Graph Traversal: DFS vs. BFS

ChatGPT 3.5

You

Give output for following:

```
def dfs(visited,graph,node):
    if node not in visited:
        print(node,end = " ")
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

def bfs(visited,graph,node,queue):
    visited.add(node)
    queue.append(node)

    while queue:
        s = queue.pop(0)
        print(s,end = " ")
        for neighbour in graph[s]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)

def main():
    visited1 = set() # TO keep track of DFS visited nodes
    visited2 = set() # TO keep track of BFS visited nodes
    queue = []      # For BFS
    n = int(input("Enter number of nodes : "))
    graph = dict()

    for i in range(1,n+1):
        edges = int(input("Enter number of edges for node {} : ".format(i)))
        graph[i] = list()
        for j in range(1,edges+1):
            node = int(input("Enter edge {} for node {} : ".format(j,i)))
            graph[i].append(node)

    print("The following is DFS")
    dfs(visited1, graph, 1)
    print()
    print("The following is BFS")
    bfs(visited2, graph, 1, queue)

if __name__=="__main__":
    main()
```

```
# graph = {
#    '1' : ['2','3'],
#    '2' : ['4', '5'],
#    '3' : ['6','7'],
#    '4' : [],
#    '5' : [],
#    '6' : [],
#    '7' : []
#    DFS : 1 2 4 5 3 6 7
#    BFS : 1 2 3 4 5 6 7
# }
```

ChatGPT

Let's simulate the code with a sample input:

```mathematica
Enter number of nodes: 3
Enter number of edges for node 1: 2
Enter edge 1 for node 1: 2
Enter edge 2 for node 1: 3
Enter number of edges for node 2: 1
Enter edge 1 for node 2: 3
Enter number of edges for node 3: 0
```

This input creates a graph where node 1 is connected to nodes 2 and 3, node 2 is connected to node 3, and node 3 has no outgoing edges.

Here's the output:

```csharp
The following is DFS
1 2 3
The following is BFS
1 2 3
```

This output represents the traversal order of both DFS and BFS algorithms on the provided graph.

You

Give the output:

```
g=0
def print_board(elements):
    for i in range(9):
        if i%3 == 0:
            print()
        if elements[i]==-1:
            print("_", end = " ")
```

```
        else:
            print(elements[i], end = " ")
    print()

def solvable(start):
    inv=0

    for i in range(9):
        if start[i] <= 1:
            continue
        for j in range(i+1,9):
            if start[j]==-1:
                continue
            if start[i]>start[j]:
                inv+=1
    if inv%2==0:
        return True
    return False

def heuristic(start,goal):
    global g
    h = 0
    for i in range(9):
        for j in range(9):
            if start[i] == goal[j] and start[i] != -1:
                h += (abs(j-i))//3 + (abs(j-i))%3
    return h + g

def moveleft(start,position):
    start[position],start[position-1]= start[position-1],start[position]

def moveright(start,position):
    start[position],start[position+1]= start[position+1],start[position]

def moveup(start,position):
    start[position],start[position-3]= start[position-3],start[position]

def movedown(start,position):
    start[position],start[position+3]= start[position+3],start[position]


def movetile(start,goal):
    emptyat= start.index(-1)
    row = emptyat//3
    col = emptyat%3
    t1,t2,t3,t4 = start[:],start[:],start[:],start[:]
    f1,f2,f3,f4 = 100,100,100,100

    if col -1 >=0:
        moveleft(t1, emptyat)
        f1 = heuristic(t1, goal)
```

```
    if col+1<3:
        moveright(t2, emptyat)
        f2 = heuristic(t2, goal)
    if row + 1 <3:
        movedown(t3, emptyat)
        f3 = heuristic(t3, goal)
    if row-1>=0:
        moveup(t4, emptyat)
        f4 = heuristic(t4, goal)

    min_heuristic = min(f1, f2,f3,f4)

    if f1==min_heuristic:
        moveleft(start, emptyat)
    elif f2==min_heuristic:
        moveright(start, emptyat)
    elif f3==min_heuristic:
        movedown(start, emptyat)
    elif f4 == min_heuristic:
        moveup(start, emptyat)


def solveEight(start,goal):
    global g
    g+=1
    movetile(start,goal)
    print_board(start)
    f = heuristic(start,goal)
    if f == g:
        print("Solved in {} moves".format(f))
        return

    solveEight(start,goal)


def main():
    global g
    start = list()
    goal = list()
    print("Enter the start state:(Enter -1 for empty):")
    for i in range(9):
        start.append(int(input()))

    print("Enter the goal state:(Enter -1 for empty):")
    for i in range(9):
        goal.append(int(input()))

    print_board(start)

    # To check if solvable
    if solvable(start):
```

```
        solveEight(start,goal)
        print("Solved in {} moves".format(g))
    else:
        print("Not possible to solve")


if __name__ == '__main__':
    main()



# Test Cases
#
# 1
# 2
# 3
# -1
# 4
# 6
# 7
# 5
# 8

# 1
# 2
# 3
# 4
# 5
# 6
# 7
# 8
# -1
```

ChatGPT

Here's the output for the given test case:

```mathematica
Enter the start state:(Enter -1 for empty):
1
2
3
-1
4
6
7
5
8
Enter the goal state:(Enter -1 for empty):
1
2
3
4
5
6
```

```
7
8
-1

1 2 3
_ 4 6
7 5 8

1 2 3
4 _ 6
7 5 8

1 2 3
4 5 6
7 _ 8

1 2 3
4 5 6
7 8 _

Solved in 4 moves
```

The program demonstrates the solution to the 8-puzzle problem, moving tiles to solve the puzzle.

You

Give the output:

```
class Graph():

    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printSolution(self, dist):
        print("Vertex \t Distance from Source")
        for node in range(self.V):
            print(node, "\t\t", dist[node])

    def minDistance(self, dist, sptSet):
        min = float('inf')
        for v in range(self.V):
            if dist[v] < min and sptSet[v] == False:
                min = dist[v]
                min_index = v
        return min_index

    def dijkstra(self, src):
        dist = [float('inf')] * self.V
        dist[src] = 0
        sptSet = [False] * self.V

        for cout in range(self.V):
            u = self.minDistance(dist, sptSet)
            sptSet[u] = True
            for v in range(self.V):
                if (self.graph[u][v] > 0 and
```

```
                sptSet[v] == False and
                dist[v] > dist[u] + self.graph[u][v]):
            dist[v] = dist[u] + self.graph[u][v]

    self.printSolution(dist)


# Taking user input for the number of vertices
V = int(input("Enter the number of vertices: "))
g = Graph(V)

# Taking user input for the adjacency matrix representation of the graph
print("Enter the adjacency matrix (separate elements with space):")
for i in range(V):
    g.graph[i] = list(map(int, input().split()))

# Taking user input for the source vertex
src = int(input("Enter the source vertex: "))

# Running Dijkstra's algorithm
g.dijkstra(src)
```

ChatGPT

Here's how the output would look when you run the provided code:

```mathematica
Enter the number of vertices: 5
Enter the adjacency matrix (separate elements with space):
0 10 5 0 0
0 0 2 1 0
0 3 0 9 2
0 0 0 0 4
7 0 0 6 0
Enter the source vertex: 0
Vertex   Distance from Source
0          0
1          8
2          5
3          14
4          7
```

This output represents the distances from the source vertex (vertex 0) to all other vertices in the graph computed using Dijkstra's algorithm.

You

Give the output:

```
# Class to represent a graph
class Graph:
    def __init__(self, vertices):
        self.V = vertices
```

```
    self.graph = []

# Function to add an edge to graph
def addEdge(self, u, v, w):
    self.graph.append([u, v, w])

def find(self, parent, i):
    if parent[i] != i:
        parent[i] = self.find(parent, parent[i])
    return parent[i]

def union(self, parent, rank, x, y):
    if rank[x] < rank[y]:
        parent[x] = y
    elif rank[x] > rank[y]:
        parent[y] = x
    else:
        parent[y] = x
        rank[x] += 1

def KruskalMST(self):
    result = []
    i = 0
    e = 0
    self.graph = sorted(self.graph, key=lambda item: item[2])

    parent = []
    rank = []

    for node in range(self.V):
        parent.append(node)
        rank.append(0)

    while e < self.V - 1:
        u, v, w = self.graph[i]
        i = i + 1
        x = self.find(parent, u)
        y = self.find(parent, v)

        if x != y:
            e = e + 1
            result.append([u, v, w])
            self.union(parent, rank, x, y)

    minimumCost = 0
    print("Edges in the constructed MST")
    for u, v, weight in result:
        minimumCost += weight
        print("%d -- %d == %d" % (u, v, weight))
    print("Minimum Spanning Tree", minimumCost)
```

```
# Taking user input for the number of vertices
V = int(input("Enter the number of vertices: "))
g = Graph(V)

# Taking user input for the edges of the graph
print("Enter the edges (vertex1 vertex2 weight): ")
for i in range(V - 1):
    u, v, w = map(int, input().split())
    g.addEdge(u, v, w)

# Running Kruskal's algorithm
g.KruskalMST()
```

ChatGPT

Here's how the output would look when you run the provided code:

```lua
Enter the number of vertices: 5
Enter the edges (vertex1 vertex2 weight):
0 1 2
0 3 6
1 2 3
1 3 8
2 4 4
Edges in the constructed MST
0 -- 1 == 2
2 -- 4 == 4
0 -- 3 == 6
1 -- 2 == 3
Minimum Spanning Tree 15
```

This output represents the edges in the constructed Minimum Spanning Tree (MST) using Kruskal's algorithm, along with the total weight of the MST.

You

Give the output:

```
import sys

class Graph():
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printMST(self, parent):
        print("Edge \tWeight")
        for i in range(1, self.V):
            print(parent[i], "-", i, "\t", self.graph[i][parent[i]])
```

```
    def minKey(self, key, mstSet):
        min = sys.maxsize
        for v in range(self.V):
            if key[v] < min and mstSet[v] == False:
                min = key[v]
                min_index = v
        return min_index

    def primMST(self):
        key = [sys.maxsize] * self.V
        parent = [None] * self.V
        key[0] = 0
        mstSet = [False] * self.V
        parent[0] = -1

        for cout in range(self.V):
            u = self.minKey(key, mstSet)
            mstSet[u] = True
            for v in range(self.V):
                if self.graph[u][v] > 0 and mstSet[v] == False \
                and key[v] > self.graph[u][v]:
                    key[v] = self.graph[u][v]
                    parent[v] = u

        self.printMST(parent)

# Taking user input for the number of vertices
V = int(input("Enter the number of vertices: "))
g = Graph(V)

# Taking user input for the adjacency matrix representation of the graph
print("Enter the adjacency matrix (separate elements with space):")
for i in range(V):
    g.graph[i] = list(map(int, input().split()))

# Running Prim's algorithm
g.primMST()
```

ChatGPT

Here's how the output would look when you run the provided code:

```mathematica
Enter the number of vertices: 5
Enter the adjacency matrix (separate elements with space):
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
Edge    Weight
0 - 1    2
1 - 2    3
```

```
0 - 3    6
1 - 4    5
```

This output represents the edges in the Minimum Spanning Tree (MST) constructed using Prim's algorithm, along with the weights of the edges.

You

Give the output:

```
# Selection sort in Python
# Time complexity: O(n*n)
# Sorting by finding min_index

def selectionSort(array, size):
    for ind in range(size):
        min_index = ind

        for j in range(ind + 1, size):
            # Select the minimum element in every iteration
            if array[j] < array[min_index]:
                min_index = j

        # Swapping the elements to sort the array
        (array[ind], array[min_index]) = (array[min_index], array[ind])

# Taking user input for the array
arr = []
size = int(input("Enter the size of the array: "))
print("Enter the elements of the array:")
for i in range(size):
    element = int(input())
    arr.append(element)

# Calling the selectionSort function
selectionSort(arr, size)

# Printing the sorted array
print('The array after sorting in Ascending Order by selection sort is:')
print(arr)
```

ChatGPT

Here's how the output would look when you run the provided code:

```sql
Enter the size of the array: 5
Enter the elements of the array:
5
3
8
1
9
```

```
The array after sorting in Ascending Order by selection sort is:
[1, 3, 5, 8, 9]
```

This output shows the sorted array in ascending order using the selection sort algorithm.

You

Give the output:

```
# Python3 program to implement greedy algorithm for graph coloring

def addEdge(adj, v, w):
    adj[v].append(w)
    adj[w].append(v)
    return adj

# Assigns colors (starting from 0) to all
# vertices and prints the assignment of colors
def greedyColoring(adj, V):
    result = [-1] * V

    # Assign the first color to first vertex
    result[0] = 0

    # A temporary array to store the available colors.
    # True value of available[cr] would mean that the
    # color cr is assigned to one of its adjacent vertices
    available = [False] * V

    # Assign colors to remaining V-1 vertices
    for u in range(1, V):
        # Process all adjacent vertices and
        # flag their colors as unavailable
        for i in adj[u]:
            if result[i] != -1:
                available[result[i]] = True

        # Find the first available color
        cr = 0
        while cr < V:
            if available[cr] == False:
                break
            cr += 1

        # Assign the found color
        result[u] = cr

        # Reset the values back to false
        # for the next iteration
        for i in adj[u]:
            if result[i] != -1:
                available[result[i]] = False
```

```
    # Print the result
    for u in range(V):
        print("Vertex", u, " ---> Color", result[u])

# Driver Code
if __name__ == '__main__':
    V = int(input("Enter the number of vertices: "))
    E = int(input("Enter the number of edges: "))

    adj = [[] for i in range(V)]
    print("Enter the edges (vertex1 vertex2): ")
    for i in range(E):
        v, w = map(int, input().split())
        adj = addEdge(adj, v, w)

    print("\nColoring of the graph: ")
    greedyColoring(adj, V)
```

ChatGPT

Here's how the output would look when you run the provided code:

```lua
Enter the number of vertices: 5
Enter the number of edges: 4
Enter the edges (vertex1 vertex2):
0 1
0 2
1 2
1 3

Coloring of the graph:
Vertex 0  ---> Color 0
Vertex 1  ---> Color 1
Vertex 2  ---> Color 2
Vertex 3  ---> Color 0
Vertex 4  ---> Color 1
```

This output represents the colors assigned to each vertex in the graph using the greedy graph coloring algorithm.

You

Give the output:

```
# Python program to solve N Queen Problem using backtracking

def printSolution(board):
    for i in range(N):
        for j in range(N):
            print(board[i][j], end=' ')
        print()
```

```
def isSafe(board, row, col):
    for i in range(col):
        if board[row][i] == 1:
            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solveNQUtil(board, col):
    if col >= N:
        return True

    for i in range(N):
        if isSafe(board, i, col):
            board[i][col] = 1
            if solveNQUtil(board, col + 1) == True:
                return True
            board[i][col] = 0

    return False

def solveNQ():
    board = [[0 for _ in range(N)] for _ in range(N)]

    if solveNQUtil(board, 0) == False:
        print("Solution does not exist")
        return False

    printSolution(board)
    return True

# Taking user input for the size of the chessboard
N = int(input("Enter the size of the chessboard (N x N): "))

# Solving the N Queen Problem
solveNQ()
```

ChatGPT

Here's how the output would look when you run the provided code:

```
mathematica
```

```
Enter the size of the chessboard (N x N): 4
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```

This output represents a possible solution to the N Queen Problem for a 4x4 chessboard. Each '1' in the output represents the position of a queen on the chessboard where no two queens are attacking each other.

You

Give the output:

```
def greet(bot_name, birth_year):
    print("Hello! My name is {0}.".format(bot_name))
    print("I was created in {0}.".format(birth_year))

def remind_name():
    print('Please, remind me your name.')
    name = input()
    print("What a great name you have, {0}!".format(name))

def guess_age():
    print('Let me guess your age.')
    print('Enter remainders of dividing your age by 3, 5 and 7.')
    rem3 = int(input())
    rem5 = int(input())
    rem7 = int(input())
    age = (rem3 * 70 + rem5 * 21 + rem7 * 15) % 105
    print("Your age is {0}; that's a good time to start programming!".format(age))

def count():
    print('Now I will prove to you that I can count to any number you want.')
    num = int(input())
    counter = 0
    while counter <= num:
        print("{0} !".format(counter))
        counter += 1

def test():
    print("Let's test your programming knowledge.")
    print("Why do we use methods?")
    print("1. To repeat a statement multiple times.")
    print("2. To decompose a program into several small subroutines.")
    print("3. To determine the execution time of a program.")
    print("4. To interrupt the execution of a program.")
    answer = 2
    guess = int(input())
    while guess != answer:
        print("Please, try again.")
        guess = int(input())
```

```
    print('Completed, have a nice day!')

def end():
    print('Congratulations, have a nice day!')

greet('Sbot', '2021') # change it as you need
remind_name()
guess_age()
count()
test()
end()
```

ChatGPT

Here's how the output would look when you run the provided code:

```css
Hello! My name is Sbot.
I was created in 2021.
Please, remind me your name.
Daniel
What a great name you have, Daniel!
Let me guess your age.
Enter remainders of dividing your age by 3, 5 and 7.
1
2
1
Your age is 22; that's a good time to start programming!
Now I will prove to you that I can count to any number you want.
5
0 !
1 !
2 !
3 !
4 !
5 !
Let's test your programming knowledge.
Why do we use methods?
1. To repeat a statement multiple times.
2. To decompose a program into several small subroutines.
3. To determine the execution time of a program.
4. To interrupt the execution of a program.
3
Please, try again.
2
Completed, have a nice day!
Congratulations, have a nice day!
```

This output represents the interaction with the bot, where it greets, asks for the name, guesses the age, counts, tests programming knowledge, and ends with congratulations.

You

Give the output:

```
QUESTIONS = [
    'Do you have cough?',
    'Do you have a sore throat?',
```

```
    'Do you have a fever?',
    'Are you noticing any unexplained excessive sweating?',
    'Do you have an itchy throat?',
    'Do you have a runny nose?',
    'Do you have a stuffy nose?',
    'Do you have a headache?',
    'Do you feel tired without actually exhausting yourself?'
]

THRESHOLD = {
    'Mild': 30,
    'Severe': 50,
    'Extreme': 75
}

def expertSystem(questions, threshold):
    score = 0

    for question in questions:
        print(question + " (Y/N) ")
        ans = input("> ")
        if ans.lower() == 'y':
            print('On a scale of 1-10 how bad is it ?')
            ip = input('> ')
            while (not ip.isnumeric()) or int(ip) < 1 or int(ip) > 10:
                print('Enter a valid input !')
                ip = input('> ')
            score += int(ip)

        print("\n\n")

    if score >= threshold['Extreme']:
        print("You are showing symptoms of having EXTREME COVID-19")
        print("Please call +91 8112233445 immediately to immediate assistance")
        print("Based on your symptoms, You will need Immediate Hospitalization")
    elif score >= threshold['Severe']:
        print("Based on your answers You are showing Symptoms of SEVERE COVID-19")
        print("You are advised to contact a COVID-19 Specialist ASAP")
        print("You are prescribed with Favipriavir, Dolo 650 / Crocin 500, Paracetamol, Brufane")
        print("Also conduct a COVID-19 Lab Test ASAP at your own convenience as this might be a false
Positive")
        print("\n\n")
        print("Lab Testing: https://www.metropolisindia.com/parameter/pune/covid-19-rt-pcr-test")
    elif score >= threshold['Mild']:
        print("Based on your answers You are showing Symptoms of VERY MILD COVID-19")
        print("Please Isolate yourself Immediately on a precautionary basis")
        print("As this has a possibility of being a false positive, please consider testing yourself")
        print("At home testing using Self-Testing kits is recommended, but you can get Lab Tests as well")
        print("\n\n")
        print("Self testing : https://www.flipkart.com/mylab-coviself-covid-19-rapid-antigen-test-
kit/p/itm4d34ea09cad97")
```

```
        print("Lab Testing : https://www.metropolisindia.com/parameter/pune/covid-19-rt-pcr-test")
    else:
        print("You are Showing NO Symptoms of COVID-19")
        print("This might be a false negative, If you feel unsure, please get Tested")
        print("As this has a possibility of being a false negative, please consider testing yourself")
        print("At home testing using Self-Testing kits is recommended")
        print("\n\n")
        print("Self testing : https://www.flipkart.com/mylab-coviself-covid-19-rapid-antigen-test-kit/p/itm4d34ea09cad97")

if '__main__' == __name__:
    print("\n\n\t\tWelcome To The COVID-19 EXPERT SYSTEM\n")
    print("\tNote : Please answer the following questions very honestly\n\n")
    expertSystem(QUESTIONS, THRESHOLD)
```

ChatGPT

Here's how the output would look when you run the provided code:

mathematica

```
                Welcome To The COVID-19 EXPERT SYSTEM

    Note : Please answer the following questions very honestly


Do you have cough? (Y/N)
> y
On a scale of 1-10 how bad is it ?
> 5



Do you have a sore throat? (Y/N)
> n



Do you have a fever? (Y/N)
> y
On a scale of 1-10 how bad is it ?
> 8



Are you noticing any unexplained excessive sweating? (Y/N)
> n



Do you have an itchy throat? (Y/N)
> n



Do you have a runny nose? (Y/N)
> n



Do you have a stuffy nose? (Y/N)
> y
On a scale of 1-10 how bad is it ?
> 4
```

```
Do you have a headache? (Y/N)
> y
On a scale of 1-10 how bad is it ?
> 7



Do you feel tired without actually exhausting yourself? (Y/N)
> y
On a scale of 1-10 how bad is it ?
> 6



Based on your answers You are showing Symptoms of SEVERE COVID-19
You are advised to contact a COVID-19 Specialist ASAP
You are prescribed with Favipriavir, Dolo 650 / Crocin 500, Paracetamol, Brufane
Also conduct a COVID-19 Lab Test ASAP at your own convenience as this might be a false Positive



Lab Testing: https://www.metropolisindia.com/parameter/pune/covid-19-rt-pcr-test
```

You

cd c:

ChatGPT

Certainly. It seems you're attempting to change the current directory to the root of the C drive. This can be achieved through the command prompt or terminal interface, depending on your operating system. In Windows, you would typically use the `cd` command followed by the directory path, such as `cd C:\`. This command instructs the system to change the current directory to the root of the C drive.

However, it's important to exercise caution when navigating system directories, especially when operating at root levels. Changes made at this level can have significant impacts on system stability and functionality. Therefore, it's advisable to ensure a clear understanding of the intended actions and their potential consequences before proceeding.

Furthermore, it's worth noting that directory navigation commands may vary slightly between operating systems and command line interfaces. Therefore, it's always beneficial to refer to the specific documentation or help resources provided by your operating system for precise instructions tailored to your environment.

Should you require further assistance or clarification on directory navigation procedures or any related matter, please feel free to ask.