C++ Implementation of Graph Algorithms (worth 10%, due June 6th 23:59PM, late submissions not accepted)

Mingyu Guo

1 Task Description

You are asked to use C++ to solve the following puzzle.

Hint: All it takes is an algorithm mentioned in this course (with a slight twist).

Update on May 10th: The graph is undirected!

2 Submission Guideline

You must follow this guideline! Your submission will be marked automatically. Failure to follow this guideline will result in 0.

Your submission should contain exactly one file: `main.cpp`.

You do not need to submit a design.

3 Puzzle

You need to redesign the road system of an imaginary country.

The country is composed of N cities (for simplicity numbered from 0 to $N-1$). Some pairs of cities are connected by bidirectional roads. We say that there is a path between different cities A and B if there exists a sequence of unique cities $C_1, C_2, \ldots, C_M$, such that $C_1 = A$ and $C_M = B$ and for each index $i < M$, there is a road between cities C_i and C_{i+1}.

The current state of the road network is miserable. Some pairs of cities are not connected by any path. On the other hand, other pairs of cities are connected by multiple different paths, and that leads to complicated traffic routing. You want to build some new roads and destroy some of the already existing roads in the country so that after the reconstruction there will exist exactly one path between every pair of distinct cities. As building new roads and destroying old ones costs a lot of money, you want to minimize the total cost spent on the reconstruction.

You are given three two-dimensional arrays:

- `country[i][j]=1 or 0`: there is an existing road between city i and j if and only if `country[i][j]=1`.
- `build[i][j]`: the cost for building a road between i and j. The values of `build[i][j]` are represented using English letters. $A, B, \ldots, Z$ represent $0, 1, \ldots, 25$ and $a, b, \ldots, z$ represent $26, 27, \ldots, 51$. For example, if `build[2][4]=b`, then that means the cost for building a road between city 2 and city 4 is 27.
- `destroy[i][j]`: the cost for destroying a road between i and j. Again, the values are represented using English letters like the above.

Your task is to find and print the minimal cost needed for the road network reconstruction.

You don't need to worry about invalid inputs.

- Sample input 1: `000,000,000 ABD,BAC,DCA ABD,BAC,DCA`

  Note: `000,000,000` describes the two-dimensional array `country`. `ABD,BAC,DCA` describes the two-dimensional array `build`. `ABD,BAC,DCA` describes the two-dimensional array `destroy`. The input format is: three strings separated by spaces; each string contains N parts separated by commas; each part contains N characters.

  Sample output 1: 3

  Comment: There are three cities, totally disconnected.

- Sample input 2: `011,101,110 ABD,BAC,DCA ABD,BAC,DCA`

  Sample output 2: 1

  Comment: Now the three cities form a connected triangle and we need to destroy one road. Optimal solution is to destroy the road between the cities 0-1 (cost 1).

- Sample input 3: (note: all inputs are on the same line. I just couldn't fit them in one line in this pdf.)

  `011000,101000,110000,000011,000101,000110`

  `ABDFFF,BACFFF,DCAFFF,FFFABD,FFFBAC,FFFDCA`

  `ABDFFF,BACFFF,DCAFFF,FFFABD,FFFBAC,FFFDCA`

  Sample output 3: 7

  Comment: We have six cities forming two separate triangles. Destroy one road in each triangle (costs 1 for each road) and then join the triangles by a new road (costs 5).

- Sample input 4: 0 A A

  Sample output 4: 0

  Comment: One city is okay just as it is.

- Sample input 5: `0001,0001,0001,1110 AfOj,fAcC,OcAP,jCPA AWFH,WAxU,FxAV,HUVA`

  Sample output 5: 0

  Comment: We have four cities, which are connected in such a way that there is exactly one path between each two cities.

  Thus there is nothing to reconstruct.

4 Marking

Marking will be done automatically. The total mark is 10 (1 for compiling and 9 for 9 test cases).

5 SVN Instructions

First of all, you need to create a directory under version control:

```
svn mkdir --parents -m "Creating ADSA Assignment 4 folder" https://version-control.adelaide.edu.au/svn/aXXXXXXX/2021/s1/adsa/assignment4/
```

aXXXXXXX should be your student ID. The directory path needs to be exactly "2021/s1/adsa/assignmentK", where "K" is the assignment number. To check out a working copy, type

```
svn checkout https://version-control.adelaide.edu.au/svn/aXXXXXXX/2021/s1/adsa/assignment4/ adsa-21-s1-assignment4/
cd adsa-21-s1-assignment4
svn add main.cpp
```

Commit the files to SVN:

```
svn commit -m "Adding ADSA assignment 4 main.cpp"
```

SVN helps keeping track of file changes (over different commits). You should commit your work early and often.

6 Websubmission

You are asked to submit via the web interface `https://cs.adelaide.edu.au/services/websubmission/`. The submission steps should be self-explanatory. Simply choose the correct semester, course, and assignment. The websubmission system will automatically fetch the latest version of your work from your SVN repository (you may also choose to submit older versions). Once your work is submitted, the system will launch a script checking the format of your submission. Click "View Feedback" to view the results. Your mark will be calculated offline after the deadline. You are welcome to resubmit for as many times as you wish (before the deadline).

We will compile your code using `g++ -o main.out -std=c++11 -O2 -Wall main.cpp`. It is your responsibility to ensure that your code compiles **on the university system**.[1]

[1] g++ has too many versions, so being able to compile on your laptop does not guarantee that it compiles on the university system. You are encouraged to debug your code on a lab computer (or use SSH).