National Yang Ming Chiao Tung University
Department of Electrical Engineering
Data Structures, Fall 2022

October 12, 2022
Instructor: Mark P.-H. Lin
TA: J. Y. Weng

Programming Assignment #2

Stacks & Queues

1 Problem Description

As a "Sky Dragon City" resident, there was usually no reason to leave your city before, but today, a title-decided baseball derby will be held at H City Baseball Stadium tonight. You definitely do not want to miss it. Therefore, you immediately read the map to find out the path to H City. Luckily, you have taken the data structure course this semester, and come up with a powerful algorithm: depth-first search.

First of all, the classes of both city and path are denoted as follows, where a city has its variables of $Index$, $Name$, and $Visit$, whose types are unsigned integer (UINT), string (STRING), and Boolean (BOOL), respectively, and a path has its $Source\ City\ Index$, $Destination\ City\ Index$, and $Distance$, whose types are all UINT.

Class CITY
  $Index$: UINT
  $Name$: STRING
  $Visit$: BOOL
end Class
Class PATH
  $Source\ City\ Index$: UINT
  $Destination\ City\ Index$: UINT
  $Distance$: UINT
end Class

The depth-first search (DFS) algorithm starts at the Sky Dragon City, as seen in Figure 1. It explores as far as possible along each path before backtracking. To implement the DFS algorithm, the stack is an appropriate data structure to store visited cities and determine which is the next city to explore. The pseudo-code of the DFS algorithm is given in Algorithm 1.

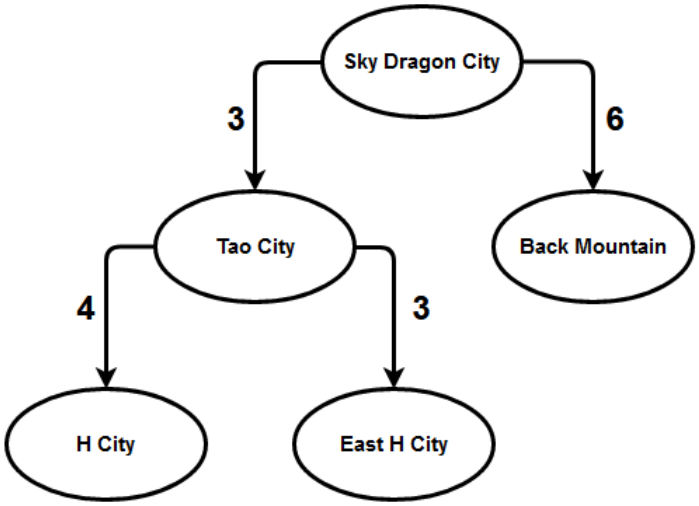

Figure 1: A city graph, consisting of 5 cities and 4 paths.

Algorithm 1 DFS Algorithm with Stack.

Input: city graph G, target city name C_t
Initialize an empty city stack S
$N \leftarrow G.starting\_point()$;
$S.push(N)$;
while city name c of $S.top()$!= C_t **do**
    if $S.empty()$ **then**
        return path not found;
    end if
    $N \leftarrow S.pop()$
    for N's all connected cities n **do**
        if n had not been visited **then**
            $S.push(n)$;
        end if
    end for
end while
return path found;

In this programming assignment, you are asked to implement the DFS algorithm with stack on a given city graph and a given target city. To simplify the problem, we have the following assumptions:

1. The DFS algorithm always starts from Sky Dragon City, with the city index, 0.
2. Every city has *exactly one* source city except Sky Dragon City.
3. Every path is *one-way* from its source city to its destination city.

Hints: You can simply construct the city graph with a 2-D array or a vector.

2 Input Format

The first line of the input file gives the number of C cities. The subsequent C lines give the city names with the indices from 0 to $C-1$. The $(C+1)^{th}$ line further gives the number of path P, and then the subsequent P lines detail the information of each path, including the *Source City Index*, *Destination City Index*, and *Distance, D*. Finally, the last line of the input file gives the target city name. Please be noted that $0 < C, P, D \leq 10000$.

A sample input file is given below:

Sample Input	Comments
5	5 cities in the city graph, with indices, 0, 1, 2, 3, and 4
Sky Dragon City	city name of city 0, this line is always Sky Dragon City
Back Mountain	city name of city 1
Tao City	city name of city 2
H City	city name of city 3
East H City	city name of city 4
4	4 paths in cities graph
0 1 6	a path from city 0 to city 1 with $Distance = 6$
0 2 3	a path from city 0 to city 2 with $Distance = 3$
2 3 4	a path from city 2 to city 3 with $Distance = 4$
2 4 3	a path from city 2 to city 4 with $Distance = 3$
H City	target city name

3 Output Format

The output file contains two parts:

1. *Stack push/pop instruments during DFS algorithm.* Output *push n* when push a city with index n into the stack and *pop n* when pop.

2. *The total distance from Sky Dragon City (city 0) to the city with search target city name.* If the city with search target city name or a valid path from Sky Dragon City to the city with search target city name is *not* exist, output -1.

Sample Output	Comments
push 0	push city 0 (Sky Dragon City) into stack
pop 0	pop city 0
push 1	push city 1 into stack
push 2	push city 2 into stack
pop 2	pop city 2
push 3	push city 3 into stack
push 4	push city 4 into stack
pop 4	pop city 4
7	total distance from Sky Dragon City to H City

4 Command-line Parameter

In order to test your program, you are asked to add the following command-line parameters to your program:

[executable file name] **[input file name]** **[output file name]**

5 Submission Information

- Your program must be written in the C/C++ language and can be compiled on the Linux platform.
- The source files of your program must be named with "[your student ID].h" and "[your student ID].cpp".
- To submit your program, please archive all source files of your program into a single zip file, named "[your student ID].zip", and upload it to E3.

6 Due Date

Be sure to upload the zip file by "Wednesday, October 26, 2022". There will be a 25% penalty per day for late submissions.

7 Grading Policy

The programming assignment will be graded based on the following rules:

- Pass sample input with compilable source code (50%)
- Pass five hidden test cases (50%)

The submitted source codes, which are either copied from or copied by others, will NOT be graded.