GB20602 - Programming Challenges

Week 0 - Course Details

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Sciences

(last updated: April 13, 2024)

Version 2024

Course Introduction

What is the goal of this course?

- Course Objective: Improve your skill by writing many programs;
- How? Write a program that solves a puzzle (programming challenge)
  1. Read and understand the puzzle;
  2. Think what program solves the puzzle;
  3. Write the program;
  4. Test the result of the program;
- Why?
  1. We use in practice the algorithms learned in the second year.
  2. We learn how the algorithm works in practice.
  3. We learn how to debug from data.

Automated Judge (AJ)

This course uses an Automated Judge (AJ) to check your homework.

- An AJ is a program that checks if code is correct:
  - Ex: AtCoder, Aizu Online Judge, Topcoder, Codeforces, etc.
- The AJ checks your program agains Secret Test Data.
- The AJ tells you if your code is correct or incorrect.

When your program is incorrect: Be Careful!

- Debug your program carefully;
- Create your own test data;
- Think before submitting again;

Class Requirements

- You have basic programming knowledge (C++, Java or Python);
  - This is a hard programming class
  - Data structures and algorithms are important
- You will do homework every week;
  - Minimum 2 programs per week;
  - Debug carefully;
- You will not copy the homework;
  - Copy homework from other students will be penalized;
  - Copy homework from the internet will be penalized;
  - Penalties include failing the course, and failing the semester;
- No final exam – only homework

Hint: Do your homework early!

Homework: What are Programming Challenges?

It is a puzzle that you solve by writing a program.

The program reads the input, and must write the correct output.

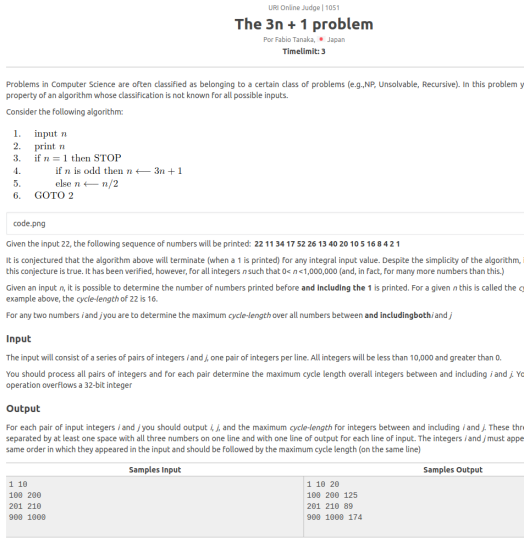
- Difficulty 1: The output must be exactly correct.
- Difficulty 2: The test input is not known.
- Difficulty 3: There is a maximum execution time.

In general, a correct program will be small (< 200 lines).

Programming Challenge Example: "The 3n+1 problem"

Contents of a Programming Challenge:

- Problem Description;
- Input Description;
- Output Description;
- Input/Output Example;



URI Online Judge | 1051

The 3n + 1 problem

Por Fabio Tanaka, Japan

Timelimit: 3

Problems in Computer Science are often classified as belonging to a certain class of problems (e.g.,NP, Unsolvable, Recursive). In this problem y property of an algorithm whose classification is not known for all possible inputs.

Consider the following algorithm:

1. input n
2. print n
3. if $n = 1$ then STOP
4. if n is odd then $n \longleftarrow 3n + 1$
5. else $n \longleftarrow n/2$
6. GOTO 2

code.png

Given the input 22, the following sequence of numbers will be printed: **22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1**

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, this conjecture is true. It has been verified, however, for all integers *n* such that 0< *n* <1,000,000 (and, in fact, for many more numbers than this.)

Given an input *n*, it is possible to determine the number of numbers printed before **and including the 1** is printed. For a given *n* this is called the c example above, the *cycle-length* of 22 is 16.

For any two numbers *i* and *j* you are to determine the maximum *cycle-length* over all numbers between **and includingboth** *i* and *j*

Input

The input will consist of a series of pairs of integers *i* and *j*, one pair of integers per line. All integers will be less than 10,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length overall integers between and including *i* and *j*. Yo operation overflows a 32-bit integer

Output

For each pair of input integers *i* and *j* you should output *i*, *j*, and the maximum *cycle-length* for integers between and including *i* and *j*. These thr separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers *i* and *j* must appe same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line)

Samples Input	Samples Output
1 10 100 200 201 210 900 1000	1 10 20 100 200 125 201 210 89 900 1000 174

Example: The 3n+1 problem

What is the problem?

What is the longest sequence generated by this function?

1. input n
2. if $n = 1$ then STOP
3. if n is odd, then $n = 3n + 1$
4. else $n = n/2$
5. GOTO 2

For example, if $i = 1$ and $j = 4$:

- n = 1: 1 END; Length 1
- n = 2: 2 1 END; Length 2
- n = 3: 3 10 5 16 8 4 2 1 END; Length 8
- n = 4: 4 2 1 END; Length 3

So between n=1 and n=4, the maximum is 8 (n = 3)

URI Online Judge | 1051

The 3n + 1 problem

Por Fabio Tanaka, Japan

Timelimit: 3

Problems in Computer Science are often classified as belonging to a certain class of problems (e.g.,NP, Unsolvable, Recursive). In this problem y property of an algorithm whose classification is not known for all possible inputs.

Consider the following algorithm:

```
1. input n
2. print n
3. if n = 1 then STOP
4.       if n is odd then n ⟵ 3n + 1
5.       else n ⟵ n/2
6. GOTO 2
```

code.png

Given the input 22, the following sequence of numbers will be printed: **22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1**

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, this conjecture is true. It has been verified, however, for all integers *n* such that 0< *n* <1,000,000 (and, in fact, for many more numbers than this.)

Given an input *n*, it is possible to determine the number of numbers printed before **and including the 1** is printed. For a given *n* this is called the c example above, the *cycle-length* of 22 is 16.

For any two numbers *i* and *j* you are to determine the maximum *cycle-length* over all numbers between **and includingboth** *i* and *j*

Input

The input will consist of a series of pairs of integers *i* and *j*, one pair of integers per line. All integers will be less than 10,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length overall integers between and including *i* and *j*. Yo operation overflows a 32-bit integer

Output

For each pair of input integers *i* and *j* you should output *i*, *j*, and the maximum *cycle-length* for integers between and including *i* and *j*. These thr separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers *i* and *j* must appe same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line)

Samples Input	Samples Output
1 10 100 200 201 210	1 10 20 100 200 125 201 210 89

Example: The 3n+1 problem

A simple program

```
int main() {
  int min, max;
  int maxcycle = 0;
  cin >> min >> max;                              // Read i and j
  for (int a = min; a <= max; i++) {              // Loop from i to j
    int cycle = 1;
    int n = a;
    while (n != 1) {
      if (n % 2 == 0) { n = n / 2; }              // calculate n
      else { n = n*3 + 1; }
      cycle++;                                    // increase cycle size
    }
    if (cycle > maxcycle) maxcycle = cycle;       // keep max cycle
  }
  cout << min << " " << max << " " << maxcycle << "\n";
  return 0;
}
```

Example: The 3n+1 problem

Simple programs, simple problems

If you try to run this program with large inputs, it will be very slow! Why?

Consider the inputs: i = 1, j = 10:

- n = 1: 1 END
- n = 2: 2 1 END ...
- n = 7: 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 END

  ...
- n = 9: 9 28 14 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1 END
- n = 10: 10 5 16 8 4 2 1 END

There is a lot of repetition! How to make the program faster?

Example: The 3n+1 problem

Memoization

We use Memoization to make the program faster

Memoization:

- Every time you finish a calculation, store the result in the memory;
- Before you begin a calculation, check if the result is not in the memory;

This technique can reduce the amount of repeated work.

In this course we will review and study many techniques like this one.
You will have to implement these techniques in the homework to make it efficient.

Topics in this course

1. Introduction
2. Data Structures
3. Search Problems
4. Dynamic Programming
5. Graphs Problems (Graph Structure)
6. Graph Problems (Graph Search and Flow)
7. String Manipulation
8. Math Problems
9. Geometry Problems
10. Final Remix

About the Lecturer

- Name: Claus Aranha;
- Country: Brazil;
- Research Topics:
  - Evolutionary Computation;
  - Artificial Life;
- Hobbies:
  - Game Programming;
  - Geocaching;
- webpage:
  `http://conclave.cs.tsukuba.ac.jp`

Why programming challenges?

- For Competitions: Competitive programming started around 1980 in the US. Today, many universities in the world compete, including Tsukuba.
- For Study: Recently, many people use Automated Judges to study and improve programming ability (AtCoder, Codeforces, etc);
- For Recruitment: Also, many companies today use programming challenges in their recruitment.
- For Fun: It is fun to solve puzzles and to program!

Extra: Join the Tsukuba ICPC Team!

What is ICPC?

If you like these contests, and want an extra challenge, please consider joining the Tsukuba ICPC team!

ICPC (International Collegiate Programming Contest) is the largest and most traditional programming competition between universities.

More than 50.000 students from all over the world participate in this competition every year.

Extra: Join the Tsukuba ICPC Team!

Program and see the world!

- Requirements: Team of 3 students, any course;
- Schedule:
  - First Online Contest in July
  - Japanese National Contest in November
  - World Final Next Year (June?)
- Contact me if you're interested!

Course Organization

Outline

What you will do every week

Monday – Get materials from manaba. Check the problems;

Tuesday – 3C205/3C206: Solve the problems, Ask questions to the teacher;

Entire Week – Submit the problems on kattis;

Next Tue – Homework Deadline

Course Dates and Deadlines

Course Dates

- 4/16, 4/23, 4/30, 5/14, 5/21, 5/28, 6/4, 6/11, 6/18, 6/25;
- No final exam;

Deadlines

- Deadline for homework: Every Tuesday, 23:00

Where to find the material?

manaba

- Official place for lecture material and videos;
- Use Forum for questions;
- Please read announcements; Please answer surveys;

github

- Lecture materials is also available on github:
- URL: `https://caranha.github.io/Programming-Challenges/`
- Not-official. Includes material from last year.
- manaba is the official version

Websites to submit homework

KATTIS online judge

- https://judge.conclave.cs.tsukuba.ac.jp/
- Create an account using your university e-mail (sXXXXXXX@u.tsukuba.ac.jp)
- We need to approve accounts, so create your account quickly!
- See Manaba for more information.

About Course Language

Natural Language

- Materials and Homework: English
- Video and manaba: Japanese
- E-mail, feedback: English/Japanese;
- If you want to help me translate the homework, contact me!

Programming Language

- The Judge accepts: C, C++, Java, Python, Ruby;
- The teacher helps with: C, C++, Java, Python;
- If you want to use another language, contact me;

Reference Books

Textbook:

- textbook: Steven Halim, Felix Halim,"Competitive Programming", 4th edition. `https://cpbook.net/`

Other books:

- Steven S. Skiena, Miguel A. Revilla,"Programming Challenges", Springer, 2003
- 秋葉拓哉、 岩田陽一、 北川宜稔,『プログラミングコンテストチャレンジブック』
- 渡部有隆、『オンラインチャレンジではじめるC/C++プログラミング入門、Online Programming Challenge!』 (ISBN978-4-8399-5110-8)
- 渡部有隆、『プログラミングコンテスト攻略のためのアルゴリズムとデータ構造』（ISBN978-4-8399-5295-2）

Grading Rules

Base Grade

Your base grade is based on the number of accepted homework programs you submit:

- C grade: 2+ accepted problems every week;
- B grade: 3+ accepted problems every week;
- A grade: 5+ accepted problems every week;
- A+ grade: 7+ accepted problems every week;

Important!!

- Every week means "X problems every week" (not “average”);
- You can submit problems late, with a penalty;

Grading Rules

Late Penalty

You can submit problems late. But there is a penalty.

If the number of total late programs $\geq$ 25% of total programs, your grade will lower 1 step.

You will not fail the course for late programs.

Example:

- Student (1) submitted 45 problems, minimum 5 problems per week. 5 problems are late. $5 \leq 45 * 0.25$, no penalty. Grade A.
- Student (2) submitted 46 problems, minimum 5 problems per week. 16 problems are late. $16 \geq 46 * 0.25$, penalty. Grade B.
- Student (3) submitted 24 problems, minimum 2 problems per week. 10 problems are late. $10 \geq 24 * 0.25$, penalty. Grade C (will not fail).

Grading

Plagiarism

The assignments are individual. You must write your programs by yourself.

You can do this

- Ask for ideas to your friends;
- Ask for ideas in the MANABA forum;
- Ask for help with a bug;

You can NOT do this

- Copy a solution from the internet;
- Copy a solution from your friends;
- Give your code to a friend;

Students who do plagiarism will fail the course, and suffer penalties from the university.

About these Slides

Individual images in some slides might have been made by other authors. Please see the following pages for details.

Image Credits I