**Soft Computing Practical Exam**

**Name**:Reeta Soni

**Enrollment number:**170382

**Objective:**

**Write a program to implement Depth Limited Search Algorithm in Python programming Language**

**Algorithm:**

**Depth-Limited Search Algorithm:**

A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

Depth-limited search can be terminated with two Conditions of failure:

Standard failure value: It indicates that the problem does not have any solution.

Cutoff failure value: It defines no solution for the problem within a given depth limit.

**Advantages:**

Depth-limited search is Memory efficient.

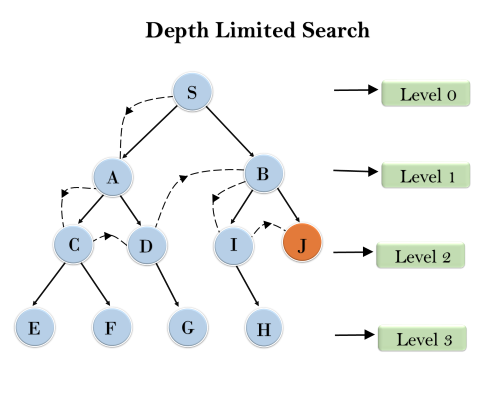
**Disadvantages:**

Depth-limited search also has a disadvantage of incompleteness.

It may not be optimal if the problem has more than one solution.

Example:

Uninformed Search Algorithms



**Code:**

**#depth-limit-search**

**test = {}**

**""""**

**SRRXG**

**RXRXR**

**RRRXR**

**XRXRR**

**RRRRX**

**"""**

**test['S'] = ['2', '6']**

**test['2'] = ['S', '3']**

**test['3'] = ['2','8']**

**test['G'] = ['10']**

**test['6'] = ['S', '11']**

**test['8'] = ['3', '13']**

**test['10'] = ['G', '15']**

**test['11'] = ['6', '12']**

**test['12'] = ['11', '13', '17']**

**test['13'] = ['8', '12']**

**test['15'] = ['10', '20']**

**test['17'] = ['12','22']**

**test['19'] = ['20', '24']**

**test['20'] = ['15','19']**

**test['21'] = ['22']**

**test['22'] = ['17','21','23']**

**test['23'] = ['22', '24']**

**test['24'] = ['19','23']**

**print (test)**

**visited = {str(i) : False for i in range(1,26)}**

**visited['S'] = False**

**visited['G'] = False**

**def dls(start, goal):**

**depth = 0**

**limit = 200**

**OPEN=[]**

**CLOSED=[]**

**OPEN.append(start)**

**visited["S"] = True**

**while OPEN != []: # Step 2**

**if depth<=limit:**

**current = OPEN.pop()**

**if current == goal:**

**print("Desitination/Goal Node Found")**

**return True**

**else:**

**lst = successors(current)**

**for i in lst:**

**if(not(visited[i])):**

**OPEN.append(i)**

**visited[i] = True**

**depth +=1**

**else:**

**print("It was not found within depth limit")**

**return False**

**return False**

**Output:**

**{'S': ['2', '6'], '2': ['S', '3'], '3': ['2', '8'], 'G': ['10'], '6': ['S', '11'], '8': ['3', '13'], '10': ['G', '15'], '11': ['6', '12'], '12': ['11', '13', '17'], '13': ['8', '12'], '15': ['10', '20'], '17': ['12', '22'], '19': ['20', '24'], '20': ['15', '19'], '21': ['22'], '22': ['17', '21', '23'], '23': ['22', '24'], '24': ['19', '23']}**