$

$N + 7 = B + 10$

$N = B + 3$

$B = 2, 3$ or 6

if $B = 2$

$N = 5 \rightarrow X$.

$B = 3$

$N = 6$

$\rightarrow$ $T = 0$ $L = 8$

$C_4 = 1$ $E = 9$.

only 2 is left

So $= 0 = 2$.

$B = 3$

$A = 4$

$D = 5$

$N = 6$

$R = 7$

6)
$$\begin{array}{cccccc} C_4 & C_3 & C_2 & C_1 & \\ & B & A & S & E \\ & B & A & L & L \\ \hline C_5 & A & M & E & S \end{array}$$

1) $C_5 = 1$

$C_4 = 1$

2) $B + B + C_3 = A + C_4$

$C_4 = 1$

$B + B + C_3 > 9$

Assume $B = 7$ and $C_3 = 0$

$7 + 7 + 0 = 14$

$A = 4$.

3) $A + A + C_2 = M + C_3$

$4 + 4 + C_2 = M + C_3$ — (1)

$C_2 = 0$ ↙ ↘ $C_2 = 1$

$C_3 = 8$ $M = 9$.

4) $S + L + C_1 = E$ — (2)

$E + L = S$ — (3)

assume $L = 5$ $E + 5 = S$

$S + 5 + C_1 = E$

Consider $C_1 = 0$

$S + 5 = E$

Suppose $E = 3$ and $S = 8$

$3 + 5 = 8$ — (3)

and $8 + 5 = 13$ — (2)

$E = 3$, $S = 8$, $L = 5$

5) in (1)

$4 + 4 + C_2 = M$ $(C_2 = 1)$

$4 + 4 + 1 = M$

$M = 9$.

finally.

$$\begin{array}{cccc} {\scriptstyle 1} & {\scriptstyle 1} & & \\ 7 & 4 & 8 & 3 \\ 7 & 4 & 5 & 5 \\ \hline 1 & 4 & 9 & 3 & 8 \end{array}$$

(*) Means-Ends Analysis :-

↳ Means end Analysis is special type of search method that allows both backward and forward searching.

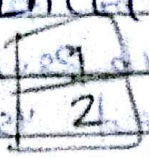
↳ The Basic principle of means end analysis is that it identifies difference between the current state and the goal state.

↳ Then it will try to find an operator that will move from current state to the goal state, if an operator cannot be found then the problem is reduced to sub problems and these sub problems are solved using the same idea. means-end analysis can be applied recursively.

↳ The rules are usually represented as a left side that describes the conditions that must be met for the rule to be applicable and a right side that describes those aspects of the problem state that will be changed by the application of the rule.

↳ A separate data structure called a difference table indexes the rules by the differences that they can be used to reduce.

Operator	Preconditions	Results
PUSH(obj, loc)	at(robot, obj) ∧ large(obj) ∧ clear(obj) ∧ armempty	at(obj, loc) ∧ at(robot, loc)
CARRY(obj, loc)	at(robot, obj) ∧ small(obj)	at(obj, loc) ∧ at(robot, loc)
WALK(loc)	none	at(robot, loc)
PICKUP(obj)	at(robot, obj)	holding(obj)
PUTDOWN(obj)	holding(obj)	¬holding(obj)
PLACE(obj1, obj2)	at(robot, obj2) ∧ holding(obj1)	on(obj1, obj2)



(fig 15 The Robot's Operators)

	Push	Carry	Walk	Pickup	Putdown	Place
Move object	*	*				
Move robot			*			
clear object				*		
Get object on object						*
Get arm empty					*	*
Be holding object				*		

(fig 16 A difference Table)

↳ Consider a simple household robot domain. The available operators are shown in fig-15 along with their preconditions and results.

↳ fig-16 shows the difference table that describes when each of the operators is appropriate.

↳ Suppose that the robot in this domain were given the problem of moving a desk with two things on it from ~~one~~ one room to another.

↳ The main difference between the start state and the goal state would be the location of the desk. ~~To~~

↳ To reduce this difference, either PUSH or CARRY could be chosen.

↳ CARRY is applicable if small table otherwise use PUSH.

Algorithm: Means-Ends Analysis

1) Till the goal is reached or no more procedures are available.

2) In this step describe the current state, the goal state and the difference between the two. ~~description of the current state or goal state, to select a promising procedure.~~

3) Use the difference between the current state and goal state, possibly with the description of the current state or goal state, to select a promising procedure.

4) Use the promising procedure and update the current state.

5) If the goal is reached, announce success; otherwise, announce failure.