CENTRO UNIVERSITÁRIO CAMPOS DE ANDRADE
CIÊNCIA DA COMPUTAÇÃO

AUGUSTO CESAR TERRES BUENO DE CAMARGO
HARON BRISAC NORONHA
MATHEUS ALMEIDA NAZÁRIO

GRAFOS APLICADOS NA COMPUTAÇÃO

Curitiba
2022

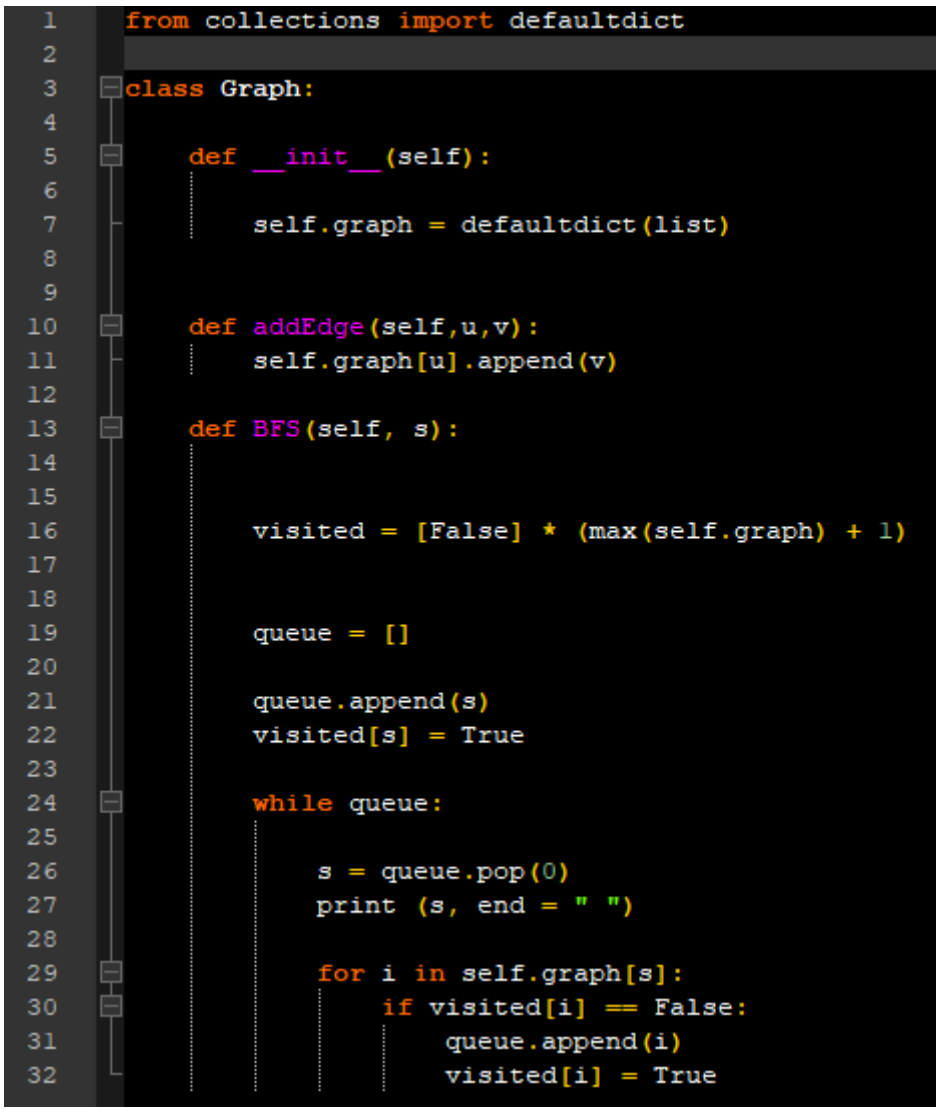
1 INTRODUÇÃO

Este trabalho visa analisar a implementação da Busca em largura (BFS) e da Busca em Profundidade (DFS), com o objetivo de compreender como funcionam e em quais casos eles são utilizados, bem como se aprofundar em um exemplo específico, estudando sua aplicação de forma a ter um entendimento melhor acerca do assunto.

2 DESENVOLVIMENTO

Foram realizados pesquisas a procura de exemplos de implementação do BFS e DFS onde foram encontrados e implementados:

Imagem 1: BFS



```
from collections import defaultdict

class Graph:

    def __init__(self):

        self.graph = defaultdict(list)


    def addEdge(self,u,v):
        self.graph[u].append(v)

    def BFS(self, s):


        visited = [False] * (max(self.graph) + 1)


        queue = []

        queue.append(s)
        visited[s] = True

        while queue:

            s = queue.pop(0)
            print (s, end = " ")

            for i in self.graph[s]:
                if visited[i] == False:
                    queue.append(i)
                    visited[i] = True
```

:

Fonte: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

A função BFS implementada começa sinalizando que nenhum vértice foi visitado, logo em seguida ele realiza e retorna as ligações achadas dentro do “FOR”

Imagem 2 : DFS

```python
from collections import defaultdict

class Graph:

    def __init__(self):

        self.graph = defaultdict(list)

    def addEdge(self, u, v):
        self.graph[u].append(v)

    def DFSUtil(self, v, visited):

        visited.add(v)
        print(v, end=' ')

        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited)

    def DFS(self, v):

        visited = set()

        self.DFSUtil(v, visited)
```

Fonte:https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/?ref=lbp

A função DFS implementada começa sinalizando que nenhum vértice foi visitado, logo em seguida ele chama uma função recursiva que verifica todos os caminho possíveis antes de retroceder e repetir o processo até que não aja nenhum nó adjacente não visitado

3 COMPARAÇÕES

Resultado BFS

```
2 0 7 1 5 6 3 4
Tempo pecorrido: 0.05857992172241211
```

Resultado DFS:

```
2 0 1 3 4 5 6 7
Tempo pecorrido: 0.0646705627441406 2
```

Enquanto BFS demorou +- 0.058s se provou mais rapido que o DFS que demorou 0.064s

4 DIFICULDADES

A maior dificuldade que tivemos foi na compreensão dos códigos achados onde realizamos diversos teste a fim de entender o funcionamento do código

5 CONCLUSÃO

Após a compreensão dos códigos nos os modificamos para a exibição do tempo decorrido ao executar o BFS ou DFS e implementamos a possibilidade de inserir grafos por um arquivo CSV

6 REFERÊNCIAS

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/