ai

May 30, 2023

```
[ ]: #DFS
graph = {
    'A':['B','C','D'],'B':['E'],'C':['D','E'],'D':[],'E':[]
}
visited = set()

def dfs(visited,graph,root):
    if root not in visited:
        print(root)
        visited.add(root)
        for neighbor in graph[root]:
            dfs(visited,graph,neighbor)

dfs(visited,graph,'A')
```

```
[ ]: #DFS
graph = {
    'a':['b','c','d'],'b':['e'],'c':['d','e'],'d':[],'e':[]
}
visited = set()

def dfs(visited,graph,root):
    if root not in visited:
        print(root)
        visited.add(root)
        for neighbor in graph[root]:
            dfs(visited,graph,neighbor)

dfs(visited,graph,'a')
```

```
[ ]: #BFS
import collections

graph = {
    0:[1,2,3],1:[0,2],2:[0,1,4],3:[0],4:[2]
}
```

```
def bfs(graph,root):
    visited = set()
    queue = collections.deque([root])
    while queue:
        vertex= queue.popleft()
        visited.add(vertex)
        for i in graph[vertex]:
            if i not in visited:
                queue.append(i)
    print(visited)

bfs(graph,0)
```

[ ]:
```
#BFS
import collections

graph = {
    0:[1,2,3],1:[0,2],2:[0,1,4],3:[0],4:[2]
}

def bfs(graph,root):
    visited = set()
    queue = collections.deque([root])

    while queue:
        vertex = queue.popleft()
        visited.add(vertex)

        for i in graph[vertex]:
            if i not in visited:
                queue.append(i)
    print(visited)

bfs(graph,0)
```

[ ]:
```
#Chatbot
from nltk.chat.util import Chat,reflections
reflections
pairs=[
    ['Hello',['Hi!, How can I help you?']],
    ['Need help',['How can I help you?']],
    ['I am Niraj',['Nice to hear that']],
    ['What is your name?',['I am Chatbot and here to help you']],
    ['What is Chatbot?',['Chatbot is python program to help you']]
]

chat=Chat(pairs,reflections)
```

```
chat.converse()
```

[ ]:
```
#chatbot
from nltk.chat.util import Chat,reflections
reflections
pairs=[
    ['Hello',['Hi!, How can I help you?']],
    ['Need help',['How can I help you?']],
    ['I am Niraj',['Nice to hear that']],
    ['What is your name?',['I am Chatbot and here to help you']],
    ['What is Chatbot?',['Chatbot is python program to help you']]
]
chat=Chat(pairs,reflections)
chat.converse()
```

[5]:
```
#n-queen
n=int(input("Enter the value of n : "))
board=[[0 for i in range(n)]for i in range(n)]
def check_column(board,row,column):
    for i in range(row,-1,-1):
        if board[i][column]==1:
            return False
    return True
def check_diagonal(board,row,column):
    for i,j in zip(range(row,-1,-1),range(column,-1,-1)):
        if board[i][j]==1:
            return False
    for i,j in zip(range(row,-1,-1),range(column,n)):
        if board[i][j]==1:
            return False
    return True
#backtracking
def nqn(board,row):
    if row==n:
        return True
    for i in range(n):
        if(check_column(board,row,i)==True and␣
 ↪check_diagonal(board,row,i)==True):
            board[row][i]=1
            if nqn(board,row+1):
                return True
            board[row][i]=0
    return False
nqn(board,0)
for row in board:
    print(row)
```

```
Enter the value of n : 8
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0]
```

[10]:
```
#n-queen
n=int(input("Enter the value of n:"))

board=[[0 for i in range(n)]for i in range(n)]

def check_column(board,row,column):
    for i in range(row,-1,-1):
        if board[i][column]==1:
            return False
    return True

def check_diagonal(board,row,column):
    for i,j in zip(range(row,-1,-1),range(column,-1,-1)):
        if board[i][j]==1:
            return False
    for i,j in zip(range(row,-1,-1),range(column,n)):
        if board[i][j]==1:
            return False
    return True
#backtrack
def nqn(board,row):
    if row==n:
        return True
    for i in range(n):
        if(check_column(board,row,i)==True and␣
 ↪check_diagonal(board,row,i)==True):
            board[row][i]=1
            if nqn(board,row+1):
                return True
            board[row][i]=0
    return False
nqn(board,0)
for row in board:
    print(row)
```

```
Enter the value of n:9
[1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0, 0]
```

```
[0, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0]
```

[6]:
```python
#sel-sort
arr=[]
n=int(input("Number of elements in array:"))
for i in range(0,n):
    l=int(input())
    arr.append(l)
for i in range(len(arr)):
    min_idx=i
    for j in range(i+1,len(arr)):
        if arr[min_idx]>arr[j]:
            min_idx = j
    arr[i], arr[min_idx]= arr[min_idx], arr[i]
print("Sorted array is ",arr)
```

```
Number of elements in array:4
45
2
78
9
Sorted array is  [2, 9, 45, 78]
```

[8]:
```python
#sel-sort
arr=[]
n=int(input("Enter no of elments :"))
for i in range(0,n):
    l=int(input())
    arr.append(l)

for i in range(len(arr)):
    min_idx=i
    for j in range(i+1,len(arr)):
        if arr[min_idx]>arr[j]:
            min_idx=j

    arr[i],arr[min_idx]=arr[min_idx],arr[i]

print("Sorted array :",arr)
```

```
Enter no of elments :4
23
```

```
78
21
3
Sorted array : [3, 21, 23, 78]
```

```
[ ]:
```