ANALYZING TRAVERSAL ALGORITHMS ON MOVIE DATA SET

Data

Analyze difference in BFS and DFS algorithms

To extract a sample data set from movie data set {dictionary} and implement the **Breadth first search (BFS)** as well as **Depth first search (DFS)** algorithm. After implementation bring out the significant differences in the algorithm's output to understand the functioning of algorithm. Mention some of the useful applications of such algorithm in real life cases.

Mini Project-1:

Team: Sushanth S | Manglam Jain | Jinen Mirje

Methodology

- Finding the shortest and optimized path using BFS and DFS algorithms
- Raw data set collection
- Extracting the selected data from the raw data set
- Understanding the movie data set
- Converting the data set into GRAPHS Traversals: Vertex and exactly in well defined order
- Implementation of BFS and DFS respectively:
    - Traversing the child nodes
    - Traversing process through iterations
    - Pseudocode
- Analysis of output (BFS , DFS)
- Application
- Conclusion

Analyzing Traversal Algorithms on Movie data set

ANALYZE DIFFERENCE IN BFS AND DFS ALGORITHMS

EXPLAINING THE DATA SET

The master movies data consists of Movie Genre, and Movie Name as keys.

Further the Movie Name contains details such as director, Id, Genre, length etc.

The list contains 146 unique movies with their attributes.

From the above dataset a subset is extracted and a sample set is formulated, which is shown below:

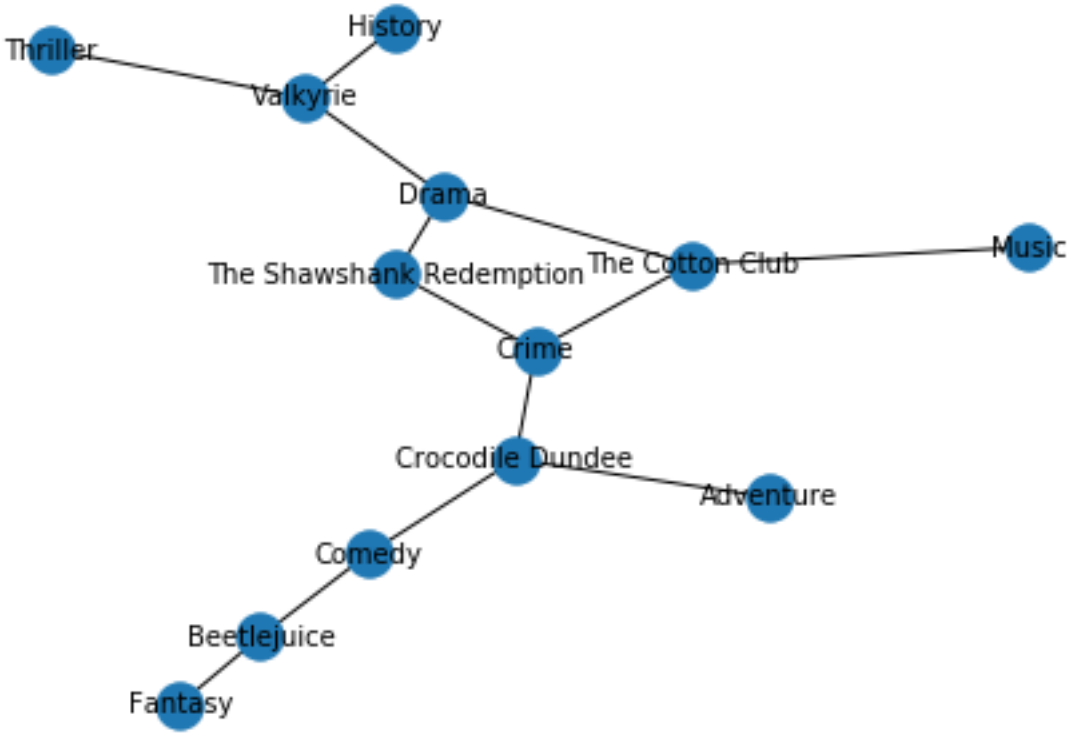
```
d = {'Beetlejuice': {'Comedy': {}, 'Fantasy': {}},
        'The Cotton Club': {'Crime': {}, 'Drama': {}, 'Music': {}},
        'The Shawshank Redemption': {'Crime': {}, 'Drama': {}},
        'Crocodile Dundee': {'Adventure': {}, 'Comedy': {}, 'Crime': {}},
        'Valkyrie': {'Drama': {}, 'History': {}, 'Thriller': {}}, 'Comedy': {'Beetlejuice': {},
        'Crocodile Dundee': {}},  'Fantasy': {'Beetlejuice': {}},  'Crime': {'The Cotton Club': {},
        'The Shawshank Redemption': {}, 'Crocodile Dundee' : {}}, 'Drama': {'The Cotton Club': {},
        'The Shawshank Redemption': {},
        'Valkyrie': {}}, 'Music': {'The Cotton Club': {}},
        'Adventure': {'Crocodile Dundee': {}}, 'History': {'Valkyrie': {}}, 'Thriller': {'Valkyrie': {}}}
```

Converting the Data in Graph

It is useful to convert the data set into a graph format. This can be done through multiple options. Either through an object-oriented programing structure or using a module NetworkX.

In our case we are converting this date to a network of nodes and edges using a network command.

The output of the code resembles as given below:



As it can be observed that all the movies with their respective genres are interconnected to form a network graph.

Implementation of BFS:

Pseudo Code for the implementation:

```
def bfs(dictionary, input_node, target_node):

    # first step is to initialize the empty queue and visited list
    visited = []
    queue = Queue()

    queue.put(input_node)
    visited.append(input_node)

    parent = {}
    parent[input_node] = None

    pathfound = False
```

```
    while not queue.empty():
        a = queue.get()
        if a == target_node:
            pathfound = True
            break

        for i in dictionary[a]:
            if i not in visited:
                queue.put(i)
                parent[i] = a
                visited.append(i)
    print(visited)
    # to get the path and represent it as output

    path = []
    if pathfound:
        path.append(target_node)
        #print(parent)
        while parent[target_node] is not None:
            path.append(parent[target_node])
            target_node = parent[target_node]
        path.reverse()
    return path
```

Implementation of DFS:

Pseudo Code for the implementation:

```
visited = set() # Set to keep track of visited nodes of graph.

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

Analysis the Output from BFS & DFS:

Same input was provided to both the algorithm to understand the path it takes:

Input:

To find the path from movie **"Crocodile Dundee"** and traverse along to reach the last node in the graph.

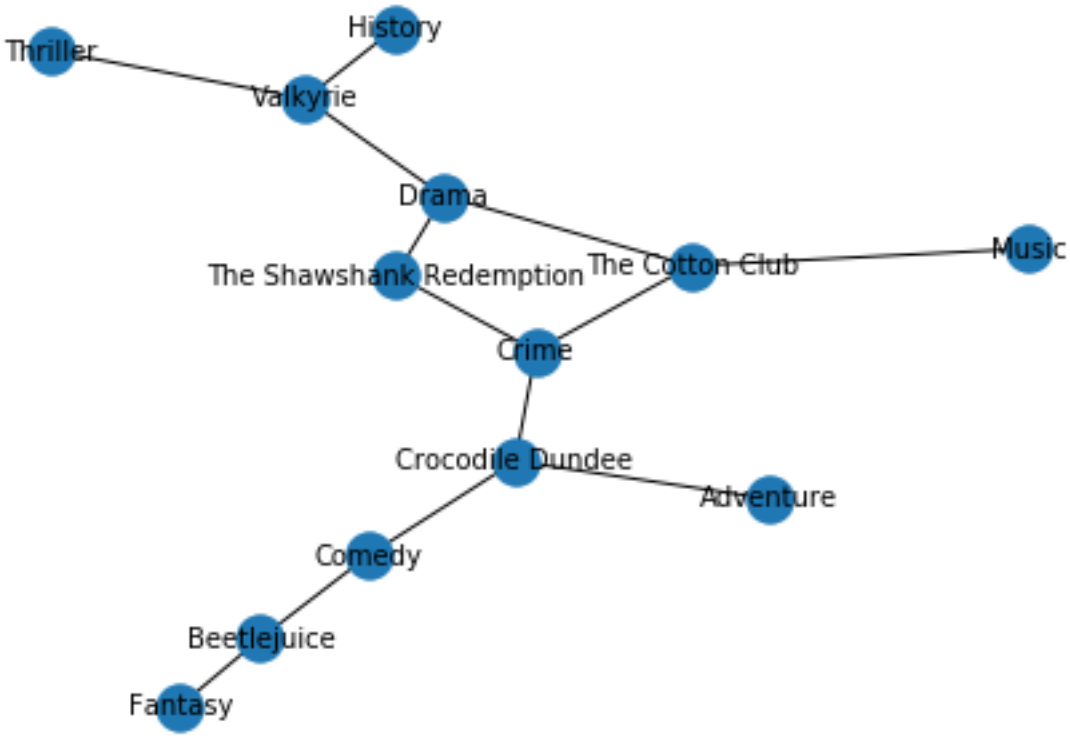
Output of BFS:

```
Crocodile Dundee, Adventure, Comedy, Crime, Beetlejuice, The Cotton Club, The S
hawshank Redemption, Fantasy, Drama, Music, Valkyrie, History, Thriller
```

Output of DFS:

```
Crocodile Dundee, Adventure, Comedy, Beetlejuice, Fantasy, Crime, The Cotton Cl
ub, Drama, The Shawshank Redemption, Valkyrie, History, Thriller, Music
```

Let us understand the functioning of both the algorithms:



BFS: From node (Crocodile Dundee as the central node) it expanded all the adjacent nodes. After every first node is completely expanded the algorithm travels to level-2 nodes that is connecting the first level nodes. So, the farthest point in the node Thriller and History is reached only at the end.

DFS: From node (Crocodile Dundee) it expanded downwards to Comedy and Adventure only. Then further in the graph Comedy can be drilled down. Only after completely reaching to the bottom the next adjacent node crime is searched. To reach the outer most node

APPLICATIONS OF BFS & DFS:

All the data that are hierarchical or follow a tree structure. can be converted into a graphical form with nodes and their vector connected with edges. Then these algorithms can be applied to find the optimum travel distance from one node to the other node. To identify the depth of relations.

Some common applications of this algorithms are:

1) Crawling the web pages: here each page is considered to be one node. When we click a button the target node is selected and displayed.
2) GPS: Each location available in the graph acts as a node.
3) Identifying the topology of a tree structure. When it is not known.

REFERENCES

- **Data set : http://github.com/erik-sytnk/movies-list**
- **https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/**
- **https://brilliant.org/wiki/depth-first-search-dfs/**
- **https://www.tutorialspoint.com/difference-between-bfs-and-dfs**

02 MAIB AI 101 Fundamentals of Artifi

Traversial algorithms.ipynb

****