

Q1

a)

i) The Nave Auction Algorithm is a technique for solving linear programs, which are optimization issues involving determining the greatest or lowest value of a linear objective function under a number of linear constraints. In order to minimize the objective function while still meeting all constraints, the method iteratively assigns variables to each constraint.

ii) Time complexity

Numeric Stability

Sensitive to the initial price

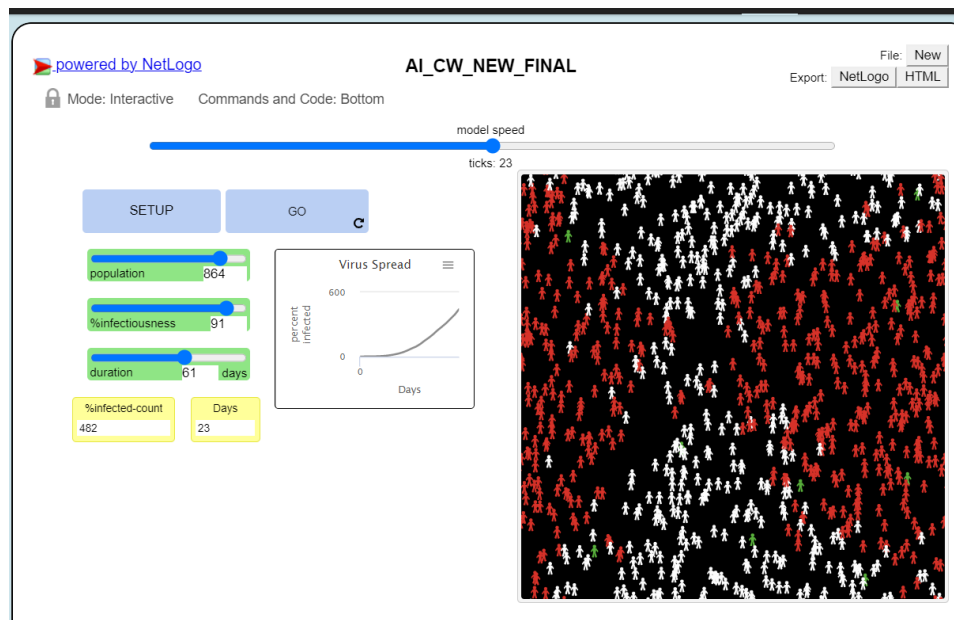
Limited Flexibility

Lack of guarantee

iii)

b)

ii) model Screen shot



iii)

Code

```
globals [%infected-count]

to setup
  clear-all
  reset-ticks
  create-turtles Population
  [
    setxy random-xcor random-ycor
    set shape "person"
    set color white
  ]
  repeat 10 [
    ask turtle random 100 [set color red]
  ]
  repeat 10 [
    ask turtle random 100 [set color green]
  ]
end

to go
  tick
  ask turtles
  [rt random 100 lt random 100 fd random 100]
  ask turtles with [color = red]
  [
    ask other turtles-here [if random 100 < %infectiousness [set color red] ]
  ]

  set %infected-count count turtles with [color = red]
  if %infected-count = count turtles [stop]
end
```

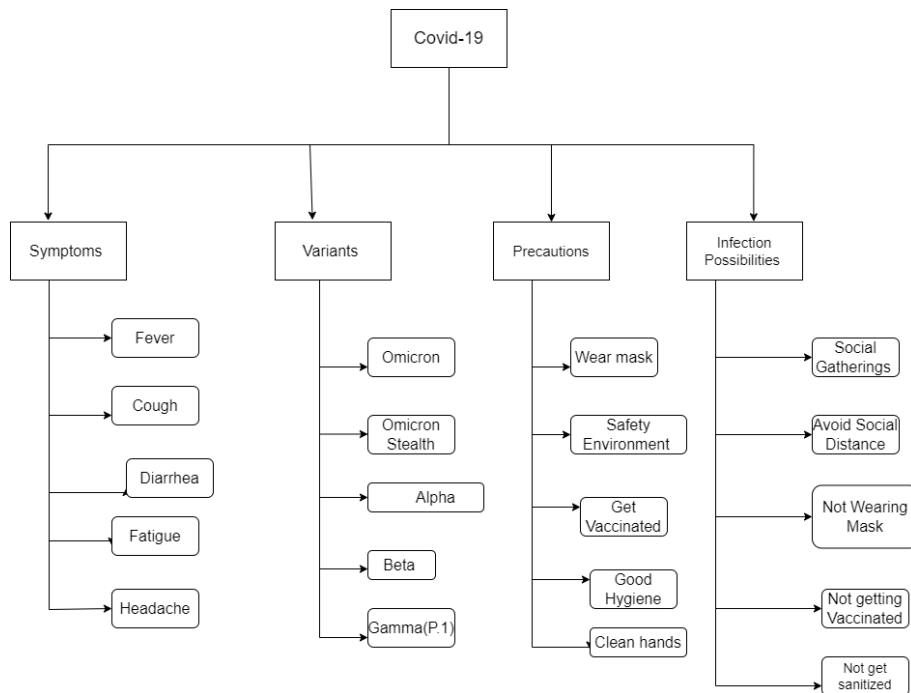
- In setup command population will be created and I indicated “white” people as healthy and “red” people as the people who infected virus and “green” people as the people who are immune. The Program will create 10 immune people and 1 infected person .
- In go command we ask turtle to move around and if red turtle is touching the white turtle we turn white turtle to red to indicate they are infected. Program will stop when every turtles turn red

Q2

1)The Scope I attempt to cover from this knowledge model is about Coronavirus and Different variants, Infection Possibilities, Precautions and Symptoms

- 1) What are the Different types of variants in Coronavirus?
- 2) What are the precautions of Coronavirus?
- 3) What are the Symptoms of Coronavirus?
- 4) What are the infection Possibilities of Coronavirus?
- 5) What is the major symptom of Coronavirus ?

2)



References

Morgan, K.K. (n.d.). *Variants of Coronavirus*. [online] WebMD. Available at: <https://www.webmd.com/covid/coronavirus-strains> [Accessed 20 Dec. 2022].

World Health Organization (2021). *Advice for Public*. [online] Who.int. Available at: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public>.

Centers for Disease Control and Prevention (2022). *Coronavirus Disease 2019 (COVID-19) – Symptoms*. [online] Centers for Disease Control and Prevention. Available at: <https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html>.

www.who.int. (n.d.). *Coronavirus disease (COVID-19): How is it transmitted?* [online] Available at: <https://www.who.int/news-room/questions-and-answers/item/coronavirus-disease-covid-19-how-is-it-transmitted#:~:text=Current%20evidence%20suggests%20that%20the>.

3)

```
3) <rdf:RDF

  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:Covid-19="http://www.example.com/Covid-19#">

  <!-- OWL Header -->
  <owl:Ontology rdf:about="http://www.example.com/Covid-19">
  <dc:title> Coronavirus Ontology</dc:title>
  <dc:description>Coursework ontology </dc:description>
  </owl:Ontology>

  <!-- Define the hasVariants property -->
  <owl:ObjectProperty rdf:about="http://www.example.com/Covid-19#hasVariants"/>

  <!-- Define the hasSymptoms property -->
  <owl:ObjectProperty rdf:about="http://www.example.com/Covid-19#hasSymptoms"/>

  <!-- Define the hasPrecautions property -->
  <owl:ObjectProperty rdf:about="http://www.example.com/Covid-19#hasPrecautions"/>

  <!-- Define the hasinfectionpossiblites property -->
  <owl:ObjectProperty rdf:about="http://www.example.com/Covid-
  19#hasInfectionPossibilities"/>

  <!-- OWL Class Definition - Covid-19 variants -->
  <owl:Class rdf:about="http://www.example.com/Covid-19#variants">
  <rdfs:label> Covid-19 type</rdfs:label>
  <rdfs:comment>The class of all Covid-19 variants.</rdfs:comment>
  </owl:Class>

  <!-- OWL Class Definition - Covid-19 Symptoms -->
  <owl:Class rdf:about="http://www.example.com/Covid-19#Symptoms">
  <rdfs:label> Covid-19 Symptoms</rdfs:label>
  <rdfs:comment>The class of all Covid-19 Symptoms.</rdfs:comment>
  </owl:Class>

  <!-- OWL Class Definition - Covid-19 Precautions -->
  <owl:Class rdf:about="http://www.example.com/Covid-19#precaution">
```

```

<rdfs:label> Covid-19 Precautions</rdfs:label>
<rdfs:comment>The class of all Covid-19 Precautions.</rdfs:comment>
</owl:Class>

<!-- OWL Class Definition - Covid-19 InfectionPossibilities -->
<owl:Class rdf:about="http://www.example.com/Covid-19#Infectionpossibilities">
<rdfs:label> Covid-19 Infection Possibilities</rdfs:label>
<rdfs:comment>The class of all Covid-19 Infection Possibilities</rdfs:comment>
</owl:Class>

<!-- OWL SubClass Definition - Omicron -->
<owl:Class rdf:about="http://www.example.com/Covid-19#Omicron">

    <!-- Omicron is a subclassification of Corona virus varients -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#variants"/>

    <rdfs:label>Omicron</rdfs:label>
    <rdfs:comment>Omicron etc</rdfs:comment>

<rdfs:subClassOf>
    <owl:Restriction>
        <owl:InfectionPossibilities
rdf:resource="http://www.example.com/Covid-19#Notwearmask"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<!-- Define the Omicron class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Omicron">
<rdf:type rdf:resource="http://www.example.com/covid-19#variants"/>
</rdf:Description>

<!-- OWL SubClass Definition - Omicron Stealth -->
<owl:Class rdf:about="http://www.example.com/Covid-19#OmicronStealth">

    <!-- OMicrona Stelth is a subclassification of Corona virus varients -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#variants"/>

    <rdfs:label>Omicron Stealth</rdfs:label>
    <rdfs:comment>Omicron Stealth</rdfs:comment>

```

```

<rdfs:subClassOf>
  <owl:Restriction>
    <owl:InfectionPossibilities
rdf:resource="http://www.example.com/Covid-19#Socialgatherings"/>
    </owl:Restriction>
  </rdfs:subClassOf>

</owl:Class>
<!-- Define the Omicron Stealth class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#OmicronStelath">
<rdf:type rdf:resource="http://www.example.com/covid-19#variants"/>
</rdf:Description>

<!-- OWL SubClass Definition - Alpha-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Alpha">

  <!-- Alpha is a subclassification of Corona virus varients -->
  <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#variants"/>

  <rdfs:label>Alpha</rdfs:label>
  <rdfs:comment>Alpha etc</rdfs:comment>

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:InfectionPossibilities
rdf:resource="http://www.example.com/Covid-19#Socialgatherings"/>
      </owl:Restriction>
    </rdfs:subClassOf>

</owl:Class>
<!-- Define the Alpha class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Alpha">
<rdf:type rdf:resource="http://www.example.com/covid-19#variants"/>
</rdf:Description>

<!-- OWL SubClass Definition - Beta-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Beta">

  <!-- Beta is a subclassification of Corona virus varients -->
  <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#variants"/>

  <rdfs:label>Beta</rdfs:label>
  <rdfs:comment>Beta etc</rdfs:comment>

```

```

    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:InfectionPossibilities
rdf:resource="http://www.example.com/Covid-19#Notgetvaccine"/>
        </owl:Restriction>
      </rdfs:subClassOf>
    </owl:Class>

<!-- Define the Beta class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Beta">
<rdf:type rdf:resource="http://www.example.com/covid-19#variants"/>
</rdf:Description>

<!-- OWL SubClass Definition - Gamma-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Gamma">

  <!-- Gamma is a subclassification of Corona virus varients -->
  <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#variants"/>

  <rdfs:label>Gamma</rdfs:label>
  <rdfs:comment>Gamma etc</rdfs:comment>

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:InfectionPossibilities
rdf:resource="http://www.example.com/Covid-19#Notgetvaccine"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

<!-- Define the Gamma class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Gamma">
<rdf:type rdf:resource="http://www.example.com/covid-19#variants"/>
</rdf:Description>

<!-- OWL SubClass Definition - Cough-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Cough">

  <!-- Cough is a subclassification of Corona virus Symptoms -->
  <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Symptoms"/>

```



```

    <rdfs:label>Cough</rdfs:label>
    <rdfs:comment>Cough etc</rdfs:comment>

</owl:Class>

<!-- Define the Cough class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Cough">
<rdf:type rdf:resource="http://www.example.com/covid-19#Symptoms"/>
</rdf:Description>

<!-- OWL SubClass Definition - Headache-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Headache">

    <!-- Headache is a subclassification of Corona virus Symptoms -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Symptoms"/>

    <rdfs:label>Headache</rdfs:label>
    <rdfs:comment>Headache etc</rdfs:comment>

</owl:Class>

<!-- Define the Headache class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Headache">
<rdf:type rdf:resource="http://www.example.com/covid-19#Symptoms"/>
</rdf:Description>

<!-- OWL SubClass Definition - Diarrhea-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Diarrhea">

    <!-- Diarrhea is a subclassification of Corona virus Symptoms -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Symptoms"/>

    <rdfs:label>Diarrhea</rdfs:label>
    <rdfs:comment>Diarrhea etc</rdfs:comment>

</owl:Class>

<!-- Define the Diarrhea class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Diarrhea">
<rdf:type rdf:resource="http://www.example.com/covid-19#Symptoms"/>
</rdf:Description>

```

```

<!-- OWL SubClass Definition - Fatigue-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Fatigue">

    <!-- Fatigue is a subclassification of Corona virus Symptoms -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Symptoms"/>

    <rdfs:label>Fatigue</rdfs:label>
    <rdfs:comment>Fatigue etc</rdfs:comment>

</owl:Class>
<!-- Define the Fatigue class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Fatigue">
<rdf:type rdf:resource="http://www.example.com/covid-19#Symptoms"/>
</rdf:Description>

<!-- OWL SubClass Definition - Fever-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Fever">

    <!-- Fever is a subclassification of Corona virus Symptoms -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Symptoms"/>

    <rdfs:label>Fever</rdfs:label>
    <rdfs:comment>Fever etc</rdfs:comment>

</owl:Class>
<!-- Define the Fever class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Fever">
<rdf:type rdf:resource="http://www.example.com/covid-19#Symptoms"/>
</rdf:Description>

<!-- OWL SubClass Definition - wear mask-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Wearmask">

    <!-- Headache is a subclassification of Corona virus Precautioons -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Precautions"/>

    <rdfs:label>Wear mask</rdfs:label>
    <rdfs:comment>Wear mask etc</rdfs:comment>

</owl:Class>

```

```

<!-- Define the Wear mask class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Wearmask">
<rdf:type rdf:resource="http://www.example.com/covid-19#Precautions"/>
</rdf:Description>

<!-- OWL SubClass Definition - Safe environment-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Safeenvironment">

    <!-- Headache is a subclassification of Corona virus Precautions -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Precautions"/>

    <rdfs:label>Safe environment</rdfs:label>
    <rdfs:comment>Safe environment etc</rdfs:comment>

</owl:Class>

<!-- Define the Safe environment class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Safeenvironment">
<rdf:type rdf:resource="http://www.example.com/covid-19#Precautions"/>
</rdf:Description>

<!-- OWL SubClass Definition - Get vaccine-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Getvaccine">

    <!-- Headache is a subclassification of Corona virus Precautioons -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Precautions"/>

    <rdfs:label>Get vaccine</rdfs:label>
    <rdfs:comment>Get vaccine etc</rdfs:comment>

</owl:Class>

<!-- Define the Get vaccine class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Getvaccine">
<rdf:type rdf:resource="http://www.example.com/covid-19#Precautions"/>
</rdf:Description>

<!-- OWL SubClass Definition - Good hyGINE-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Goodhygiene">

    <!-- Good hygiene is a subclassification of Corona virus Precautioons -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Precautions"/>

```

```

    <rdfs:label>Good hygiene</rdfs:label>
    <rdfs:comment>Good hygiene etc</rdfs:comment>

</owl:Class>

<!-- Define the Good hygiene class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Goodhygiene">
<rdf:type rdf:resource="http://www.example.com/covid-19#Precautions"/>
</rdf:Description>

<!-- OWL SubClass Definition - Wash hand-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Washhands">

    <!-- Wash hand is a subclassification of Corona virus Precautioons -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-19#Precautions"/>

    <rdfs:label>Washhands</rdfs:label>
    <rdfs:comment>Washhands etc</rdfs:comment>

</owl:Class>

<!-- Define the wash hands class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#washhands">
<rdf:type rdf:resource="http://www.example.com/covid-19#Precautions"/>
</rdf:Description>

<!-- OWL SubClass Definition - Social gathering-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Socialgatherings">

    <!-- Headache is a subclassification of Corona virus Infectionpossibilities -
->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-
19#Infectionpossibilities"/>

    <rdfs:label>Social gatherings</rdfs:label>
    <rdfs:comment>Socialgatherings etc</rdfs:comment>

</owl:Class>

<!-- Define the Social Gatherings class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Socialgatherings">
<rdf:type rdf:resource="http://www.example.com/covid-19#InfectionPossibilities"/>

```

```

</rdf:Description>

<!-- OWL SubClass Definition - Not wear mask-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Notwearmask">

    <!-- Not wear mask is a subclassification of Corona virus
    Infectionpossibilities -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-
    19#Infectionpossibilities"/>

    <rdfs:label>Not wear mask</rdfs:label>
    <rdfs:comment>Not wear mask etc</rdfs:comment>

</owl:Class>

<!-- Define the Not wear mask class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Notwearmask">
<rdf:type rdf:resource="http://www.example.com/covid-19#InfectionPossibilities"/>
</rdf:Description>

<!-- OWL SubClass Definition - Notgetvaccine-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Notgetvaccine">

    <!-- Notgetvaccine is a subclassification of Corona virus
    Infectionpossibilities -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-
    19#Infectionpossibilities"/>

    <rdfs:label>Not get vaccine</rdfs:label>
    <rdfs:comment>Not get vaccine etc</rdfs:comment>

</owl:Class>

<!-- Define the Not get vaccine class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Notgetvaccine">
<rdf:type rdf:resource="http://www.example.com/covid-19#InfectionPossibilities"/>
</rdf:Description>

<!-- OWL SubClass Definition - Avoid social distnace-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Avoidsocialdistance">

    <!-- Avoidsocialdistance is a subclassification of Corona virus
    Infectionpossibilities -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-
    19#Infectionpossibilities"/>

```

```

    <rdfs:label>Avoidsocialdistance</rdfs:label>
    <rdfs:comment>Avoidsocialdistance etc</rdfs:comment>

</owl:Class>

<!-- Define the Avoid Social Distance class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Avoidsocialdistance">
<rdf:type rdf:resource="http://www.example.com/covid-19#InfectionPossibilities"/>
</rdf:Description>

<!-- OWL SubClass Definition - Not get sanitize-->
<owl:Class rdf:about="http://www.example.com/Covid-19#Notgetsanitize">

    <!-- Avoidsocialdistance is a subclassification of Corona virus
Infectionpossibilities -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Covid-
19#Infectionpossibilities"/>

    <rdfs:label>Notgetsanitize</rdfs:label>
    <rdfs:comment>Notgetsanitize</rdfs:comment>

</owl:Class>

<!-- Define the Avoid Social Distance class instance -->
<rdf:Description rdf:about="http://www.example.com/Covid-19#Notgetsanitize">
<rdf:type rdf:resource="http://www.example.com/covid-19#InfectionPossibilities"/>
</rdf:Description>

</rdf:RDF>

```

4)

SPARQL query

PREFIX Covid-19: <http://www.example.com/Covid-19#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

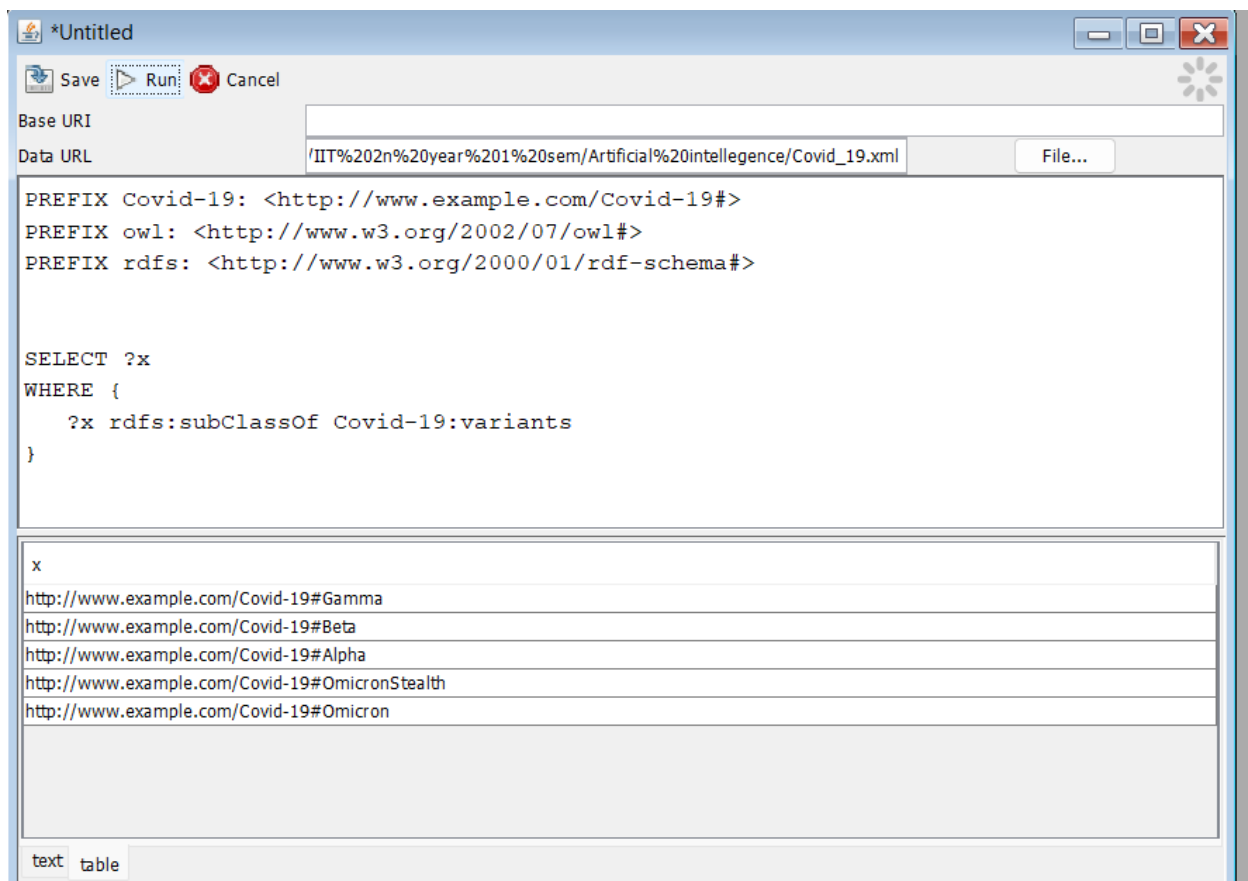
SELECT ?x

WHERE {

 ?x rdfs:subClassOf Covid-19:variants

}

Output



SPARQL query PREFIX Covid-19: <http://www.example.com/Covid-19#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

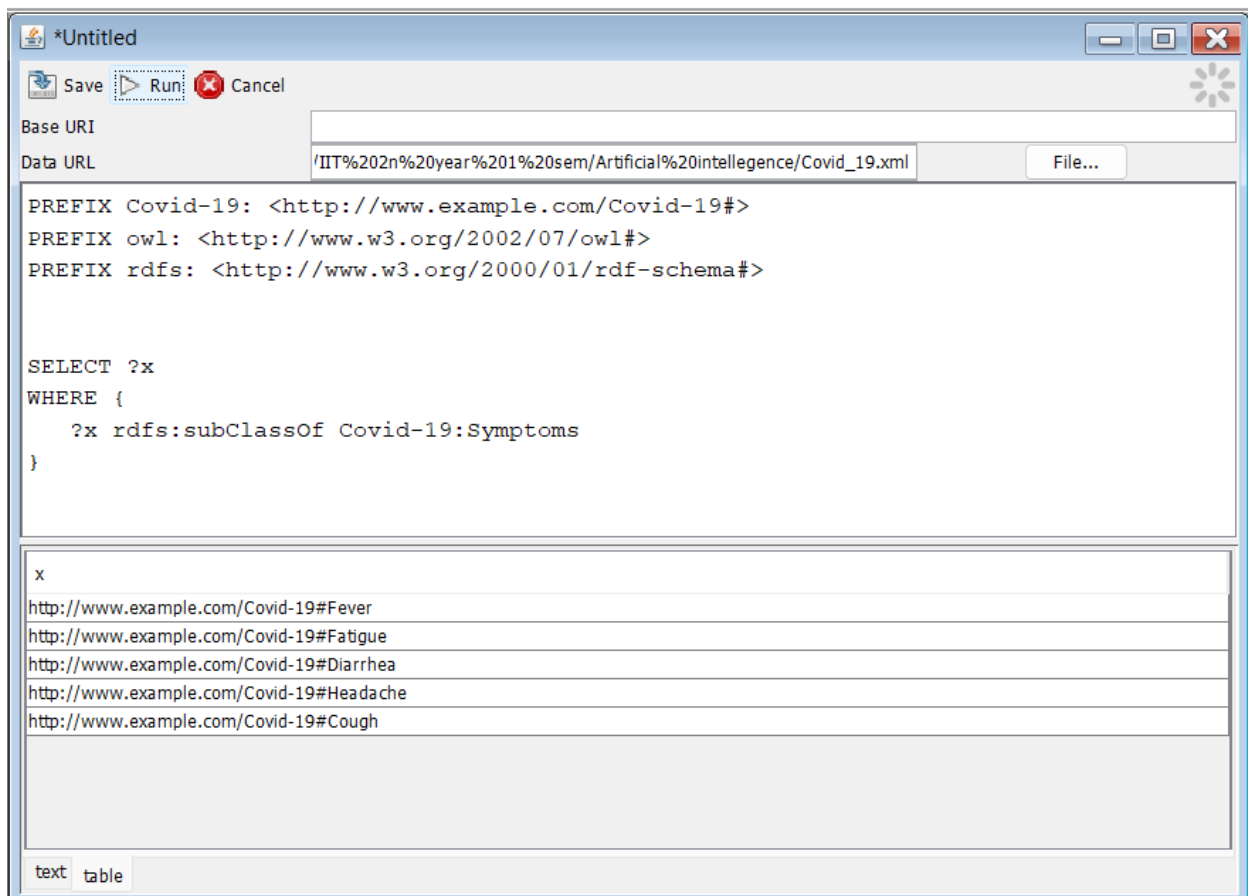
SELECT ?x

WHERE {

 ?x rdfs:subClassOf Covid-19:Symptoms

}

Output



SPAQL query

PREFIX Covid-19: <http://www.example.com/Covid-19#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

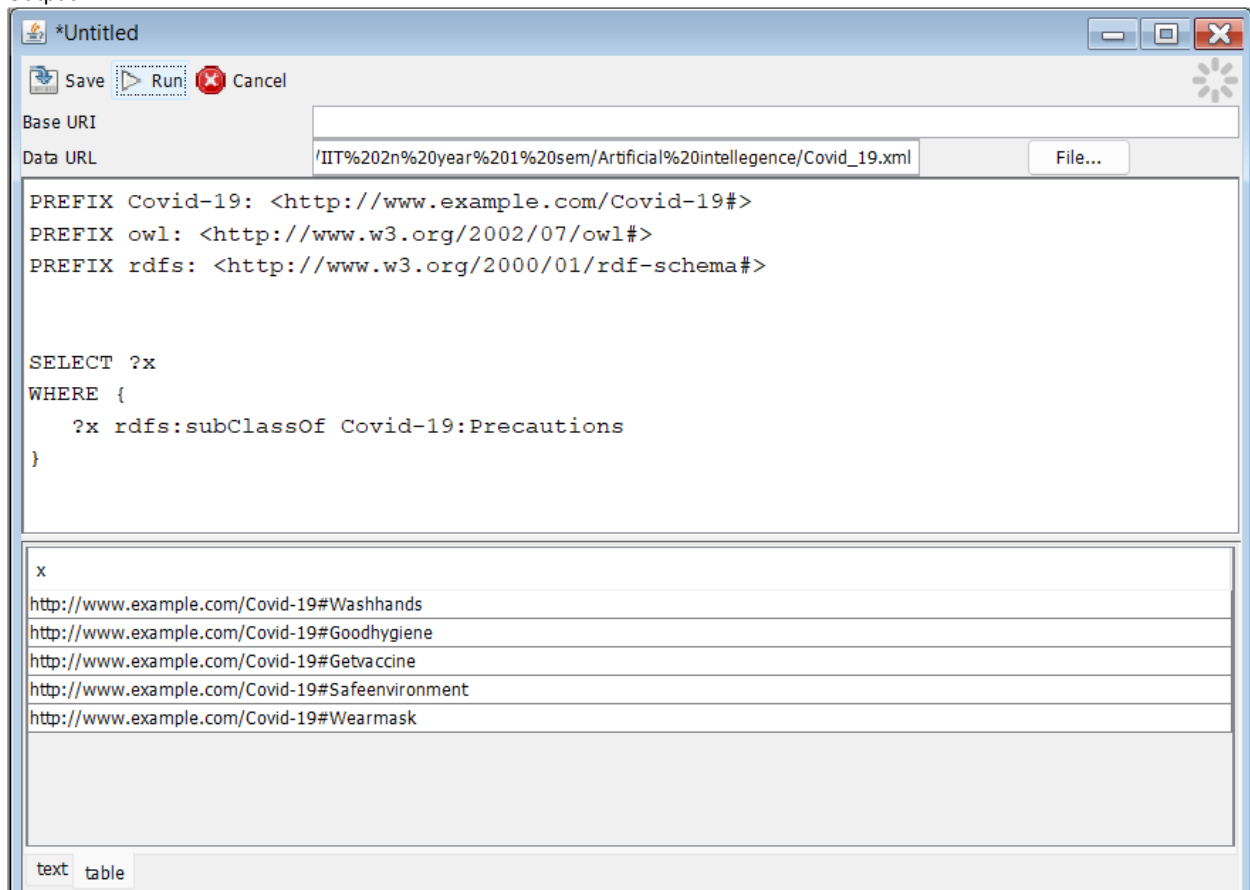
SELECT ?x

WHERE {

 ?x rdfs:subClassOf Covid-19:Precautions

}

Output



SPAQL query

PREFIX Covid-19: <http://www.example.com/Covid-19#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x

WHERE {

 ?x rdfs:subClassOf Covid-19:variants .

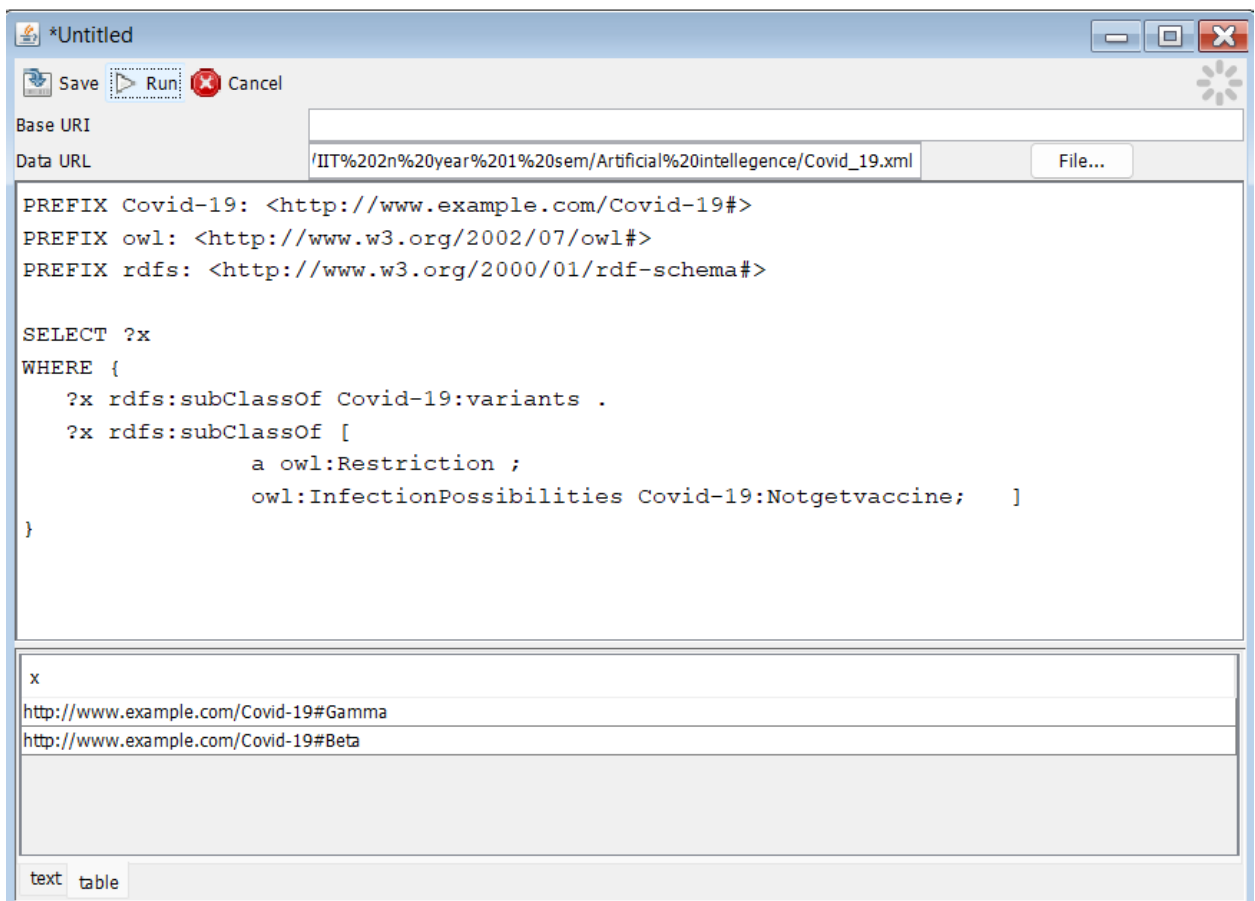
 ?x rdfs:subClassOf [

 a owl:Restriction ;

 owl:InfectionPossibilities Covid-19:Notgetvaccine;]

}

Output



*Untitled

Save Run Cancel

Base URI

Data URL /IIT%20n%20year%201%20sem/Artificial%20intelligence/Covid_19.xml File...

```
PREFIX Covid-19: <http://www.example.com/Covid-19#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x
WHERE {
    ?x rdfs:subClassOf Covid-19:variants .
    ?x rdfs:subClassOf [
        a owl:Restriction ;
        owl:InfectionPossibilities Covid-19:Notgetvaccine;   ]
}
```

x	
http://www.example.com/Covid-19#Gamma	
http://www.example.com/Covid-19#Beta	

text table

SPARQL Query

PREFIX Covid-19: <http://www.example.com/Covid-19#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x

WHERE {

?x rdfs:subClassOf Covid-19:variants .

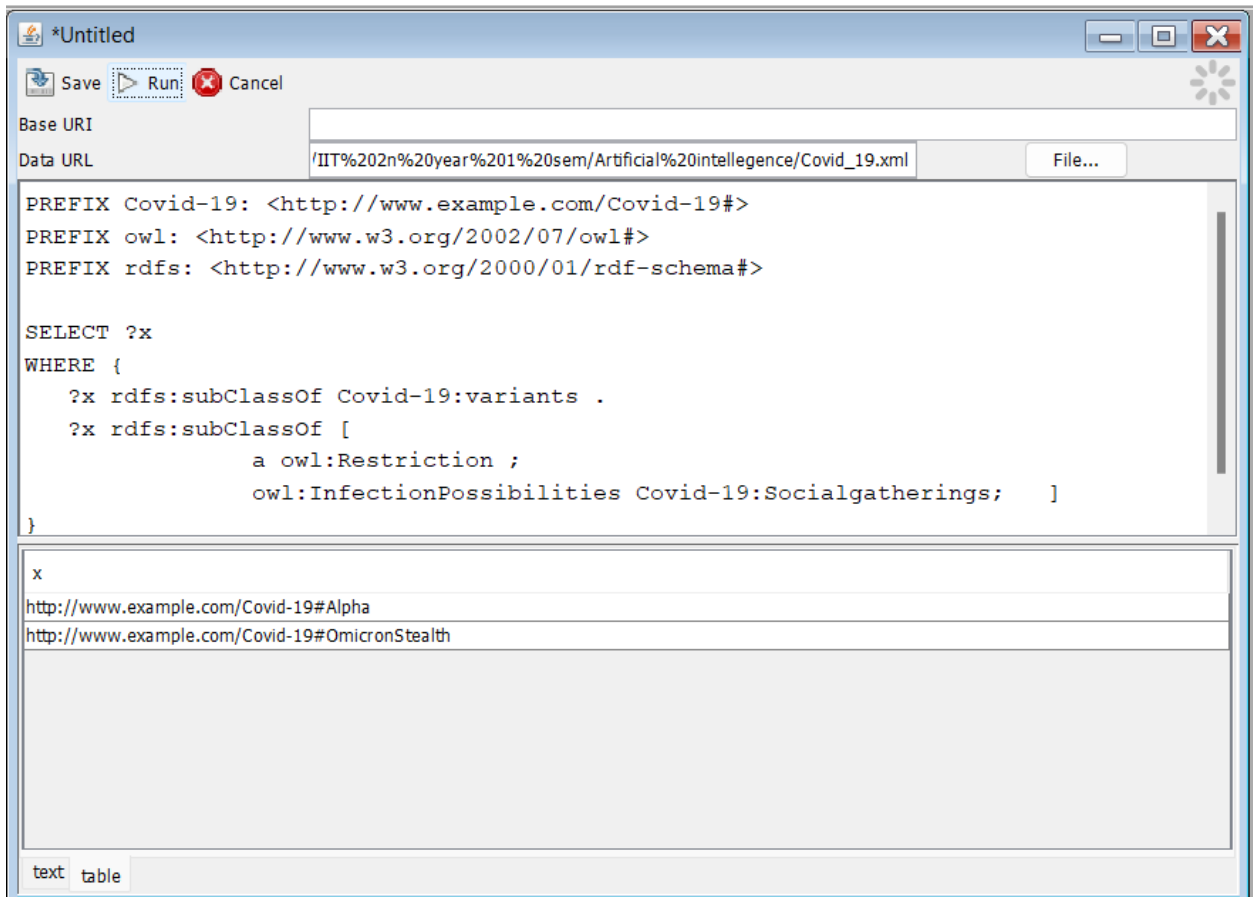
?x rdfs:subClassOf [

a owl:Restriction ;

owl:InfectionPossibilities Covid-19:Socialgatherings;]

}

Output



Q3

Task 1

```
import random

print("DFS\n")

#Creating the 6 by 6 maze
X=6
Y=6

maze = []
for i in range(0, Y):
    line = []
```

```

    for j in range(0, X):
        line.append(0)
    maze.append(line)

#initializing starting point and End point
Starting_point=5
Goal_point=9

#Randomly Selecting a position for start point and end point
Start_colum=random.randint(0,1)
Start_row=random.randint(0,5)
Goal_colum=random.randint(4,5)
Goal_row=random.randint(0,5)

#insitialzing Starting point and end point to randomly selected position
maze[Start_row][Start_colum]=Starting_point
maze[Goal_row][Goal_colum]=Goal_point
position=(Start_row,Start_colum)
wall_count=0

#Creating wall
while wall_count<4:
    wall_row = random.randint(0,5)
    wall_colum=random.randint(0,5)

#Check if the position is a already a wall or staring point or end point
    if (maze[wall_row][wall_colum]==5) or maze[wall_row][wall_colum]==9 or
maze[wall_row][wall_colum]==1:
        continue
    maze[wall_row][wall_colum] = 1
    wall_count+=1

for i in maze:
    print(i)

```

Output

```

[1, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 0, 0]
[0, 5, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 9]

```

Task 2

```

#Function for moves in DFS
def move(A,B,maze):

    next_move=[]#storing every move to this Stack
    if A-1>=0 and maze[A-1][B]!=1:
        next_move.append((A-1,B))

    if A+1<len(maze) and maze[A+1][B]!=1:
        next_move.append((A + 1, B))

    if B+1<len(maze) and maze[A][B+1]!=1:
        next_move.append((A, B+1))

    if B-1>=0 and maze[A][B-1]!=1:
        next_move.append((A, B-1))

    if (A-1>=0 and B-1>=0) and (maze[A-1][B-1]!=1):
        next_move.append((A-1, B-1))

    if (A+1<len(maze) and B+1<len(maze)) and (maze[A + 1][B + 1] != 1):
        next_move.append((A+1, B+1))

    if (A+1<len(maze) and B-1>=0) and (maze[A + 1][B - 1] != 1):
        next_move.append((A+1, B-1))

    if (A-1>=0 and B+1<len(maze)) and (maze[A - 1][B + 1] != 1):
        next_move.append((A-1, B+1))

    return next_move

#Fuction for DFS
def DFS(maze,stack,visited):
    #Initialzing Starting position and End point for varialbes
    start=(Start_row,Start_colum)
    Goal=(Goal_row,Goal_column)
    stack.append(start)#append above positon to the Stack
    while stack:
        n= stack.pop()

```

```

        print("Popping",n)#popping the positon that on top of the stack

        if n==Goal:#Checks if the popped item is the goal position
            print("Goal found")
            print("This is the Final path",visited)
            Total_Node=len(visited)
            print("The time taken to find the goal is",Total_Node,"minutes")
            return
#initializing index num 1 and 2 to A,B
    A=n[0]
    B=n[1]
    next_steps = move(A,B, maze)
    print("Next Steps is",next_steps)
    for x in next_steps:
        if x in visited:
            print(x," is already visited, skipping...")#If that position
is in Visited list skip that node
            print("\n")
            continue
        visited.add(x)
        stack.append(x)
        print("Visited nodes:",visited)
        print("Stack",stack)

stack=[]
visited=set()

move(Start_row,Start_colum,maze)
DFS(maze,stack,visited)

```

Output

```

Popping (4, 1)
Next Steps is [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (3, 2)]
Visited nodes: {(3, 1)}
Stack [(3, 1)]
Visited nodes: {(3, 1), (5, 1)}
Stack [(3, 1), (5, 1)]
Visited nodes: {(3, 1), (5, 1), (4, 2)}
Stack [(3, 1), (5, 1), (4, 2)]
Visited nodes: {(3, 1), (4, 0), (5, 1), (4, 2)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0)]
Visited nodes: {(4, 0), (3, 1), (5, 1), (4, 2), (5, 2)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2)]
Visited nodes: {(4, 0), (3, 1), (5, 1), (4, 2), (5, 0), (3, 2), (5, 2)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (3, 2)]
Popping (3, 2)
Next Steps is [(2, 2), (4, 2), (3, 3), (3, 1), (2, 1), (4, 3), (4, 1), (2, 3)]
Visited nodes: {(4, 0), (3, 1), (5, 1), (4, 2), (5, 0), (2, 2), (3, 2), (5, 2)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (2, 2), (3, 3)]
(4, 2) is already visited, skipping...

Visited nodes: {(4, 0), (3, 1), (5, 1), (4, 2), (3, 3), (5, 0), (2, 2), (3, 2), (5, 2)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (2, 2), (3, 3)]
(3, 1) is already visited, skipping...

```



```

Visited nodes: {(4, 0), (3, 4), (4, 3), (3, 1), (5, 1), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (4, 5), (3, 3), (5, 0), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (3, 5), (5, 2), (4, 4), (0, 3), (1, 4), (2, 3)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (2, 2), (3, 3), (2, 1), (4, 3), (4, 1), (1, 3), (3, 4), (0, 4), (1, 5), (0, 3), (4, 5), (4, 4)]
Popping (4, 4)
Next Steps is [(3, 4), (5, 4), (4, 5), (4, 3), (3, 3), (5, 5), (5, 3), (3, 5)]
(3, 4) is already visited, skipping...

Visited nodes: {(4, 0), (3, 4), (4, 3), (3, 1), (5, 4), (5, 1), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (4, 5), (3, 3), (5, 0), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (3, 5), (5, 2), (4, 4), (0, 3), (1, 4), (2, 3)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (2, 2), (3, 3), (2, 1), (4, 3), (4, 1), (1, 3), (3, 4), (0, 4), (1, 5), (0, 3), (4, 5), (5, 4), (5, 5)]
(4, 5) is already visited, skipping...

(4, 3) is already visited, skipping...

(3, 3) is already visited, skipping...

Visited nodes: {(4, 0), (3, 4), (4, 3), (3, 1), (5, 4), (5, 1), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (4, 5), (3, 3), (5, 0), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (3, 5), (5, 2), (4, 4), (5, 5), (0, 3), (1, 4), (2, 3)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (2, 2), (3, 3), (2, 1), (4, 3), (4, 1), (1, 3), (3, 4), (0, 4), (1, 5), (0, 3), (4, 5), (5, 4), (5, 5)]
Visited nodes: {(4, 0), (3, 4), (4, 3), (3, 1), (5, 4), (5, 1), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (4, 5), (3, 3), (5, 0), (5, 3), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (3, 5), (5, 2), (4, 4), (5, 5), (0, 3), (1, 4), (2, 3)}
Stack [(3, 1), (5, 1), (4, 2), (4, 0), (5, 2), (5, 0), (2, 2), (3, 3), (2, 1), (4, 3), (4, 1), (1, 3), (3, 4), (0, 4), (1, 5), (0, 3), (4, 5), (5, 4), (5, 5), (5, 3)]
(3, 5) is already visited, skipping...

```

```

Popping (5, 3)
Next Steps is [(4, 3), (5, 4), (5, 2), (4, 2), (4, 4)]
(4, 3) is already visited, skipping...

(5, 4) is already visited, skipping...

(5, 2) is already visited, skipping...

(4, 2) is already visited, skipping...

(4, 4) is already visited, skipping...

Popping (5, 5)
Goal found
This is the Final path {(4, 0), (3, 4), (4, 3), (3, 1), (5, 4), (5, 1), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (4, 5), (3, 3), (5, 0), (5, 3), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (3, 5), (5, 2), (4, 4), (5, 5), (0, 3), (1, 4), (2, 3)}
The time taken to find the goal is 27 minutes

```

Task 3

```

def Heuristic_Cost_Calculator():
    GR=Goal_row
    GC=Goal_column
    global heuristic_Values
    heuristic_Values={}

    for row in range(0, 6):
        for column in range(0, 6):
            Position = ((GR - row, GC - column))
            Chebyshev_Distance = max(Position)
            heuristic_Values[row,column]=abs(Chebyshev_Distance)

    print("These are the heuristic Value",heuristic_Values)

Heuristic_Cost_Calculator()

```

Output

Astar search

These are the heuristic Value {(0, 0): 5, (0, 1): 5, (0, 2): 5, (0, 3): 5, (0, 4): 5, (0, 5): 5, (1, 0): 5, (1, 1): 4, (1, 2): 4, (1, 3): 4, (1, 4): 4, (1, 5): 4, (2, 0): 5, (2, 1): 4, (2, 2): 3, (2, 3): 3, (2, 4): 3, (2, 5): 3, (3, 0): 5, (3, 1): 4, (3, 2): 3, (3, 3): 2, (3, 4): 2, (3, 5): 2, (4, 0): 5, (4, 1): 4, (4, 2): 3, (4, 3): 2, (4, 4): 1, (4, 5): 1, (5, 0): 5, (5, 1): 4, (5, 2): 3, (5, 3): 2, (5, 4): 1, (5, 5): 0}

Printing (0, 1)

Task 4

```
def move(A,B,maze):
    next_minimum_move={} # storing every move to this Stack
    if A - 1 >= 0 and maze[A - 1][B] != 1:
        Heuristic_cost=heuristic_Values.get((A-1,B))
        F_cost=Heuristic_cost+1
        next_minimum_move[(A-1,B)]=F_cost
    if B + 1 < len(maze) and maze[A][B + 1] != 1:
        Heuristic_cost = heuristic_Values.get((A , B+1))
        F_cost = Heuristic_cost + 1
        next_minimum_move[(A , B+1)] = F_cost
    if A + 1 < len(maze) and maze[A + 1][B] != 1:
        Heuristic_cost = heuristic_Values.get((A + 1, B))
        F_cost = Heuristic_cost + 1
        next_minimum_move[(A + 1, B)] = F_cost
    if B - 1 >= 0 and maze[A][B - 1] != 1:
        Heuristic_cost = heuristic_Values.get((A, B-1))
        F_cost = Heuristic_cost + 1
        next_minimum_move[(A, B-1)] = F_cost
    if (A - 1 >= 0 and B - 1 >= 0) and (maze[A - 1][B - 1] != 1):
        Heuristic_cost = heuristic_Values.get((A - 1, B - 1))
        F_cost = Heuristic_cost + 1
        next_minimum_move[(A - 1, B - 1)] = F_cost
    if (A + 1 < len(maze) and B + 1 < len(maze)) and (maze[A + 1][B + 1] != 1):
        Heuristic_cost = heuristic_Values.get((A + 1, B + 1))
        F_cost = Heuristic_cost + 1
        next_minimum_move[(A + 1, B + 1)] = F_cost
    if (A + 1 < len(maze) and B - 1 >= 0) and (maze[A + 1][B - 1] != 1):
        Heuristic_cost = heuristic_Values.get((A + 1, B - 1))
        F_cost = Heuristic_cost + 1
        next_minimum_move[(A + 1, B - 1)] = F_cost
    if (A - 1 >= 0 and B + 1 < len(maze)) and (maze[A - 1][B + 1] != 1):
        Heuristic_cost = heuristic_Values.get((A - 1, B + 1))
        F_cost = Heuristic_cost + 1
        next_minimum_move[(A - 1, B + 1)] = F_cost
```

```

        minimum_path=min(next_minimum_move,key=next_minimum_move.get)

    return minimum_path

def AStarSearch(maze,stack,visited):
    start = (Start_row, Start_colum)
    Goal = (Goal_row, Goal_column)
    stack.append(start)
    while stack:
        n = stack.pop()
        print("Popping", n)
        if n==Goal:#Checks if the popped item is the goal position
            print("Goal found")
            print("This is the Final path",visited)
            Total_Node=len(visited)
            print("The time taken to find the goal is",Total_Node,"minutes")
            return
        A=n[0]
        B=n[1]
        next_steps = move(A,B, maze)
        print("Next minimum path is ", next_steps)
        visited.add(next_steps)
        stack.append(next_steps)
        print("Visited nodes:", visited)
        print("Stack", stack)

stack=[]
visited=set()

move(Start_row,Start_colum,maze)
AStarSearch(maze,stack,visited)

```

Output


```

Popping (2, 4)
Next Steps is [(1, 4), (3, 4), (2, 3), (1, 3), (3, 5), (3, 3), (1, 5)]
(1, 4) is already visited, skipping...

Visited nodes: {(3, 4), (3, 1), (0, 2), (0, 5), (2, 2), (1, 3), (4, 2), (3, 0), (3, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(2, 1), (4, 1), (3, 2), (3, 0), (2, 0), (4, 2), (1, 2), (2, 3), (1, 1), (3, 3), (3, 1), (0, 3), (1, 4), (0, 2), (3, 4)]
(2, 3) is already visited, skipping...

(1, 3) is already visited, skipping...

Visited nodes: {(3, 4), (3, 1), (0, 2), (0, 5), (2, 2), (1, 3), (4, 2), (3, 0), (3, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (3, 5), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(2, 1), (4, 1), (3, 2), (3, 0), (2, 0), (4, 2), (1, 2), (2, 3), (1, 1), (3, 3), (3, 1), (0, 3), (1, 4), (0, 2), (3, 4), (3, 5)]
(3, 3) is already visited, skipping...

(1, 5) is already visited, skipping...

Popping (3, 5)
Goal found
This is the Final path {(3, 4), (3, 1), (0, 2), (0, 5), (2, 2), (1, 3), (4, 2), (3, 0), (3, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (3, 2), (4, 1), (3, 5), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
The time taken to find the goal is 22 minutes
*****

```

```

*****
Astar search
These are the heuristic Value {(0, 0): 5, (0, 1): 4, (0, 2): 3, (0, 3): 3, (0, 4): 3, (0, 5): 3, (1, 0): 5, (1, 1): 4, (1, 2): 3, (1, 3): 2, (1, 4): 2, (1, 5): 2, (2, 0): 5, (2, 1): 4, (2, 2): 3, (2, 3): 2, (2, 4): 1, (2, 5): 1, (3, 0): 5, (3, 1): 4, (3, 2): 3, (3, 3): 2, (3, 4): 1, (3, 5): 0, (4, 0): 5, (4, 1): 4, (4, 2): 3, (4, 3): 2, (4, 4): 1, (4, 5): 0, (5, 0): 5, (5, 1): 4, (5, 2): 3, (5, 3): 2, (5, 4): 1, (5, 5): 0}
Popping (3, 1)
Next minimum path is (3, 2)
Visited nodes: {(3, 2)}
Stack [(3, 2)]
Popping (3, 2)
Next minimum path is (3, 3)
Visited nodes: {(3, 2), (3, 3)}
Stack [(3, 3)]
Popping (3, 3)
Next minimum path is (3, 4)
Visited nodes: {(3, 2), (3, 3), (3, 4)}
Stack [(3, 4)]
Popping (3, 4)
Next minimum path is (3, 5)
Visited nodes: {(3, 2), (3, 3), (3, 4), (3, 5)}
Stack [(3, 5)]
Popping (3, 5)
Goal found
This is the Final path {(3, 2), (3, 3), (3, 4), (3, 5)}
The time taken to find the goal is 4 minutes

Process finished with exit code 0

```

2nd random maze

Output

```
C:\Users\SINGER\AppData\Local\Microsoft\WindowsApps\python.exe "D:/IIT/IIT 2n year 1 sem/Artificial intelligence/DFS_AND_ASTAR.py"
DFS
```

```
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 1, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 9]
[5, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
Popping (4, 0)
Next Steps is [(3, 0), (5, 0), (5, 1), (3, 1)]
Visited nodes: {(3, 0)}
Stack [(3, 0)]
Visited nodes: {(5, 0), (3, 0)}
Stack [(3, 0), (5, 0)]
Visited nodes: {(5, 0), (5, 1), (3, 0)}
Stack [(3, 0), (5, 0), (5, 1)]
Visited nodes: {(3, 1), (5, 0), (5, 1), (3, 0)}
Stack [(3, 0), (5, 0), (5, 1), (3, 1)]
Popping (3, 1)
Next Steps is [(2, 1), (3, 2), (3, 0), (2, 0), (4, 2), (4, 0), (2, 2)]
Visited nodes: {(2, 1), (3, 1), (5, 1), (3, 0), (5, 0)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1)]
Visited nodes: {(2, 1), (3, 1), (5, 1), (3, 0), (5, 0), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2)]
(3, 0) is already visited, skipping...
```

```
Visited nodes: {(2, 1), (3, 1), (2, 0), (5, 1), (3, 0), (5, 0), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0)]
```

```
Visited nodes: {(2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (5, 0), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2)]
Visited nodes: {(4, 0), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (5, 0), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2), (4, 0)]
Visited nodes: {(4, 0), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (5, 0), (2, 2), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2), (4, 0), (2, 2)]
Popping (2, 2)
Next Steps is [(1, 2), (3, 2), (2, 3), (2, 1), (1, 1), (3, 3), (3, 1)]
Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (5, 0), (2, 2), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2), (4, 0), (1, 2)]
(3, 2) is already visited, skipping...
```

```
Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 3), (5, 0), (2, 2), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3)]
(2, 1) is already visited, skipping...
```

```
Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (1, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 3), (5, 0), (2, 2), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1)]
Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (1, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 3), (3, 3), (5, 0), (2, 2), (3, 2)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3)]
(3, 1) is already visited, skipping...
```

```
Popping (3, 3)
Next Steps is [(2, 3), (4, 3), (3, 4), (3, 2), (2, 2), (4, 4), (4, 2), (2, 4)]
(2, 3) is already visited, skipping...
```

```

Visited nodes: {(4, 0), (3, 4), (4, 3), (3, 1), (5, 1), (0, 5), (2, 2), (2, 5), (4, 2), (3, 0), (3, 3), (5, 0), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (3, 2), (3, 5), (4, 4),
(1, 1), (2, 0), (2, 3)}
Stack [(3, 0), (5, 0), (5, 1), (2, 1), (3, 2), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (4, 3), (3, 4), (4, 4), (2, 5), (3, 5), (0, 5), (0, 4)]
(2, 4) is already visited, skipping...

Popping (0, 4)
Next Steps is [(0, 5), (1, 5)]
(0, 5) is already visited, skipping...

(1, 5) is already visited, skipping...

Popping (0, 5)
Next Steps is [(1, 5), (0, 4)]
(1, 5) is already visited, skipping...

(0, 4) is already visited, skipping...

Popping (3, 5)
Goal found
This is the Final path {(4, 0), (3, 4), (4, 3), (3, 1), (5, 1), (0, 5), (2, 2), (2, 5), (4, 2), (3, 0), (3, 3), (5, 0), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (3, 2), (3, 5),
(4, 4), (1, 1), (2, 0), (2, 3)}
The time taken to find the goal is 23 minutes
*****

```

```

Astar search
These are the heuristic Value {(0, 0): 5, (0, 1): 4, (0, 2): 3, (0, 3): 3, (0, 4): 3, (0, 5): 3, (1, 0): 5, (1, 1): 4, (1, 2): 3, (1, 3): 2, (1, 4): 2, (1, 5): 2, (2, 0): 5, (2,
1): 4, (2, 2): 3, (2, 3): 2, (2, 4): 1, (2, 5): 1, (3, 0): 5, (3, 1): 4, (3, 2): 3, (3, 3): 2, (3, 4): 1, (3, 5): 0, (4, 0): 5, (4, 1): 4, (4, 2): 3, (4, 3): 2, (4, 4): 1, (4,
5): 0, (5, 0): 5, (5, 1): 4, (5, 2): 3, (5, 3): 2, (5, 4): 1, (5, 5): 0}
Popping (4, 0)
Next minimum path is (5, 1)
Visited nodes: {(5, 1)}
Stack [(5, 1)]
Popping (5, 1)
Next minimum path is (5, 2)
Visited nodes: {(5, 1), (5, 2)}
Stack [(5, 2)]
Popping (5, 2)
Next minimum path is (5, 3)
Visited nodes: {(5, 3), (5, 1), (5, 2)}
Stack [(5, 3)]
Popping (5, 3)
Next minimum path is (5, 4)
Visited nodes: {(5, 3), (5, 4), (5, 1), (5, 2)}
Stack [(5, 4)]
Popping (5, 4)
Next minimum path is (5, 5)
Visited nodes: {(5, 5), (5, 4), (5, 1), (5, 3), (5, 2)}
Stack [(5, 5)]
Popping (5, 5)
Next minimum path is (4, 5)
Visited nodes: {(5, 5), (5, 4), (5, 1), (4, 5), (5, 3), (5, 2)}
Stack [(4, 5)]
Popping (4, 5)
Next minimum path is (3, 5)

```

```

Visited nodes: {(5, 5), (5, 4), (5, 1), (4, 5), (5, 3), (3, 5), (5, 2)}
Stack [(3, 5)]
Popping (3, 5)
Goal found
This is the Final path {(5, 5), (5, 4), (5, 1), (4, 5), (5, 3), (3, 5), (5, 2)}
The time taken to find the goal is 7 minutes

Process finished with exit code 0

```

3rd random maze

Output


```

DFS

[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 1, 0]
[5, 0, 0, 1, 0, 0]
[1, 0, 0, 0, 0, 9]
Popping (4, 0)
Next Steps is [(3, 0), (4, 1), (5, 1), (3, 1)]
Visited nodes: {(3, 0)}
Stack [(3, 0)]
Visited nodes: {(4, 1), (3, 0)}
Stack [(3, 0), (4, 1)]
Visited nodes: {(4, 1), (5, 1), (3, 0)}
Stack [(3, 0), (4, 1), (5, 1)]
Visited nodes: {(3, 1), (4, 1), (5, 1), (3, 0)}
Stack [(3, 0), (4, 1), (5, 1), (3, 1)]
Popping (3, 1)
Next Steps is [(2, 1), (4, 1), (3, 0), (2, 0), (4, 2), (4, 0), (2, 2)]
Visited nodes: {(2, 1), (3, 1), (5, 1), (3, 0), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1)]
(4, 1) is already visited, skipping...

(3, 0) is already visited, skipping...

```

```

Visited nodes: {(2, 1), (3, 1), (2, 0), (5, 1), (3, 0), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0)]
Visited nodes: {(2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2)]
Visited nodes: {(4, 0), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0)]
Visited nodes: {(4, 0), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 2), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (2, 2)]
Popping (2, 2)
Next Steps is [(1, 2), (2, 3), (2, 1), (1, 1), (3, 3), (3, 1), (1, 3)]
Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 2), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2)]
Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 3), (2, 2), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3)]
(2, 1) is already visited, skipping...

Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (1, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 3), (2, 2), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1)]
Visited nodes: {(4, 0), (1, 2), (2, 1), (3, 1), (1, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 3), (3, 3), (2, 2), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3)]
(3, 1) is already visited, skipping...

Visited nodes: {(1, 3), (4, 0), (1, 2), (2, 1), (3, 1), (1, 1), (2, 0), (5, 1), (4, 2), (3, 0), (2, 3), (3, 3), (2, 2), (4, 1)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3), (1, 3)]
Popping (1, 3)
Next Steps is [(0, 3), (2, 3), (1, 4), (1, 2), (0, 2), (2, 4), (2, 2), (0, 4)]

```

```

(0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3), (0, 3), (1, 4), (0, 2), (2, 4), (0, 5), (4, 5)]
(2, 4) is already visited, skipping...

Visited nodes: {(4, 0), (3, 1), (5, 1), (0, 2), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (3, 0), (4, 5), (3, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (4, 1), (3, 5), (4, 4), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3), (0, 3), (1, 4), (0, 2), (2, 4), (0, 5), (4, 5), (4, 4)]
Popping (4, 4)
Next Steps is [(5, 4), (4, 5), (3, 3), (5, 5), (5, 3), (3, 5)]
Visited nodes: {(4, 0), (3, 1), (5, 4), (5, 1), (0, 2), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (3, 0), (4, 5), (3, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (4, 1), (3, 5), (4, 4), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3), (0, 3), (1, 4), (0, 2), (2, 4), (0, 5), (4, 5), (5, 4)]
(4, 5) is already visited, skipping...

(3, 3) is already visited, skipping...

Visited nodes: {(4, 0), (3, 1), (5, 4), (5, 1), (0, 2), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (3, 0), (4, 5), (3, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (4, 1), (3, 5), (4, 4), (5, 5), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3), (0, 3), (1, 4), (0, 2), (2, 4), (0, 5), (4, 5), (5, 4), (5, 5)]
Visited nodes: {(4, 0), (3, 1), (5, 4), (5, 1), (0, 2), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (3, 0), (4, 5), (3, 3), (5, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (4, 1), (3, 5), (4, 4), (5, 5), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3), (0, 3), (1, 4), (0, 2), (2, 4), (0, 5), (4, 5), (5, 4), (5, 5), (5, 3)]
(3, 5) is already visited, skipping...

Popping (5, 3)
Next Steps is [(5, 4), (5, 2), (4, 2), (4, 4)]

```

```

Visited nodes: {(4, 0), (3, 1), (5, 4), (5, 1), (0, 2), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (3, 0), (4, 5), (3, 3), (5, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (4, 1), (3, 5), (5, 2), (4, 4), (5, 5), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
Stack [(3, 0), (4, 1), (5, 1), (2, 1), (2, 0), (4, 2), (4, 0), (1, 2), (2, 3), (1, 1), (3, 3), (0, 3), (1, 4), (0, 2), (2, 4), (0, 5), (4, 5), (5, 4), (5, 5), (5, 2)]
(4, 2) is already visited, skipping...

(4, 4) is already visited, skipping...

Popping (5, 2)
Next Steps is [(4, 2), (5, 3), (5, 1), (4, 1)]
(4, 2) is already visited, skipping...

(5, 3) is already visited, skipping...

(5, 1) is already visited, skipping...

(4, 1) is already visited, skipping...

Popping (5, 5)
Goal found
This is the Final path {(4, 0), (3, 1), (5, 4), (5, 1), (0, 2), (0, 5), (2, 2), (2, 5), (1, 3), (4, 2), (3, 0), (4, 5), (3, 3), (5, 3), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (4, 1), (3, 5), (5, 2), (4, 4), (5, 5), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}
The time taken to find the goal is 29 minutes

```

```

Astar search
These are the heuristic Value {(0, 0): 5, (0, 1): 5, (0, 2): 5, (0, 3): 5, (0, 4): 5, (0, 5): 5, (1, 0): 5, (1, 1): 4, (1, 2): 4, (1, 3): 4, (1, 4): 4, (1, 5): 4, (2, 0): 5, (2, 1): 4, (2, 2): 3, (2, 3): 3, (2, 4): 3, (2, 5): 3, (3, 0): 5, (3, 1): 4, (3, 2): 3, (3, 3): 2, (3, 4): 2, (3, 5): 2, (4, 0): 5, (4, 1): 4, (4, 2): 3, (4, 3): 2, (4, 4): 1, (4, 5): 1, (5, 0): 5, (5, 1): 4, (5, 2): 3, (5, 3): 2, (5, 4): 1, (5, 5): 0}
Popping (4, 0)
Next minimum path is (4, 1)
Visited nodes: {(4, 1)}
Stack [(4, 1)]
Popping (4, 1)
Next minimum path is (4, 2)
Visited nodes: {(4, 1), (4, 2)}
Stack [(4, 2)]
Popping (4, 2)
Next minimum path is (5, 3)
Visited nodes: {(5, 3), (4, 1), (4, 2)}
Stack [(5, 3)]
Popping (5, 3)
Next minimum path is (5, 4)
Visited nodes: {(5, 3), (5, 4), (4, 1), (4, 2)}
Stack [(5, 4)]
Popping (5, 4)
Next minimum path is (5, 5)
Visited nodes: {(5, 5), (5, 4), (4, 2), (5, 3), (4, 1)}
Stack [(5, 5)]
Popping (5, 5)
Goal found
This is the Final path {(5, 5), (5, 4), (4, 2), (5, 3), (4, 1)}
The time taken to find the goal is 5 minutes

```

- In first random maze dfs search took 22 minutes to find the goal while Astar search took only 4minutes
- In second random maze dfs search took 23 minutes to find the goal while Astar search took only 7 minutes
- In Third random maze dfs search took 29 minutes to find the goal while Astar search took only 5minutes