MC 312 LIST OF PRACTICALS

Program 1: Write a program to solve the 8-Puzzle problem using Generate and Test Strategy.

(as discussed in the class, given a current state, generate all the possible next states, also known as neighbouring states, and check whether any one of them is final state or not, if not, repeat the procedure with neighbouring states).

You can take the current state as:

8	1	3
4		5
2	7	6

And the final state must be:

1	2	3
8		4
7	6	5

Program 2: Write a program to solve the 8-Puzzle problem using DFID (Depth First Iterative Deepening) Strategy.

You can take the current state as:

8	1	3
4		5
2	7	6

And the final state must be:

1	2	3
8		4
7	6	5

Program 3: Write a program to solve the 3- SAT Problem using Variable Neighbourhood Descent Algorithm.

3-SAT Problem is a problem that asks what is the fastest algorithm to tell for a given formula in Boolean Algebra (with unknown variable) is satisfiable.

You can take any function in CNF form of your choice or you can take the function F as:

F= (b V c') Λ (c V d') Λ (b') Λ (a' V e') Λ (c' V e) Λ (c' V d').

Let the initial state be {a=1, b=1, c=1, d=1, e=1} and let the heuristic function be number of clauses that are true in any given state.

You need to apply different move gen functions at every level, starting from most sparse function to the most dense function.

Program 4: Write a program to solve the 3- SAT Problem using Stochastic Hill Climbing Algorithm.

You can take any function in CNF form of your choice or you can take the function F as:

F= (b V c') Λ (c V d') Λ (b') Λ (a' V e') Λ (c' V e) Λ (c' V d').

Let the initial state be {a=1, b=1, c=1, d=1, e=1} and let the heuristic function be number of clauses that are true in any given state.

You can use the SIGMOID function as discussed in the class to calculate the probability at every state. Let the value of parameter "T" be 10, and you can choose any probability threshold of your choice for comparison.

Program 5: Write a program to solve the 8-Puzzle problem using A* algorithm.

You can take the current state as:

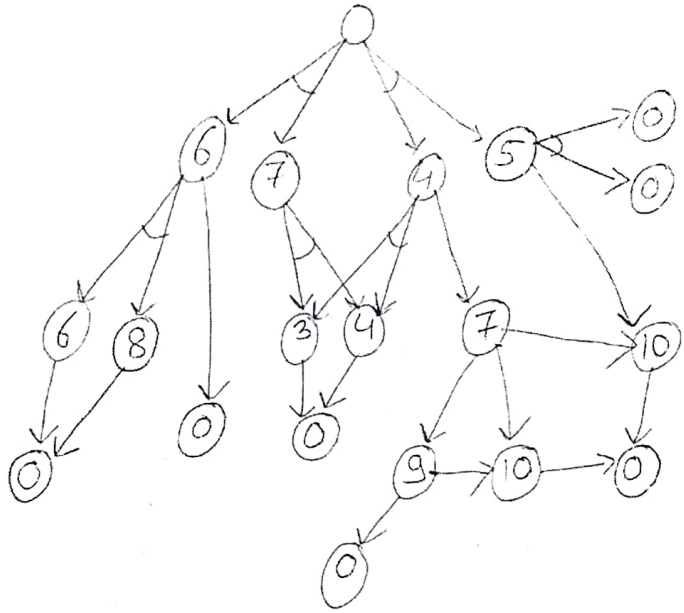
2	8	3
1	6	4
7		5

And the final state must be:

1	2	3
8		4
7	6	5

Take g(n) function as depth of node and h(n) as number of misplaced tiles. At every iteration, the node with minimum value of f(n) (i.e., f(n) = g(n)+h(n)) is selected.

Program-6: Consider the given AND OR graph below. Write a program to solve the above problem represented by the AND OR graph using AO* search algorithm. Assume all edge weights are one. And the number inside every node represents the estimated time required to solve the sub problem at that node. The number 0 inside the node represents that sub problem is completely solved.



Program 7: WAP to find maximum of two/three numbers.

Program 8: WAP to find factorial of a number

Program 9: WAP to find sum of first N numbers

Program 10: WAP to find Fibonacci sequence upto Nth term.

Program 11: Consider the following statements of a knowledge database:

a. Prakash likes food if the food is edible and it tastes sweet.
b. Chocolates tastes sweet.
c. Toffees tastes sweet.
d. Gourd tastes bitter.
e. Chocolates are edible.
f. Toffees are edible.
g. Gourd is edible.

WAP to represent the above statements in Prolog. Further ask a question/query in Prolog such that the output of the question should be:

```
Chocolates
Toffees
No
```