CS 170 Algorithms
Spring 2014 Elchanan Mossel

Final Contest

Problem Description

Given a set of E linear equations mod P over V variables, find an assignment to the variables such that the maximum number of equations are satisfied (it may be impossible to satisfy them all). Each equation will involve exactly two different variables. P is assumed to be prime.

Submission instructions

The contest website will be open by Wednesday.

Phase 1 (Input files and coding): May 8th

We'll open this by Wednesday, May 7th.For this phase, generate a single input file with $2 \leq V \leq 10^3$ and $1 \leq E \leq 10^5$ and P less than 9000. Also come up with E equations in the format specified under input format (next page). Make sure to follow the input format exactly, or your file will be discarded. Don't forget to start coding during this phase. You will turn in a single text file for this phase.

Phase 2 (Output files + Source code) May 9th

We will release the input files generated by each group that are valid. After we release all the input files, you will have about 24 hours to download the files, run your code, and submit the output file and source code. Make sure you only have a single output file (see output format specification for more details). The input files will be named "i.in", where i is the number of the instance. Make sure that the ith line of your output corresponds to this input file. You will also need to create a README file that includes all of your team members, and briefly explains what your code does and what you've tried.

Input format

There will be multiple files. This format describes the contents of one file.
Line 1: Two space separated integers V, E, P representing the number of variables, the number of equations, and your prime. Your variables will be $x_0, x_1, \ldots, x_{V-1}$.
Line 2...$E+1$: 5 space-separated integers a, b, c, d, e, with $1 \leq a, c \leq P-1$, $0 \leq e \leq P-1$, and $0 \leq b, d \leq V-1$ with $b \neq d$. This line represents the equation $a \cdot x_b + c \cdot x_d + e \equiv 0 \pmod{P}$.
You may assume all input files will follow this format this exactly, so there's no need to check these conditions explicitly

Example Input

```
4 4 3
1 0 2 3 2
2 3 1 2 1
1 1 1 0 1
2 2 2 3 0
```

We have four variables x_0, x_1, x_2, x_3 and four equations and $P = 3$. The input corresponds to the equations:

$$\begin{aligned} x_0 + 2x_3 + 2 &\equiv 0 \pmod{3} \\ 2x_3 + x_2 + 1 &\equiv 0 \pmod{3} \\ x_1 + x_0 + 1 &\equiv 0 \pmod{3} \\ 2x_2 + 2x_3 + 0 &\equiv 0 \pmod{3} \end{aligned}$$

Output format

You will submit **one** output file. Each line represents the solution for an individual input file.
Line i: This line corresponds to the answer to the ith input file. You should have V space-separated integers corresponding to the assignments to the variables $x_0, \ldots, x_{V-1}$. These integers should be in $\{0, 1, \ldots, P-1\}$.

Example Output

(Note that this file would have more lines if we had more input files)

```
0 2 1 2
```

This corresponds to setting $x_0 = 0, x_1 = 2, x_2 = 1, x_3 = 2$. This would satisfy all four equations, which is the best we could possibly get, and your raw score for this test case is 4. If you do not wish to solve an input file for any reason (i.e. not enough time), leave the line blank.

Scoring details

We will score only your output files automatically. We will not automatically run your source code, but we may have a look at it.

Each file will have a "raw" score, which is the number of equations your solution satisfies. If your output line is not in the right format, you will receive a raw score of zero for that instance. Suppose that several groups have submitted solutions for the ith input file. Then, let M be the maximum raw score of any one groups. Let m be the your raw score. Your normalized score for that file will be m/M. If $m = M$, your "best" score for that file will be 1 and 0 otherwise. If $M = 0$, then everybody will get a zero for both normalized and best score. We will first break ties by the sum of your best scores across all files, then by the sum of your normalized scores across all files. If there are still ties, we will break ties based on the first input file, then the second, and so on. We hope we do not have to break ties any further.

Grading Scheme

Let K be the highest total normalized score. Let the sum of your normalized score be k. Then, you will get $2k/K$ points for this assignment (capped at 1). These points will be added to your course grade (this is essentially worth at most 1% extra for your course grade).

Prizes

Each member of the winning team will receive a signed copy of the textbook.

Miscellaneous Rules and Reminders

- You may work in teams of at most 5.
- You may use any language.
- You may use any publicly available libraries, but please make sure to cite your sources if you decide to copy something.
- If your input file does not conform to the specifications, you will not get any credit for it. Make sure you carefully read the next sections for more details
- Similarly, if your output does not conform to the specifications, you will not get credit for that instance.
- Starter code for reading and writing to files has been included at the end of this pdf

Starter code

Here's some code in popular languages to get you started if you need help to read in and read out to files (only Python, Java, and C/C++). Remember, you can do this in any language you like.

Python

```
T = 20 # number of test cases
fout = open ("answer.out", "w")
for t in xrange(1, T+1):
    fin = open(str(t) + ".in", "r")
    V, E, P = [int(x) for x in fin.readline().split()]
    for i in xrange(E):
        a,b,c,d,e = [int(x) for x in fin.readline().split()]
        # process input
    assign = [0] * V
    # find an assignment
    fout.write("%s\n" % " ".join(map(str, assign)))
fout.close()
```

Java

```
// useful imports
import java.io.*;
import java.util.*;
public class Main {
    public static void main (String[] args) throws IOException {
        int T = 20; // number of test cases
        PrintWriter out = new PrintWriter (new FileWriter (new File ("answer.out")));
        for (int t = 1; t <= T; t++) {
            Scanner in = new Scanner (new File (t + ".in"));
            int V = in.nextInt(), E = in.nextInt(), P = in.nextInt();
            for (int i = 0; i < E; i++) {
                    int a = in.nextInt(), b = in.nextInt(), c = in.nextInt(),
                        d = in.nextInt(), e = in.nextInt();
                    // process input
            }
            int[] assign = new int[V];
            // find an assignment
            out.print(assign[0]);
            for (int i = 1; i < V; i++)
                out.print (" " + assign[i]);
           out.println();
        }
       out.close();
    }
}
```

C/C++

```
// useful imports
#include <stdio.h>
#include <string.h>
#include <algorithm>
int main() {
    int T = 20; // number of test cases
    FILE * fout = fopen ("answer.out", "w");
    for (int t = 1; t <= T; t++) {
        char filename[10];
        sprintf (filename, "%d.in", t);
        FILE * fin = fopen(filename, "r");

        int V, E, P;
        fscanf(fin, "%d%d%d", &V, &E, &P);
        for (int i = 0; i < E; i++) {
            int a, b, c, d, e;
            fscanf(fin, "%d%d%d%d%d", &a, &b, &c, &d, &e);
            // process input
        }
        int assign[V];
        memset (assign, 0, sizeof(assign));
        // find an assignment

        fprintf(fout, "%d", assign[0]);
        for (int i = 1; i < V ;i++)
            fprintf(fout, " %d", assign[i]);
        fprintf(fout, "\n");
    }
    fclose(fout);
    return 0;
}
```

Miscellaneous

The first 1000 prime numbers can be found here `http://primes.utm.edu/lists/small/1000.txt`