CS3230 Design and Analysis of Algorithms

# Programming Assignment 0
**(Ungraded)**

**Released** August 17, 2020 8:00 AM

**Due** August 31, 2020 11:59 PM

## 1 What?

Over the course of this semester, you will encounter and learn a good number of algorithms, data structures and problem-solving paradigms.

To ensure that you are well-equipped with not only the theoretical knowledge of these concepts, but also the practical skills necessary to solve real problems, a number of programming assignments will be given.

There will be **two (2)** programming assignments (not including this one, which is ungraded) over the course of the semester, each carrying a weight of **five percent (5%)** of the total marks allocated for this module.

**Programming Assignment 1** will released on **August 31, 8:00 AM** and will be due on **October 5, 11:59 PM.**

**Programming Assignment 2** will released on **October 5, 8:00 AM** and will be due on **November 13, 11:59 PM.**

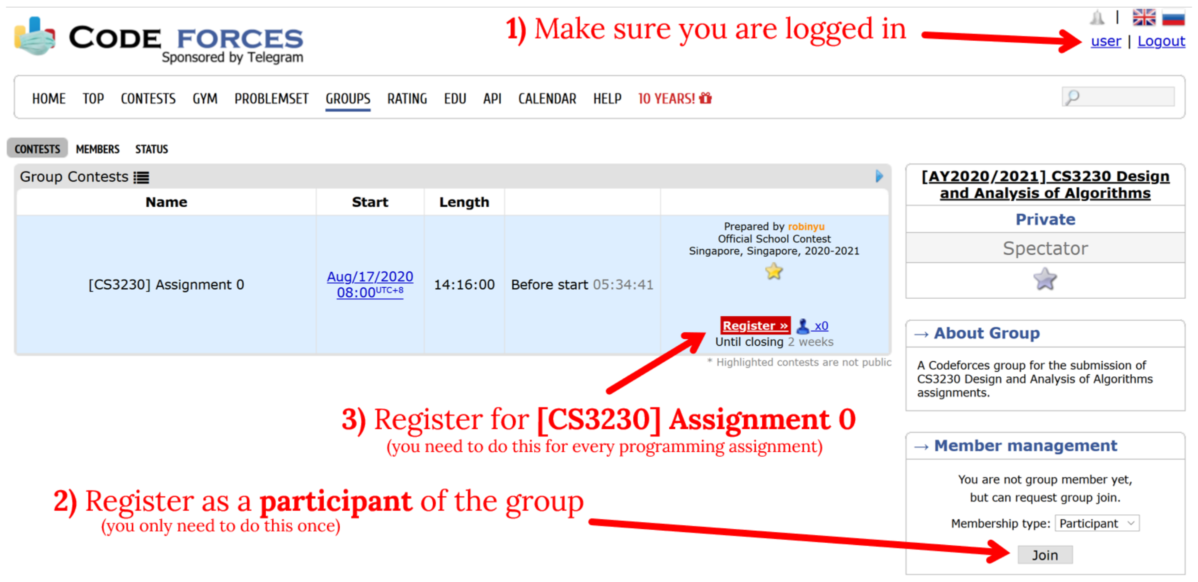
Each of these programming assignments will consist of one or more algorithmic problems, and your task is to write *correct* and *efficient* code to solve these problems.

## 2 Instructions

First, register an account on Codeforces. You may, but do not need to, use your NUS e-mail when registering, and you may use an existing account if you already have one. You can also use any available handle (username). This account will be used for all assignments, so please remember your credentials or store them in your favorite password manager.

Second, fill out the quiz "Codeforces Handle" on LumiNUS with your Codeforces handle. This will allow us to link your Codeforces account to you.

Third, while logged in, register as a participant in this Codeforces group. (Link: `https://codeforces.com/group/b0nFDNKNrz`) This will give you access to the submission portals.



1) Make sure you are logged in

Group Contests

| Name | Start | Length | | |
|---|---|---|---|---|
| [CS3230] Assignment 0 | Aug/17/2020 08:00UTC+8 | 14:16:00 | Before start 05:34:41 | Prepared by robinyu<br>Official School Contest<br>Singapore, Singapore, 2020-2021<br>Register » x0<br>Until closing 2 weeks |

* Highlighted contests are not public

[AY2020/2021] CS3230 Design and Analysis of Algorithms

Private

Spectator

→ About Group

A Codeforces group for the submission of CS3230 Design and Analysis of Algorithms assignments.

→ Member management

You are not group member yet, but can request group join.

Membership type: Participant

3) Register for **[CS3230] Assignment 0**
(you need to do this for every programming assignment)

2) Register as a **participant** of the group
(you only need to do this once)

Once each assignment opens, you will be able to register for it and submit your solutions for the problems there.

You may submit as many times as you want for any problem before the deadline passes. The grading system is automated and you will immediately know your result; we will take the best result out of all your submissions.

You may submit in any programming language supported by Codeforces, i.e. just about every major programming language, including but not limited to C++, Java, Python 2 and 3, Perl, Ruby, Haskell, Go and Kotlin; see Codeforces for the full list.[1]

**Unfortunately, because the grading system is automated and some languages are inherently slower than others, we do not guarantee that it is possible to score full marks using every programming language, even with the optimal algorithm.** Our model solutions will use C++, but reasonable leeway will be given to account for slower languages.

[1] Unfortunately, Codeforces does not support esoteric languages, so Malbolge, GolfScript, FiM++ etc. are out of the window. You are free to try them for fun anyway!

Additionally, in the interest of fairness and practicality, no marks will be given to wrong solutions, no matter how "close" to a correct solution they are (e.g. off-by-one errors, silly bugs, "just" a few milliseconds too slow, etc.) You are highly encouraged to attempt the problems early and seek guidance when you get stuck.

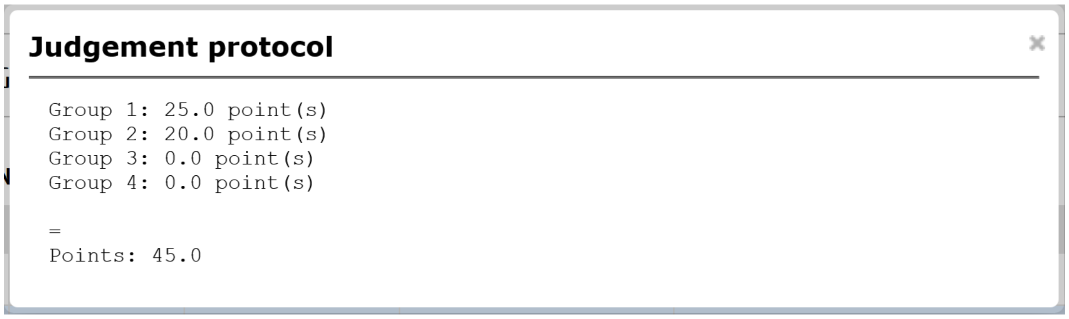
## 2.1 How Grading Works

Your code must read from *standard input* and print to *standard output;* the expected input and output formats will be given in the problem. **Do not print anything not explicitly mentioned in the output format.** This includes messages such as `please input a and b` or `the answer is:`. The judging system is fairly primitive and will not be able to distinguish between these messages and your actual answer, so these will be marked incorrect.

Your code will be run against a number of *secret test cases.* Each secret test case will satisfy the provided constraints and will be in the required format. If your code produces the correct output within the given time and memory limits, your code is said to have *passed* the test case.

There may be multiple groups of secret test cases with varying constraints (for example, some groups may have smaller limits to allow for suboptimal solutions to score partial marks). These groups and corresponding constraints will be indicated in the problem. Your code needs to pass all the test cases in a given group to receive the marks allocated to that group; your total marks will be the sum of all the marks from each group.

You can view which groups your code passed (but not which specific test cases your code passed or failed) by clicking on the verdict on the submissions page.



```
Group 1: 25.0 point(s)
Group 2: 20.0 point(s)
Group 3: 0.0 point(s)
Group 4: 0.0 point(s)

=
Points: 45.0
```

The message `0.0 point(s)` indicates that your code failed at least one test case in that group; otherwise, all test cases were passed and you receive the corresponding marks for that group. No other information is given about the failed test cases, so you will need to think carefully about where your code might be wrong or insufficiently efficient.

Some problems may have different grading schemes. These will be clearly indicated.

To familiarize yourself with the system, you are encouraged to try the two ungraded example problems given in this programming assignment. The same problems are reproduced on the Codeforces submission portal.

# 3 Problems

## A + B (70 marks)

**Time limit: 1 second**
**Memory limit: 256 MB**

You are given two integers $a$ and $b$. Find their sum.

### Input

The first and only line of input contains two integers, $a$ and $b$.

### Output

Output a single integer on a line by itself, the sum $a + b$.

### Scoring

For all groups, $1 \leq a, b$.

| **Group** | **Marks** | $a$ | $b$ |
|---|---|---|---|
| **1** | 25 | $a \leq 10^9$ | $b \leq 10^9$ |
| **2** | 20 | $a \leq 10^{18}$ | $b \leq 10^{18}$ |
| **3** | 10 | $a \leq 10^{200000}$ | $b = 1$ |
| **4** | 15 | $a \leq 10^{200000}$ | $b \leq 10^{200000}$ |

### Example

#### Input

```
351 69
```

#### Output

```
420
```

## **Shortest Way Home** (30 marks)

**Time limit: 2 seconds**
**Memory limit: 256 MB**

Pete is the mayor of a small town.

In his town, there are $n$ junctions, labeled from $1$ to $n$, and $m$ bidirectional roads. Each road connects two distinct junctions, with the $i^{\text{th}}$ road connecting the junctions labeled $a_i$ and $b_i$ and being $d_i$ meters long.

Every day, Pete goes to work. His home is located at the junction labeled $1$, and his office is located at the junction labeled $n$. He always takes the shortest path to work.

Recently, he has felt that the commute takes too much time, and decided to request the construction of *one* new road. He should choose two distinct junctions $a'$ and $b'$ and build a bidirectional road between them which is some positive integer $d'$ meters long, such that the length of the shortest path to work becomes *strictly* shorter than what it is currently.

He does not want to build a road between a pair of junctions that already have a road between them, as that would be seen as wasteful (even if the new road might be shorter).

How many ways are there to build this new road? Two ways are different if the unordered pair of junctions $\{a', b'\}$ the road connects are different or the length of the road $d'$ is different.

Since this number can be quite large, output only the remainder after dividing it by $10^9 + 7$.

### **Input**

The first line of input contains two integers $n$ and $m$, the number of junctions and the number of existing bidirectional roads, respectively.

The next $m$ lines of input each contain three integers. In particular, the $i^{\text{th}}$ of these lines contains $a_i$, $b_i$ and $d_i$, the junctions this road connects and the length of this road in meters, respectively.

### **Output**

Output a single integer on a line by itself, the number of ways to build the new road.

Since this number can be quite large, output only the remainder after dividing it by $10^9 + 7$.

### **Scoring**

For all groups,

- $2 \le n$;

- $1 \leq m \leq \min\left(400\,000, \frac{n(n-1)}{2}\right)$;
- $1 \leq a_i, b_i \leq n$;
- $a_i \neq b_i$;
- $1 \leq d_i \leq 10^9$;
- There is at most one road between any two junctions;
- It is possible to travel from Pete's home to his office using the existing roads.

| **Group** | **Marks** | $n$ |
|---|---|---|
| **1** | 8 | $n \leq 50$ |
| **2** | 6 | $n \leq 2\,000$ |
| **3** | 16 | $n \leq 200\,000$ |

**Example**

**Input**

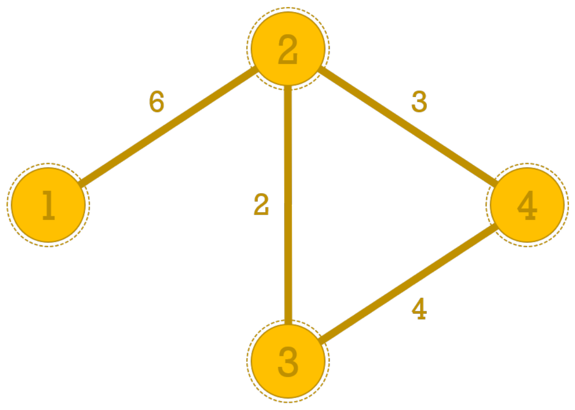
```
4 4
1 2 6
2 3 2
2 4 3
3 4 4
```

**Output**

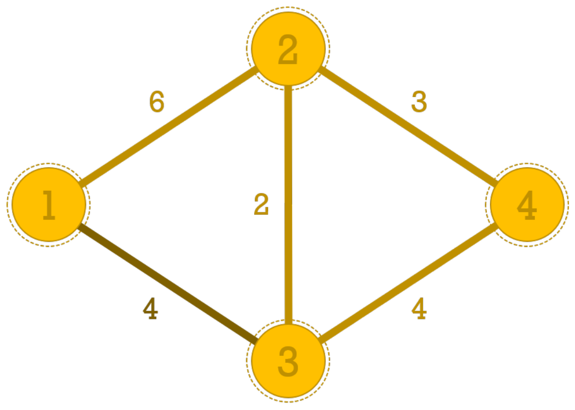
```
12
```

**Note**

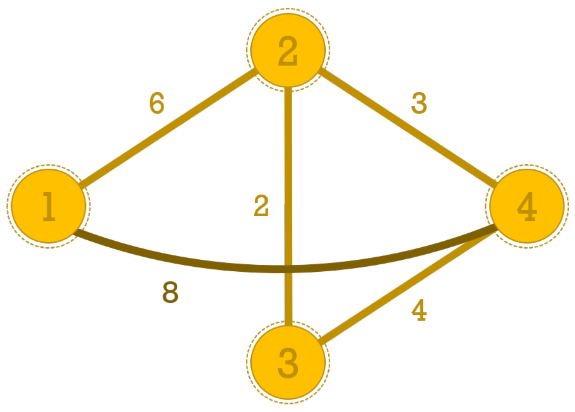
The following image illustrates the example. The current shortest path is $1 \to 2 \to 4$ with a total length of $9$:



We can add a road between junctions $1$ and $3$ with any positive integer length up to $4$ and get a strictly shorter path $1 \to 3 \to 4$:



Alternatively, we can add a road directly between junctions $1$ and $4$ with any positive integer length up to $8$ and get a strictly shorter path $1 \to 4$:



Note that these are the only pairs of junctions not already connected by roads, so these are the only new roads we need to consider. Therefore, the total number of ways is $12$.

## Warning

You may discuss the *high-level conceptual ideas* of the assignments with other students taking this module this semester, or through the LumiNUS forums, but **all the code you submit must be your own.**

**Do not:**

1. Post or discuss the assignments on Stack Overflow, Theoretical Computer Science Stack Exchange, the comments on Terence Tao's blog or any other website accessible publicly, or with anyone except the professor, teaching assistants and students taking this module this semester;
2. Share code snippets relating to the assignments with anyone;
3. Publish your code on a public GitHub repository or use online compilers like Ideone which make your code public during the course of the semester;[2]
4. Copy code from online sources or your friends;
5. Engage in pair / triple / $n$-tuple, $n \geq 2$ programming;
6. Perform any other action which violates the spirit of these rules.

When in doubt, ask us first.

We will conduct extensive checks for evidence of violations of these rules. If the preponderance of the evidence suggests that the rules have been violated, sanctions may be levied, including but not limited to: a zero for the assignment, failing the class, or escalation to the Office of Student Affairs for further disciplinary action.

If we find considerable similarities between the submissions of multiple students, **all affected students** are liable to receive sanctions. This includes not only the one(s) doing the copying but also the one(s) who allowed his/her/their code to be copied, so exercise due diligence to keep your code private.

Please, just do not cheat. If you cheat, we will have to do additional paperwork, and we do not really want to do that.

[2] If you like, you may publish your code after the CS3230 final has concluded.