[CENG 315 ALL Sections] Algorithms

Description | Submission view

THE5

Available from: Saturday, December 9, 2023, 12:00 PM
Due date: Sunday, December 10, 2023, 11:59 PM
Requested files: the5.cpp, test.cpp, the5.h (Download)
Type of work: Individual work

Your friend is writing a new compiled programming language called **Smurf.** He believes that it will be one of the best programming languages ever. He has almost finished implementing the compiler but is struggling with one last issue.

In Smurf, programmers can write their codes into multiple files and can import other files or libraries. Before the compiler compiles a file, the files that the file to be compiled depends on must be already compiled. Therefore, the compiler should know the compilation order of files. Also, for some programs, compilation may not be possible because of cyclic dependencies (an example of cyclic dependency, two programs may be trying to import each other). In the case of cyclic dependency, the compiler should warn the user by giving the sets of files that cause the cyclic dependency. You offer to help your friend by writing a function to help this compiling process.

Problem

In this exam, you are expected to implement the ***run*** function shown below. If the program you are trying to compile is compilable, you should assign true to the **isCompilable** variable, and assign the order of compilation shown by the file IDs to the **compileOrder** variable. If it is not compilable due to cyclic dependencies, you should assign the IDs of cyclic dependent files to the **cyclicDependencies** variable, a vector including vectors for each cyclic-dependent file group.

```
void run(const std::vector<std::vector<int>>& dependencyMatrix, bool& isCompilable,

        std::vector<int>& compileOrder, std::vector<std::vector<int>>& cyclicDependencies)
```

- ***dependencyMatrix:*** Square matrix with dimensions FxF, where F is the number of files. If dependencyMatrix[i][j] is 1, then file ***j is dependent on file i***. Otherwise, it is 0.
- ***isCompilable:*** If compilation is possible, you should assign true, otherwise false.
- **compileOrder:** If compilation is possible, you should assign compilation order to this variable.
- **cyclicDependencies:** If compilation is not possible, you should find cyclic-dependent file groups and update this vector to include the file groups by their IDs. ***Cyclic-dependent file group is a group of files that each file in the group is directly or indirectly dependent on every other file in the group***.

Constraints and Hints:

- If the compilation order does not matter for two files, you should sort them via indexes. (Smaller index, higher priority)
- Self-import is also possible. You should check them. Also, there may be indirect dependencies.(i.e 1 -> 2 -> 3 -> 1)
- The return order of cyclic-dependent files is not important and each file can be an element of at most one cyclic-dependent file group.
- The number of files is less than 100.

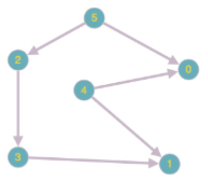
Evaluation:

- After your exam, black-box evaluation will be carried out. You will get full points if you return the correct order or the correct set of cyclic-dependent files. The grade you see in the VPL contains 50% of your final grade. We will evaluate your grades with different inputs after the end of the exam.

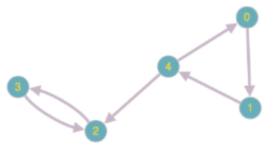
Example IO:

1)

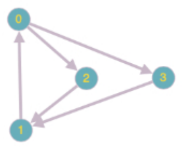
```
Dependency matrix:

0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 1 0 0
0 1 0 0 0 0
1 1 0 0 0 0
1 0 1 0 0 0

Output:

Compilable = true

Compilation Order = 4 -> 5 -> 0 -> 2 -> 3 -> 1
```



```
2)

Dependency matrix:

0 1 0 0 0
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
1 0 1 0 0

Output:

Compilable = false

Cyclic Dependencies = [0, 1, 4], [2, 3]
```



```
3)

Dependency matrix:

0 0 1 1
1 0 0 0
0 1 0 0
0 1 0 0

Output:

Compilable = false

Cyclic Dependencies = [0, 1, 2, 3]
```



Specifications:

- There is 1 task to be solved in **36 hours** in this take-home exam.
- You will implement your solutions in ***the5.cpp*** file.
- You are free to add other functions to *the5.cpp*
- **Do not change** the first line of the5.cpp, which is #include "the5.h"
- *<vector>, <utility> and <memory>* are included in "the5.h" for your convenience, you can use them freely.
- **Do not change** the arguments and the return value of the function ***run()*** in the file the5.cpp
- **Do not include** any other library or write include anywhere in your the5.cpp file (not even in comments).
- You are given ***test.cpp*** file to test your work on **ODTUClass** or your **locale**. You can, and you are, encouraged to modify this file to add different test cases.
- If you want to test your work and see your outputs you can compile your work on your locale as:

```
>g++ test.cpp the5.cpp -Wall -std=c++11 -o test

> ./test
```

- You can test your the5.cpp on the virtual lab environment. If you click **run**, your function will be compiled and **executed with test.cpp**. If you click **evaluate**, you will get **feedback** for your current work and your work will be **temporarily graded** for a limited number of inputs.
- The grade you see in lab is not your final grade, **your code will be reevaluated with different inputs** after the exam.
- I/O used in VPL can be found here.

The system has the following limits:

- a maximum execution time of 16 seconds (your program needs to return in less than one second per test case on average)
- a 1 GB maximum memory limit,
- an execution file size of 4M.
- Solutions with longer running times will not be graded.
- If you are sure that your solution works in the expected complexity, but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.

Requested files

the5.cpp

```
#include "the5.h"

// do not add extra libraries here

```

```cpp
void run(const std::vector<std::vector<int>>& dependencyMatrix,
         bool& isCompilable,
         std::vector<int>& compileOrder,
         std::vector<std::vector<int>>& cyclicDependencies){

         }
```

test.cpp

```cpp
#include <iostream>
#include <fstream>
#include "the5.h"


void test(){
    char inp_file[] = "inp01.txt"; // 01 to 10 is available
    int size;
    std::ifstream infile (inp_file);
    if(!infile.is_open()){
        std::cout << "Input file cannot be opened" << std::endl;
        std::cout << "File name: " << inp_file << std::endl;
        return;
    }
    infile >> size;
    std::vector<std::vector<int>> input_array(size, std::vector<int>(size, 0));
    std::vector<int> compilation_order;
    std::vector<std::vector<int>> cyclic_dependencies;
    bool compilable = false;

    for(int idx=0; idx < size ; idx++)
        for(int idy=0; idy < size; idy++)
            infile >> input_array[idx][idy];

    std::cout << "Number of files: " << size << std::endl <<
                 "Dependency matrix: [" << std::endl;
    for(int idx=0; idx < size; idx++){
        std::cout << "                        [";
        for(int idy=0; idy < size-1; idy++){
            std::cout << input_array[idx][idy] << ", ";
        }
        std::cout << input_array[idx][size-1] << "]" << std::endl;
    }
    std::cout << "                   ]"<< std::endl;

    run(input_array, compilable, compilation_order, cyclic_dependencies);
    if(compilable){
        std::cout << "Compilation is possible." << std::endl;
        std::cout <<"Sorted order: {";
        for(int idx=0; idx < size-1; idx++) std::cout << compilation_order[idx] << ", ";
        std::cout << compilation_order[size - 1] << "}" << std::endl;
    }
    else{
        std::cout << "Compilation is not possible. Cyclic dependencies: " << std::endl;
        for(int idx=0; idx < (int)cyclic_dependencies.size(); idx++){
            std::cout << (idx+1) << ": ";
            for(int idy=0; idy < (int)cyclic_dependencies[idx].size()-1; idy++){
                std::cout << cyclic_dependencies[idx][idy] << " ";
            }
            std::cout << cyclic_dependencies[idx].back() << std::endl;

        }
    }
    std::cout << "---------------------------------------------" << std::endl;
    infile.close();
}

int main(){
    test();
    return 0;
}
```

the5.h

```cpp
#ifndef THE5_THE5_H
#define THE5_THE5_H
#include <vector>
#include <memory>
#include <utility>
//this file will be overwritten during execution
//changes made in this file will not affectyour grade or execution
//it is shared with you so that you can work on a local environment if needed

void run(const std::vector<std::vector<int>>& dependencyMatrix,
         bool& isCompilable,
         std::vector<int>& compileOrder,
         std::vector<std::vector<int>>& cyclicDependencies);
#endif //THE5_THE5_H
```

VPL