**Experiment No. 04**

**Aim:** Implementation of Hill Climbing Algorithm

# Theory:

Hill climbing is a local search algorithm used for mathematical optimization problems. It starts with an arbitrary solution to a problem and iteratively makes small improvements to it, moving towards a better solution. At each iteration, the algorithm evaluates the neighboring solutions and selects the one that improves the current solution the most. This process continues until no further improvements can be made, and the algorithm reaches a local optimum.

Here is a brief overview of the hill climbing algorithm:

1. **Initialization**: Start with an initial solution to the problem.
2. **Evaluation**: Evaluate the current solution by calculating its fitness or cost.
3. **Neighbor Generation**: Generate neighboring solutions by making small modifications to the current solution. Neighbors are potential solutions that are close to the current one.
4. **Selection**: Choose the best neighboring solution based on its fitness or cost.
5. **Termination**: Check if the stopping criteria are met. This could be a maximum number of iterations, reaching a satisfactory solution, or other criteria.
6. **Update**: If a better solution is found, update the current solution with the selected neighbor and repeat from step 2.

Hill climbing has several variants, including steepest ascent hill climbing (where the steepest ascent neighbor is always selected) and steepest descent hill climbing (where the steepest descent neighbor is always selected).

# Code:

# import copy

# visited\_states = []

# def gn(curr\_state,prev\_heu,goal\_state):

# global visited\_states

# state = copy.deepcopy(curr\_state)

# for i in range(len(state)):

# temp = copy.deepcopy(state)

# if len(temp[i]) > 0:

# elem = temp[i].pop()

# for j in range(len(temp)):

# temp1 = copy.deepcopy(temp)

# if j != i:

# temp1[j] = temp1[j] + [elem]

# if (temp1 not in visited\_states):

# curr\_heu=heuristic(temp1,goal\_state)

# if curr\_heu>prev\_heu:

# child = copy.deepcopy(temp1)

# return child

# return 0

# def heuristic(curr\_state,goal\_state):

# goal\_=goal\_state[3]

# val=0

# for i in range(len(curr\_state)):

# check\_val=curr\_state[i]

# if len(check\_val)>0:

# for j in range(len(check\_val)):

# if check\_val[j]!=goal\_[j]:

# # val-=1

# val-=j

# else:

# # val+=1

# val+=j

# return val

# def sln(init\_state,goal\_state):

# global visited\_states

# if (init\_state == goal\_state):

# print (goal\_state)

# print("solution found!")

# return

# current\_state = copy.deepcopy(init\_state)

# while(True):

# visited\_states.append(copy.deepcopy(current\_state))

# print(current\_state)

# prev\_heu=heuristic(current\_state,goal\_state)

# child = gn(current\_state,prev\_heu,goal\_state)

# if child==0:

# print("Final state - ",current\_state)

# return

# current\_state = copy.deepcopy(child)

# 

# def main():

# global visited\_states

# initial = [[],[],[],['B','C','D','A']]

# goal = [[],[],[],['A','B','C','D']]

# sln(initial,goal)

# main()

# 

**Conclusion:** In conclusion, Hill climbing is a local search algorithm that iteratively improves a solution by selecting the best neighboring solution. Conclusions depend on factors such as sensitivity to the initial solution, the presence of local optima, and the efficiency of local exploration. The algorithm may find satisfactory solutions quickly but is susceptible to getting stuck in local optima.