AI314 - Autonomous Multiagent Systems

GREEN HOUSE WITH MULTIAGENT (ROBOTS)

Spraying Agent

Names	IDs
Ahmed Mohamed Abdel-Rashied	20180028
Eslam Nasser Ghoneim	20180047
Aoss Majed Sultan Zaid	20180432

MAY 22, 2021

SPRING 2021

Dr. Mohamed A. Wahby Shalaby

Contents

Abilities

Pesticide Spraying

Spray Pesticide to clear plants by killing Insects

Water Spraying

Spray water Spraying water with accurate dates and quantities, taking into account not wasting water

Fertilizer Spraying

Fertilizer Spraying on the plants in a regular manner in specific quantities to make the plants grow faster

Older versions

The wheeled spraying machine uses the same Micothon technology as the automatic robot, but is operated by hand – you walk through the glasshouses with the wheeled machine. Thegreat advantage of a Micothon wheeled machine compared to a standard wheeled machine is the patented air-assisted system used during the spraying. This guarantees the best possible coverage of foliage and less use of herbicides and pesticides.

The advantages:

Easy to use

Reduces your annual ppp use by as much as 50%

The machine is propelled automatically so it does not require muscle power to move thespraying robot

Saves an average of 25 hours of labour per hectare

Speed can be adjusted from 0 to 40 metres per minute

Cable rewinds automatically while the spraying robot moves to another row

Long life and low maintenance thanks to the use of stainless steel

Reduces growth inhibition in your crops by up to 2.5%.

Rationality feature

Performance

Spray each group of squares at the specified time and quantity. Move between groups in an orderly way

knowledge of environment

Distribution and the initial location of the agent are known. Spray type and quantities are known foreach group. Steps is known (Right, Left, go forward)

Precept Sequence	Actions
[Group A, box 1]	Spray, Right
[Group A, box 2]	Spray, Right
[Group A, box 3]	Spray, Right
[Group A, box 4]	Spray, go forward
[Group A, box 1]	Spray, Left
[Group A, box 2]	Spray, Left
[Group A, box 3]	Spray, Left
[Group A, box 4]	Spray, save group

PEAS:

Performance measure: Safe, fast, Lower labor costs , Reducing the waste of pesticide costs, obviously reducing human health risks

Environment: Green house ,user interfaces

Actuators: Stop button, large reservoir, pump, four steering valves, internal microprocessor receivinginput signals, worm drive, wheel, two stores

Sensors: Shock sensors front and back, Cameras, engine sensors, keyboard, Internet

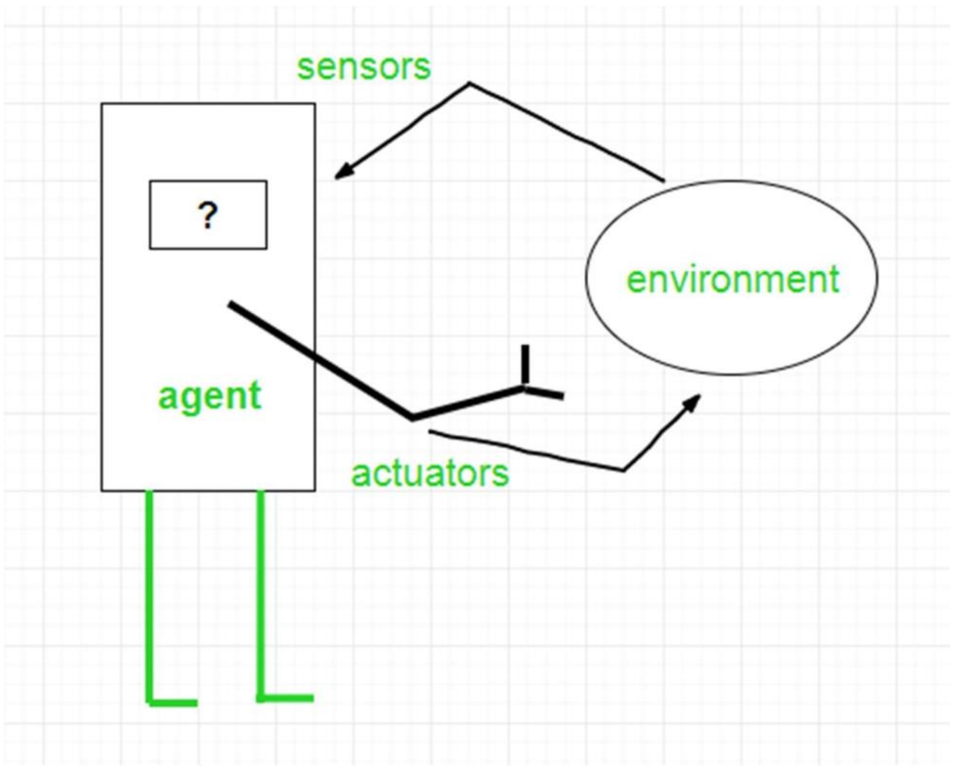
Agent	Performance measure	Environment	Actuators	Sensors
Automated Spraying Agent	Safe, fast, Lower labor costs , Reducing the waste of pesticide costs,	Green house ,user interfaces	Stop button, large reservoir, pump, four steering valves, internal microprocessor receiving input	Shock sensors front and back, Cameras, engine sensors,

	obviously reducing human health risks		signals, worm drive, wheel, two stores	keyboard

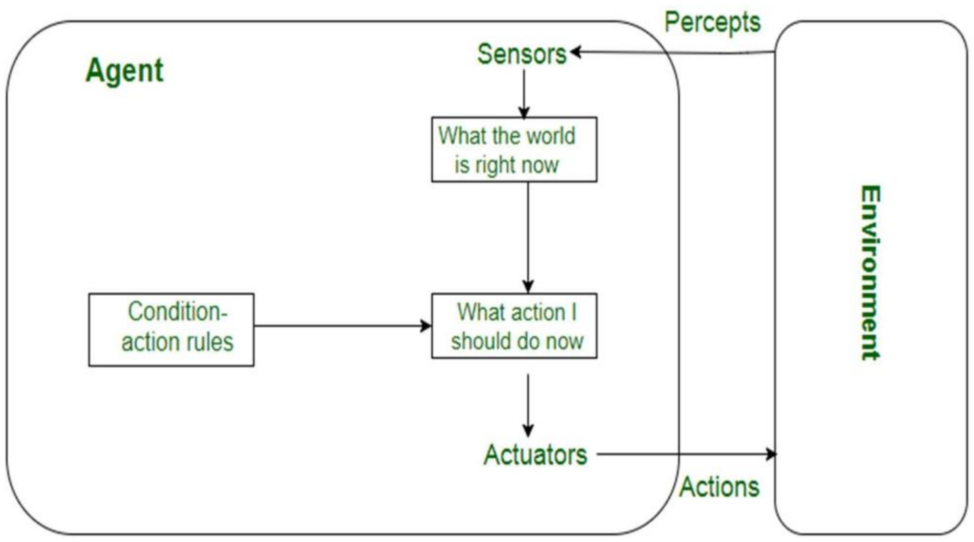
Environment type:

1. **Fully observable:**
   Because All data are known and the environment is fully accessible.
2. **Deterministic:**
   Because All steps are known.
3. **Sequential**
   Because is Dependent on previous actions.
4. **Semidynamic :**
   Because the environment itself does not change with the passage of time but the agent's performance score does.
5. **Discrete:**
   Because A limited number of distinct, clearly defined percepts and actions.
6. **Single agent:**
   Because An agent operating by itself in an environment .

Types of Agents:



Simple reflex agents



we can summarize part of the table by formulating commonly occurring patterns as condition action rules:

We used these types because the robot takes the data and then dictates the tank to what it needs, afterthat, it goes to the group that hirs it on it and then walks on each square after that, it goes to the group after it The group sprinkles and comes back

STATE-SPACE formulation

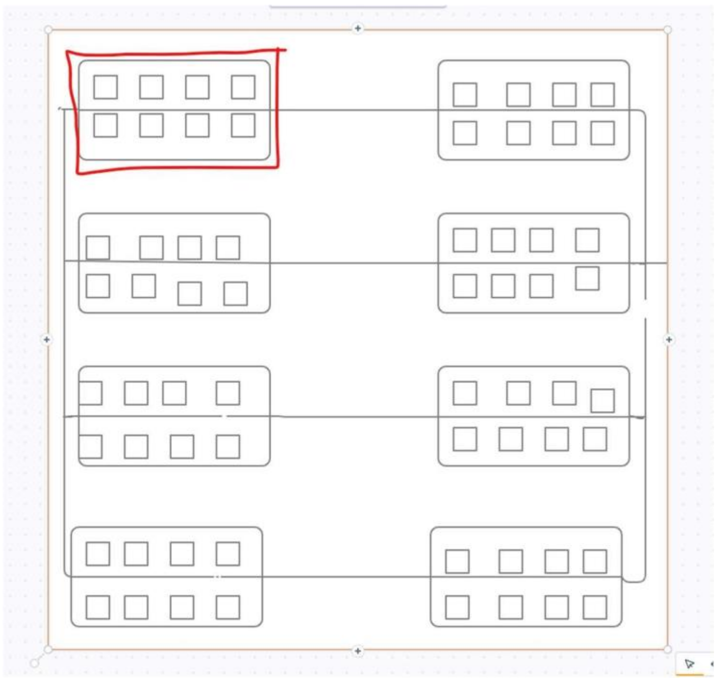
1. The initial state
   The agent starts from group 1 every time until he reaches the goal and returns to the same group again.
2. The possible actions
   - Introduction
     Agent will move above groups starting from group 1. Agent path will be in the middle of each group horizontally.
   - Spraying
     Agent can spray right, left, or in both directions. Two-way spraying is used for water or spraying the whole group, while one-way spraying is used when defining small targets, for example spraying an insecticide of some kind in a square or two out of 8 in group 6
   - Moves
     Agent can move in middle of each group horizontally. Each group is connected to both horizontal and vertical groups in order in the same line.
3. The transition model
   Agent can move from any group to horizontal and vertical group
   Example:

1	5
2	6
3	7
4	8

   Agent will start from group 1 it can move to group 5 horizontally or group 2 vertically. It can't move from group 1 to group 6 directly.
4. The goal test
   Each group will be saved when all the inner squares to be sprayed are completed
5. The path cost
   This process is done by calculating the length and width of each group and the spaces between each group, horizontally and vertically. Also, the length and width of each square within the group and the spaces between them

   EX:
   Each group will be 2.5m*15m Each square is 1*1m. space among squares are 0.5m.
   Space between each group horizontally 2m and vertically 1m. So Move horizontally from group1 to 5 will cost 15+15+2=32m, While Move vertically from group 1 to 2 will cost 2.5+2.5+1=6m

Diagrams of the complete state space



The red state is the initial state. Agent can move to any line in the cycle
Each group contains squares. Agent will move in middle. It can spray right, left or in both sides

This is another diagram shows all possible directed ways (arrows) that agent can move in

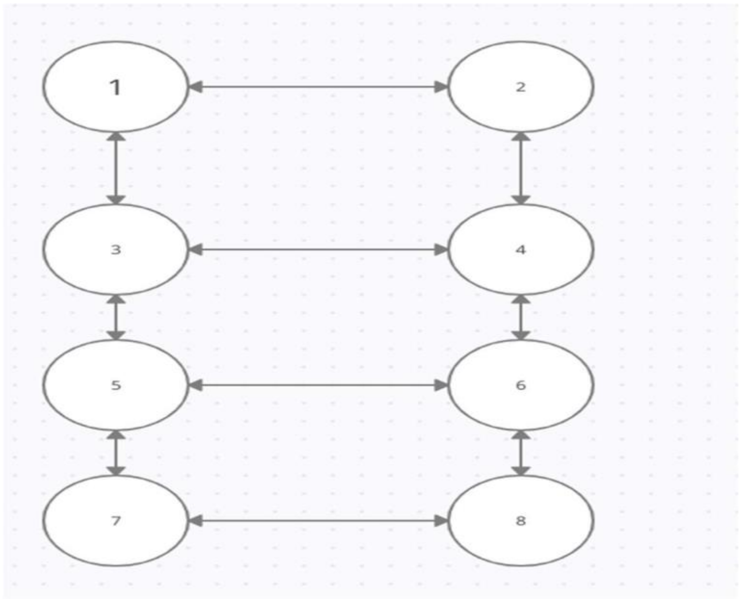
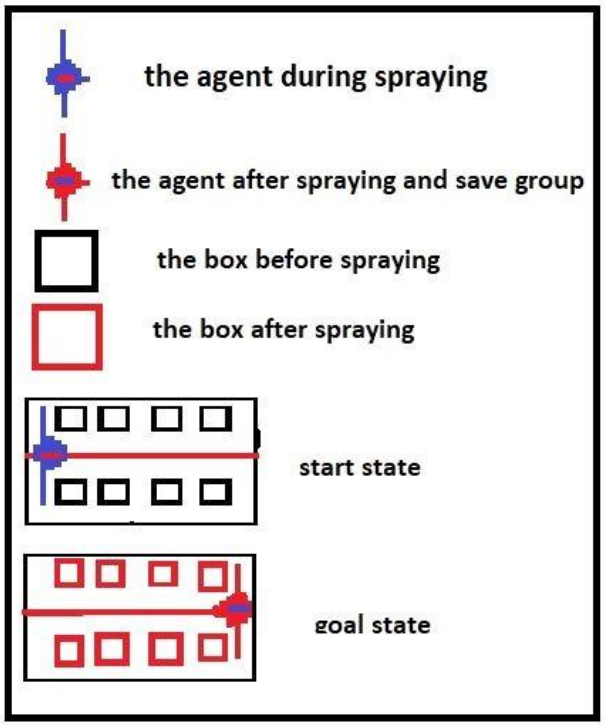
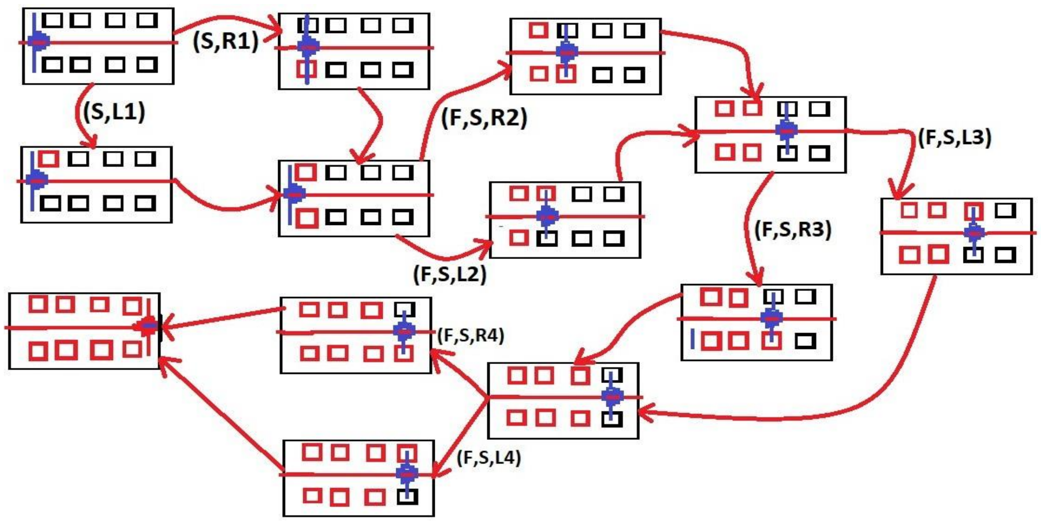


Diagram of spraying:



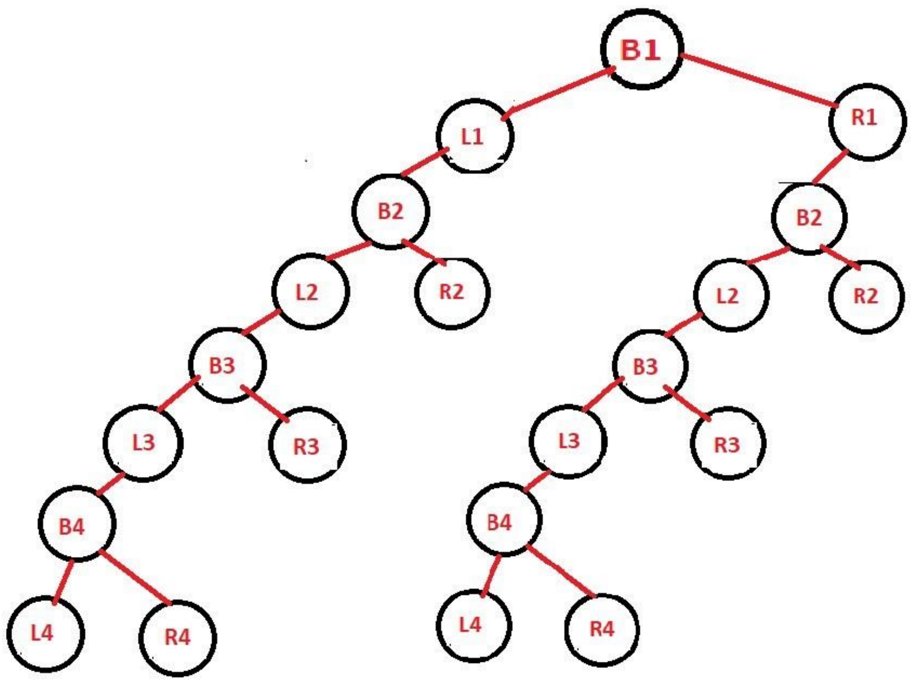


S spraying
LN L is a left , N is number of box
RN R is a right , N is number of
boxF go forword

Search :

In spraying:

- Tree search:



B is the

boxL is a

left

R is a right

If we search about goal (R_4):

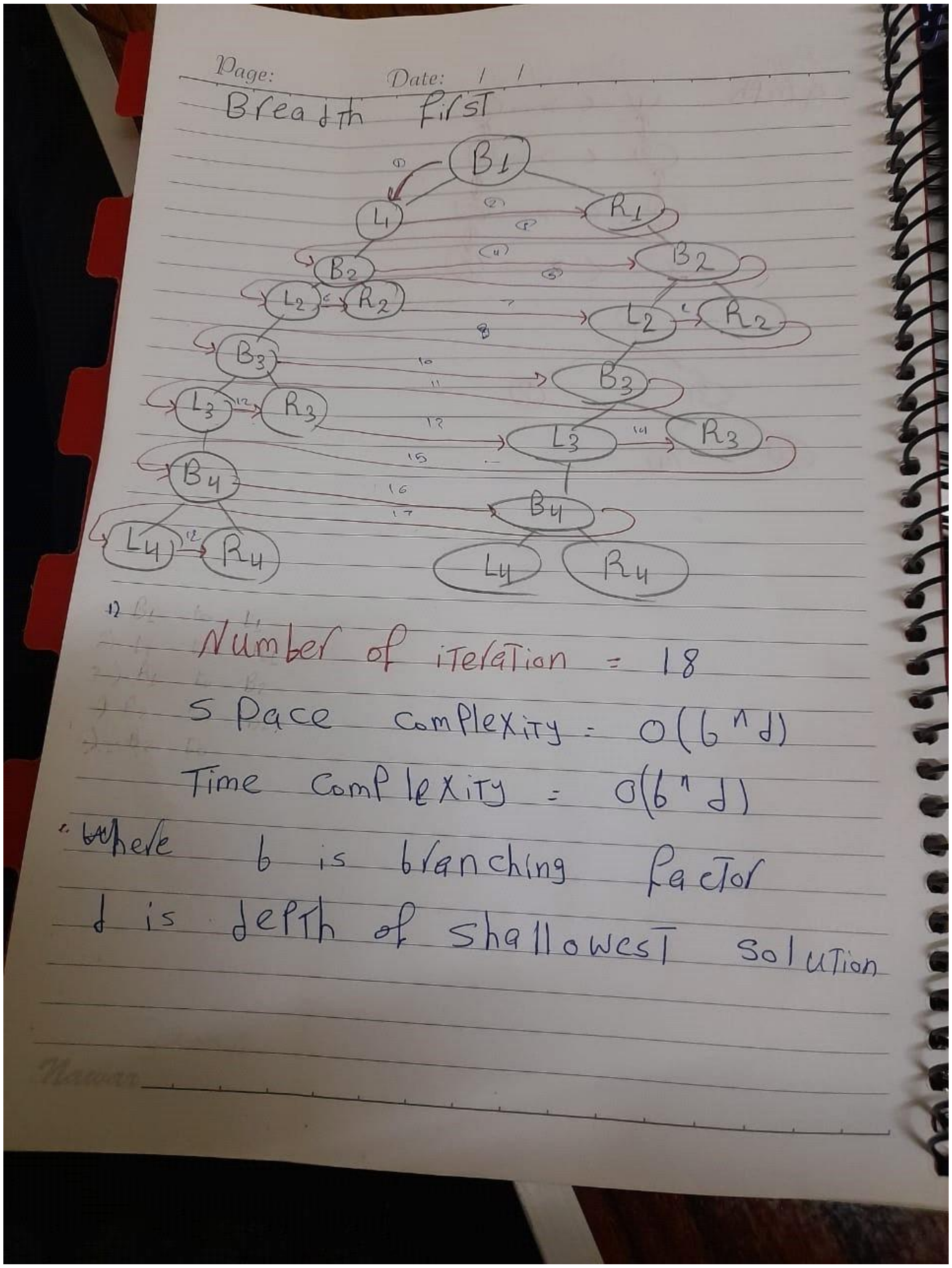
- B1
- B1L1
- B1L1B2
- B1L1B2L2
- B1L1B2L2B3
- B1L1B2L2B3L3
- B1L1B2L2B3L3B4
- B1L1B2L2B3L3B4R4

OR

- B1
- B1R1
- B1R1B2
- B1R1B2L2
- B1R1B2L2B3
- B1R1B2L2B3L3
- B1R1B2L2B3L3B4
- B1R1B2L2B3L3B4R4

we can use any way because both have same way.

Breadth first:



Breadth first

Number of iteration = 18

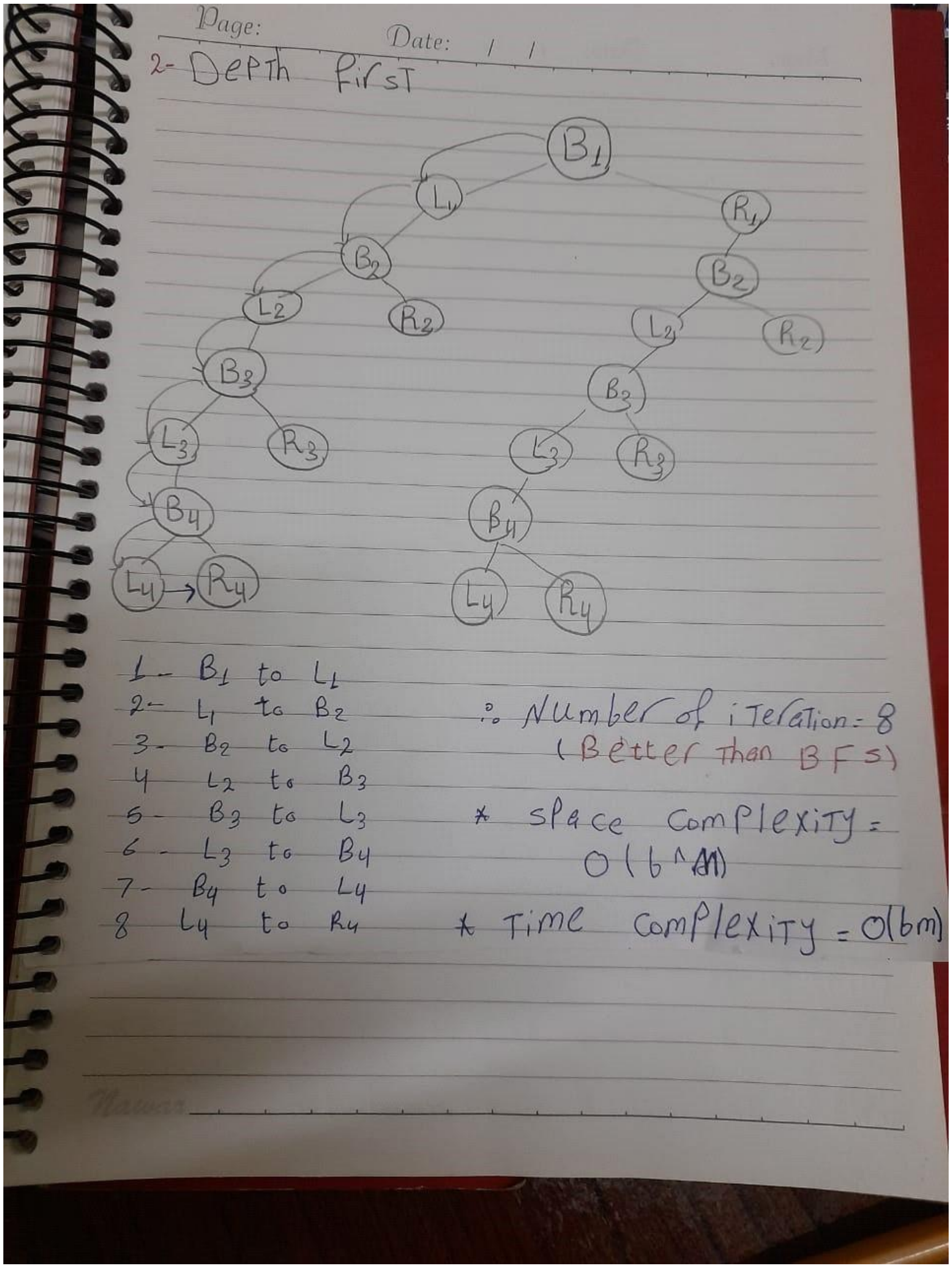
Space complexity : $O(b^d)$

Time complexity : $O(b^d)$

where b is branching factor

d is depth of shallowest solution

Depth first :


B1
L1
R1
B2
L2
R2
B3
L3
R3
B4
L4
R4
B1
R1
B2
L2
R2
B3
L3
R3
B4
L4
R4

Page: Date: / /

2- Depth first

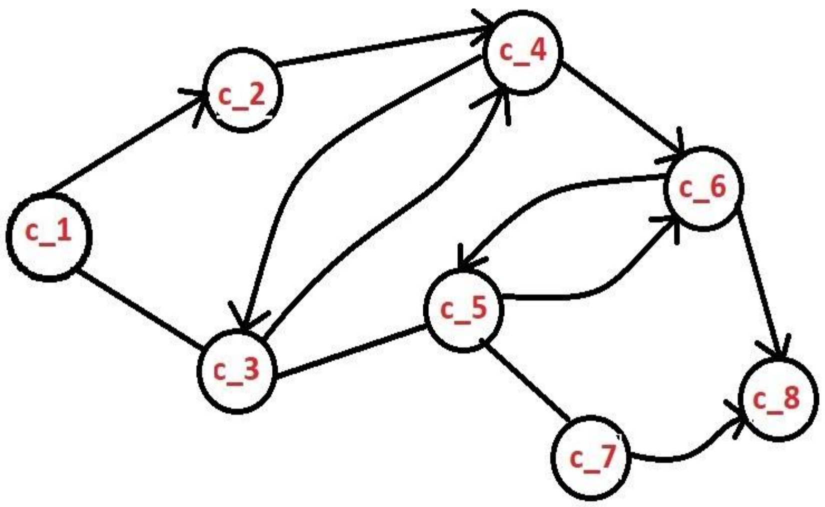
1- B_1 to L_1
2- L_1 to B_2
3- B_2 to L_2
4 L_2 to B_3
5- B_3 to L_3
6- L_3 to B_4
7- B_4 to L_4
8 L_4 to R_4

∴ Number of iteration = 8
(Better than BFS)

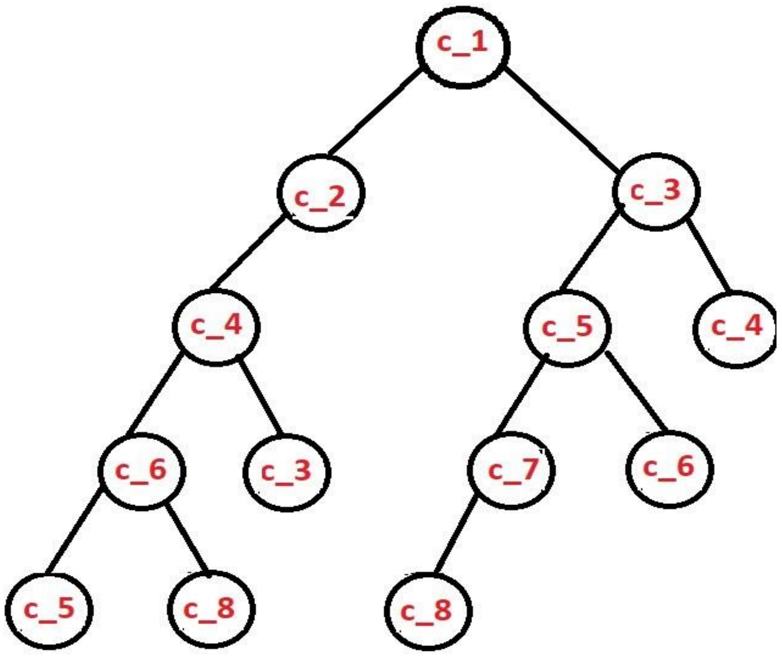
* space complexity = $O(b^M)$

* Time complexity = $O(bm)$

In move between classes or groups:Graph:



Tree search :



If we search about goal (c_8):

- C_1
- C_1C_2
- C_1C_2C_4
- C_1C_2C_4C_6
- C_1C_2C_4C_6C

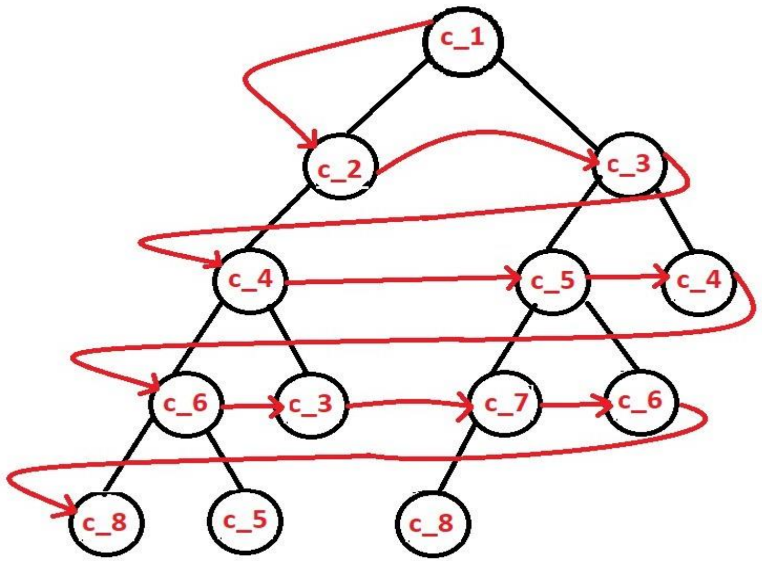
_8OR

- C_1
- C_1C_3
- C_1C_3C_5
- C_1C_3C_5C_7
- C_1C_3C_5C_7C_8

-we can use any way because both have same cost.

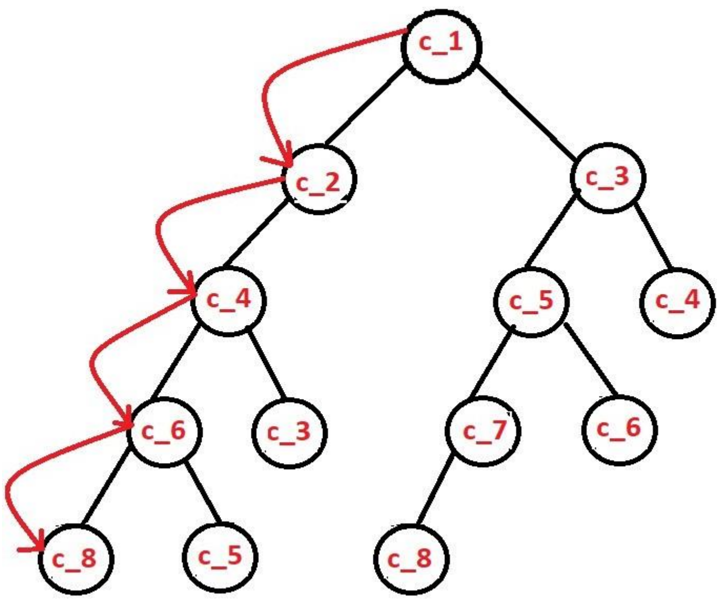
If we use Uninformed (Blind) Search Strategies :

- **Breadth first:**
- **Tree:**



- Number of iterations =10
- space complexity =O(b^d)
- time complexity = O(b^d)

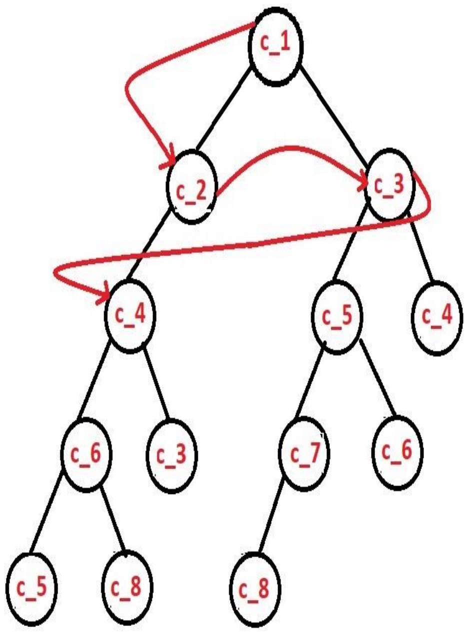
- **Depth first :**



Number of iterations =4 space complexity =O(b^m) time complexity = O(bm)

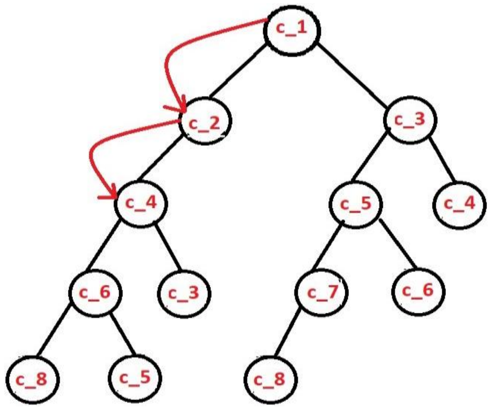
- **We notice that in the Breadth it takes a longer way than the Depth because the Depth check the left branch first until it gets the target. If it does not get the goal, it searches the right branchafter it has finished from the left branch, while Breadth checks alevel by level to get the goal.**
- **another example:**
- **If we search about goal (c_4):**

Breadth first :



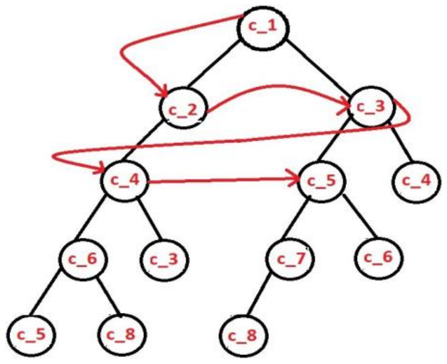
Number of iterations =3 space complexity =O(b^d) time complexity = O(b^d)

Depth first :



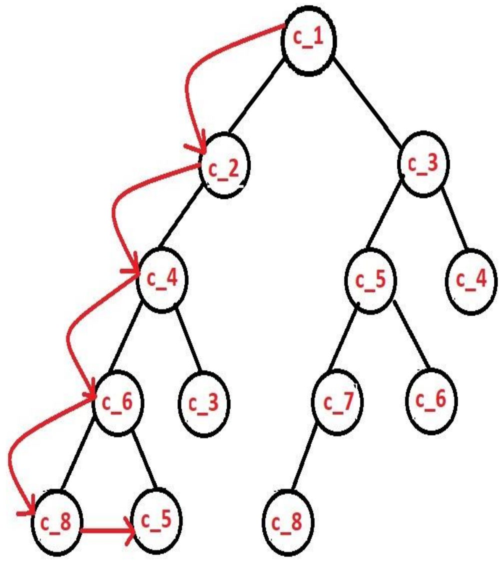
Number of iterations =2 space complexity =O(b^m) time complexity = O(bm)

- **We also notice that in the Breadth it takes a longer way than the Depth.**
- **However, this does not always happen. For example, when searching for the target(c_5), we will notice that Breadths better than Depth. But in this we will use the Depth, if we search about goal (c_5):**
    - **Breadth first:**



Number of iteration =4
space complexity
=O(b^d)time complexity
= O(b^d)

- **Depth first :**



Number of iteration =5
space complexity
=O(b^m)time complexity
= O(bm)

by comparison:

Breadth-First Search:

-How much space does the fringe take?

- **Has roughly the last tier, so O(b^5**

)Is it complete?

- **the node must be finite if a solution exists, so yes!Is it optimal?**

- **yes, it is optimalDepth-**

First Search:

How much space does the fringe take?

- **Only has siblings on path to root, so O(b^5)Is it complete?**

- **node could be infinite, so only if we preventcycles**

- **Is it optimal?**

- **No, it finds the “leftmost” solution,regardless of depth or cost**

Full graph

```
import networkx as nx
import matplotlib.pyplot as plt

graph=nx.DiGraph()
graph.add_edges_from([
                        ('A','B'),('A','C')
                       ,('B','A'),('B','D')
                       ,('C','A'),('C','D'),('C','E')
                       ,('D','B'),('D','C'),('D','F')
                       ,('E','C'),('E','F'),('E','G')
                       ,('F','D'),('F','E'),('F','H')
                       ,('G','E'),('G','H'),('G','I')
                       ,('H','F'),('H','G'),('H','J')
                       ,('I','G'),('I','J')
                       ,('J','H'),('J','I')
                                        ] )

pos =nx.spring_layout(graph)
nx.draw_networkx_nodes(graph,pos, node_size=100)
nx.draw_networkx_edges(graph,pos , edgelist=graph.edges(),edge_color='black')
nx.draw_networkx_labels(graph,pos)

plt.show()
```

code to search

```
import collections
import heapq

def spray(edges, source, target):
  # create graph
    graph = collections.defaultdict(list)
    for l, r, c in edges:
        graph[l].append((c,r))
        #create queue to know visited nodes
    queue, visited = [(0, source, [])], set()
    # biuld heap
    heapq.heapify(queue)
    # traverse graph with BFS
    while queue:
        (cost, node, path) = heapq.heappop(queue)
        # visit the node if it was not visited before
        if node not in visited:
            visited.add(node)
            path = path + [node]
            # hit the target
            if node == target:
                return (cost, path)
            # visit neighbours
            for c, neighbour in graph[node]:
                if neighbour not in visited:
                    heapq.heappush(queue, (cost+c, neighbour, path))
    return float("inf")

if __name__ == "__main__":
    edges = [
        ('A','B',32),
        ('A','C',6),
        ('B','A',32),
        ('B','D',6),
        ('C','A',6),
        ('C','D',32),
        ('C','E',6),
        ('D','B',6),
        ('D','C',32),
        ('D','F',6),
        ('E','C',6),
        ('E','F',32),
        ('E','G',6),
        ('F','D',6),
        ('F','E',32),
        ('F','H',6),
        ('G','E',6),
        ('G','H',32),
        ('G','I',6),
        ('H','F',6),
        ('H','G',32),
        ('H','J',6),
        ('I','G',6),
        ('I','J',32),
        ('J','H',6),
        ('J','I',32)
    ]
    start= str(input("Enter initial state:\t"))
    end=str(input("Enter end state:\t"))
    spray_type=str(input("Enter spray type:\t"))
    start=start.capitalize()
    end=end.capitalize()
    start=start[0]
    end=end[0]
    data=spray(edges, start, end)
    print (data,"\t spray ", spray_type)
```

path GUI

```
graph=nx.DiGraph()
data=data[1]
print('path: ',data)
data2=[]
i=1
while i < len(data):
    data2.append((data[i-1],data[i]))
    i+=1
graph.add_edges_from(data2 )
pos =nx.spring_layout(graph)
nx.draw_networkx_nodes(graph,pos, node_size=100)
nx.draw_networkx_edges(graph,pos , edgelist=graph.edges(),edge_color='black')
nx.draw_networkx_labels(graph,pos)

plt.show()
```

Code:

Link with GUI

https://colab.research.google.com/drive/1jEjNKhZdctkIofI82774G_0vN9XDfSEE?usp=sharing

Resources

https://www.researchgate.net/publication/320716348_Autonomous_Pesticide_Spraying_Robot_for_use_in_a_Greenhouse