Algorithms for Programming Contests - Week 1

Prof. Dr. Javier Esparza
A. R. Balasubramanian, Christoph Welzel,
Stefan Jaax, Tobias Meggendorfer,
Mikhail Raskin,
conpra@in.tum.de

21.04.2020

General Concept

- Theoretical background about several concepts and algorithms.
- Implementing algorithms is a useful skill, for CS research, other research, software development... sometimes for hobbies.
- Contest-format algorithm implementation practical course.

The Problem Sets

- Available on the course homepage, accessible with password only (you can also see the current tasks in the contest system — zip file of sample inputs only available on the course page). This semester: also Moodle.
- Usually, five problems per week:
    - two easy (4 Points each),
    - two medium (6 Points each),
    - one hard (8 Points).
- Each week's problems are (mostly) about the topics, algorithms and concepts from that week's lecture.
- There will be hints in the lectures.

Have a Problem with a Problem?

Should difficulties arise: Ask questions! Write a clarification request! If needed we could find a time for a call.

Grading

Your final grade will be determined by how many points you earned, as well as an oral exam at the end of the semester. The oral exam will account for at most 40% of the grade (using rounding to the nearest). The (tentative) key for grading the problems is the following:

Percentage	Grade
$\geq 90\%$	1.0
$\geq 85\%$	1.3
$\geq 80\%$	1.7
$\geq 75\%$	2.0
$\geq 70\%$	2.3
$\geq 65\%$	2.7
$\geq 60\%$	3.0
$\geq 55\%$	3.3
$\geq 50\%$	3.7
$\geq 40\%$	4.0

Topics (preliminary list)

1. Introduction
2. Data Structures (UF, Binary Search, Graphs)
3. Graphs, Minimum Spanning Trees, DFS, BFS
4. Shortest Paths
5. Maximum Flows
6. Brute Force / Backtracking
7. Greedy
8. Dynamic Programming
9. Number Theory
10. Trees
11. Geometry
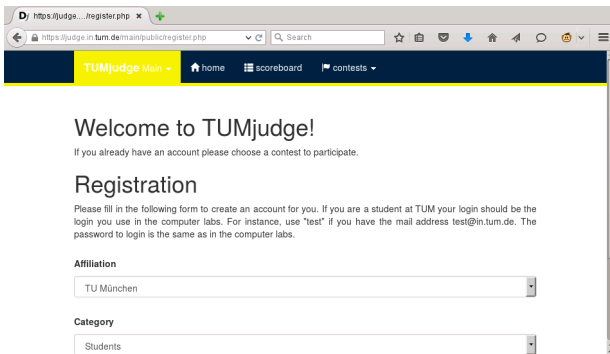12. Contest
13. Conclusion

Official DOMjudge System

- In use for programming contests such as the GCPC or the ICPC.
- On the web:

TUMjudge

https://judge.in.tum.de/conpra/

Registration

- Registration necessary.

Login

- Authentification necessary.
- Works with LDAP and is identical with the Webmail.IN.TUM.de account.
    - Login is "name" in name@in.tum.de.
    - Password is the corresponding password.
- Forgot your password and want to change / reset it?
    - RBG can reset your password, we cannot!
    - We can do something ad-hoc to create you an account.
      (we'd send you to RBG, but you might not be allowed to enter)

Overview

The overview page shows several pieces of information:

- the personal scoreboard of the current contest,
- an overview of already submitted programs,
- an overview of clarifications (more on that later on).

Overview

Problem structure

A problem consists of several parts:

- name, abbreviation, difficulty,
- problem author,
- problem statement,
- input format specification,
- output format specification,
- constraints,
- sample input and output.

SS15N01A   Hello World!

Author: Stefan Toman

This is probably the first problem you will solve and it should help you set up and test your system. Solve this problem first to make sure everything is in place.

We would like to introduce you to Lea. You will meet her in many of the problems you will solve. After reading all of them you will know her quite well.

Lea is a very friendly person who likes to say hello to everybody, but she doesn't want to say the same thing to every person she meets. Therefore, she never knows what to say. For greeting Bob it is appropriate to say "Hello Bob!", whereas for greeting Peter it is better to say "Hello Peter!". Help her and tell her which sentence to use.

Input

The first line of the input contains an integer t. t test cases follow.

Each test case consists of a single line containing a name *name*.

Output

For each test case, print a line containing "Case #i: Hello *name*!" where i is its number, starting at 1. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$.
- No name will contain whitespaces.
- The names' lengths will be at most 100.

Sample Data

Input
1 2
2 Bob
3 Peter

Output
1 Case #1: Hello Bob!
2 Case #2: Hello Peter!

Submitting programs

Submitting program is done on the TUMjudge web interface entirely.

- No files to be sent via e-mail etc.
- Only source code files are uploaded, no .class files or similar.

Submit

- Choose files to be uploaded by Drag-and-Drop or in the menu "Choose Files",
- Choose problem,
- Choose language (unless already chosen automatically),
- "submit",
- F5, F5, F5, F5, ... (the page actually reloads automatically)

Submitting programs

Judging

The TUMjudge

- compiles,
- executes,
- tests

the submission against several test cases. As long as the TUMjudge is working on a submission, the submission's status is "PENDING".
The submission is treated instantaneously and the TUMjudge (usually) announces its verdict within a few moments.

Judging

The following verdicts can occur:

CORRECT

The submission successfully solved all the test cases.

COMPILER-ERROR

The submission could not be compiled. The exact error message can be seen on the submission's detail page.

NO-OUTPUT

The submission does not produce any output. Be sure to output to "standard out".

Judging

TIMELIMIT

The submission runs longer than the maximal allowed time and was terminated.
Possible reasons:

- The submission runs in an endless loop.
- The submission is not efficient enough.

RUN-ERROR

An error occured during the submission's execution.
Possible reasons:

- Division by 0.
- Incorrectly addressing memory locations, e.g. ArrayIndexOutOfBounds.
- Using more memory than the allowed memory limit.

Judging

WRONG-ANSWER

The submission's output is incorrect.
Possible reasons:

- The answer is just wrong.
- The answer does not conform to the output format specification given on the problem set.
- The answer is not exact enough (e.g. with floating point answers with a desired precision).

TOO-LATE

The program was submitted after the submission deadline. It is stored in the system, but no longer processed.

Scoreboard

Different background colors indicate different outcomes:

Problem solved.

Problem solved first.

Incorrect submission(s).

Submission in pending status.

No submissions.

Scoreboard

Order (tie-breakers):

1. number of solved problems,

2. score:
   - per problem: (number of incorrect submissions) $*$ (penalty time) $+$ (time for the first correct submission),
   - penalty time $= 600$, i.e., 10 hours,
   - e.g. $3/1820$ indicates: the problem was solved with 3 submissions, with a total penalty time of 1820.

You can submit any number of times to solve a problem!

Each week's score itself (apart from the number of problem solved) does not change the grading at the end of the semester, it only affects the position in the scoreboard of that week.

Scoreboard

Anybody who does not want to be seen in the public scoreboard, must choose the invisibility option during the registration (in "Category").

- We can switch you between visible and invisible in public scoreboard if you made a mistake.

Clarifications

- Messages to the system administrators, i.e., the teaching assistants and/or tutors.
- Sent via the "request clarification" form on the overview page.
- Used for questions about the problems or about the system in general.
- Please choose a subject accordingly: either "general" or the specific problem.
- Depending on the actual question, the answer is only visible to the persons who sent the question, or it is published to all users of the system.
- The answer (along with the question) can be seen on the right side on the overview page.

Clarifications

Discussion forum

Sometimes you want to discuss with other students instead of sending clarification requests.
Please discuss problem statements, corner cases, algorithms and approaches. The code should be your own.

You are welcome to use the forum instance with a subsection dedicated to the course: `https://judge.in.tum.de/discuss/c/conpra`

Discussion forum

ICPC@TUM Discuss

ConPra ▶ | **Latest** | New | Unread | Top

Welcome to our community! These are the most popular recent topics.

Topic	Replies	Activity
★ About the ConPra category	0	Apr '16

There are no more ConPra topics. **Why not create a topic?**

Discussion forum

You are welcome to use the forum instance with a subsection dedicated to the course: https://judge.in.tum.de/discuss/c/conpra

You are welcome to discuss and post test cases if you want to
(But please do not post and do not ask for code snippets)

Restrictions

- Compilation of a submission may take no longer than 30 seconds. After that time, compilation is aborted and the verdict will be a COMPILER-ERROR.

- The maximal allowed size of a source code file is 256 KB. Bigger submissions will not be accepted.

- During the execution of a submissions, up to 1.5 GB of memory is available. This includes source code, variables, stack, Java VM (up to 0,35 GB),... If a submission tries to address more memory, it will be terminated and the verdict will be a RUN-ERROR.

- It is not allowed to use multi threading. Each submission has only one CPU core fully at its disposal.

Restrictions

Tampering with the system in any way will be penalized!
Do not fool the system!

- Do not open files, input is always in "standard in".
- Do not address files locally on the system! This is not possible anyways.
- Do not open network connections.
- . . .

Furthermore, please keep the number of submissions at an acceptable level as to not unnecessarily slow judging for all participants.

Java Submission

```java
import java.util.Scanner;

public class JavaSubmission {
    public static void main(String[] args) {
        // create scanner object
        Scanner s = new Scanner(System.in);

        // loop over all test cases
        int t = s.nextInt();
        for(int i = 1; i <= t; i++) {

            // read several types of input
            boolean b = s.nextBoolean();
            String st = s.next();

            // output: use the possibility you like more
            System.out.println("Case #"+i+": "+st);
            System.out.format("Case %d#: %s\n", i, st);
        }
        s.close();
    }
}
```

C++ Submission

```cpp
#include <iostream>
#include <stdio.h>

int main() {
    // loop over all test cases
    int t;
    scanf("%d", t);
    for(int i = 1; i <= t; i++) {

        // read several types of input
        int j;
        std::string s1;
        char s2[101];
        // use the possibility you like more
        std::cin >> j >> s1;
        scanf("%d %100s", &j, s2);
        // output: use the possibility you like more
        std::cout << "Case #" << i << ": " << s1 << std::endl;
        printf("Case #%d: %s %d", i, s2, j);
    }
    return 0;
}
```

Python Submission

```python
import sys

if __name__ == '__main__':
    case_count = int(sys.stdin.readline())
    for case_number in range(1, case_count + 1):
        n, k = map(int, sys.stdin.readline().split())
        print("Case #{0}: {1}".format(case_number, n + k))
```

Julia Submission (experimental)

```
module add_pairs

function __init__()
        n = parse(Int,readline())
        for k = 1:n
                numbers = map(x->parse(Int,x),
                        split(readline()," "))
                println("Case #",k,": ",sum(numbers))
        end
end

end
```

Understanding Problems

- Read the problem statement very carefully.
- Also the constraints, think about special cases:
    - E.g. if there are negative values or 0 allowed, then there is probably a test case for that.
    - E.g. special characters or a space when dealing with strings.
    - ...

Solving Problems

- Code efficiently.
  - Think about which data types to use.
  - Sometimes arrays might not have to be two- or three-dimensional.
  - Sometimes objects add runtime overhead without making code clearer.
  - Implement algorithms given in the lecture with their amortized running times.
- Look carefully at the input and output specifications and let your program be conform to those!
- Remove all debug messages before submitting.
- Write comments!

Debugging your programs

- RUNTIME-ERROR supersedes WRONG-ANSWER
- For C++ and memory corruption: Valgrind
- Generating completely random input can be faster than waiting for us to answer
    - ... do not only write tests by hand, write a generator
- Shortest possible input?
- If you get TIMELIMIT, look at your program and try constructing its worst input
- Small function mean less variables in scope (more mistakes become compile-time errors)
    - ... and small functions can be tested separately
- We may omit the very worst test cases — so if optimisation is useful *almost* always, use it
- If someting is formally allowed to be zero, we might have such a case

Code from the Internet

- You are allowed to download and include a library, but you need to cite the correct source.
- Do this by putting a comment in your code stating the url or similar.
- Including other people's published code without a reference, or using other students' code, is a rule violation
- However, we advise you to code on your own as it improves the understanding about the algorithms involved. You probably need this in subsequent problems anyway.

Half Points

If the judge does not accept your solution but you are sure you solved it correctly, use the "request clarification" option. In your request include:

- the name of the problem (by selecting it in the subject field),
- a verbose description of your approach to solve the problem,
- the time you submitted the solution we should judge.

We will check your submission and award you half the points if there is only a minor flaw in your code.

We might use submissions from half-point requests (anonymised) as realistic examples of subtle problems in the lectures and debugging demo sessions.

Big Inputs / Outputs

Let's speak about actual contents of the course!

Big Inputs / Outputs

- Some problems require you to read/write a substantial amount of input/output.
- Even without doing anything else, this can take longer than the allowed time limit when not handled correctly!

Speed up your code easily!

Use the faster readers / writers when handling big data.

Java - Input

Always remember to throw IOExceptions!

BufferedReader

```
InputStreamReader r = new InputStreamReader(System.in);
BufferedReader in = new BufferedReader(r);
String line = in.readLine();
String[] parts = line.split(" ");
int n = Integer.parseInt(parts[0]);
double d = Double.parseDouble(parts[1]);
```

Scanner

```
Scanner s = new Scanner(System.in);
int n = s.nextInt();
double d = s.nextDouble();
```

Java - Input

Scanner is much more convenient to use but slower!

Input Size	Scanner	BufferedReader
5 Mio Integers	3321 ms	431 ms
50 Mio Integers	30988 ms	3937 ms

C++ - Input

cin

```
#include <iostream.h>
...
int n;
double d;
cin >> n >> d;
```

scanf

```
int n;
double d;
scanf("%d %i", &n, &d);
```

C++ - Input

Input Size	cin	scanf
5 Mio Integers	1887 ms	552 ms
50 Mio Integers	18789 ms	5467 ms

cin synchronizes with stdio buffers.
Turning this off can make it even faster than scanf.

```
std::ios_base::sync_with_stdio(false);
```

Java - Output

Assume that we want to print the integer answer x and the case header for case i.

println

```
System.out.println("Case #" + i + ": " + x);
```

println

```
System.out.format("Case #%d: %d\n", i, x);
```

BufferedWriter

```
OutputStreamWriter s = new OutputStreamWriter(System.out);
BufferedWriter out = new BufferedWriter(s);
out.write("Case #" + i = ": " + x + "\n");
```

Java - Output

StringBuilder (println)

```
StringBuilder sb = new StringBuilder():
sb.append("Case #");
sb.append(i);
sb.append(": ");
sb.append(x);
sb.append("\n");
System.out.println(sb);
```

Java - Output

StringBuilder (BufferedWriter)

```
StringBuilder sb = new StringBuilder():
sb.append("Case #");
sb.append(i);
sb.append(": ");
sb.append(x);
sb.append("\n");
OutputStreamWriter s = new OutputStreamWriter(System.out);
BufferedWriter out = new BufferedWriter(s);
out.write(sb);
```

Java - Output

Choose a method that is convenient and matches the expected size of the output.

Output size	println	format	BufferedWriter
5 Mio Integers	27220 ms	37250 ms	558 ms
50 Mio Integers	-	-	5057 ms

Output Size	StringBuilder(println)	StringBuilder (BufferedWriter)
5 Mio Integers	440 ms	470 ms
50 Mio Integers	4057 ms	4234 ms

C++ - Output

cout

```
#include  <iostream.h>
...
cout << "Case #" << i << ": " << x << endl;
```

printf

```
printf("Case #%d: %i\n", i, x);
```

C++ - Output

Output Size	cout	printf
5 Mio Integers	11927 ms	492 ms
50 Mio Integers	-	4919 ms

Again, cout can be improved by using the following line.

```
std::ios_base::sync_with_stdio(false);
```

Julia, Python

There might be some problems too large for Python

- we tried to avoid this in the course
- Python gets slightly more time than Java in ConPra

Julia setup is experimental/work-in-progress
No advice right now — maybe later