

[CENG 315 All Sections] Algorithms


[Dashboard](#) / [My courses](#) / [571 - Computer Engineering](#) / [CENG 315 All Sections](#) / [January 5 - January 11](#) / [THE6](#)

Navigation

- ▾ [Dashboard](#)
- 🏠 [Site home](#)
- [Site pages](#)
- ▾ [My courses](#)
- ▾ [571 - Computer Engineering](#)
- [\[CENG 351 All Sections\]](#)
- [CENG 300 All Sections](#)
- [CENG 300 Section 4](#)
- ▾ [CENG 315 All Sections](#)
- [Participants](#)
- 🛡 [Badges](#)
- ✅ [Competencies](#)
- 📊 [Grades](#)
- [General](#)
- [October 13 - October 19](#)
- [October 20 - October 26](#)
- [October 27 - November 2](#)
- [November 3 - November 9](#)
- [November 10 - November 16](#)
- [November 17 - November 23](#)
- [November 24 - November 30](#)
- [December 1 - December 7](#)
- [December 8 - December 14](#)
- [December 15 - December 21](#)
- [December 22 - December 28](#)
- [December 29 - January 4](#)
- ▾ [January 5 - January 11](#)
- 📺 [Lecture - Shortest Paths & Maximum Flow](#)
- 📅 [Week 12 Annotations](#)
- 💬 [THE 6 Discussion Forum](#)
- ▾ [THE6](#)
- 📄 [Description](#)
- 📄 [Submission view](#)
- [January 12 - January 18](#)
- [January 19 - January 25](#)
- [CENG 315 Section 3](#)
- [CENG 331 All Sections](#)
- [CENG 331 Section 2](#)
- [CENG 351 Section 3](#)
- [651 - Music and Fine Arts](#)
- [612 - Modern Languages \(Persian\)](#)
- [642 - Turkish Language](#)

-  Description
-  Submission view

THE6

📅 **Available from:** Friday, January 7, 2022, 11:59 AM
🕒 **Due date:** Friday, January 7, 2022, 11:59 PM
📎 **Requested files:** the6.cpp  [Download](#)
Type of work: 👤 Individual work

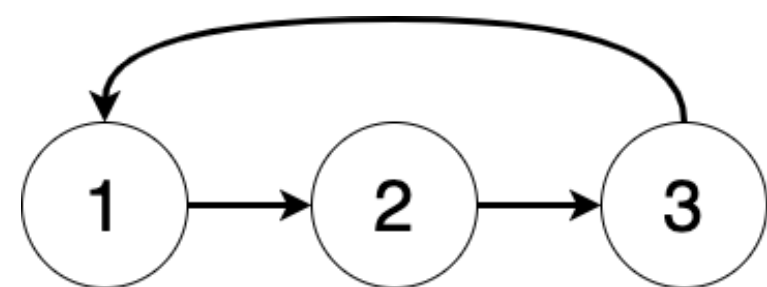
PROBLEM DEFINITION

In this exam, you will try to guess the result of a race you are listening on the radio. You know the number of racers participating in the race (N) but you do not know the initial ordering of the racers at the race start. Commentators are giving updates about the race like "Racer 3 passed Racer 8". However, your radio is broken and sometimes you lose the signal and miss some updates from the commentators. Commentators are not giving any contradictory updates, however, since you miss some updates they give, the updates you have heard may contain contradictory statements. For example, in a race between N racers, assume that first you hear the update "Racer 1 passed Racer 2", then after a while and possibly missing some updates you hear "Racer 2 passed Racer 1". During the time between the two updates racers 1 and 2 may have passed other racers and may have passed each other multiple times. Therefore, you should not discard any previous updates when you encounter a contradictory one.

Note that if you never miss an update and if you know the initial ordering of the racers you can deterministically find out the end result of the race, but since you miss some updates and the initial order is unknown there *may be more than one possible result* you can generate or it *may be impossible* to generate a result at all. A possible result is defined as a result which satisfies all the updates you have heard. For example consider a race with 3 racers, if you hear the updates "Racer 1 passed Racer 2" and "Racer 2 passed Racer 3", the race result 1,2,3 is a possible ordering as it satisfies both updates. Now assume that you heard a third update "Racer 3 passed Racer 1". In this case it becomes impossible to generate a possible result because however you order racers 1 and 3 in the result, the result only satisfies two of the three updates:

- 1, 2, 3 -> contradicts "Racer 3 passed Racer 1"
- 3, 1, 2 -> contradicts the "Racer 2 passed Racer 3"
- ...

Note that if the updates you have heard creates a loop, it becomes impossible to generate a result. In the above example, the loop can be visualized as



After hearing a bunch of these updates (and missing some) you want to generate a possible end result for the race from the updates you have heard or if it is not possible you want to show a loop that makes it impossible to generate a result.

Each racer has a unique integer id and each race has N racers. You can hear M updates and updates are given to you as a vector of integer pairs. For example $[(3, 5), (1, 2)]$, contains two updates saying "Racer 3 passed Racer 5" and "Racer 1 passed Racer 2". And you are expected write a function that returns a boolean value and a vector pair. Boolean value should be true if a race result can be generated and false otherwise. The vector should contain a possible race result if boolean value is true or the racers that form a loop if the boolean value is false.

You are expected to implement the below function:

```
std::pair<bool, std::vector<int>> RaceResult(int N, std::vector<std::pair<int, int>> updates);
```

Inputs:

- **N:** Number of racers participating in the race
- **updates:** List of updates you hear. Each update is a int pair $\{a, b\}$ expressing the update "Racer a passed Racer b ".

Outputs:

- Pair of a **bool** and a **vector**:
 - **bool** is true if there is a possible ordering, false if it is impossible
 - **vector** contains a possible result if possible (**bool** value is **true**), or a loop of racers that make it impossible to generate a result (**bool** value is **false**)

Example IO:

Input: $N=5$, **updates**=[[0, 2], {0, 1}, {1, 3}, {1, 2}, {2, 3}]
Return Value: {true, [0, 1, 2, 3, 4]}
Explanation: In the ordering 0 comes before 1 and 2 satisfying first two updates. 1 comes before 2 and 3 satisfying next two updates. 2 comes before 3 satisfying the last update. *There may be more than one correct answer, see examples below.*

Input: $N=5$, **updates**=[[0, 2], {0, 1}, {1, 3}, {1, 2}, {2, 3}, {3, 0}]
Return Value: {false, [0, 2, 3, 0]}
Explanation: In the given updates 0th, 4th, 5th updates create the loop 0->2->3->0. Therefore, an ordering is not possible. Notice that the first and the last element of the returned vector is the same. *There may be more than one correct answer, see examples below.*

Input: $N=10$, **updates**=[[0, 5], {1, 0}, {1, 7}, {2, 4}, {2, 0}, {3, 4}, {3, 5}, {3, 7}, {3, 6}, {4, 7}, {4, 1}, {6, 2}, {6, 5}, {6, 9}, {7, 9}, {7, 5}, {8, 3}]
Return Value: {true, [8, 3, 6, 2, 4, 1, 0, 7, 9, 5]} or {true, [8, 3, 6, 2, 4, 1, 0, 7, 5, 9]} or {true, [8, 3, 6, 2, 4, 1, 7, 0, 9, 5]} or {true, [8, 3, 6, 2, 4, 1, 7, 0, 5, 9]} or {true, [8, 3, 6, 2, 4, 1, 7, 9, 0, 5]}.
Explanation: Any one of these values is accepted. You only need to return a single answer.

Input: $N=10$, **updates**=[[0, 5], {1, 0}, {1, 7}, {2, 4}, {2, 0}, {3, 4}, {3, 5}, {3, 7}, {3, 6}, {4, 7}, {4, 1}, {6, 2}, {6, 5}, {6, 9}, {7, 9}, {7, 5}, {7, 3}, {8, 3}]
Return Value: {false, [3, 4, 7, 3]} or {false, [3, 6, 2, 4, 7, 3]} or {false, [3, 6, 2, 4, 1, 7, 3]} or {false, [3, 4, 1, 7, 3]}.
Explanation: Any of one of these values is accepted. You only need to return a single answer. Again, notice that the first and the last element of the returned vector is the same.

Specifications:

- There is 1 **task** to be solved in **12 hours** in this take home exam.
- You will implement your solutions in **the6.cpp** file.
- You are free to add other functions to **the6.cpp**
- Do **not** change the first line of **the6.cpp**, which is **#include "the6.h"**
- Some headers are defined in the **"the6.cpp"** for your convenience. Here is the full content of **"the6.h"** file:

```
#ifndef _THE_6_H_
#define _THE_6_H_

#include <limits>
#include <cmath>
#include <cstdio>
#include <iostream>
#include <queue>
#include <stack>
#include <unordered_map>
#include <utility>
#include <vector>

std::pair<bool, std::vector<int>> RaceResult(int N, std::vector<std::pair<int, int>> updates);

#endif
```

- Do **not** change the arguments and return value of the function **RaceResult** in the file **the6.cpp**
- Do **not** include any other library or write include anywhere in your **the6.cpp** file (not even in comments).
- Your code will be compiled with **-Wall** and **-std=c++11** flags.
- You can test your **the6.cpp** on virtual lab environment. If you click **run**, your function will be compiled and executed with a sample main function. If you click **evaluate**, you will get a feedback for your current work and your work will be **temporarily** graded for **limited** number of inputs.
- The grade you see in lab is **not** your final grade, your code will be reevaluated with **completely different** inputs after the exam.

The system has the following limits:

- a maximum execution time of 32 seconds (your functions should return in less than 1 seconds for the largest inputs)
- a 192 MB maximum memory limit
- an execution file size of 1M.
- Solutions with longer running times will not be graded.
- If you are sure that your solution works in the expected complexity constraints but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.

Constraints:

- There can be N racers in a race where $10 \leq N \leq 10000$.
- You can hear M updates where $N \leq M \leq N^2$.
- There may be more than one correct answer for each test instance. You only need to return **a single one** of the correct answers. **Hint:** Even if there are multiple correct answers, the possibility of finding a solution between the correct answers **do not** change (i.e., all correct answers have the same bool value in returned pair).

Evaluation:

- After your exam, black box evaluation will be carried out. You will get full points if you fill the return value as explained above.

Requested files

the6.cpp

```
1 #include "the6.h"
2 // Don't add any additional includes
3
4 /*
5  * N: Number of racers
6  * updates: Updates you have heard
7  */
8
9 std::pair<bool, std::vector<int>>
10 RaceResult(int N, std::vector<std::pair<int, int>> updates) {
11     return {true, {}}; // This is a dummy return value. YOU SHOULD CHANGE THIS!
12 }
13
```

VPL

◀ [THE 6 Discussion Forum](#)

Jump to...

[Lecture - String Matching & Huffman Code](#)