

Ahsanullah University of Science and Technology



Department of Computer Science and Engineering

Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE 4108

Course Title: Artificial Intelligence Lab

Assignment No: 02

Date of Submission: 08/01/2022

Submitted to:

Mr. Faisal Muhammad Shah
Associate Professor, Department of CSE, AUST.

Mr. Md. Siam Ansary
Lecturer, Department of CSE, AUST.

Submitted By:

Name: Mahin Opu

Student ID: 17.02.04.006

Q1: Define a recursive procedure in Python to find the sum of 1st n terms of an equal-interval series given the 1st term and the interval.

Python Code:

```
def Sum(first_term, interval, num_of_terms):  
    if num_of_terms == 0:  
        return 0  
    else:  
        return first_term + Sum((first_term + interval), interval, num_of_terms - 1)
```

```
R= int(input("How many times to check: "))
```

```
for i in range(R):
```

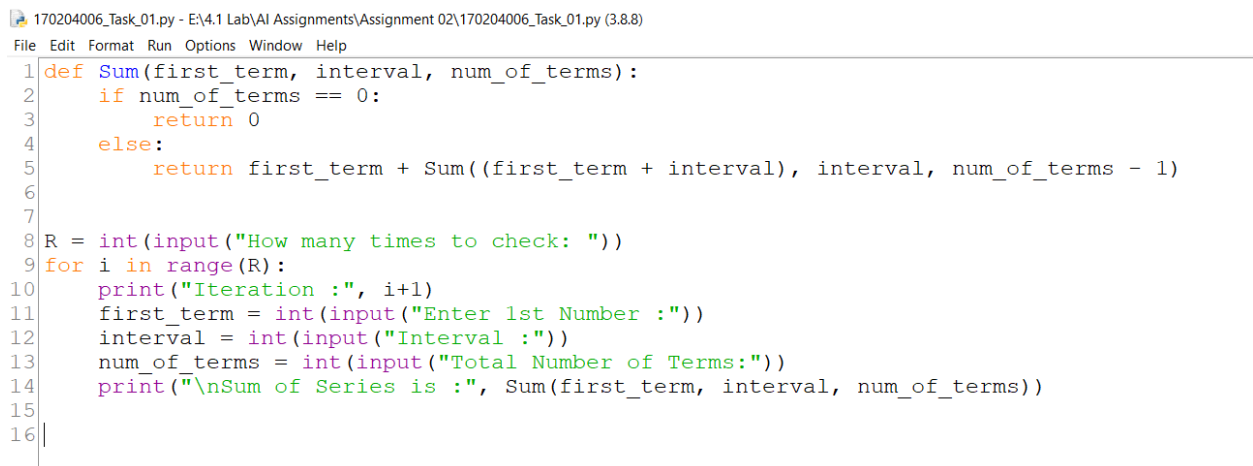
```
    print("Iteration :", i+1)
```

```
    first_term = int(input("Enter 1st Number :"))
```

```
    interval = int(input("Interval :"))
```

```
    num_of_terms = int(input("Total Number of Terms:"))
```

```
    print("\nSum of Series is :", Sum(first_term, interval, num_of_terms))
```



The screenshot shows a Python IDE window titled "170204006_Task_01.py - E:\4.1 Lab\AI Assignments\Assignment 02\170204006_Task_01.py (3.8.8)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code is as follows:

```
1 def Sum(first_term, interval, num_of_terms):  
2     if num_of_terms == 0:  
3         return 0  
4     else:  
5         return first_term + Sum((first_term + interval), interval, num_of_terms - 1)  
6  
7  
8 R = int(input("How many times to check: "))  
9 for i in range(R):  
10     print("Iteration :", i+1)  
11     first_term = int(input("Enter 1st Number :"))  
12     interval = int(input("Interval :"))  
13     num_of_terms = int(input("Total Number of Terms:"))  
14     print("\nSum of Series is :", Sum(first_term, interval, num_of_terms))  
15  
16
```

```

IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\4.1 Lab\AI Assignments\Assignment 02\170204006_Task_01.py =====
How many times to check: 2
Iteration : 1
Enter 1st Number :3
Interval :2
Total Number of Terms:4

Sum of Series is : 24
Iteration : 2
Enter 1st Number :4
Interval :2
Total Number of Terms:3

Sum of Series is : 18
>>> |

```

Q2: Define a recursive procedure in Python to find the length of a path between two vertices of a directed weighted graph.

Python Code:

```

graph=[
    ('I', 'A', 25),
    ('A', 'C', 21),
    ('I', 'B', 41),
    ('A', 'D', 28),
    ('B', 'D', 27),
    ('B', 'E', 39),
    ('B', 'F', 21),
    ('C', 'D', 33),
    ('C', 'G', 38),
    ('D', 'G', 36),
    ('E', 'G', 31)

```

```
]
```

```
visited = [0] * len(graph)
```

```
all_paths = []
```

```
def pathFind(start,end, weight=[]):
```

```
    if start == end:
```

```
        all_paths.append(list(weight))
```

```
    i = 0
```

```
    child = "
```

```
    while i <= len(graph)-1:
```

```
        if visited[i] == 0 and graph[i][0] == start:
```

```
            visited[i] = 1
```

```
            child = graph[i][1]
```

```
            weight.append(( start,child,graph[i][2] , i))
```

```
            pathFind(child,end)
```

```
i+=1
```

```
if len(weight) >= 1:
```

```
    visited[weight[len(weight)-1][3]] = 0
```

```
    weight.pop()
```

```
start = 'I'
```

```
end = 'G'
```

```
pathFind(start,end)
```

```
print(f"\nStart node = {start} and End node = {end} \n")
```

```
for i,target_list in enumerate( all_paths ,1):
```

```
    print(
```

```
        f"Path values {i} = {target_list} Length = { sum( [ p[2] for p in target_list] )  
    }"
```

```
    )
```

170204006_Task_02.py - E:\4.1 Lab\AI Assignments\Assignment 02\170204006_Task_02.py (3.8.8)

File Edit Format Run Options Window Help

```
graph=[
    ('I', 'A', 25),
    ('A', 'C', 21),
    ('I', 'B', 41),
    ('A', 'D', 28),
    ('B', 'D', 27),
    ('B', 'E', 39),
    ('B', 'F', 21),
    ('C', 'D', 33),
    ('C', 'G', 38),
    ('D', 'G', 36),
    ('E', 'G', 31)
]

visited = [0] * len(graph)
all_paths = []

def pathFind(start,end, weight=[]):

    if start == end:
        all_paths.append(list(weight))

    i = 0
    child = ''
    while i <= len(graph)-1:
        if visited[i] == 0 and graph[i][0] == start:
            visited[i] = 1
            child = graph[i][1]
            weight.append(( start,child,graph[i][2] , i))
            pathFind(child,end)

            i+=1

    if len(weight) >= 1:
        visited[weight[len(weight)-1][3]] = 0
        weight.pop()
    |
```

```

        weight.pop()
start = 'I'
end = 'G'
pathFind(start,end)

|
print(f"\nStart node = {start} and End node = {end} \n")
for i,target_list in enumerate( all_paths ,1):
    print(
        f"Path values {i} = {target_list} Length = { sum( [ p[2] for p in target_list] ) }"
    )

```

IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: E:\4.1 Lab\AI Assignments\Assignment 02\170204006_Task_02.py =====

Start node = I and End node = G

Path values 1 = [('I', 'A', 25, 0), ('A', 'C', 21, 1), ('C', 'D', 33, 7), ('D', 'G', 36, 9)] Length = 115
Path values 2 = [('I', 'A', 25, 0), ('A', 'C', 21, 1), ('C', 'G', 38, 8)] Length = 84
Path values 3 = [('I', 'A', 25, 0), ('A', 'D', 28, 3), ('D', 'G', 36, 9)] Length = 89
Path values 4 = [('I', 'B', 41, 2), ('B', 'D', 27, 4), ('D', 'G', 36, 9)] Length = 104
Path values 5 = [('I', 'B', 41, 2), ('B', 'E', 39, 5), ('E', 'G', 31, 10)] Length = 111

>>>

Q3: Write a program in Python to calculate the heuristic for 8 puzzle problem where the heuristic is the Manhattan distance of the tiles.

Python Code:

```

gtp=[
    (1,1,1),
    (2,1,2),
    (3,1,3),
    (4,2,3),
    (5,3,3),
    (6,3,2),
    (7,3,1),
    (8,2,1)
]
gblnk = (2,1)

tp=[
    (1,1,2),

```

```
(2,1,3),
(3,2,1),
(4,2,3),
(5,3,3),
(6,2,2),
(7,3,2),
(8,1,1)
]
blnk = (3,1)

print('\n')
i,h=0,0
L = []

while i<=(len(gtp) - 1):
    val = abs( gtp[i][1] - tp[i][1] ) + abs( gtp[i][2] - tp[i][2] )
    h += val
    L.append(val)
    i=i+1

print("T = ' , L)
print('H2(Heuristics value): ',h)
print('\n')
```



```
1 gtp=[
2     (1,1,1),
3     (2,1,2),
4     (3,1,3),
5     (4,2,3),
6     (5,3,3),
7     (6,3,2),
8     (7,3,1),
9     (8,2,1)
10 ]
11 gblnk = (2,1)
12
13 tp=[
14     (1,1,2),
15     (2,1,3),
16     (3,2,1),
17     (4,2,3),
18     (5,3,3),
19     (6,2,2),
20     (7,3,2),
21     (8,1,1)
22 ]
23 blnk = (3,1)
24
25 print('\n')
26 i,h=0,0
27 L = []
28
29 while i<=(len(gtp) - 1):
30     val = abs( gtp[i][1] - tp[i][1] ) + abs( gtp[i][2] - tp[i][2] )
31     h += val
32     L.append(val)
33     i=i+1
34
35 print('T = ' , L)
36 print('H2(Heuristics value): ',h)
37 print('\n')
38
```

```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\4.1 Lab\AI Assignments\Assignment 02\170204006_Task_03.py =====
T = [1, 1, 3, 0, 0, 1, 1, 1]
H2(Heuristics value): 8
>>>
```

Q4: Write a program in Python to calculate the heuristic for 8 queen problem where the heuristic is the number of attacking pairs.

Python Code:

```
ROW = 8
COL = 8
board = [
    (0,'Q',0,0,0,0,0,'Q'),
    (0,0,0,0,0,0,0,0),
    (0,0,0,0,0,'Q',0,0),
    (0,0,0,0,'Q',0,0,0),
    (0,0,'Q',0,0,0,0,0),
    ('Q',0,0,0,0,0,0,0),
    (0,0,0,'Q',0,0,0,0),
    (0,0,0,0,0,0,'Q',0),
]

L = []

for i in range(ROW):
    for j in range(COL):
        if board[j][i] == 'Q':
            L.append([j , (j,i)])
```

```
print("\nL = ', L)
```

```
right = 0
```

```
for queen in L:
```

```
    position = queen[1]
```

```
    row = position[0]
```

```
    col = position[1]
```

```
    range_start = col+1
```

```
    range_end = COL
```

```
    for i in range(range_start,range_end):
```

```
        if board[row][i] == 'Q' :
```

```
            right += 1
```

```
print("\nRight (face to face in the row) = ',right)
```

```
dia_down = 0
```

```
for queen in L:
```

```
    position = queen[1]
```

```
    row = position[0]
```

```
    col = position[1]
```

```
    range_start = row + 1
```

```
    range_end = COL - col
```

```
    j = 1
```

```
    for i in range(range_start,range_end):
```

```
        if board[row+j][col+j] == 'Q' :
```

```
            dia_down += 1
```

```
        j += 1
```

```
print('Diagonally down (face to face diagonally down )= ',dia_down)
```

```
dia_up = 0
```

```
for queen in L:
```

```
    position = queen[1]
```

```
    row = position[0]
```

```
    col = position[1]
```

```
    range_start = 0
```

```
    range_end = row
```

```
    j = 1
```

```

for i in reversed(range(range_start,range_end)):
    if i == -1 or col+j == COL:
        break
    if board[i][col+j] == 'Q':
        dia_up += 1
    j += 1

```

```

print('Diagonally up ((face to face diagonally up) = ',dia_up)

```

```

print(f"\nh(l) = {right + dia_down + dia_up}")

```

170204006_Task_04.py - E:\4.1 Lab\AI Assignments\Assignment 02\170204006_Task_04.py (3.8.8)

```

File Edit Format Run Options Window Help
1 ROW = 8
2 COL = 8
3 board = [
4     (0,'Q',0,0,0,0,0,0,'Q'),
5     (0,0,0,0,0,0,0,0,0),
6     (0,0,0,0,0,'Q',0,0,0),
7     (0,0,0,0,'Q',0,0,0,0),
8     (0,0,'Q',0,0,0,0,0,0),
9     ('Q',0,0,0,0,0,0,0,0),
10    (0,0,0,'Q',0,0,0,0,0),
11    (0,0,0,0,0,0,'Q',0,0),
12 ]
13
14 L = []
15
16 for i in range(ROW):
17     for j in range(COL):
18         if board[j][i] == 'Q':
19             L.append([j , (j,i)])
20
21 print('\nL = ', L)
22
23 right = 0
24 for queen in L:
25     position = queen[1]
26     row = position[0]
27     col = position[1]
28     range_start = col+1
29     range_end = COL
30     for i in range(range_start,range_end):
31         if board[row][i] == 'Q':
32             right += 1
33
34 print('\nRight (face to face in the row) = ',right)
35
36 dia_down = 0
37 for queen in L:

```

File Edit Format Run Options Window Help

```
37 for queen in L:
38     position = queen[1]
39     row = position[0]
40     col = position[1]
41     range_start = row + 1
42     range_end = COL - col
43     j = 1
44     for i in range(range_start, range_end):
45         if board[row+j][col+j] == 'Q':
46             dia_down += 1
47
48     j += 1
49
50 print('Diagonally down (face to face diagonally down) = ', dia_down)
51
52
53 dia_up = 0
54 for queen in L:
55     position = queen[1]
56     row = position[0]
57     col = position[1]
58     range_start = 0
59     range_end = row
60     j = 1
61
62     for i in reversed(range(range_start, range_end)):
63         if i == -1 or col+j == COL:
64             break
65         if board[i][col+j] == 'Q':
66             dia_up += 1
67         j += 1
68
69 print('Diagonally up ((face to face diagonally up) = ', dia_up)
70 print(f"\nh(1) = {right + dia_down + dia_up}")
71
72
73
```

IDLE Shell 3.8.8

File Edit Shell Debug Options Window Help

Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

==== RESTART: E:\4.1 Lab\AI Assignments\Assignment 02\170204006_Task_04.py ====

L = [[5, (5, 0)], [0, (0, 1)], [4, (4, 2)], [6, (6, 3)], [3, (3, 4)], [2, (2, 5)], [7, (7, 6)], [0, (0, 7)]]

Right (face to face in the row) = 1

Diagonally down (face to face diagonally down) = 1

Diagonally up ((face to face diagonally up) = 3

h(1) = 5

>>> |