ITU Computer Engineering Department
BLG 223E Data Structures, Summer 2021
Homework #8
Due August 13, 2021 23:59

There's no typo, homework duration is two days!!!

Definition

You are asked to implement a program that finds all the walks in a graph that follows a requested pattern. Please pay attention to the following details in your implementations:

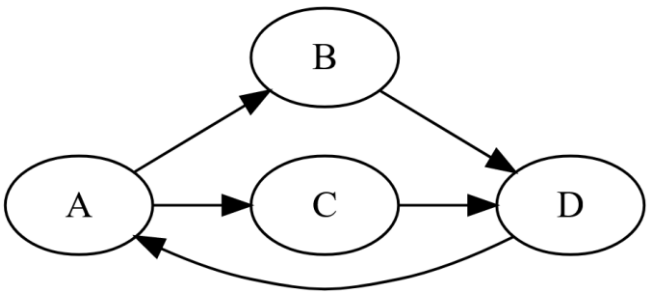
- Your program should read a graph from a file as input followed by the pattern definition and enlist all the walks that is consistent with the pattern.
- For this homework, efficiency is important, please try to optimize your program run, especially by using graph theoretical concepts as much as possible.
- Hamiltonian path problem is partly similar with this homework. You may check out this web page: https://www.geeksforgeeks.org/hamiltonian-path-using-dynamic-programming/
- Problem gets really huge for walks with even small lengths such as 6, be careful, trying out all the combinations takes a long time.

Input-Output

Your program should accept a filename as a command line parameter. In the accepted file:

- First line is going to contain a single number that gives the dimension of the square adjacency matrix. Let's assume this number is N
- Next N lines presents rows of the adjacency matrix. Each cell in the matrix is 1 if there's an edge between two vertices, 0 otherwise.
- You should assume that the first row starts with capital English letter 'A', second row as 'B' and do the vertice labeling accordingly.
- Next N lines present the pattern that is required to hold in the walks. Each line contains a single character (vertice label) followed by a single integer such as "A 2". It is expected for the specific vertice to repeat as much as the integer in the walk. For the previous example vertice A should be visited exactly twice in the produced walks.
- Getting a 0 as the repetition number means that vertice will not be present in the walk.
- Getting a -1 as the repetition number means that there's no bound for the number of repetitions for that vertice. It may be repeated any number of times or not be presented in the walk at all.
- The last line of file contains a single integer that gives you the walk length (in terms of vertices) that you need to produce.
- Your program should output to command line alphabetically sorted walks.

Input will not contain syntactical or semantic errors. See the provided test cases for examples. An example graph, a possible input file fort he graph and the output fort he input file is provided below.



```
4
0 1 1 0
0 0 0 1
0 0 0 1
1 0 0 0
A 2
B -1
C -1
D 2
6
```

Output: ABDABD ABDACD ACDABD ACDACD BDABDA BDACDA CDABDA CDACDA DABDAB DABDAC DACDAB DACDAC

Deliver

Please zip and deliver the directory structure defined below:

- HW8: Topmost folder, that will contain all the folders in your submission. No other files should be present under this folder in your submission.
- HW8/src: Contains all the *.cpp files
- HW8/src/main.cpp: Contains your main function and other code you want to deliver.
- HW8/bin: An empty directory that will contain objective files when your project is compiled

Please check the calico test file in the homework definition to see how your files will be compiled and tested.

GRADING

Specific to this homework you will be graded as follows:

- 1/15 for a compilable code
- 6/15 for passing provided test cases
- 8/15 for the performance grade of your code. Your code's performance will be measured by issuing time command of Linux and getting the user time. Performance grading will be done by sorting the running time of the 7/15 graded submissions in the previous two items. Runtime performance will be measured for a different test case than the ones delivered with the homework.

Restrictions and Guidelines

- Compilation environment: Only the code that can be compiled by the environment of the container definition provided in ninova will be accepted.
- Testing of your program will be performed using Calico (https://calico.readthedocs.io/en/latest/). Test cases that will be used to test your homework is provided as an attachment in ninova.
- **STL usage is allowed. Using any non-standard library is strongly prohibited.**
- **This homework is for individual submissions**. Any kind of code sharing or code adaptation from an external source is strictly forbidden. Submitted code will undergo a plagiarism check process and plagiarsim penalties may be given if the submitted code's similarity is approved by the instuctor.
- Make sure you write your name and number in all of the files of your project, in the following format:
  /* @Author
  Student Name:<studentname>
  Student ID :<studentid>
  Date:<date>*/
- Only electronic submissions through Ninova will be accepted no later than deadline
- Use comments wherever necessary in your code to explain what you did.