EXPERIMENT 17

Python program for implementing dfs

|  |
| --- |
| class Graph: |
|  | def \_\_init\_\_(self, num\_vertices): |
|  | self.num\_vertices = num\_vertices |
|  | self.adj\_matrix = [[-1] \* num\_vertices for \_ in range(num\_vertices)] |
|  |  |
|  | def add\_edge(self, v1, v2, weight=1): |
|  | self.adj\_matrix[v1][v2] = weight |
|  |  |
|  | def remove\_edge(self, v1, v2): |
|  | self.adj\_matrix[v1][v2] = -1 |
|  |  |
|  | def \_\_dfs\_helper(self, temp, v, visited): |
|  | temp.append(v) |
|  | visited[v] = True |
|  |  |
|  | for i in range(self.num\_vertices): |
|  | if self.adj\_matrix[v][i] > 0 and not visited[i]: |
|  | temp = self.\_\_dfs\_helper(temp, i, visited) |
|  | return temp |
|  |  |
|  | def dfs(self, v): |
|  | visited = [False for \_ in range(self.num\_vertices)] |
|  | return self.\_\_dfs\_helper([], v, visited) |
|  |  |
|  | g = Graph(4) |
|  | g.add\_edge(0, 1) |
|  | g.add\_edge(0, 2) |
|  | g.add\_edge(1, 2) |
|  | g.add\_edge(2, 0) |
|  | g.add\_edge(2, 3) |
|  | g.add\_edge(3, 3) |
|  |  |
|  | print(g.dfs(2)) |