GB21802 - Programming Challenges

Week 0 - Introduction

Claus Aranha
caranha@cs.tsukuba.ac.jp

Department of Computer Science

2020/4/14
(last updated: March 8, 2020)

About the Lecturer

- Name: Claus Aranha;
- Country: Brazil;
- Research:
  - Evolutionary Algorithms;
  - Artificial Life;
- Hobbies:
  - Programming Games;
  - Watching Stars;
- webpage:

  `http://conclave.cs.tsukuba.ac.jp`

What is a Programming Challenge? - 2

- Program Challenges are good for Practicing Algorithms
- Program Challenges are good for Rapid Prototyping
- Program Challenges are used for Work Recruiting
- Program Challenges are also very fun puzzles!

This course is about Programming Challenges

The Goal of this course is solve programming challenges to become better at programming.

We will study algorithms and techniques that are common in programming competitions

First Things First: Important Notices

Manaba Page

All lecture notes and announcements for this course will be done through MANABA. Access the url below:

`https://manaba.tsukuba.ac.jp/ct/course_1149028`
Registration Code: 7255921

Language

- Lectures: Japanese
- Slides and materials: English
- Exercises: English
- Questions, Mails and Homework: Any language

What is a Programming Challenge? - 0

Consider a string with N characters chosen from G, C, T, A. Ex:

$$GACACATACAGATTACATTACAGA \ldots GATACCAGATA$$

When you receive pair of indexes s and e, calculate the number of "CA"s between N_s and N_e. Ex:

-
-
-

What is a Programming Challenge? - 0

Consider a string with N characters chosen from G, C, T, A. Ex:

GACACATACAGATTACATTACAGA ... GATACCAGATA

When you receive pair of indexes s and e, calculate the number of "CA"s between N_s and N_e. Ex:

- s = 0, e = 12 3 repetitions
-
-

What is a Programming Challenge? - 0

Consider a string with N characters chosen from G, C, T, A. Ex:

GACACATACAGATTACATTACAGA ... GATACCAGATA

When you receive pair of indexes s and e, calculate the number of "CA"s between N_s and N_e. Ex:

- s = 0, e = 12 — 3 repetitions
- s = 15, e = 20 — 1 repetition
-

What is a Programming Challenge? - 0

Consider a string with N characters chosen from G, C, T, A. Ex:

GACACATACAGATTACATTACAGA ... GATACCAGATA

When you receive pair of indexes s and e, calculate the number of "CA"s between N_s and N_e. Ex:

- s = 0, e = 12 — 3 repetitions
- s = 15, e = 20 — 1 repetition
- s = 4, e = 10 — 2 repetitions

What is a Programming Challenge? - 1

GACACATACAGATTACATTACAGA ... GATACCAGATA

Algorithm Idea 1

Start a counter $c = 0$, loop from N_s to N_e, and every time you find "CA", add to the counter.

Problem: If we have many queries, can we do faster?

What is a Programming Challenge? - 1

GACACATACAGATTACATTACAGA ... GATACCAGATA
00011222233333344444555 ... 88888899999

Algorithm Idea 1

Start a counter $c = 0$, loop from N_s to N_e, and every time you find "CA", add to the counter.

Problem: If we have many queries, can we do faster?

Algorithm Idea 2

Creat an auxiliary array A, that keeps the Cummulative sum of "CA"s from N_0 to N_i.

If we want to know the answer, we calculate $A_e - A_s$.

What is this course about?

You have learned many programming techniques...

...but can you use them?

Course Objective: Learning by Practice

- Every week: Solve 8 programming challenges;
- Choose and implement the **best algorithm** for each problem;
- Be careful with max time, and max memory;
- We will discuss algorithms, techniques and tricks;

Course Goal:

Improve programming abilities, techniques and familiarity.

Warnings about this class

1- Heavy Workload

- Starts easy, but hard in the end;
- A few hours/week of homework;
- Lots of debugging;

- Hint: Do your homework early!

2- Course Language

- My Japanese is not very good ;-) Let's talk in C++!
- All the course materials are in English;
- You can make your homework in Japanese;

- Practice some English in this course too! :-)

What is a “Programming Challenge”?

A puzzle that you solve by programming.

Parts of a Programming Challenge:

- Description;
- Standard input;
- Standard output;
- Examples;

Task: Write a program that:

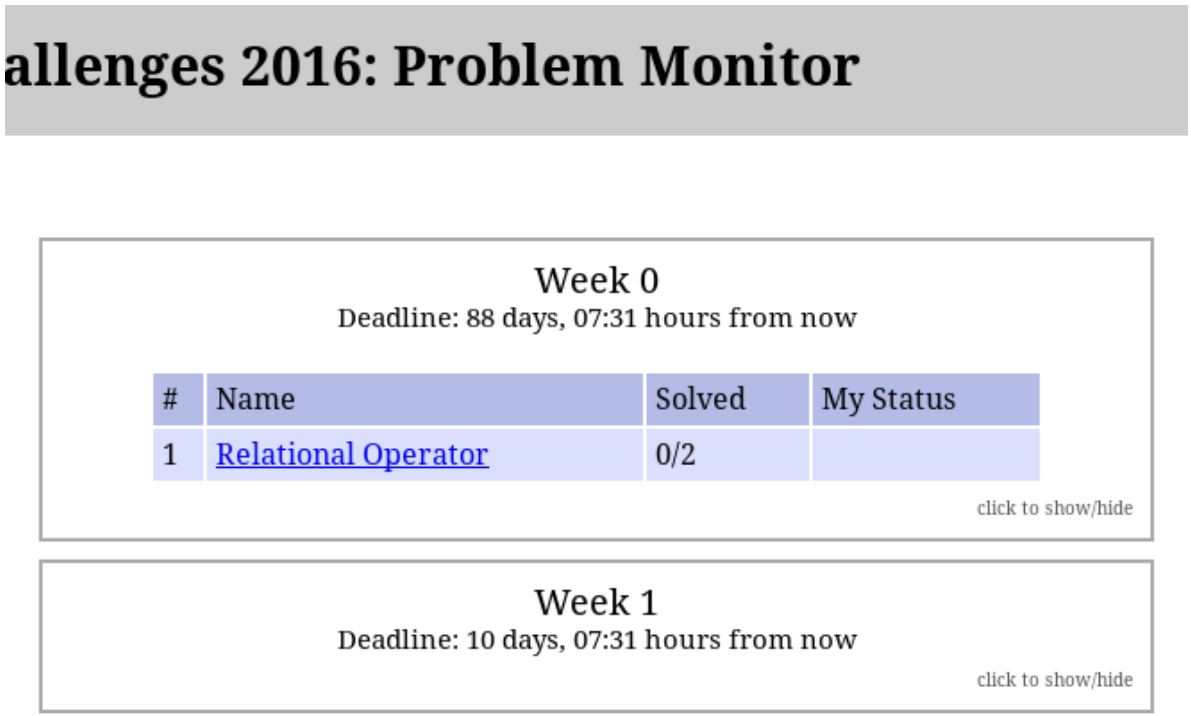
- Reads the input;
- Prints the correct output;
- And nothing else!

Tutorial: “Relational Operator” (1)

All challenges are listed at the page:

`http://conclave.cs.tsukuba.ac.jp/lecture/monitor.html`

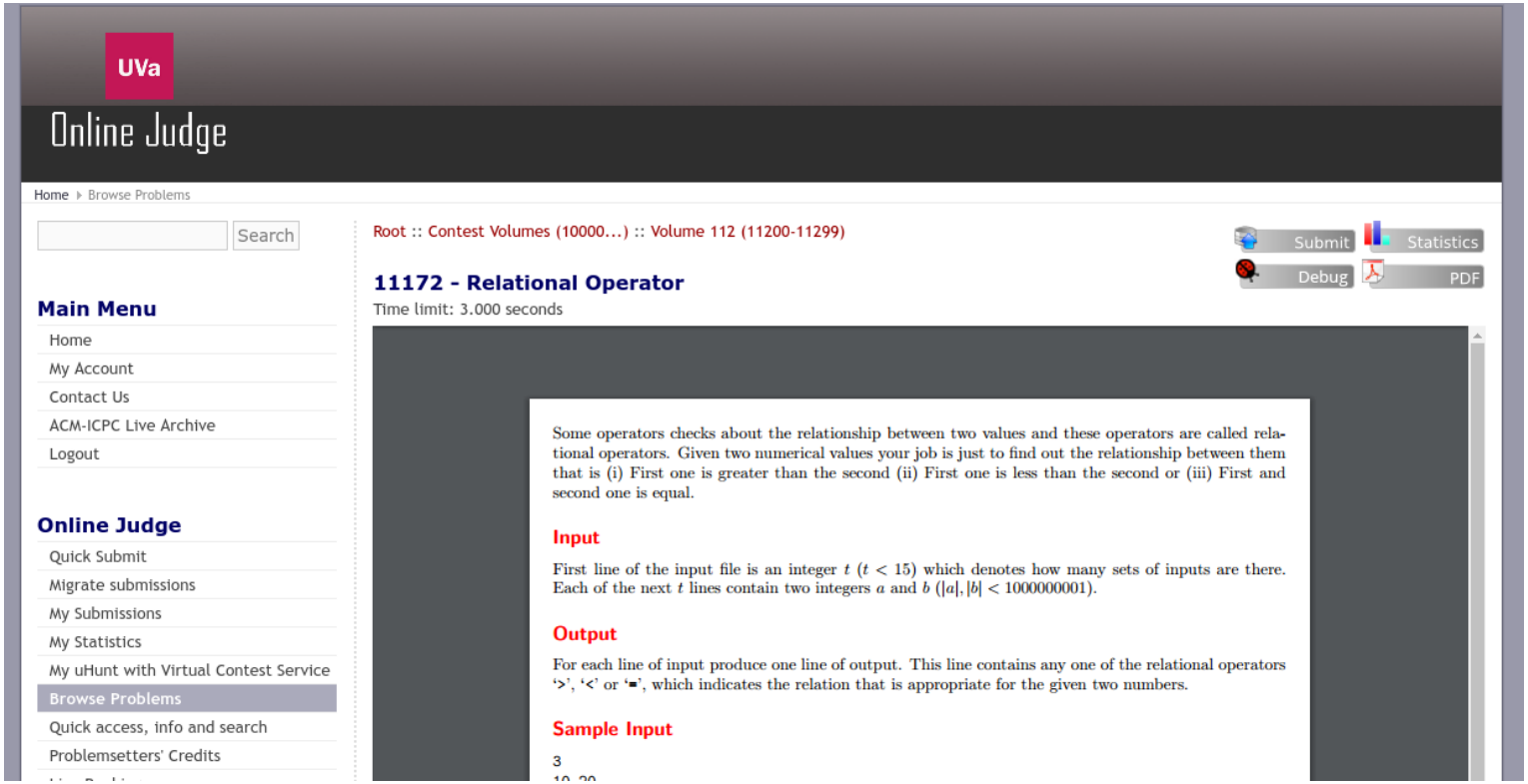
Let’s click on "Relational Operator"



Tutorial: “Relational Operator” (2)

The link takes you to the UVA (University of Valladolid) homepage (Please use an ad blocker!).

Here you can see the problem description, and the links to submit the problem.



Tutorial: “Relational Operator” (3)

Description

Some operators checks about the relationship between two values and these operators are called relational operators. Given two numerical values your job is just to find out the relationship between them that is (i) First one is greater than the second (ii) First one is less than the second or (iii) First and second one is equal.

- Reading the description can be the hardest part!
- For this challenge, you just need to:
  - If the first number is bigger than the second; print ">"
  - If the first number is smaller than the second; print "<"
  - If both numbers are equal, print "="

Very easy!

Tutorial: "Relational Operator" (4)

Input and Output

Input

First line of the input file is an integer $t (t < 15)$ which denotes how many sets of inputs are there. Each of the next t lines contain two integers a and b (|a|, $|b| < 1000000001$).

Output

For each line of input produce one line of output. This line contains any one of the relational operators ">", "<" or "=", which indicates the relation that is appropriate for the given two numbers.

- It is important to read the size of the problem!
- Pay attention to the shape of the output!

Tutorial: “Relational Operator” (5)

Examples

Sample Input

```
3
10 20
20 10
10 10
```

Sample Output

```
<
>
=
```

- Use the samples to test/debug your program.
- Use your own samples too!
- Use the samples to understand the challenge.

Solution: C++

```
// UVA 11172 - Relational Operator
// Test if a is bigger, smaller or equal to b

#include <iostream>
using namespace std;

int main()
{
    int n; long a, b;

    cin >> n;
    for (; n > 0; n--)
    {
        cin >> a >> b;
        if (a > b) cout << ">\n";
        if (a < b) cout << "<\n";
        if (a == b) cout << "=\n";
    }
}
```

Solution: Python

```
n = int(input())

while (n > 0):
    line = input()
    tokens = line.split()
    a,b = int(tokens[0]),int(tokens[1])

    if a > b: print(">")
    if a < b: print("<")
    if a == b: print("=")

    n -= 1
```

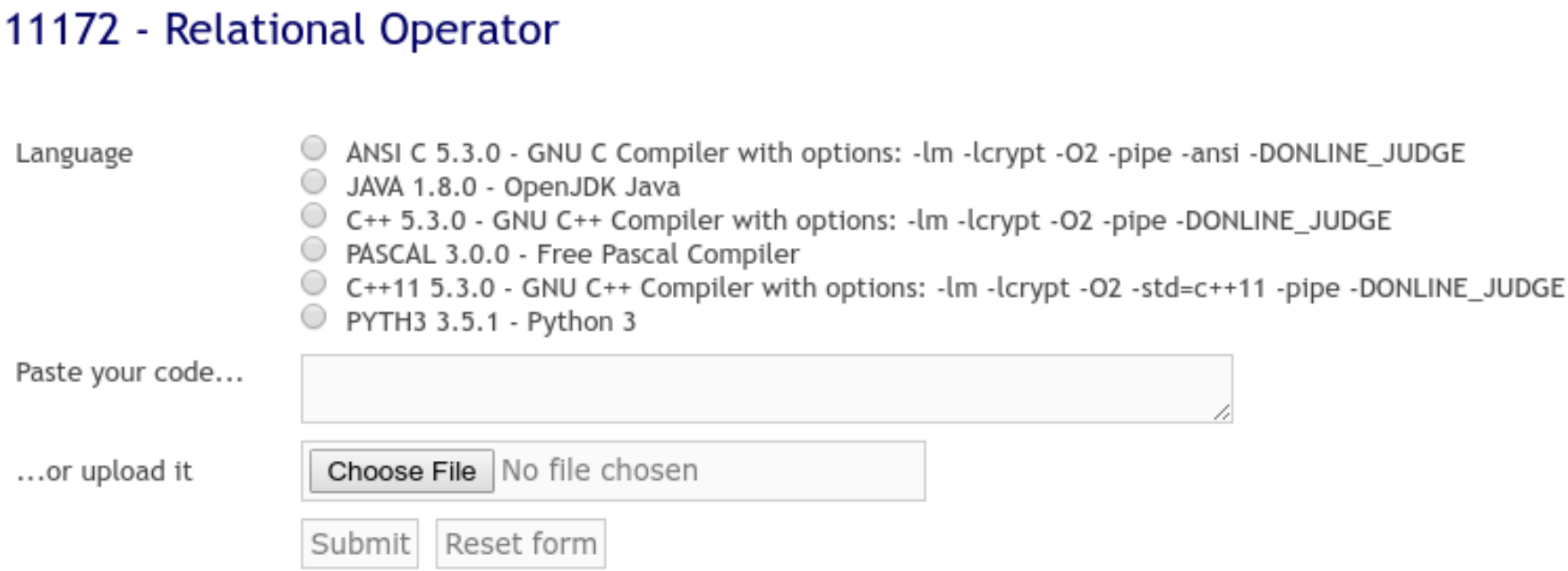
Solution: Java

```
import java.io.*;
class Main
{
   public static void main(String args[])
   {
      BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
      BufferedWriter stdout = new BufferedWriter(new OutputStreamWriter(System.out));
      try {
      String line;
      line = stdin.readLine();
      int n = Integer.parseInt(line);

      for (int i = 0; i < n; i++)
      {
         line = stdin.readLine();
         String[] tokens = line.split("\\s+");
         long a = Integer.parseInt(tokens[0]);
         long b = Integer.parseInt(tokens[1]);

         if (a > b)
            stdout.write(">\n");
         if (a < b)
            stdout.write("<\n");
         if (a == b)
            stdout.write("=\n");
         stdout.flush();
      }
      stdout.close();
      } catch (IOException ioe) { System.out.println("I/O Exception");}
   }
}
```

Java Solution – Keep in Mind

- All code must be in the one file;
- The static main method must be in Main class.
- Do not use public classes. Even Main must be non public.
- Use Buffered I/O for faster input/output.

Submitting the problem to UVA



- After you complete the program, use the submit button in the UVA page;
- Choose your language and submit your code;
- You can choose C, C++, Java or Python.

Submitting the problem to UVA

My Submissions

#		Problem	Verdict	Language	Run Time	Submission Date
17182419	1124	Celebrity jeopardy	Accepted	C++	0.000	2016-04-11 06:42:30
17181459	10141	Request for Proposal	Compilation error	C++11	0.000	2016-04-11 01:36:46
17181444	11498	Division of Nlogonia	Accepted	C++11	0.000	2016-04-11 01:30:43
17071417	102	Ecological Bin Packing	Compilation error	C++11	0.000	2016-03-23 09:21:55
17070667	161	Traffic Lights	Accepted	C++	0.000	2016-03-23 07:24:56
16607686	489	Hangman Judge	Accepted	C++	0.349	2015-12-20 03:52:45
16607670	489	Hangman Judge	Wrong answer	C++	0.335	2015-12-20 03:47:01
16607649	489	Hangman Judge	Runtime error	C++	0.000	2015-12-20 03:40:51

- UVA has an Automated Judge.
- After 2 5 minutes, you should receive an e-mail with the result.
- All your results can be seen at "My Submissions" Page.

Submitting the problem to UVA

These are the possible results:

- Accepted: Your program is correct! Congratulations!
- Presentation Error: Small mistake in number of spaces. Congratulations!
- Wrong Answer: Your program is incorrect. Time to debug.
- Time Limit Exceeded: Your program is too slow.
- Memory limit exceeded: Your program uses too much memory.
- Runtime Error: Your program crashed (segmentation fault!)

Problem Monitor Page

g Challenges 2016: Problem Monitor

Week 0

Deadline: 11 days, 00:18 hours from now

#	Name	Solved	My Status
1	Division of Nlogonia	1/2	Accepted
2	Cancer or Scorpio	0/2	Not submitted
3	The 3n + 1 problem	0/2	Not submitted
4	Request for Proposal	0/2	Not accepted

click to show/hide

- All Problems and results;
- All Deadlines;

Submitting the problem to MANABA

After you finish the problems listed in the monitor, you need to submit your source code and a comment file as a zip package to MANABA.

s2015XXXXXX-weekYY.zip

- problem1.cpp
- problem2.cpp
- problem5.cpp
- kaisetsu.txt

Attention

Submission to the UVA judge without a submission to MANABA will not be accepted!

Course Rules

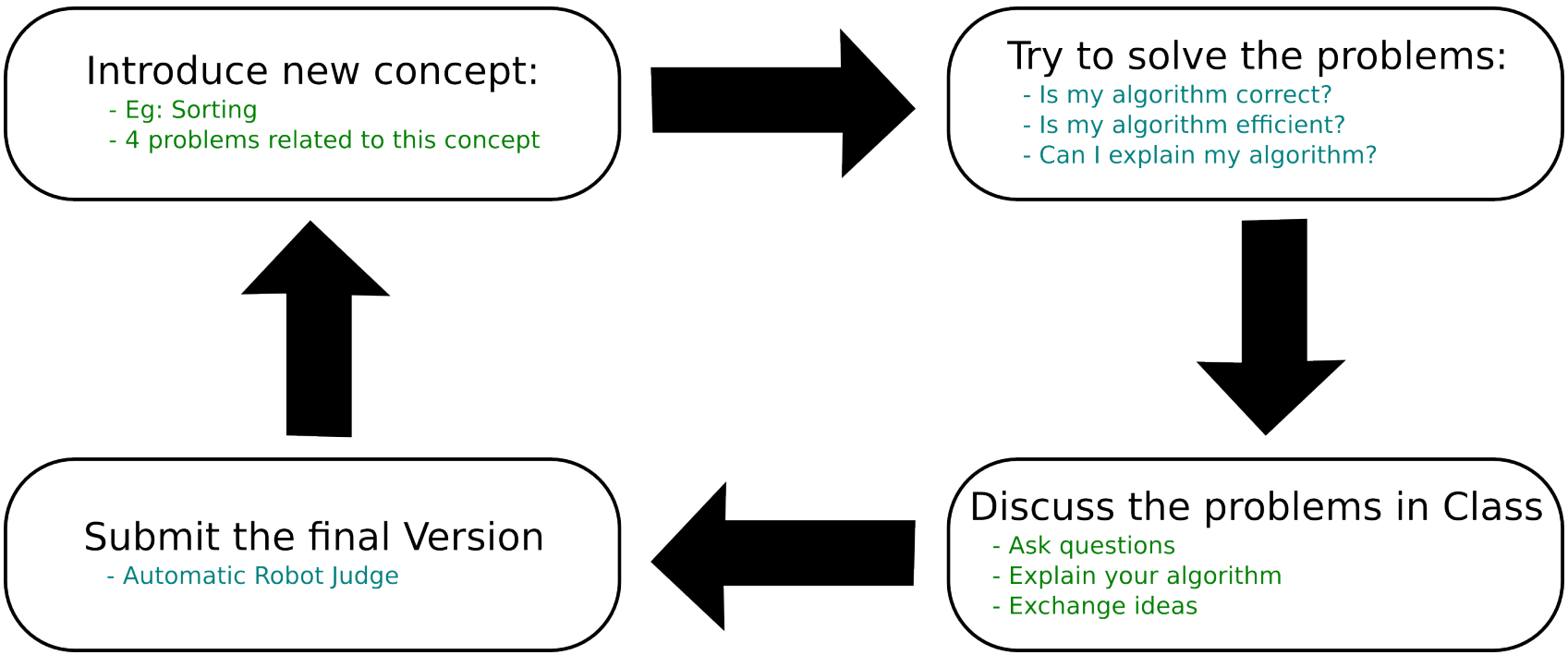
Two classes per week

- Each week has a theme (Graphs, Maths, etc...)
- Friday Class: Introduction
- Monday Class: Problem Solving and Q&A

Solving Problems

- Every week there are 8 programming assignments;
- Deadline is Thursday 23:59
  - UVA Submission
  - MANABA Submission

Outline


Introduce new concept:
- Eg: Sorting
- 4 problems related to this concept
Try to solve the problems:
- Is my algorithm correct?
- Is my algorithm efficient?
- Can I explain my algorithm?
Discuss the problems in Class
- Ask questions
- Explain your algorithm
- Exchange ideas
Submit the final Version
- Automatic Robot Judge

Grading Algorithm

Your Grade: **Base Grade** +Bonus -Penalty

- **Base Grade**:
  - You solved 2/8 problem/week: "C"
  - You solved 3/8 problems/week: "B"
  - You solved 4/8 problems/week: "A"
- +Bonus
  - 5% of best student in each category.
  - Special collaboration to the class.
- -Penalty
  - More than 25% of homework submitted late.

Evaluating and Grading: KAISETSU file

- Submit a TEXT (not word) file with your impressions of the problems, class, life, together with your code.
- Kaisetsu file can be in any Japanese or English.

Example

```
Name: Claus, ID: 98884735
# Problem 1:
This problem was very easy, but I did a stupid mistake
and it took me three hours to solve. I found the solution
when I tried a new test case.

# Problem 2:
This problem was very hard, and I was very hungry, so I
gave up.
```

Evaluation and Grading (5) – about plagiarism

The assignments are individual. Use your own strength to solve the programs.

GOOD

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

BAD

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

Plagiarism will result in course failure, and possibly worse.

Useful Links

- Manaba Page: All the class material will be here. Access Code is: 7255921
- UVA Online Judge: Use this page to submit your problems. Make an account and list the username on MANABA
- Problem Monitor: Use this page to check deadlines and weekly problems.
- Github Repository: Working directory for lecture notes. Send me PR, issues!
- uDebug: Web service that generates test inputs and test outputs for UVA problems. Useful tool for this course.

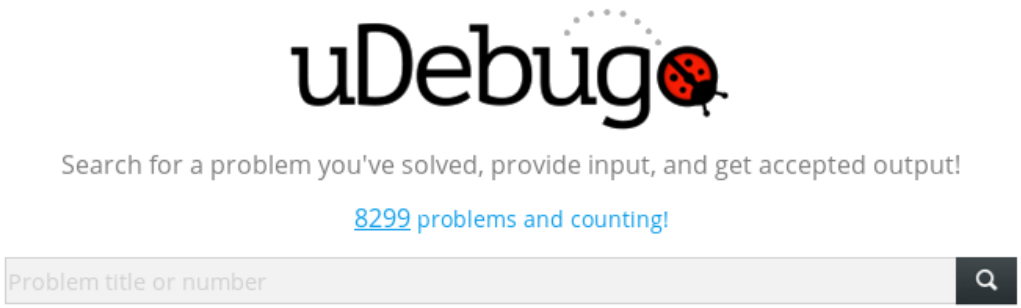
Course Book

- Competitive Programming, 3rd Edition (http://cpbook.net)
- For suggestions of books in Japanese, please check the Manaba materials!

uDebug Tool

uDebug generates outputs for many different debugs. It can help you check why your program is wrong.

`https://www.udebug.com/`



Contact the professor

- e-mail: caranha@cs.tsukuba.ac.jp
- website: `http://conclave.cs.tsukuba.ac.jp`

- Room: SB1012 – Send an e-mail and we can talk!

Both English and Japanese are okay!

What to do this weekend?

- Create an account on UVA (if you already have an account, you can use that)
- Submit your account name to the MANABA
- Ask any other questions you want to know!

Participate in ICPC!

- Fun challenges to choose the world champion!
- Teams of 3 people!
- Registration Deadline Early of June!

Today's Class: Ad Hoc Problems

- Ad Hoc: Problems that don't have a single algorithm;
- Common forms of Input and Output
- Debugging and Creating Test Cases
- Some Problem Analysis

Reading the Problem: 3n+1

For any two numbers *i* and *j*, calculate the Maximum Cycle Length.

```
Algorithm A(n)
1. print n
2. if n == 1 then STOP
3. if n is odd then n = 3n + 1
4. else n = n/2
5. GOTO 2
```

```
Example: A(22)
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Size: 16
```

Problem 3n+1 – Solution 1

```
while true:
  try:
    line = input()
    max = 0
    tk = line.split()
    i, j = int(tk[0]), int(tk[1])
    for n in range(i, j+1):
      count = 1
      while n != 1:
        if n % 2 == 1: x = 3 * x + 1
        else: n = n / 2
        count += 1
      if count > max: max = count
      print (line, max)
  except EOFError: break
```

Solution 1: Problem!

- Solution 1 solves all Sample Input correctly.
- But we still get Wrong Answer!
- Why??? :-(

Solution 1: Traps!

- Solution 1 solves all Sample Input correctly.
- If we try the input: 20 10 – output is nothing!

```
for x in range(i, j+1):    <-- Error is here!
    ...
    print (line, max)
```

- The sample input has no examples with "$i > j$"!

Solution 1: Traps!

Input Description

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 over all integers between and including i and j.

You can assume that no operation overflows a 32-bit integer.

The only rules are those expressely written!

Solution 1: Traps!!!!!

- First rule of Programming Challenges:
  Assume Worst Case
- Always try the worst case input!
  - Number is Negative; Number is zero;
  - Number is out of order (or in order);
  - Number is repeated;
  - Graph is unconnected; Graph is Fully connected;
  - Lines are parallel; Points are in the same place;
  - Area is 0; Angle is 0;
  - Input is very long;
  - Input is very short;
- If it is not against the rules: It will happen!
- This also happens in programs in the real world.

Problem 3n+1 – Solution 2

```
while true:
  try:
    line = input()
    max = 0
    tk = line.split()
    i, j = int(tk[0]), int(tk[1])
    for n in range(min(i, j), max(i, j)+1): # FIXED
      count = 0
      while n != 1:
        if n % 2 == 1: n = 3 * n + 1
        else: n = n / 2
        count += 1
      if count > max: max = count
      print (line, max)
  except EOFError: break
```

Solution 2: Time Limited Exceeded!

- All the inputs are correct.
- But now we have Time Limited Exceeded
- Why?

Solution 2: Cost Calculation

Input Description

All integers will be less than 10,000 and greater than 0.

You can assume that no operation overflows a 32-bit integer.

- What happens when the input is 1 10000?

```
1 10000 262
```

- The longest sequence in 1, 10000 has 262 steps.
- But we calculate all sequences between 1, 10000.
- Worst case: 10000*262 = 2,000,000 steps!

- For only one query!

Solution 2: Memoization

- Let's think about A();

```
A(22):  22 11 34 17 52 26 13 40 20 10 5 16 8 4 2
A(11):  11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
A(17):  17 52 26 13 40 20 10 5 16 8 4 2 1
A(13):  13 40 20 10 5 16 8 4 2 1
A(10):  10 5 16 8 4 2 1
A(8) :  8 4 2 1
```

- Do we need to Recalculate every time we see a new number?

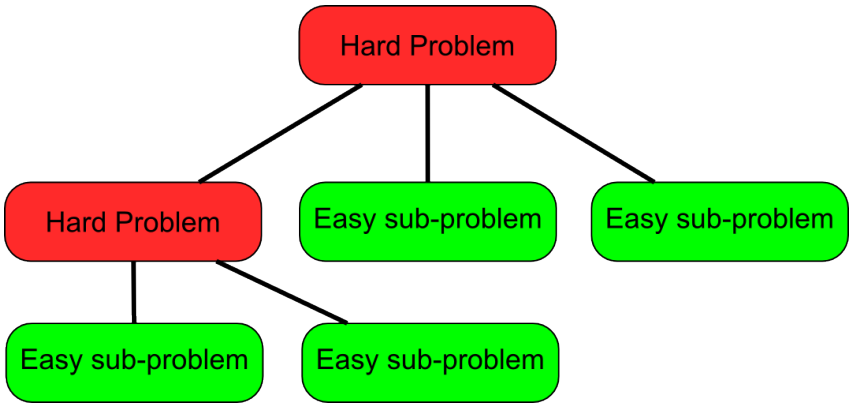
Good Technique: Memoization

If we know that we will use a result again in the future, we should store this result in the memory.

Solution 2: Memoization Idea

```
table = {}
table[1] = 1

def A(n):
  if n in table.keys():
    return table[n]
  else:
    if n % 2 == 1:
      table[n] = 1 + A(3*n + 1)
    else:
      table[n] = 1 + A(n/2)
  return table[n]
```

A Programming Challenge Workflow



Trick to solve problems without bugs: Break the problem down

- How to calculate the function $A()$;
- Imagine the worst possible cases ($i > j$);
- Calculate cost of solution;
- Improve speed with memoization;

A Programming Challenge Workflow

Common Steps for Programming challenges:

- Task 1: Read the problem description;
- Task 2: Read the input/output;
- Task 3: Think about the algorithm;
- Task 4: Write the Code;
- Task 5: Test the program on example data;
- Task 6: Test the program on hidden data;

Task 1: Understanding the Problem Description

The English description is so hard!

Don't Worry:

1. Separate the text into flavor and rules;
2. Sometimes it is easy to read the input/output first, and then the text;
3. Problems with a lot of flavor are usually not very hard.;

Example: Problem 11559 – Event Planning

Flavor:

As you didn't show up to the yearly general meeting of the Nordic Club of Pin Collectors, you were unanimously elected to organize this years excursion to Pin City. You are free to choose from a number of weekends this autumn, and have to find a suitable hotel to stay at, preferably as cheap as possible.

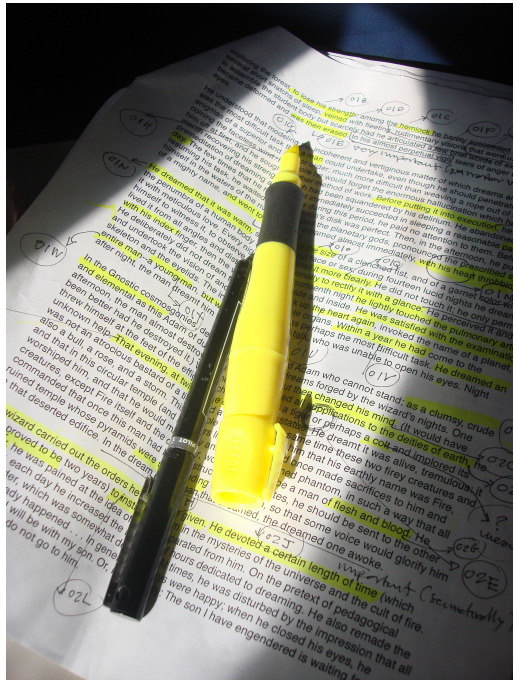
rules

You have some constraints: The total cost of the trip must be within budget, of course. All participants must stay at the same hotel, to avoid last years catastrophe, where some members got lost in the city, never being seen again.

Keywords: constraints, minimum, maximum, cost, rules, number, etc...

Hints for hard to read problems

- First, look at the sample input and output;
- Write the idea of the problem on paper
- Use the Paper: mark keywords;
- Use the Paper: cut flavor;
- Read the problem again!;
- Do not begin programming until you understand the problem!



Task 2: Reading the Input/Output

The Input Description is very important (as we saw in 3n+1)

- What is the size of the data?
- What are the limits of the data?
- What is the format of the data?
- What is the stop condition?

Task 2: Input Size

The input size shows how big the problem gets.

- Small Problem: Full Search; Simulation;
- Big Problem: Complex Algorithms; Pruning;

Keep in Mind!

- The average time limit in UVA is 1-3 seconds.
- Expect maybe 10.000.000 operations per second.

Task 2: Input Size – Examples

n < 24

Exponential algorithms will work ($O(2^n)$).
Or sometimes you can just calculate all solutions.

n = 500

Cubic algorithms don't work anymore ($O(n^3) = 125.000.000$)
Maybe $O(n^2 \log n)$ will still work.

n = 10.000

A square algorithm ($O(n^2)$) might still work.
But beware any big constants!

n = 1.000.000

$O(n \log n) = 13.000.000$
We might need a linear algorithm!

Task 2: Input Format

Three common patterns for input format:

- Read N, then read N queries;
- Read until a special condition;
- Read until EOF;

Task 2: Input Format

Read *N*, and then read *N* queries;

Remember *N* when calculating the size of the problem!

Example: Cost Cutting

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;

    for (; n > 0; n--)
    {
       // Do something
    }
}
```

Task 2: Input Format

Read Until a Special Condition.
Be careful: You can have many queries before the condition!

Example: Request for Proposal:
The input ends with a line containing two zeroes.

```
int main()
{
    cin >> n >> p;
    while (n!=0 || p!=0)
    {
        // do something!
        cin >> n >> p;
    }
}
```

Task 2: Input Format

Read until EOF.

Functions in C and Java return FALSE when they read EOF. Python requires an exception. Very common in UVA.

Example: 3N+1 Problem, Jolly Jumpers

```
int main()
{
    int a, b;
    while (cin >> a >> b;)
    {
        // Do something!
    }
}
```

Task 2: Output Format

The UVA judge decides the result based on a simple diff.

Be very careful that the output is exactly right!

The Judge is like an angry client. It wants the output EXACTLY how it stated.

Task 2: Output Format – Checklist

1. DID YOU REMOVE DEBUG OUTPUT?
2. DID YOU REMOVE DEBUG OUTPUT?
3. Easy mistakes: UPPERCASE x lowercase, spleling mitsakes;
4. Boring mistakes: plural: 1 hour or 2 hours;
5. What is the precision of float? (3.051 or 3.05)
6. Round up or Round down? ($3.62 \rightarrow 3$ or 4)
7. Multiple solutions: Which one do you output (usually ortographical sort)

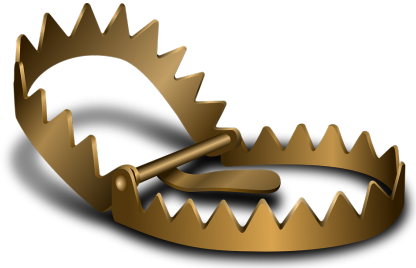
Task 2: Input/Output – Traps!

Example: 3n+1 Problem

- *i* and *j* can come in any order.

Common Traps

- Negative numbers, zeros;
- Duplicated input, empty input;
- No solutions, multiple solutions;
- Other special cases;



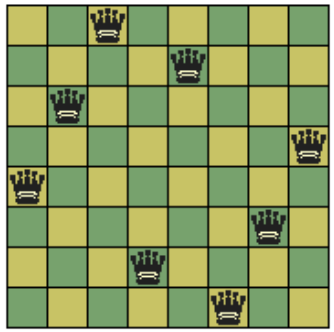
Task 3: Choosing the algorithm

The important part of choosing the algorithm is counting the time

- An algorithm with k-nested loops of and n commands has $O(nk)$ complexity;
- A recursive algorithm with b recursive calls per level, and L levels, it should have $O(bL)$ complexity;
- An algorithm with p nested loops of size n is $O(n^p)$
- An algorithm processing a $n * n$ matrix in O(k) per cell runs in $O(kn^2)$ time.

Use pruning to reduce the complexity of your algorithm!
Also don't forget the number of queries!

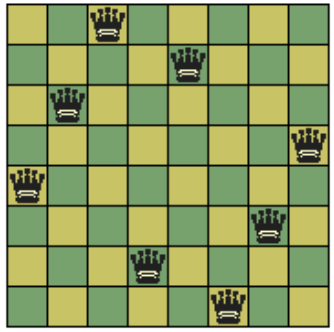
Task 3: Example of Pruning – 8 queen problem



You want to find a position for 8 queens where no queen attack another queen.

```
Solution 1: All options:
Queen1: a1 or a2 or a3 or a4 ... or h5 or h6 or h7
Queen2: Same as Queen 1
Queen3: Same as Queen 2
...
Queen8: Same as Queen 7

Total Solutions: 64 x 64 x 64 x 64 = 64^8 ~ 10^14
```

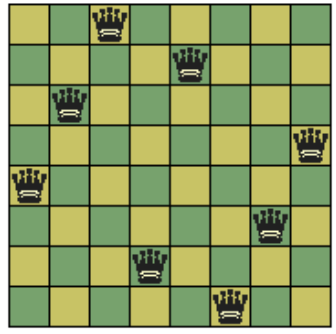
Task 3: Example of Pruning – 8 queen problem



You want to find a position for 8 queens where no queen attack another queen.

```
Solution 2: One queen / column
Queen1: a1 or a2 or a3 ...
Queen2: b1 or b2 or b3 ...
Queen3: c1 or c2 or c3 ...
...
Queen8: h1 or h2 or h3 ...

Total Solutions: 8 x 8 x 8 ... = 8^8 ~ 10^7
```

Task 3: Example of Pruning – 8 queen problem



You want to find a position for 8 queens where no queen attack another queen.

```
Solution 3: One queen / row x column
If Q1 is a1, Q2 in b2, Q3 must be c3-7...

A solution is the order of rows: Ex: 1-3-5-2-7-4-8-

Total Solutions: 8 x 7 x 6 x 5 ... = 40320
```

Task 4: Coding

Do you understand The Problem and The Algorithm?

NOW you can start writing your program.

If you start your program before you understand the solution, you will create many more bugs.

Task 4: Coding

Hint 1: "The Library"

Create a file with code examples that you often use.

- Input/output functions;
- Common data structures;
- Difficult algorithms;

Hint 2: Use paper

- Writing your idea on paper help you visualize;
- Sometimes you can find bugs this way;

Task 4: Coding

Hint 3: Programmer Efficiency

Everyone knows about CPU efficiency or Memory efficiency.

But Programmer Efficiency is very important too: Don't get tired/confused!

- Use standard library and macros;
- Your program just need to solve THIS problem;
- Use simple structures and algorithms;
- TDD mindset;

Task 5,6: Test and Hidden Data

The example data **is not** all the data!

Example Data

- Useful to test input/output;
- Read the example data to understand the problem;

Hidden Data

- Used by the UVA judge;
- Contain bigger data sets;
- Includes special cases;

Generate your own set of hidden data before submitting!

What data to generate?

- Datas with multiple entries (to check for initialization);
- Datas with maximum size (can be trivial cases);
- Random data;
- Border cases (maximum and minimum values in input range);
- Worst cases (depends on the problem);

The uDebug Website

uDebugo

Problem title or number

11947 - Cancer or Scorpio [Problem Statement]

Category: UVa Online Judge
Type of Problem: Single Output Problem
Solution by: forthright48
Random Input by: Morass

INPUT

```
1000
01012000
01022000
01032000
01042000
01052000
01062000
01072000
01082000
01092000
01102000
```

Go! Critical Input Random Input

ACCEPTED OUTPUT

```
1 10/07/2000 libra
2 10/08/2000 libra
3 10/09/2000 libra
```

YOUR OUTPUT

To Summarize

Mental framework to solve problems:

1. Read the problem carefully to avoid traps;
2. Think of the algorithm and data structure;
3. Keep the size of the problem in mind;
4. Keep your code simple;
5. Create special test cases;

Now go solve the other problems!

Thanks for Listening!

Any questions?

BONUS – Calculating Complexity in Research Experiments

BONUS – A very simple program

Ackermann's Function

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0, n = 0 \\ A(m-1,A(m,n-1)) & \text{if } m > 0, n > 0 \end{cases}$$

- $A(0,n) = 1$ step
- $A(1,n) = 2n+2$ steps
- $A(2,n) = n^2$ steps
- $A(3,n) = 2^{n+3} - 3$!
- $A(4,n) = 2^{2^{\cdots^2}} - 3$!!! (exponential tower of n+3)
- $A(5,n) =$!!!!!!!!!!