GB20602 - Programming Challenges

Week 0 - Course Details

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Sciences

(last updated: April 15, 2022)

Version 2022

What is the objective of this course

Outline

Class Objective

- Course Objective: Improve your algorithm skill by writing many programs;
- Key Idea: Write programs that solve challenges (puzzles)
  1. Read the problem and understand the puzzle;
  2. Choose the algorithm and implement;
  3. Run the program and check the correct result;
  4. Repeat many times
- In this lecture, we want to use in practice the algorithms that we learned in the 1st and 2nd year.

Algorithms: Theory vs. Implementation

Why is this a good idea? When we apply a textbook algorithm to a puzzle, we learn many new things:

- Input/Output: How to transform the input data to a format that the algorithm understands;
- Input Size: If the input is very large, a wrong program or algorithm can be very slow;
- Special Cases: How to think about special cases that can cause bugs or errors in the program;
- Debugging: How to find the source of bugs in your program, specially if the bug is because of a **special case?**;

Automated Judging

In this lecture we use **Automated Judges (AJ)** to check if your homework is correct.

- An AJ is a website that proposes programming puzzles, and receive solutions;
    - Ex: AtCoder, Aizu Online Judge, Topcoder, Codeforces, etc.
- The AJ receives your program; compile it; and grade it;
- The AJ test your program with a set of Hidden Inputs
- The AJ gives you the result: Correct or Incorrect
    - If the result is incorrect, it will not tell you why;
    - You have to create debug cases by yourself;
    - But, you can submit a new version as many times as you want;

What this class expects of you

Estimated classwork:

- You need basic programming knowledge (in C, C++, Java or Python);
- Complete 2 or more program assignments per week;
  - Average of 4 hours of study per week;
  - A lot of time with debug and creating test cases;
  - (Atcoder difficulty: 300 to 500)
- Homework starts easy, but becomes harder later in the course;
- No final exam: only assignments!

Hint: Do your homework early!

What kind of assignment?

A "Programming Challenge" is a puzzle problem that needs to be solved by writing a program.

The program **reads the input**, and must write the **correct output**, following the rule of the puzzle **exactly**.

You don't know all the input. You must make sure that your program can solve **Any acceptable input**.

There is a **running time limit** so your program must be efficient.

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

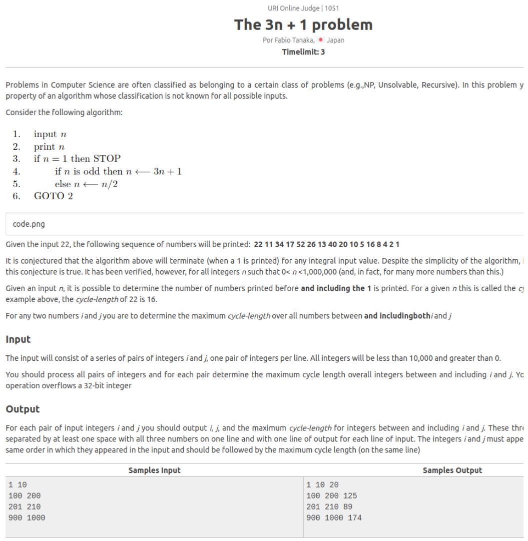
Why programming challenges?

- **For Competitions**: Around 1980, Programming Contests started as a way for Computer Sciences universities to compete. (IOI, ICPC, etc)
- **For Study**: After 2000, many people use online Automated Judges to study programming and prove their ability; (TopCoder, Codeforces, AtCoder, etc);
- **For Recruitment**: Recently, many companies use Programming Challenges to test the programming ability of candidates. candidates. (Google, Facebook, etc)
- **For Fun**: Some people just like to have an excuse to program!

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, i
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	1 10 20
100 200	100 200 125
201 210	201 210 89
900 1000	900 1000 174

Example: The 3n+1 problem

What is the problem?

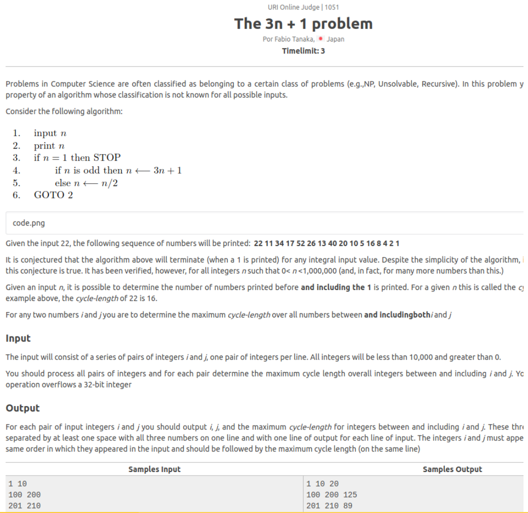
The problem wants the longest sequence size generated by the following algorithm:

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

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 the **maximum length** is 8 (for 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 $n \longleftarrow 3n + 1$
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, i
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;
  for (int i = min; i <= max; i++) {
    int cycle = 1;
    int n = i;
    while (n != 1) {
      if (n % 2 == 0) { n = n / 2; }
      else { n = n*3 + 1; }
      cycle++;
    }
    if (cycle > maxcycle) maxcycle = 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?

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

A lot of work is repeated without necessity. How to make the program faster?

Example: The 3n+1 problem

Memoization

A technique called **Memoization** can solve the "repeated work" problem.

Memoization:

- Every time you finish a calculation, store the result in the memory;
- Before you begin a calculation, check if it 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`

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:
  - National Preliminary Competition in July
  - Japanese Regional Competition in October
  - Asian Semi-final in December
  - World Final April next year

(Dates may change this year because of nCov-19)

How the course is organized

Outline

What you will do every week

(manaba)	Get the week PDF and study the lecture;
(manaba)	Watch the lecture video;
(kattis)	Check the Programming Homework Exercises;
(TEAMS/3C205)	Ask questions to the professor / Write your programs;
(kattis)	Submit your programs at the URI page;
(manaba)	Write the attendance survey;

Course Dates and Deadlines

Course Dates

- 4/19, 4/26, **5/6**, 5/10, 5/17, 5/31, 6/7, 6/14, 6/21, 6/28;
- No final exam;
- Office hours on Tue 12:15 to 15:00 (3C205 and TEAMS);

Deadlines

- The deadline for homework: **1 Week, 23:00**
- The deadline for late homework: 7/11
- Final Grades will be published around 7/15

Dates subject to changes.

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://tsukuba.kattis.com`
- Please create an account here.
- See the kattis video 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:

- **A+ grade:** 6+ accepted problems every week;
- **A grade:** 4+ accepted problems every week;
- **B grade:** 3+ accepted problems every week;
- **C grade:** 2+ accepted problems every week;

Important!!

- Every week means "every week";
- Every week **does not mean** "average";
- You can submit problems late, but there is 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 40 problems, minimum 4 problems per exercise. 5 problems are late. $5 \leq 40 * 0.25$, no penalty. Grade A.
- Student (2) submitted 44 problems, minimum 4 problems per exercise. 16 problems are late. $16 \geq 44 * 0.25$, **penalty**. Grade B.
- Student (3) submitted 24 problems, minimum 2 problems per exercise. 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

Image Credits I