Project 3: Assembling the BatSignal

CMPE 250, Data Structures and Algorithms, Fall 2020

Instructor: H. B. Yılmaz

TAs: Berk Atıl, Rıza Özçelik

Due: **January 4, 2021, Monday, 23.55**

"Everything is impossible until somebody does it" – Bruce Wayne

1 Description

Joker plundered Gotham City and the city is in big trouble! Commissioner Gordon went to the root of the Police Department to make the apparent move: lighting the BatSignal to call BatMan. But wait! There is nothing on the roof but a note: "Did you lose BatSignal? I know where it is! It is in pieces and distributed to every single road in your damn city! Good job calling BatMan! :)".

Figure 1: Current situation in Gotham City

Having read the note, Gordon feels desperate. He cannot handle Joker by himself and definitely needs to call BatMan. To light up the BatSignal, he needs to collect the pieces of BatSignal and assemble them on the rooftop. Thus, he has to find a quick way to traverse every single road in Gotham City and come back to the headquarters. Can you help him by drawing such a route?

You know what? To make things faster, we will help you as well by providing the algorithm to find a route as desired. Keep reading!

2 Hierholzer's Algorithm

A graph is called Eulerian if it contains a circuit that contains every edge exactly once. A directed graph is Eulerian if and only if in-degree of a vertex v is equal to its out-degree, for all v in the vertex set of G.

Hierholzer's algorithm finds an Eulerian circuit of an Eulerian graph by iteratively finding and merging tours (a walk with no repeated edges). It picks a starting vertex for the circuit and traverses the non-traversed edges arbitrarily, until it is stuck in a vertex,[1] completing a tour. When it is stuck, it merges the found tour with the known Eulerian circuit (initially empty) and finds a vertex in the **current circuit** with non-traversed outgoing edges. It starts a new tour from this vertex and merges the new tour with the known Eulerian circuit again. The iterations continue until the Eulerian circuit is fully constructed. Note that it can always merge the known Eulerian circuit with the found tour, since they share at least one common vertex (starting vertex of the tour) and their edge sets are disjoint (we traverse only non-traversed edges).

Depending on three parameters, Hierholzer's Algorithm can find different Eulerian circuits. We explain and fix these parameters as follows to ease the grading.

1. **Starting vertex:** We provide a vertex ID in the input file to start and end the circuit.
2. **Edge to traverse when there are multiple options:** Traverse the edge that ends in the vertex with the lowest ID.
3. **Vertex to start a new tour:** Use the vertex that is closest to the beginning of the current circuit.

You are expected to strictly obey these rules. You can see the pseudocode of the algorithm in Algorithm 1.

Algorithm 1 Hierholzer's Algorithm
Input: A graph $G = (V, E)$ and a starting vertex v
Output: An Eulerian circuit of G if G is Eulerian, [] otherwise. $v \in V$

```
1:  if not isEulerian(G) then
2:      return [ ]
3:  eulerianCircuit ← [v]
4:  while eulerianCircuit.length <= |E| do
5:      tour ← [ ]
6:      while v.hasNonUsedEdge() do
7:          (v, u) ← v.getFirstNonUsedEdge()
8:          mark (v, u) as used
9:          v ← u
10:         tour.append(v)
11:     eulerianCircuit.merge(tour)
12:     v ← eulerianCircuit.findFirstVertexinTheCircuitWithAnUnusedEdge()
13: return eulerianCircuit
```

[1] The algorithm is stuck in a vertex if every outgoing edge of the vertex is already traversed.

3 Input & Output

Your code must read the name of the input and output files from the command line. We will run your code as follows:

- g++ *.cpp *.h -std=c++11 -o project3
- ./project3 inputFile outputFile

For sure, you do not have to create any ".h" file if you do not need it. In that case, your code will be compiled as follows: g++ *.cpp -std=c++11 -o project3
Make sure that your final submission compiles and runs with these commands.

3.1 Input

The map of Gotham City is provided in the following format:

- The first line contains an integer V that denotes the total number of crossing points of roads (vertices).
- The next V lines contains an integer (Vertex ID), another integer (D^+) which is the out-degree of this vertex, and D^+ integers that represent the Vertex IDs to which there is an edge from this vertex.
- The last line contains an integer denoting the ID of the starting vertex.

Table 1 demonstrates an example input and Figure 2 illustrates the schematic description of the input.

Input File	Output File
6 0 2 1 1 1 3 0 2 3 2 1 3 3 2 4 5 4 1 5 5 2 0 1 0	0 1 3 5 1 0 1 2 3 4 5 0

Table 1: Sample input and output file

3.2 Output

Gordon wants you to find a route that contains all of the roads exactly once. Hence you will print a sequence of vertex IDs separated by space. Table 1 illustrates an example output file and Figure 3 demonstrates the order of the edges as they are included in the circuit. **Note that the edge inclusion order and vertex order in the output is different since tours are merged to produce the final circuit.**

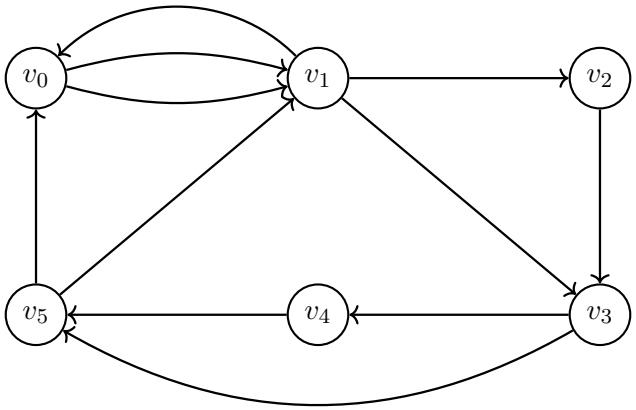

Figure 2: The schematic description of the input file as a graph.

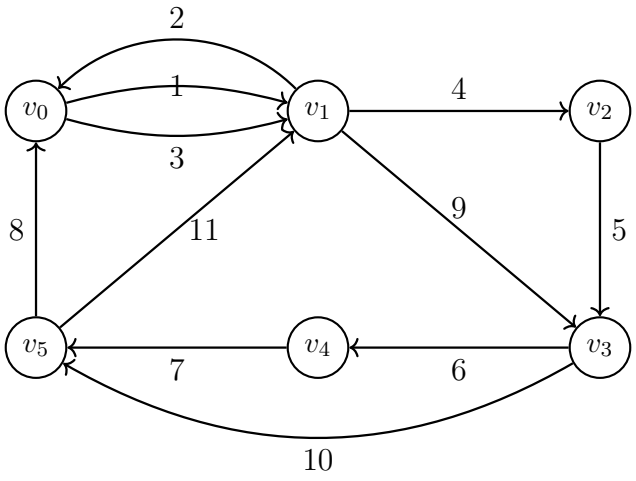

Figure 3: The inclusion order of the edges to final circuit.

Some Remarks Regarding The Project

- ID of the crossing points start from **0** and ends at $V - 1$.
- It is possible that there is no path that traverses all of the edges exactly once. In this case, you should print **"no path"** and do not run the Hierholzer's algorithm at all.
- **Follow the algorithm as described in the pseudocode instead of different versions you can find online.** Note that you are encouraged to study the algorithm further but not allowed to use any existing code. You are highly recommended to close a web page immediately if contains an implementation, since you might be inadvertently affected by that specific implementation.
- All roads are **one-way**, i.e. directed.

Grading

We will automatically grade your submission with multiple test cases besides the ones we already provided. Therefore, **ensure that your submission is runnable with the commands we provided and satisfies all project requirements.** The auto-runnability of the submission will constitute **10/100** of your final grade and passing the test cases will earn your **90/100** more points. **We expect you to print the Eulerian circuit created by the algorithm we provided, not any of them, to pass all test cases with full points.** If your submission is not auto-runnable, then **you will receive 0 at first and then have to issue an objection**.

Submission Details

You are supposed to use GitHub Classroom for the submission same as the previous projects. No other type of submission will be accepted. Also, pay attention to the following points:

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. **Make sure you write and submit your own code**.
- You can add as many files as you can as long as they are in the same folder with main.cpp.
- You are allowed to use **only** the standard C++ library. All requirements of the project must be satisfied with your authentic code, not by any external code. **Otherwise, you will get 0**.
- You are expected to use C++ as powerful, steady, and flexible as possible. Use mechanisms that affect these issues positively.
- Make sure you document your code with necessary inline comments and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long. This is very important for partial grading.
- Try to write as **efficient** (both in terms of space and time) as possible since we will **limit the run times** of the submissions to detect the correctness of your implementation.