CSE 221 Algorithm

Fall 2023

Lab Assignment 4

Fathin Ishrak

ID: 2030102X

Section: 13

task 1 (a)

The code defines a function "create_adjacency_matrix" that generates an adjacency matrix with N+1 rows and columns. It reads input from a file containing vertex count N, edge count M and edge details. It then populates the adjacency matrix based on the provided edges, ~~and this~~

task 1 (b)

The code reads input from "input(b).txt", creates an adjacency matrix based on the edges, and writes an adjacency list format into "output1(b).txt". Each row in the output represents a vertex and its connected edges, with an empty row for vertex "0".

task 2

The "bfs" function performs a breadth-first search traversal starting from a given city "start", maintaining a set of visited cities and a queue to explore neighbors. The "main" function reads input data from the file to construct the graph, performs BFS from city 1 and writes the BFS traversal order into an output file.

task 3

The "dfs" function recursively explores each node's neighbors starting from city 1, marking visited nodes and recording the traversal order. It reads graph information from the input file and initializes an empty set for visited nodes and a ~~set~~ list to store the traversal order. Finally writes the DFS traversal path into the output file by converting the traversal_order list into a space-separated string of integers

task 4

The "has_cycle" function employs DFS and auxiliary methods to detect cycles within a directed graph, recording node visits and recursion stack to identify cycles. It reads the input file containing a directed graph's information and finally outputs a corresponding "Yes" or "No" result based on cycle presence, using DFS traversal algorithm.

task 5

Import necessary libraries and defines a function to compute the shortest path from city 1 to a destination city 0. Reads the input file to create a graph and executes Dijkstra's algorithm to find the shortest path. Determine the minimum time and shortest path, writing the results to the output path.

task 8

The code reads the number of test cases from the input file and iterate through each case. Then it creates a dictionary to count occurrences of competitors based on dual fights. Finally determine the maximum count of rivals for each case and write it along with the case number to an output file.

task 6

This code builds a 2D array using numpy. which includes path(.) as 0, diamond (D) as 1 ~~and~~, obstacle (#) as -1. and a visited array with all the initial values are false. ~~The dfs function~~ This code use DFS to find the maximum number of diamonds in a grid. It iterates through the grid, explores reachable cells and counts the maximum diamonds encountered during the exploration. The count is written to an output file after processing the input grid.

task 7

This code finds two cities, city_A and city_B, that allow the maximum number of cities to be visited without using the same road twice. It uses DFS to calculate the maximum number of cities reachable from each city. The function find_maximum_cities iterates through the graph and records the pair of city (city_A and city_B) that yields the highest count. Finally, it writes these cities to an output file.