Ch : 2 Problems, Problem space, and Search

PAGE NO.:
DATE:

So solve real life problem

To Build a System to solve a particular problem. we need to do four things :

1) Define the problem precisely.

This definition must include precise Specifications of what initial situation and final situations Constitute acceptable solutions to the problem.

2) Analyze the problem.

It includes finding the key features of the problem. (for solving the problem)

3) Isolate and represent the task knowledge.

that is necessary to solve the problem.

4) choose the best problem-solving technique.

and apply it to the particular problem.

.	.	o
.	x	.
.	.	.

→ Problem space / problem state space

⇓

All possible configuration of the relevant object

→ Defining the problem as a state space search:

1) Chess game :

first of all consider chess game i.e. Problem of playing chess. To Construct a Program

Teacher's Sign

Scanned by CamScanner

that could play chess. We have to specify the starting position of the chess board, the rules that define legal moves and goal position (state (The goal of the winning the game, if possible, must be made clear)

Initial State:

8 x 8 matrix where content of each element of the matrix is the symbol corresponding to a piece of chessboard.
(8x8 array square in which each element square (x, y), where x varying from 1 to 8 and y varying from 1 to 8).

Operators (Rules):

Legal moves are defined by a set of rules, consisting of two parts:
A left side that gives the current position and

A right side describes the change to be made to the board position to reflect the move. (Action).

goal state:

The goal can be any board position in which the opponent does not have a legal move and his or her "king" is under attack.

Teacher's Sign

Scanned by CamScanner

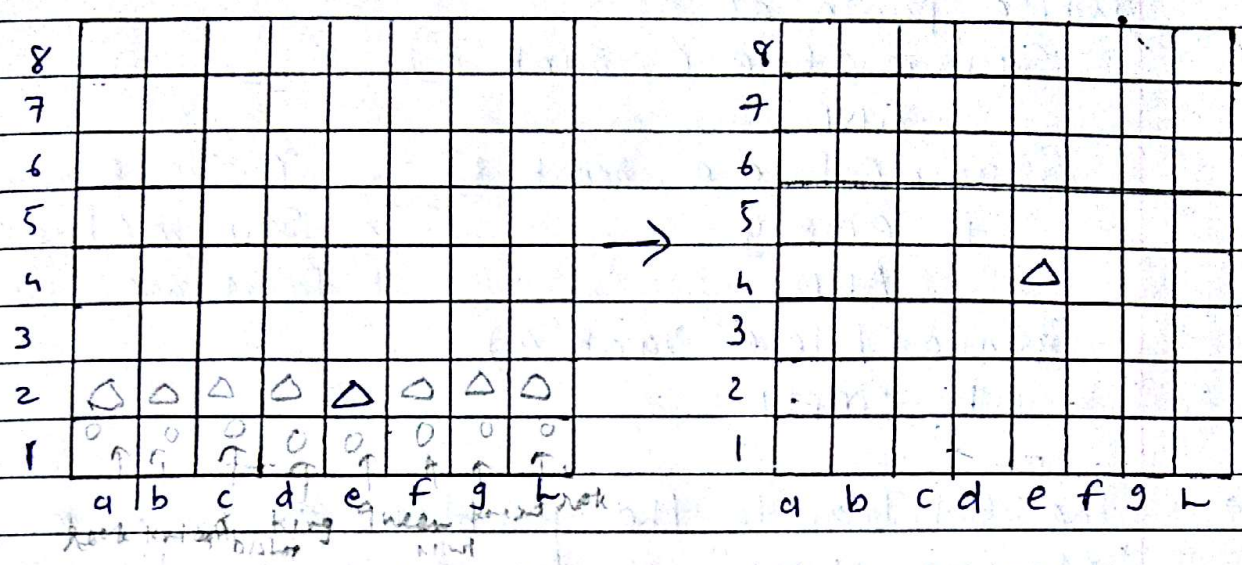


fig: 1 [one legal chess move]

→ However, if we write rules like the one above.

→ Using so many rules poses two serious practical difficulties:

1) No person could ever supply a complete set of such rules. It would take too long and could certainly not be done without mistakes.

2) No program could easily handle all those rules.

In order to minimize such problems, it is useful to introduce some convenient notation for describing patterns and substitutions:

for example the rule describe in fig-1, as well as many like it, could be written as shown in fig-2.

Scanned by CamScanner

DATE:

White pawn at
Square (file e, rank 2)
AND
Square (file e, rank 3) → move pawn from
is empty Square (file e, rank 2)
AND to Square (file e, rank 4)
Square (file e, rank 4)
is empty

The Solution to the ~~problem of~~ problem of "playing chess" is to move in the state space. Start from the initial state then using operators (rules) go on to the new states till the goal state is reached.

The state space representation is easier for the chess problem because the problem itself is highly structured.

Scanned by CamScanner

set of all possible states for a given problem is called state space of problem. which rules can be applied to reach the goals

DELUXE
PAGE NO.:
DATE:

2 A Water Jug problem :

→ you are given two jugs. a 4-gallon one and a 3-gallon one.

ve: → There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into 4-gallon jug ?

→ The state space for this problem can be described as the set of ordered pairs of integers (x, y). such that x = 0, 1, 2, 3 or 4. and y = 0, 1, 2 or 3.

→ x represents the number of gallons of water in the 4-gallon Jug

→ y represents the quantity of water in the 3-gallon Jug.

Initial State (0, 0)

Goal State (2, y)

→ The Solution to the water Jug Problem is obtained by moving from initial state to the final state using operators.

→ But How to define these operators.

→ There were some assumptions present in the problem.

Teacher's Sign

Scanned by CamScanner

DATE:

It was not mentioned in the problem that

(1) Water can be poured out from the Jugs on to the ground.

(2) We can pour water from one Jug to another and that there are no other measuring devices available:

Operators/Production rules: for water Jug Problem

	Current State	Next State	Description
1	(x,y) if $(x<4)$	$\rightarrow (4,y)$	Fill the 4-gallon Jug.
2	(x,y) if $y<3$	$\rightarrow (x,3)$	Fill the 3-gallon Jug.
3	(x,y) if $x>0$	$\rightarrow (x-d,y)$	Pour some water out of the 4-gallon jug.
4	(x,y) if $y>0$	$\rightarrow (x,y-d)$	Pour some water out of the 3-gallon Jug.
5	(x,y) if $x>0$	$\rightarrow (0,y)$	Empty the 4-gallon Jug on the ground.
6	(x,y) if $y>0$	$\rightarrow (x,0)$	Empty the 3-gallon Jug on the ground.
7	(x,y) if $x+y\geq 4$ and $y>0$	$\rightarrow (4,y-(4-x))$	Pour water from the 3-gallon Jug into the 4-gallon Jug until the 4-gallon Jug is full.

$(3,2) \xrightarrow{32} \geq 4 \rightarrow (4,2-(4-3)) = (4,\underset{\downarrow \; y=1}{1})$

$(3,1) \xrightarrow{34} \geq 4 \rightarrow (4,1-(4-3)) = (4,\underset{\downarrow \; y=0}{0})$

Teacher's Sign

Scanned by CamScanner

	current state	Next state	Description
8	(x,y) if $x+y \geq 3$ and $x>0$	$\rightarrow (x-(3-y), 3)$	pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full.
	$(2,2) \rightarrow \geq 3$	$(2-(3-2), 3) \rightarrow (1, 3)$ $\downarrow$ $x=1$	
	$(3\ 2) \rightarrow \geq 3$	$(3-(3-2), 3) \rightarrow (2, 3)$ $\downarrow$ $x=2$	
9	(x,y) if $x+y \leq 4$ and $y>0$	$(x+y, 0)$	pour all the water from the 3-gallon jug into the 4-gallon jug
10	(x,y) if $x+y \leq 3$ and $x>0$	$(0, x+y)$	pour all the water from the 4-gallon jug into the 3-gallon jug.
11	$(0,2)$	$(2,0)$	pour the 2-gallons from the 3-gallon jug into the 4-gallon jug.
12	$(2,y)$	$(0,y)$	Empty the 2 gallons in the 4-gallon jug on the ground.

The process of converting informal problem statement into formal problem description is called operationalization.

Scanned by CamScanner

PAGE NO.:
DATE:

Que: How can you get exactly 2-gallons of water into the 4-gallon Jug?

Solution:

gallons in the 4-gallon Jug	gallons in the 3-gallon Jug	Rule applied.
0	0	
		2 1
0 4	3 0	
		9 8
3 1	0 3	
		2 6
3 1	3 0	
		7 10
4 0	2 1	
		5 or 12 1
0 4	2 1	
		9 or 11 8
2 2	0 3	

→ Some Issues in operationalization of water Jug problem:

1) Role of the Conditions that occur in the left Side of rules.

– for ex: rule 1: "fill the 4-gallon Jug"
Should be: "If the 4-gallon jug is not already full, fill it".

It is stupid to fill 4-gallon Jug even if it is already full since there is no change in the problem state results

Q-B Explain State Space with the use of water Jug Problem.

PAGE NO.:
DATE:

2) Should Rule 3 and 4 be included or not be included in the list of operators?

- Emptying an unmeasured amount of water onto the ground will never get us any closer to a solution.

3) Special Case Rules:
- Rule 11 is a special case of Rule 9.
- Rule 12 is a special case of Rule 5.

ex: read book

→ These rules do not actually add power to the system since the operations they describe are already provided by rule 9 (for rule 11) and by rule 5 (for rule 12).

Steps for formal Description of problem:

1) Define a state space that contains all the possible configurations of the relevant objects.

2) Specify one or more states within that space that describe possible situations from which the problem-solving process may start. These states are called the Initial states.

3) Specify one or more states that would be acceptable as solutions to the problem. These states are called goal states.

4) Specify a set of rules that describe the actions (operators) available.

Scanned by CamScanner

PAGE NO.:
DATE:

2.2 Production System:

Production systems provide a structure, which is useful to structure AI programs in a way that facilitates describing and performing the search process.

Definition:

A production System consists of:

→ A set of rules, each consisting of a left side that determines the applicability of the rule and a right side that describes the operation to be performed if the rule is applied.

→ One or more knowledge / databases that contain whatever information is appropriate for the particular task. Some parts of the database may be permanent, while other parts of it may be related ~~to~~ only to the solution of the current problem.

→ A Control Strategy that ~~contain~~ specifies the order in which the rules will be compared to the database and a way of resolving the conflicts that arise when several rules match at once.

→ A rule applier.

Scanned by CamScanner

Control Strategies:

Which rule to apply next in the process of search for a solution if multiple rules are applicable. This is decided by Control Strategy

Requirement of good Control strategy:

→ It should cause motion:

The first requirement of a good Control Strategy is that it causes motion.

Consider the Water Jug problem: Suppose we implemented the simple Control strategy of Starting each time at the top of the list of rules and choosing the first applicable one. If we did that, we would never solve the problem.

— We would Continue indefinitely filling the 4-gallon jug with Water. Control Strategies that do not cause motion will never lead to a solution.

→ It should be Systematic.

— The Second requirement of a good Control strategy is that it be systematic. Water Jug problem: On each cycle choose at random from among applicable rules.

— If the Control strategy is not Systematic, we may explore a particular useless Sequence of operators several times before we find a solution.

Teacher's Sign _____

Scanned by CamScanner

BFS:

One systematic Control strategy for the Water Jug Problem is the following:

- Construct a tree with the Initial state as its root. (0,0)

- generate all the offspring of the root by applying each of the applicable rule to the initial state.

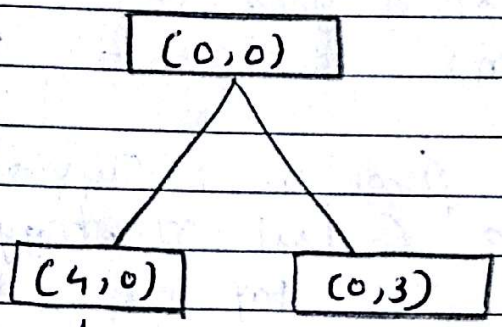


fig: One level of BFS.

Now for each leaf node, generate all its successors by applying all the rules that are appropriate.

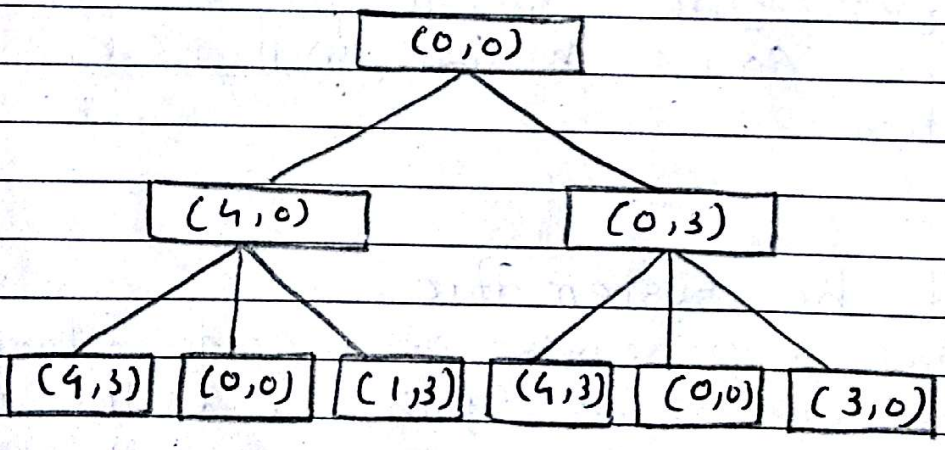
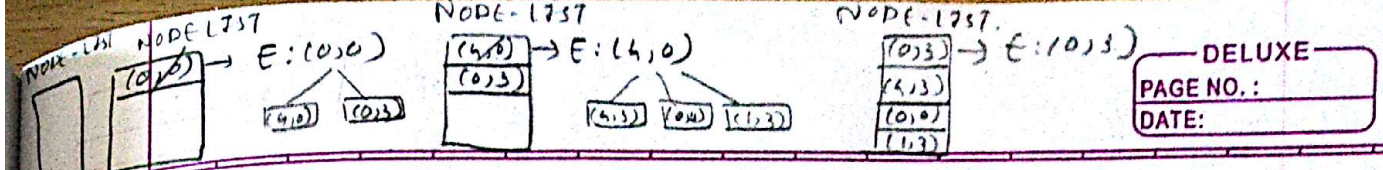


fig: Two levels of BFS tree

Scanned by CamScanner



Continue this process until some rule produces a goal state. This process, called breadth first search (BFS)

Algorithm : Breadth first Search

1) create a variable called NODE-LIST and set it to the Initial State.

2) Until a goal state is found or NODE-LIST is empty :
   a) Remove the first element from NODE-LIST and call it E. If NODE-LIST was empty, quit.
   b) for each way that each rule can match the state described in E do :

      i) Apply the rule to generate a new state.
      ii) If the new state is a goal state, quit and return this state.
      iii) otherwise, add the new state to the end of NODE-LIST.

Advantages of BFS :

1) If there is a solution then BFS is guaranteed to find it.
   - If there are multiple solutions, then a minimum solution (min. no of steps) will be found.

   - This is guarnted by the fact that longer paths are never explored until all shorter ones have already been examined.

Teacher's Sign _____

Scanned by CamScanner

- This Constrasts with depth-first-Search Which may find a long path to a Solution in one part of the tree, when a shorter path exists in Some other unexplored path of the tree.

2) BFS will not follow a path Blindly. This Constsucts with depth first Searching, which (DFS) may follow a Single, unfruitful path for a very long time that will not lead to a Solution.

Depth First Search (DFS).

→ We Could pursue a Single branch of the tree until it yields a Solution or until a decision to terminate the path is made.

→ It makes Sense to terminate a path if it reaches a dead-end, produces a previous state or becomes longer than some pre specified "futility" limit. In such a Case Backtracking occurs.

→ The most recently Created state from Which alternative moves are available will be revisited and a new state will be created. This form of Backtracking is Called Chronological backtracking.

The most recent step is always the first to be undone. This form of back tracking is Called backtracking.

Teacher's Sign

Scanned by CamScanner

Algorithm: Depth-First Search.

1 If the initial state is goal state, quit and return success.

2 Otherwise, do the following until success or failure is signaled:

a) generate a successor, E, of the initial state. If there are no more successors, signal failure.

b) call depth-first search with E as the initial state

c) If success is returned, signal success. Otherwise continue in this loop.

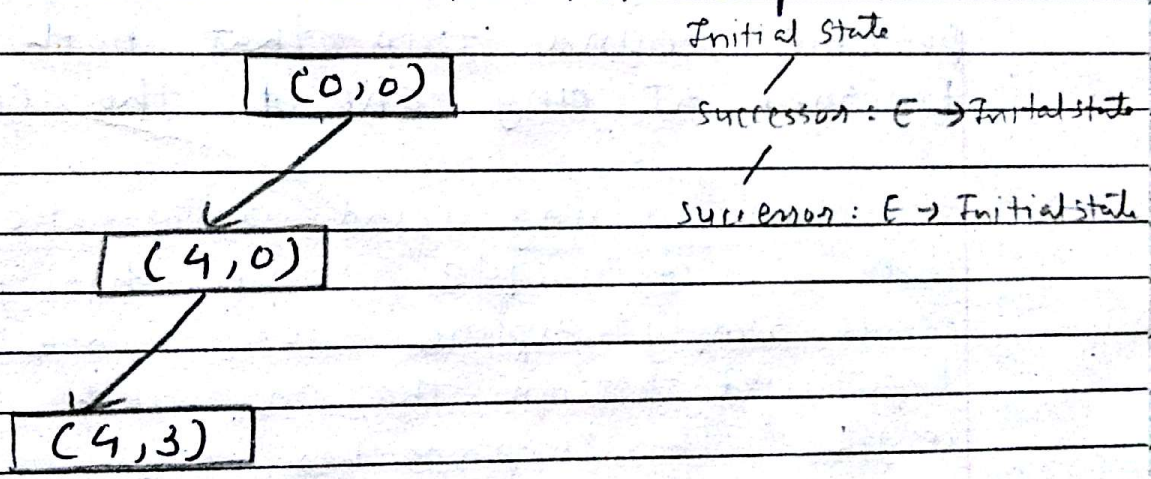


fig: DFS tree for the water jug problem.

Advantages of Depth-First Search:

1) Depth-First search requires less memory. Since only the nodes on the current path are stored. While in BFS all of the tree that has so far been generated must be stored.

Scanned by CamScanner

or if care is taken in ordering successor states

2) By chance it may find a solution without examining much of the search space at all, while in BFS in which all parts of the tree must be examined to level n before any nodes on level n+1 can be examined.
DFS can stop when one of them is found.

The Travelling Salesman Problem:

A Salesman has a list of cities, each of which he must visit exactly once. There are direct roads between each pair of cities on the list. Find the route the salesman should follow for the shortest possible round trip that both starts and finishes at any one of the cities.

→ A motion-causing and systematic control structure could, in principle, solve this problem. It would simply explore all possible paths in the tree and return the one with shortest path length. This approaches are used for very short list of cities.

→ But for large number of cities in traveling salesman problem then motion-causing and systematic control structure could not applied

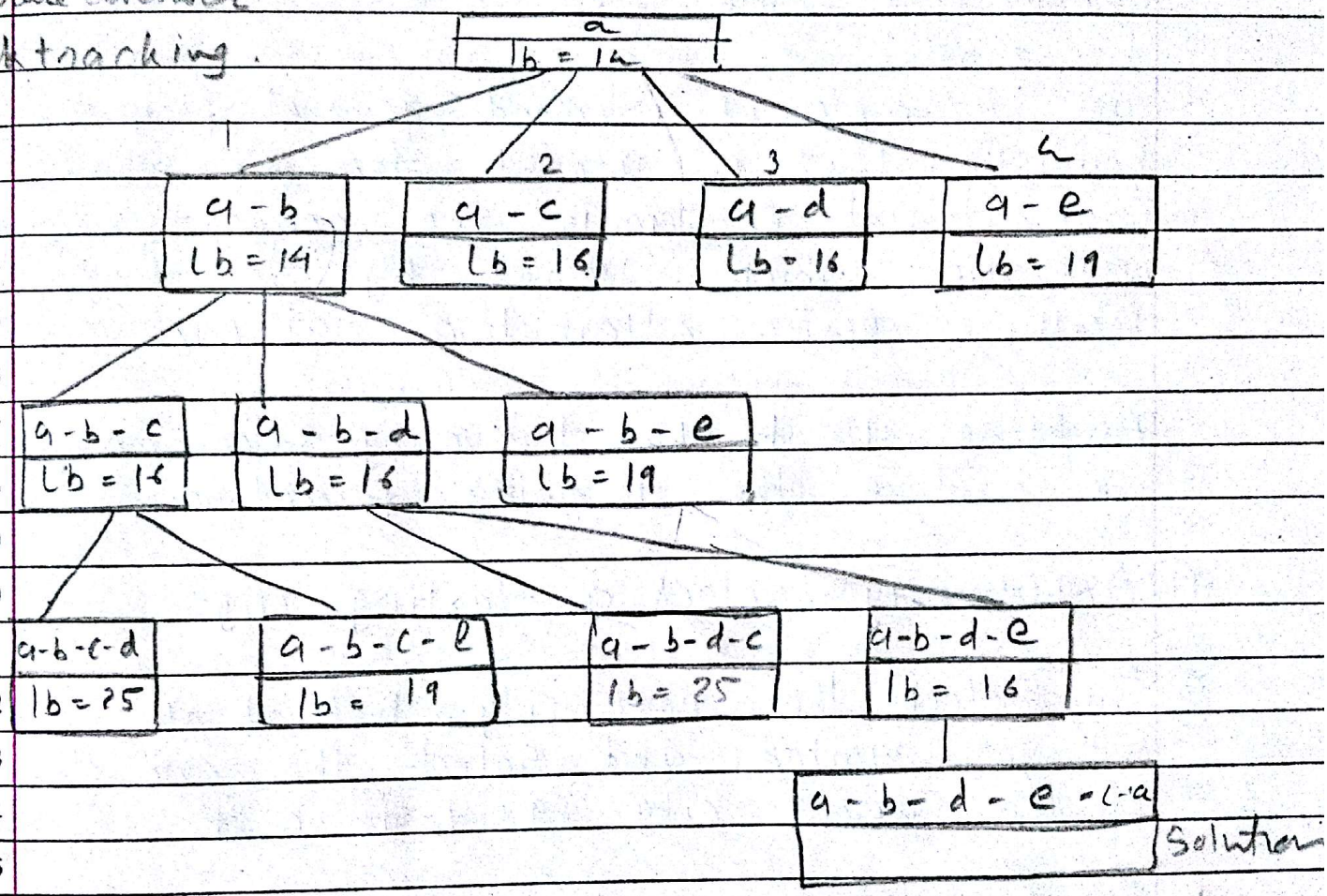
→ We can beat the above techniques using a technique called "branch and bound search technique"

Scanned by CamScanner

Initially get one solution path. this will be the shortest path found so far. Consider other paths keep track of the shortest path. While considering any other path as soon as the length becomes more than of the shortest path found so far, discard the path.

(*) Heuristic Search: example:

- Accurate answer
- Backtracking.



Heuristic Search:

→ Heuristic is a technique that improves the efficiency of a search process by sacrificing claims of completeness.

→ Heuristics are like tour guides.

Teacher's Sign

Scanned by CamScanner

Using good heuristics. we can hope to get good solutions to hard problems.

They are good to the extent that they point in generally interesting directions. They are bad to the extent that they may miss points of interest to particular individuals.

Using good heuristics, solution to hard problems such as traveling salesman, in less than exponential time.

one example of a good general purpose heuristic that is useful for a variety of combinatorial problems is the nearest neighbor heuristic. which works by selecting the locally superior alternative at each step.

Applying it to the traveling salesman problem: we produce the following procedure:

1. Arbitrarily select a starting city.

2. To select the next city, look at all cities not yet visited, and select the one closest to the current city. go to it next.

3. Repeat step 2 until all cities have been visited.

This procedure executes in time proportional to N^2.

Advantages of Heuristic:

1. Most of the time, we require a good solution not optimum.

Scanned by CamScanner

DELUXE
PAGE NO.:
DATE:

(2) Solution obtained by heuristics may not be very good in worst case, but worst case rarely arises.

(3) Trying to understand whether a Heuristic will work or not gives better understanding of the problem.

→ There are two major ways in which domain specific, heuristic knowledge can be incorporated into a rule based search procedures:

(1) In the rules themselves.

(2) As a heuristic function that evaluates individual problem states and determines how ~~describ~~ desirable they are.

Heuristic function: is a function that maps from problem state descriptions to measures of desirability. (usually represented as numbers)

→ Heuristic function helps in guiding the search process toward a solution by suggesting which path to follow first when more than one is available.

Purpose:

→ Sometimes very simple heuristic function can provide a fairly good estimate of whether path is any good or not.

Teacher's Sign

Scanned by CamScanner

DATE:

heuristic search ex: 8- puzzle:

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

Initial state

2	8	3
1	6	4
7		5

final state:

1	2	3
8		4
7	6	5

h = 4

2	8	3
1	6	4
7		5

h = 5

2	8	3
1	6	4
	7	5

h = 3

2	8	3
1		4
7	6	5

h = 5

2	8	3
1	6	4
7	5	

h = 3

2	8	3
1		4
7	6	5

h = 3

2		3
1	8	4
7	6	5

h = 4

2	8	3
1	4	
7	6	5

h = 3

	8	3
2	1	4
7	6	5

h = 4

2	8	3
7	1	4
	6	5

h = 2

	2	3
1	8	4
7	6	5

h = 4

2	3	
1	8	4
7	6	5

h = 1

1	2	3
	8	4
7	6	5

goal state →

1	2	3
8		4
7	6	5

h = 2.

1	2	3
7	8	4
	6	5

Scanned by CamScanner

Some Simple heuristic functions for few problems

Chess : the material advantage of our side over the opponent

Traveling Salesman : the sum of the distance so far.

Tic-Tac-Toe : 1 for each row in which we could win and in which we have already one piece plus 2 for each such row in which we have two pieces.

2.3 Problem Characteristics :

In order to choose the most appropriate method for a particular problem, it is necessary to analyze the problem along several key dimensions

1. Is the problem decomposable into a set of independent small or easier sub problems ?

2. Can solution steps be ignored or at least undone if they prove unwise ?

3. Is the problem's universe predictable ?

4. Is a good solution to the problem obvious without comparision to all other possible solutions ?

5. Is the desired solution a state of the world or a path to a state ?

Scanned by CamScanner

(6) Is a large amount of knowledge absolutely required to solve the problem, or is knowledge important only to constrain the search?

(7) Can a computer that is simply given the problem return the solution, or will the solution of the problem require interaction between the computer and a person?

1) Is the Problem Decomposable?

Suppose we want to solve the problem of computing the expression

ex: $\int (x^2 + 3x + \sin^2 x \cdot \cos^2 x)\, dx$

We can solve this problem by breaking it down into three smaller problems, each of which we can then solve by using a smaller collection of specific rules.

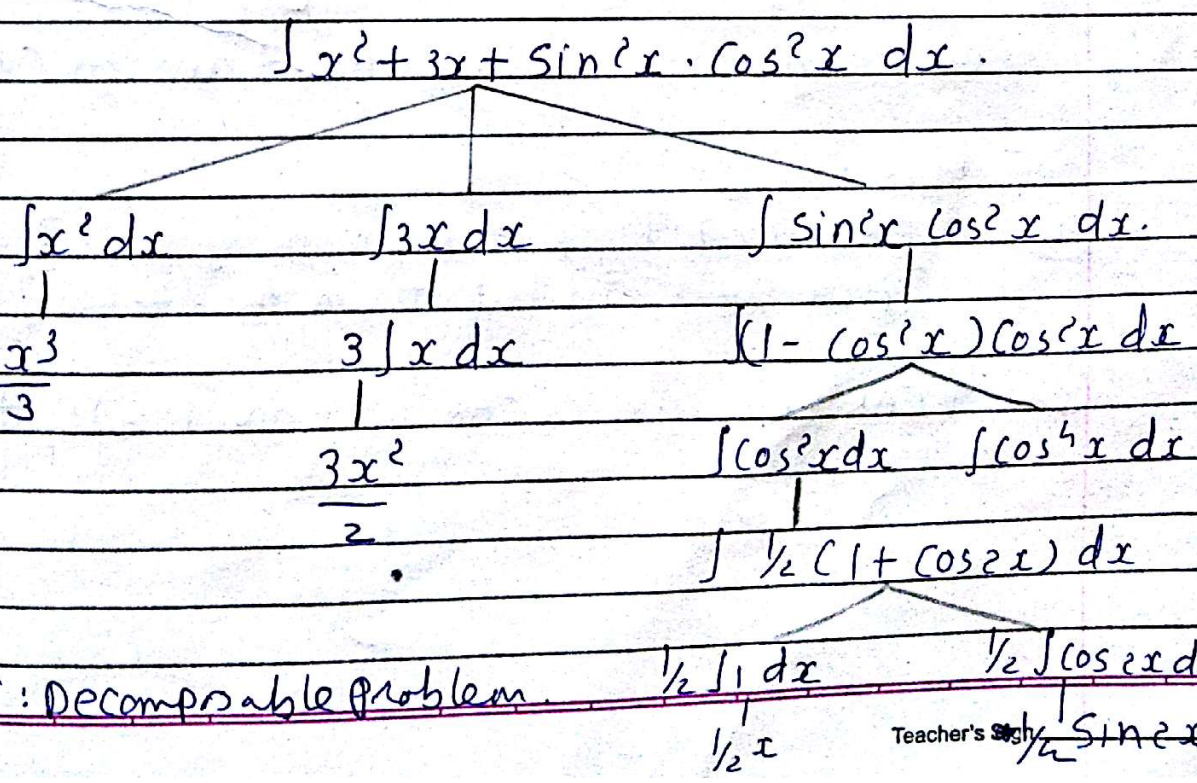


fig 6: Decomposable problem.

Scanned by CamScanner

→ fig 6 shows the problem tree that will be generated by the process of problem decomposition that works as follows:

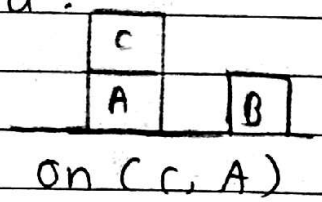
→ At each step, it checks to see whether the problem it is working on is immediately solvable. If so, then the answer is returned directly.

→ If the problem is not easily solvable the integrator checks to see whether it can decompose the problem into smaller problems. If it can, it creates those problems and calls itself recursively on them.

→ Using this technique, we can often solve very large problems easily.

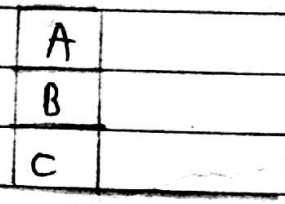
Another example: **Blocks World Problem**

Start:



on (C, A)

goal:



on(B, C) and on (A, B)

operators:

1. CLEAR (x) [block x has nothing on it] → ON(x, Table) [pick up x and put it on the table]

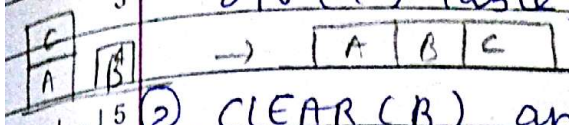
2. CLEAR (x) and CLEAR (y) → ON(x, y) [put x on y]

Scanned by CamScanner

Solution:

① CLEAR(C) is true

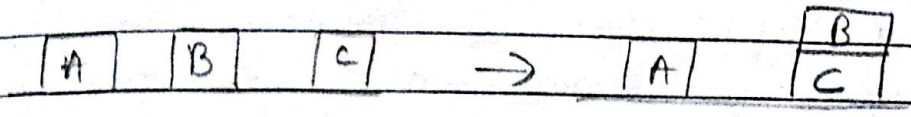
ON(C, Table) is the operation performed



Initial State

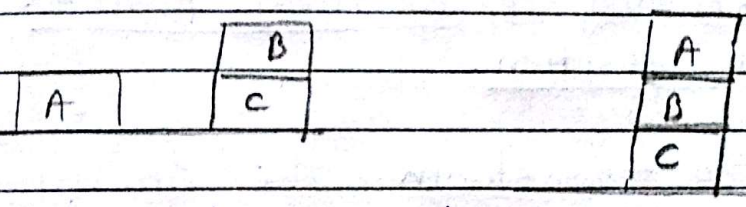
② CLEAR(B) and CLEAR(C) are true

∴ ON(B, C) is the operation performed.



③ CLEAR(A) and CLEAR(B) are true

∴ ON(A, B) is the operation performed.

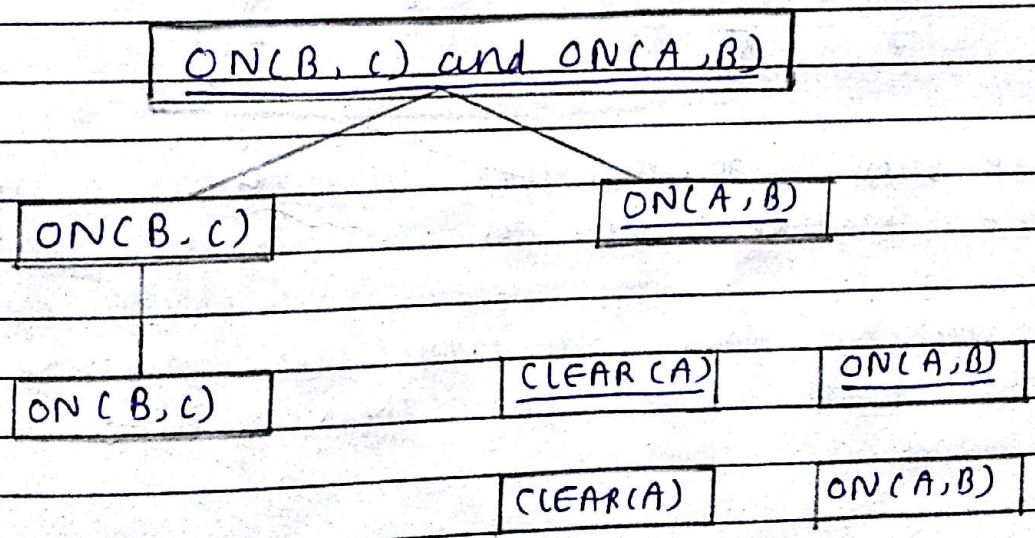


goal state.

→ Problem is solved

→ But can we decompose this problem?

Solution tree:



→ In fig, goals are underlined. states that have been achieved are not underlined.

→ The idea of this solution is to reduce problem of getting B on C and A on B to two separate problems.

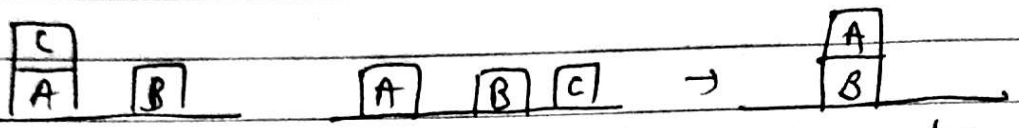
Teacher's Sign

Scanned by CamScanner

PAGE NO.:
DATE:

The first of these new problems, getting B on is simple.

The second subgoal is not simple since, the only operators we have allow to pick up single blocks at a time.



If we now try to combine the two subsolutions into one solution, we will fail.
So, This is a non decomposible problem.

Scanned by CamScanner

2) Can Solution steps Be ignored or Undone?

Consider one problem :

The 8-puzzle :

The 8-puzzle is a square tray in which are placed eight square tiles. The remaining ninth square is uncovered. Each tile has a number on it. A tile that is adjacent to the blank space can be slid into that space.

A game consists of a <u>starting position</u> and a specified <u>goal position</u>.

The goal is to transform the starting position into the goal position by sliding the tiles around.

ex :

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

→ We slide 5 to the left, which is a mistake that we realize later on.

→ Still we can rectify our mistake undoing the first move and then sliding 6 down-loaded.

→ Control structure that is used in solving the puzzle must be able to do backtracking. The order in which steps are taken must be stored.

Scanned by CamScanner

PAGE NO.:
DATE:

→ An additional step must be performed to undo each incorrect step.

Three class problem :

1) Ignorable : (e.g. Theorem proving)

→ in which Solution steps can be ignored.

→ Ignorable problems can be solved using a simple control structure that never backtracks.

→ No Backtracking required.

2) Recoverable : (e.g. 8-puzzle)

→ in which Solution steps can be undone (reverse)

→ Backtracking required.

→ Recoverable problems can be solved by a slightly more complicated control strategy that does sometimes make mistakes. Backtracking will be necessary to recover from such mistakes.

→ So the control structure must be implemented using a push-down stack. in which decisions are recorded in case they need to be undone later.

3) Irrecoverable : (e.g. chess)

→ in which Solution steps can not be undone

Scanned by CamScanner

→ great amount of planning required.

→ Silly moves can not be undone, at Best playing can be used to go a State (Better) from the current State.

→ Irreloverable problems will need to be solved by a System that extends a great deal of effort making each decision, since the decision must be final.

3) Is the Universe predictable?

ex:1 Again suppose that we are playing with the 8-puzzle. Every time we make a move, we know exactly what will happen.

→ This means that it is possible to plan an entire sequence of moves. And be Confident that we know what the resulting State will be.

→ We can use planning to avoid having to undo actual moves, although it will still be necessary to backtrack past those moves one at a time during the planning process.

→ Thus a Control structure that allows backtracking will be necessary.

→ However, in games other than the 8-puzzle this planning process may not be possible

ex:2 → Suppose we want to play bridge one of the

Scanned by CamScanner

DELUXE
PAGE NO.:
DATE:

decisions we will have to make is which card to play on the first trick. What we would like to do is to plan the entire hand before making that first play.

ex: chess:

→ But now it is not possible to do such planning with certainty since we can not know exactly where all cards are or what the other players will b do on their turns. The best we can do is to investigate several plans and use probabilities of the various outcomes to choose a plan that has the highest estimated probability of leading to a good score on the hand.

→ These two games illustrate the difference between certain outcome (e.g. puzzle) and uncertain-outcome (e.g. bridge) problems.

4) Is a good Solution Absolute or Relative?

ex:1 → Consider the problem of answering questions based on a database of simple facts, such as the following:

1. Marcus was a man.
2. Marcus was a pompeian.
3. Marcus was born in 40 A.D.
4. All men are mortal
5. All pompeians died when the volcano erupted in 79 A.D.
6. No Mortal lives longer than 150 years.
7. It is now 1991 A.D.

Scanned by CamScanner

PAGE NO.:
DATE:

Suppose we ask the question "Is Marcus alive"? By representing each of these facts in a formal language, such as predicate logic and then using formal inference methods we can fairly easily derive an answer to the question.

The answer, as shown below.

Solution: 1

	Justification
1. Marcus was a man	axiom 1
4. All men are mortal	axiom 4
8. Marcus is mortal	from 1, 4
3. Marcus was born in 40 A.D.	axiom 3
7. It is now 1991 A.D.	axiom 7
9. Marcus' age is 1951 years	from 3, 7
6. No mortal lives longer than 150 years	axiom 6
10. Marcus is dead	8, 6, 9

OR

Solution: 2

7. It is now 1991 A.D.	axiom 7
5. All pompeians died in 79 A.D.	axiom 5
11. All pompeians are dead now.	from 7, 5
2. Marcus was a pompeian	axiom 2
12. Marcus is dead	from 11, 2

[Two ways of deciding that Marcus is dead.

Scanned by CamScanner

ex: 2

Consider the traveling Salesman problem. our goal is to find the shortest route that visits each city exactly once. suppose the cities to be visited and the distances between them are as shown in fig: 10

	Boston	New york	Miami	Dallas	San francisco
Boston		250	1450	1700	3000
New york	250		1200	1500	2900
Miami	1450	1200		1600	3300
Dallas	1700	1500	1600		1700
San francisco	3000	2900	3300	1700	

fig 10 [An Instance of the TSP]

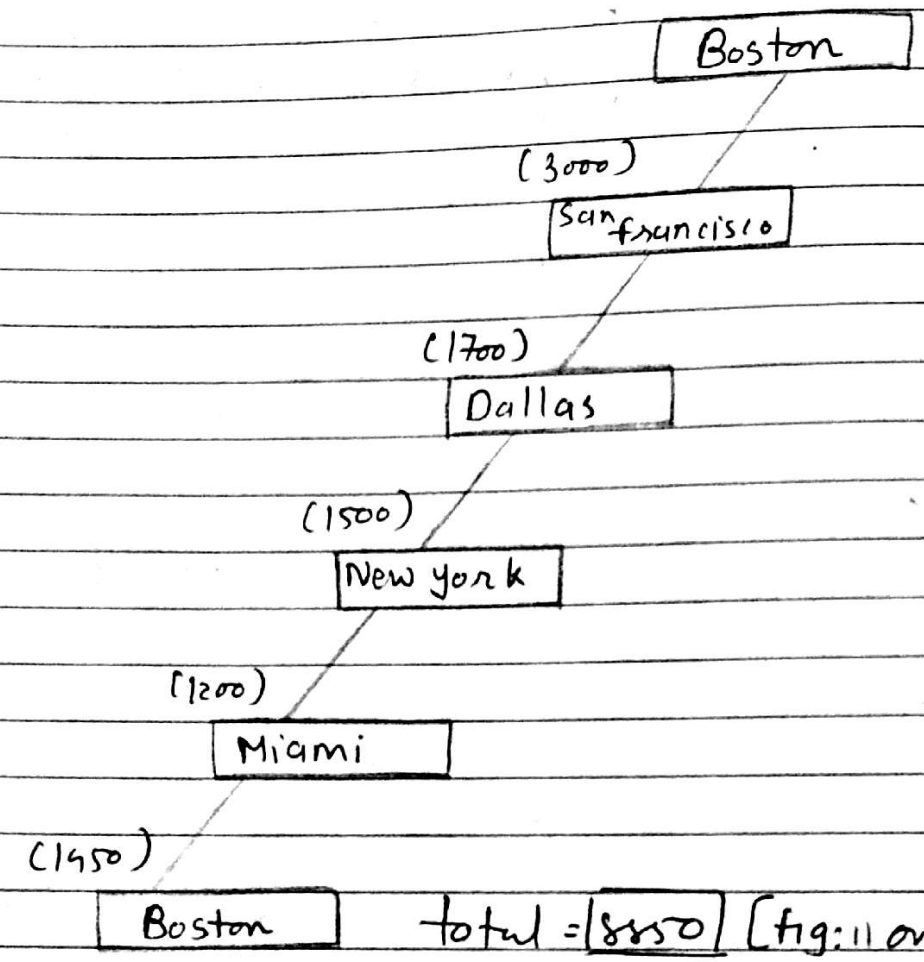
→ One place the salesman could start is Boston. In that case, one path that might be followed is the one shown in fig 11 which is 8850 miles long.

→ But is this the solution to the problem? The answer is that we cannot be sure unless we also try all other paths to make sure that none of them is shorter.

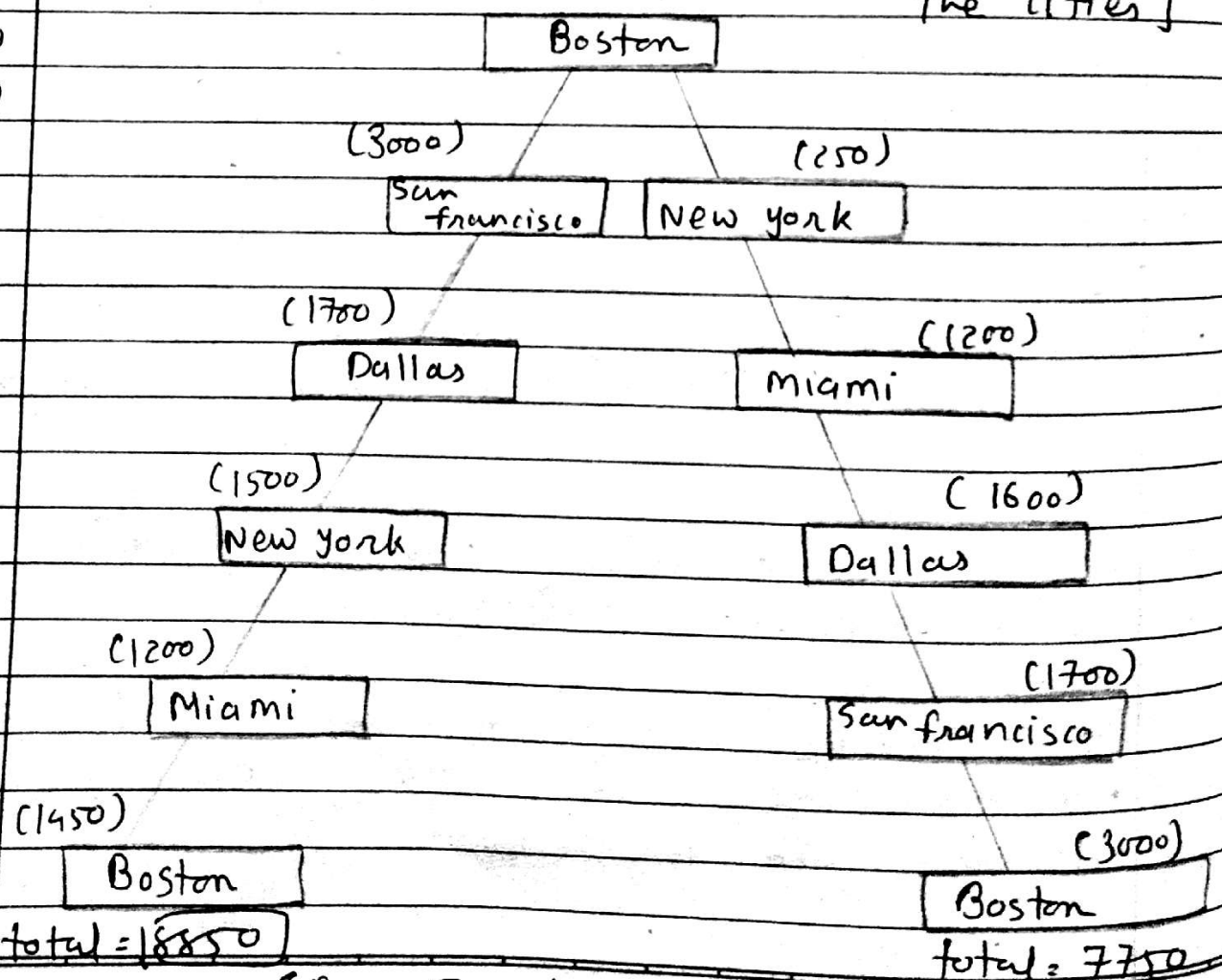
→ In this case, as can be seen from fig 12, the first path is definitely not the solution to the salesman's problem.

→ [fig 11 and fig 12] These two examples illustrate the difference between any path problems and best path problems.

Scanned by CamScanner



[fig: 11 one path among the cities]



[fig: 12 Two paths among the cities]

Scanned by CamScanner

5) Is the Solution a state or a path?

ex:1 → Consider the problem of finding interpretation for the sentence.

Solution is a state → The bank president ate a dish of pasta salad with the fork.

There are several components of this sentence, each of which, in isolation, may have more than one interpretation.

Some of the sources of ambiguity in this sentence are the following:

The word "bank" may refer either to a financial institution or to a side of river. But only one of these may have a president

→ The word "dish" is the object of the verb "eat". It is possible that a dish was eaten. But it is more likely that the pasta salad in the dish was eaten.

→ pasta salad is a salad containing pasta. But there are other ways meanings can be formed from pairs of nouns. for example, dog food normally contain dogs.

→ The phrase "with the fork" could modify several parts of the sentence. In this case, it modifies the verb "eat". But, if the phrase had been "with vegetables" then the modification structure would be different still.

Teacher's Sign

Scanned by CamScanner

In this example the Solution is a state i.e. We only need the meaning of a sentence, Not the way in which interpretation was found.

Water Jug problem:

ex:2 Solution is path

Contrast this with the Water Jug problem. Here It is not Sufficient to report that we have solved the problem and that the final State is (2,0)

→ for this kind of problem, what we really must report is not the final state but the path that we found to that state. Thus a statement to this problem must be a Sequence of operations that produces the final State.

→ These two examples, natural language understanding and the Water Jug problem illustrate the difference between problems whose Solution is a state of the World. and problems whose Solution is a path to a State.

6) What is the role of knowledge?

ex:1 Consider the problem of playing chess:

Assumption: Suppose you had unlimited Computing power available.

How much knowledge required:

1) A Set of rules for determining legal moves
2) Some Simple Control mechanism that implements an appropriate Search procedure.

Scanned by CamScanner

Additional knowledge required:
good strategy and tactics could help to constrain the search and speed up the execution of the program.

ex:2 Now consider the problem of scanning daily newspapers to decide which are supporting the democrates and which are supporting the Republicans in some upcoming election.

Assumption: you had unlimited computing power available.

Knowledge required:

→ (1) The names of the candidates in each party.

→ (2) The fact that if the major thing you want to see done is have taxes lowered, you are probably supporting the Republicans

→ (3) The fact that if the major thing you want to see ~~was~~done is improved education for minority students, you are probably supporting the democrats.

→ (4) The fact that if you are opposed to big government, you are probably supporting the Republicans.

difference: problem 1: lot of knowledge required to constrain the search for a solution.
problem 2: lot of knowledge required to recognize a solution.

Teacher's Sign

Scanned by CamScanner

7) Does the task require interaction with a person

→ Sometimes it is useful to program computers to solve problems in ways that the majority of people would not be able to understand.

→ This is fine if the level of the interaction between the computer and its human users is problem-in, solution-out

→ But increasingly we are building programs that require intermediate interaction with people, both to provide additional input to the program and to provide additional reassurance

Problem classification :

There are two types of problems :

1) Solitary : In which the computer is given a problem description and produces an answer with no intermediate communication (No intermediate communication with the user)

2) Conversational : In which there is intermediate communication between a person and the computer, either to provide additional assistance to the computer or to provide additional information to user or both (Intermediate communication is required to get reassurance or to help the system)

Scanned by CamScanner

PAGE NO.:
DATE:

(*) Production System Characteristics:

Knowledge representation is a set of production rules or operators, a database which is modified in accordance with the rules and a production System Interpreter is used to Controls the operation of the rules. i.e. the 'Control strategy' of a production System, determining the order in which production rules are applied. Such type of System is called a <u>production system</u>

→ Types of production System:

1) Monotonic Production System:

Can't change the sequence of order in which rule applied

In this type of production system application of a rule will never avoid the later application of another rule hence these rules are independent.

2) Non-Monotonic Production System:

not necessary to apply in rule order

In this type of production System changes will occur in problems but it will reversed whereas the order of operations is not critical.

3) Partially Commutative production System:

Partially change

5-3=2
5-2=3

In this case the property that if application of meticulous sequence of rules transforms state x to state y, then any permutation of those rules allowable, also transforms state x into state y.

4) Commutative Production System:

It is a production system i.e. both monotonic and partially commutative.

Teacher's Sign

Scanned by CamScanner

PAGE NO.:
DATE:

Four Categories of production System:

1) Partially Commutative Monotonic: Theorem proving

2) Partially Commutative Non Monotonic: Robot Navigation

3) Non partially Commutative Monotonic: chemical synthesis

4) Non partially Commutative Non Monotonic: Bridge

(1) Partially Commutative, Monotonic:

→ Useful for solving ignorable problems.

→ Ex: Theorem proving

→ They can be implemented without the ability to backtrack to previous states when it is discovered that an incorrect path has been followed.

→ They are good for problems where things get created.

(2) Non Monotonic, Partially Commutative:

→ Useful for problems in which changes occur but can be reversed and in which order of operations is not critical.

→ Example: Robot Navigation, 8-puzzle, blocks World

Suppose the robot has the following operations: go North(N), go East(E), go South(S), go west(W). To reach its goal, it does not matter whether the robot exec N-N-E or N-E-N.

Teacher's Sign

Scanned by CamScanner

Non partially Commutative, Monotonic:
Irreversible.
Can not change the sequence of order in which rule applied.
ex: chemical synthesis.

Non partially Commutative, non Monotonic:
Irreversible
not necessary to apply the rules in order.
ex: Chess game.

Scanned by CamScanner

DELUXE
PAGE NO.:
DATE:

(*) Issues in the Design of Search Programs:

following are Some issues will arise while solving different general purpose search Techniques:

(1) The Direction in which to Conduct the Search (either forward or Backward reasoning) We can search the forward by using the state space from start state to goal state or we can search backward from the goal

(2) How to select applicable rules (matching). generally production System Spends most of their time for rules to apply. So it is difficult to have efficient procedure for matching the rules against ~~the rules against~~ the state.

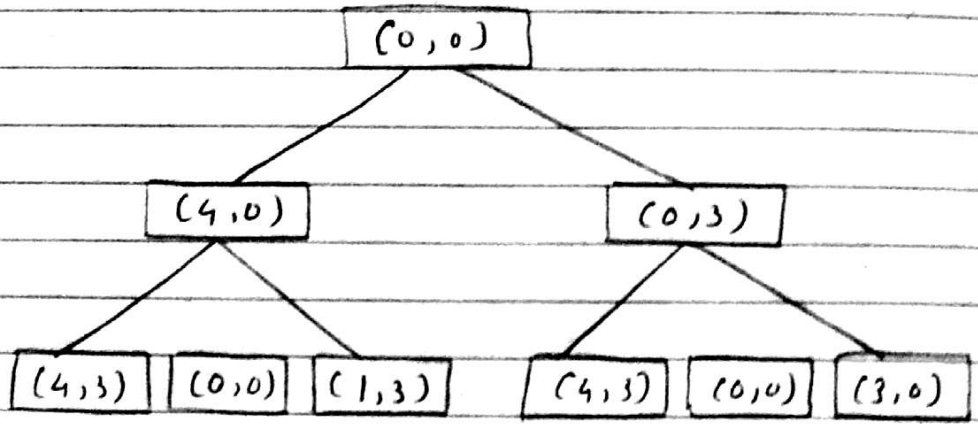
(3) How to represent each node of the search process (the knowledge representation problem and the frame problem) for problems like chess, a node can be fully represented by a simple array. In more complex problem solving, however it is inefficient and/or impossible to represent all of the facts in the world and to determine all of the side effects an action may have.

(4) One other issue we should consider at this point is that of Search trees versus Search graphs. We can think of production rules as generating nodes in a search tree. Each node can be expanded in turn, generating a set of successors.

Teacher's Sign

Scanned by CamScanner

This process continues until a node representing a solution is found.



[fig: 13 A Search tree for the water Jug Problem]

→ The tree shown in fig 13 the node (4,3), representing 4-gallons of water in one Jug and 3-gallons in the other, can be generated either by first filling the 4-gallon Jug and then the 3-gallon one or by filling them in the opposite order.

→ On the third level, the node (0,0) appears. But this is same as the top node, which has already been expanded. So we would like to eliminate them and continue only along the branches.

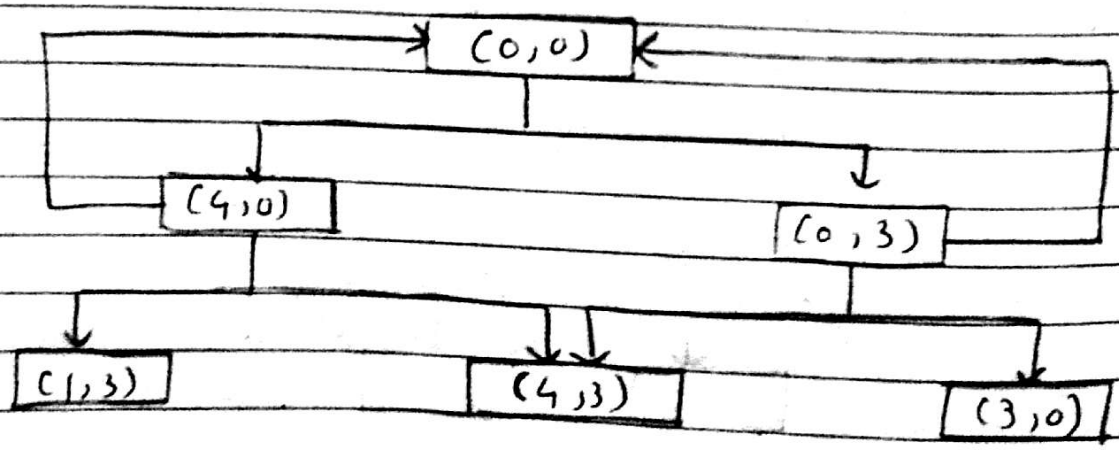


fig: A search graph for the water Jug problem]

Teacher's Sign

Scanned by CamScanner

→ Instead of traversing a Search tree, we traverse a directed graph. This graph differs from a tree in that Several paths may Come together at a node.

Algorithm : Check duplicate nodes

1. Examine the Set of nodes that have been created so far to see if the new node already exists

2. If it does not already exist – Simply add it to the graph just as for a tree.

3. If it does already exist – then do the following:

a) Set the node that is being expanded to point to the already existing-node corresponding to its successor rather than to the new one. The new one can simply be thrown away.

b) If you are keeping track of the best path to each node, then check:
if the new path is better or worse than the old one.

→ If worse : do nothing

→ If better : record the new path as the correct path to use to get to the node and propagate the corresponding change in cost down through successor nodes as necessary.

Teacher's Sign

Scanned by CamScanner

PAGE NO.:
DATE:

Treating the search process as a graph search rather than as a tree search reduces the amount of effort that is spent exploring essentially the same path several times. But it requires additional effort each time a node is generated to see if it has been generated before.

Additional Problems:

1) The Missionaries and Cannibals problem.

Three missionaries and three cannibals find themselves on one side of river.

They have agreed that they would all like to get to the other side. But Missionaries are not sure what else the cannibals have agreed to.

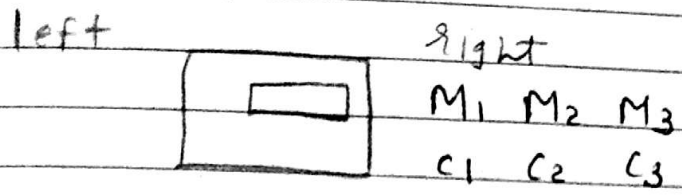
So the missionaries want to manage the trip across the river in such a way that the number of missionaries on either side of the river is never less than the number of cannibals who are on the same side.

The only boat available holds only two people at a time. How can everyone gets across the river without the missionaries risking being eaten?

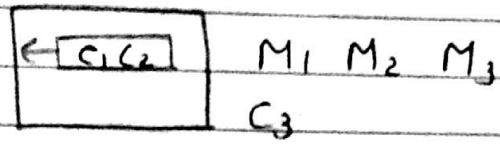
Scanned by CamScanner

Solution :

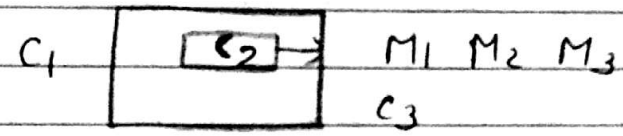
Initial Condition :



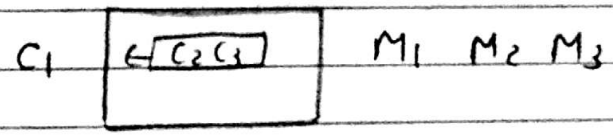
→ Move 2 Cannibals to the right



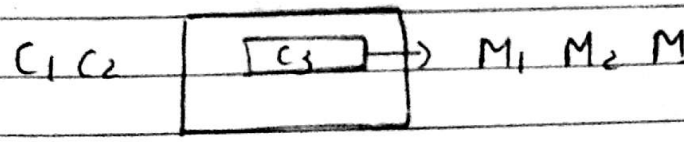
→ Move 1 Cannibal back to the right



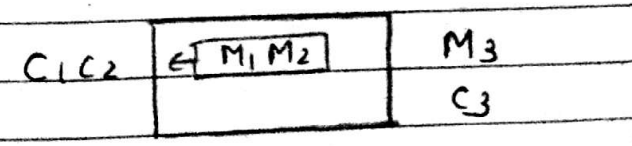
→ Move 2 Cannibals to the left.



→ Move 1 Cannibal back to right.



→ Move 2 missionaries to the left.



→ Move 1 missionary and 1 Cannibal back to the right.

Scanned by CamScanner

DATE:

M_1 C_1 | M_2C_2 → | M_3 C_3

→ Move 2 missionaries to the left

M_1 C_1 | ← M_2 M_3 | C_2 C_3

→ Move 1 Cannibal back to the right

M_1 M_2 M_3 | C_1 → | C_2 C_3

→ Move 2 Cannibals to the left

M_1 M_2 M_3 | ← C_1 C_2 | C_3

→ Move 1 Cannibal back to the right

M_1 M_2 M_3 C_1 | C_2 → | C_3

→ Move 2 Cannibals to the left

M_1 M_2 M_3 C_1 | ← C_2 C_3 |

→ Final position

M_1 M_2 M_3 C_1 C_2 C_3 |

Scanned by CamScanner

PAGE NO.:
DATE:

2) Tower of Hanoi Problem

→ The Tower of Hanoi puzzle was invented by the french Mathematician edouard Lous in 1883.

→ He was inspired by a legend that tells of a Hindu temple where the puzzle was presented to young priests.

→ At the beginning of time, the priests were given three poles and a stack of 64 gold disks, each with a hole in the center and each of different radius.

→ Initially all of the disks were on one pole, each disk resting on the one just larger that it.

→ The priest' task is to move all of the disks from one of the three poles to another.

→ ① only one disk may be moved at a time and all the other disks must be on one of the pole.

→ ② They could never place a larger disk on top of a smaller one.

→ Third pole be used as a temporary resting place for the disks.

→ What is the quickest way for the priests to accomplish their mission?

Teacher's Sign

Scanned by CamScanner

PAGE NO.:
DATE:

→ The priests worked very efficiently, day and night, moving one disk every second.

→ When they finished their work, the legend said, the world will end.

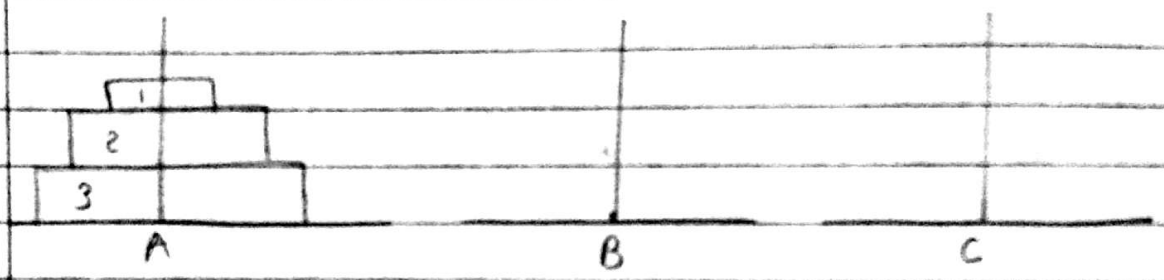
→ The number of moves required to correctly move a tower of 64 disk is $= 2^n - 1$

$= 2^{64} - 1$

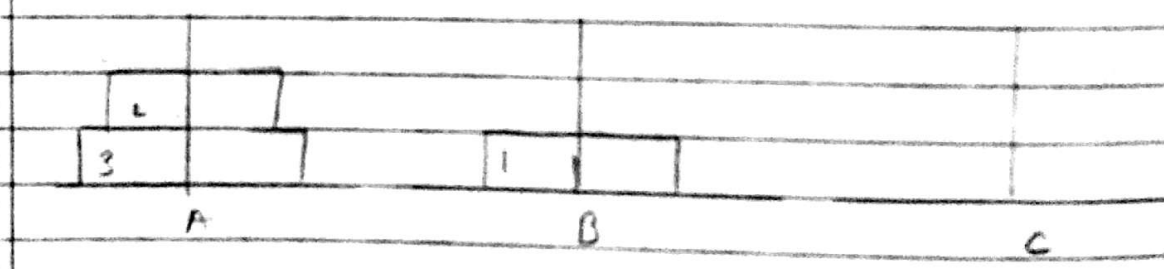
$= 18,446,744,073,709,551,615$

→ At a rate of one move per second that is 584,942,417,355 years

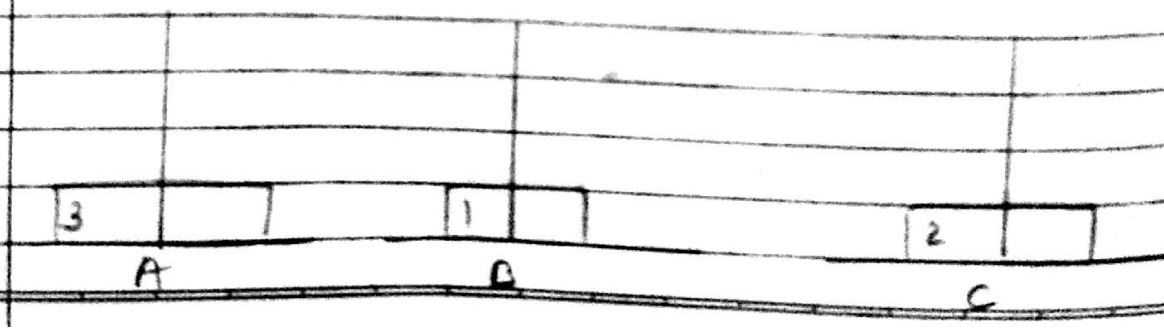
Solution: (for 3 disk) →



Step: 1

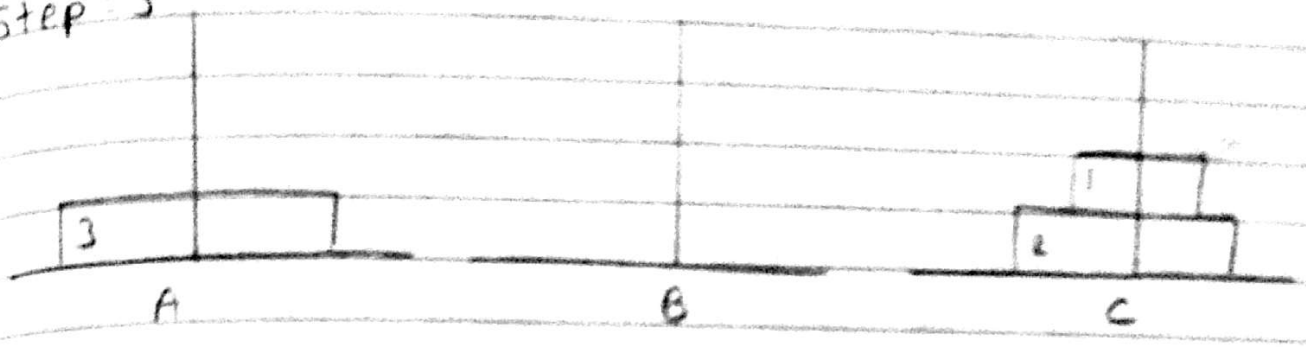


Step: 2

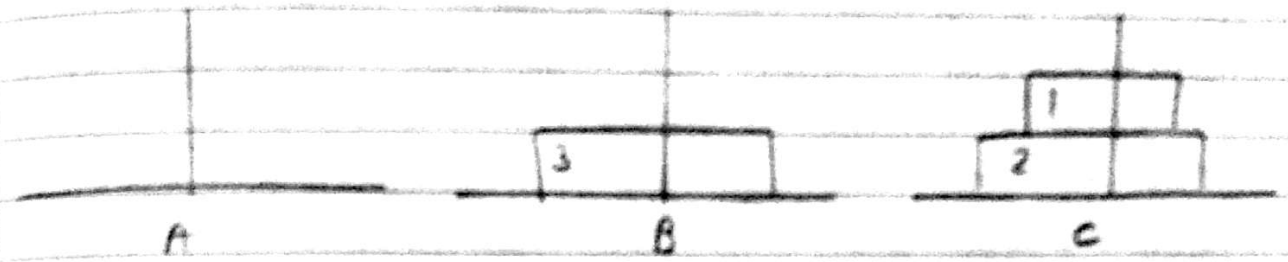


Scanned by CamScanner

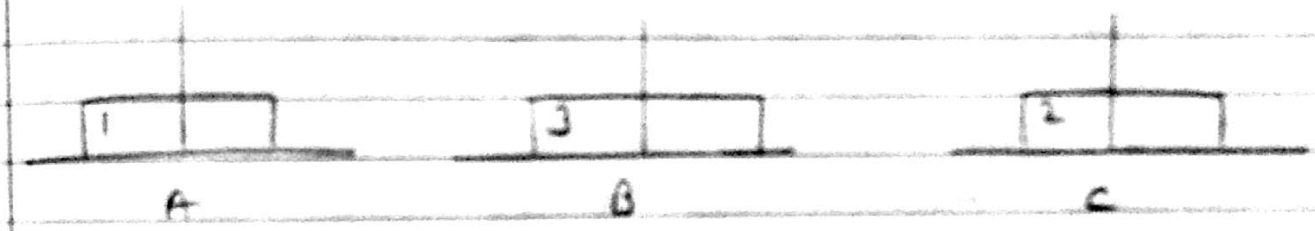
Step 3



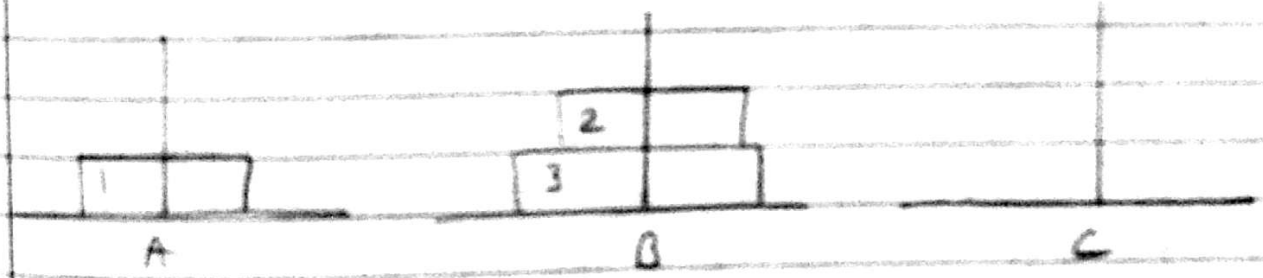
Step 4



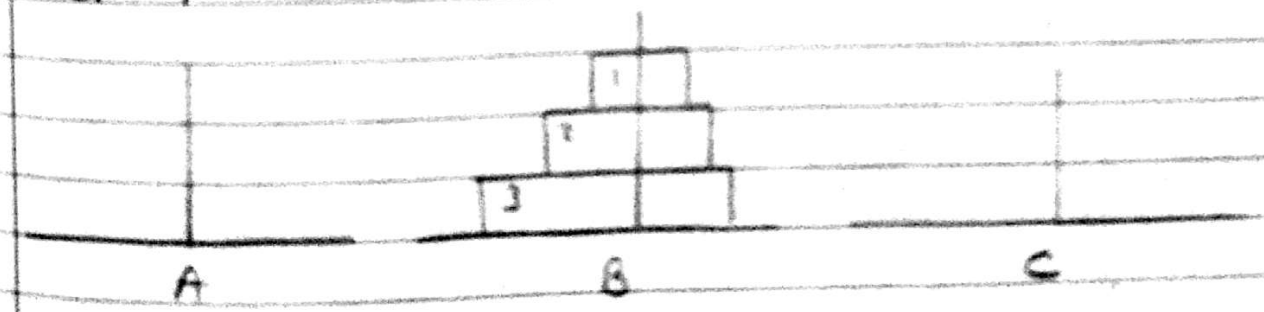
Step : 5



Step 6



Step : 7



3 disk = $2^3 - 1 = 8 - 1 = 7$ number of moves

Scanned by CamScanner

PAGE NO.:
DATE:

3) The monkey and bananas problem:

→ A Hungry Monkey finds himself in a room in which a Bunch of bananas is hanging from ceiling.

→ The monkey, unfortunately, can not, reach the bananas. However, in the room there are also a chair and a stick.

→ The celling is just the right height so that a monkey standing on a chair could knock the bananas down with the stick.

→ The monkey knows how to move around, carry other ~~kein~~ things around, reach for the bananas and wave a stick in the air. What is the best sequence of actions for the monkey to take to acquire lunch?

Solution:

Initial state:

→ Monkey on ground with empty hand bananas suspended.

goal state:

→ Monkey eating.

Actions:

→ climb chair / get off

→ grab x, wave x, eat x.

Teacher's Sign

Scanned by CamScanner

PAGE NO.:
DATE:

Monkey on ground with empty hands.

climb chair

grab stick

Monkey on chair with empty hands

get off chair

drop stick

Monkey on ground with empty hands

Monkey with stick

climb chair

Monkey on chair with stick

wave stick

Banana falls to ground

get off chair

Monkey on ground

drop stick

grab banana

Monkey on ground with banana

eat banana

Monkey eating Banana

Scanned by CamScanner