Competitive Programming

Frank Takes

LIACS, Leiden University

`https://liacs.leidenuniv.nl/~takesfw/CP`

Lecture 1 — Introduction to Competitive Programming

About this course

- Competitive Programming

About this course

- Competitive Programming: problem solving, algorithm selection, algorithm design, data structure optimization, complexity analysis, . . .

About this course

- Competitive Programming: problem solving, algorithm selection, algorithm design, data structure optimization, complexity analysis, . . .
- . . . in a competitive context

About this course

- Competitive Programming: problem solving, algorithm selection, algorithm design, data structure optimization, complexity analysis, . . .
- . . . in a competitive context, i.e., with
  - limited CPU time
  - limited memory consumption
  - a fixed amount of problem solving time (optional)
  - others competing with you (more optional)

About this course

- Competitive Programming: problem solving, algorithm selection, algorithm design, data structure optimization, complexity analysis, ...
- ... in a competitive context, i.e., with
  - limited CPU time
  - limited memory consumption
  - a fixed amount of problem solving time (optional)
  - others competing with you (more optional)
- This is not software engineering, but algorithmic problem solving.

Course information

- Lectures: Thursdays, 9:15 to 11:00 in Snellius room 408
- Sometimes including a lab session in room 302-304
- Period: February 6 — April 30, 2020; not on April 23
- Prerequisites: Algorithms, Datastructures (bachelor)
- Required skills: C++ (or Java)
- Course website:
  `https://liacs.leidenuniv.nl/~takesfw/CP`

Course format

- 13 weeks: presentations by lecturer and students
- No exam
- Books as reference material
- Grade composition

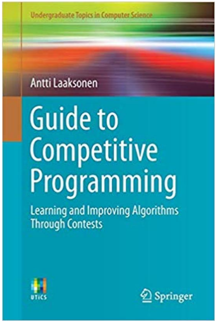
one individual assignment	20%
a presentation and report	35%
three programming contests	$3 \times 15 = 45\%$

- All five grades have to be > 5
- Final grades are rounded to nearest element in $\{1, 2, 3, 4, 5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10\}$
- 6 ECTS

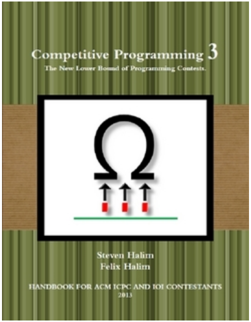
Course team

- Lecturer: dr. Frank Takes
  `f.w.takes@liacs.leidenuniv.nl`, room 157b
- Assistant: Ludo Pulles BSc
  `l.n.pulles@umail.leidenuniv.nl`

Books


Undergraduate Topics in Computer Science
Antti Laaksonen
Guide to Competitive Programming
Learning and Improving Algorithms Through Contests
UTICS
Springer

Antti Laaksonen, *Guide to Competitive Programming*, Springer, 2017.


Competitive Programming 3
The New Lower Bound of Programming Contests
Ω
Steven Halim
Felix Halim
HANDBOOK FOR ACM ICPC AND IOI CONTESTANTS
2013

Steven Halim and Felix Halim, *Competitive Programming 3*, Lulu.com, 2013.

Before we start . . .

- Deadlines and assignment (retake) deadlines are hard set as of next week
- Individual assignments must be made alone
- Team work should be balanced
- Plagiarism = instant removal from course
- This is a brand new course taught for the first time in Spring 2020; there will be errors, hickups, etc.
- Please contribute! Feedback is very welcome

To be announced (before) next week

- Individual assignment
- Degree of individual vs. teamwork in contests
- List of topics for presentation and report
- Deadlines for individual assignment and programming contests

Competitive programming

Why competive programming?

- Problem solving using algorithms
- Think conceptually and practically about
  - Time complexity
  - Space complexity
  - Data structures
- Recognize different problem types
- Increase available knowledge of algorithms and programming skills
- Learn to think, communicate and discuss about
  - algorithmic problems
  - specific solutions to these problems
  - generic types of solutions

Example problem

Consider an algorithm that takes as input a positive integer n. Then, repeatedly, if n is even, it is divided by 2, and if n is odd, the algorithm multiplies it by 3 and adds 1. It stops after n has become equal to 1. For example, the sequence for $n = 3$ is:

$$3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

Input: The only input line contains an integer n with $1 \leq n \leq 1,000,000$.
Output: One line, containing the subsequent values of n during the execution of this algorithm, separated by a space.
Example input:

```
3
```

Example output:

```
3 10 5 16 8 4 2 1
```

Problem structure

- Problem description; a little story
- Usually at least one example
- Constraints on the variables
- Example input
- Example output

Problem structure

- Problem description; a little story
- Usually at least one example
- Constraints on the variables
- Example input
- Example output
- Usually, many more testcases than the examples are used to test a submitted solution.

Example solution

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

int main() {
    int n;
    cin >> n;
    cout << n;
    while (n != 1) {
        if (n % 2 == 0)
            n /= 2;
        else
            n = n * 3 + 1;
        cout << " " << n;
    } // while
    cout << "\n";
    return 0;
} // main
```

Example solution

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

int main() {
    int n;
    cin >> n;
    cout << n;
    while (n != 1) {
        if (n % 2 == 0)
            n /= 2;
        else
            n = n * 3 + 1;
        cout << " " << n;
    } // while
    cout << "\n";
    return 0;
} // main
```

What is wrong?

Example solution

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

int main() {
    int n;
    cin >> n;
    cout << n;
    while (n != 1) {
        if (n % 2 == 0)
            n /= 2;
        else
            n = n * 3 + 1;
        cout << " " << n;
    } // while
    cout << "\n";
    return 0;
} // main
```

What is wrong? `int n` should be `long long n`, as possibly $n >$ `INT_MAX`

Solution structure

- Usually, the first variable is the number of testcases t
- Then for each test case, read one or more variables
- You may need to store the input data
- Output typically goes on a new line for each testcase
- Be careful with extra whitespace ...

```
int main() {
    int t, n, m;
    cin >> t;
    while(t--) { // for each of the t testcases...
        cin >> n >> m; // read dimensions of the problem

        // do some computation here

        cout << "Your solution, however complex or simple." << endl;
    } // while
    return 0;
} // main
```

Realistic solution

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

int main() {
    int t;
    long long n;
    cin >> t;
    while(t--) {
        cin >> n;
        cout << n;
        while (n != 1) {
            if (n % 2 == 0)
                n /= 2;
            else
                n = n * 3 + 1;
            cout << " " << n;
        } // while
        cout << "\n";
    } // while
    return 0;
```

Testing (1)

in.txt

```
3
8
42
15
```

out.txt

```
8 4 2 1
42 21 64 32 16 8 4 2 1
15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
```

Testing (2)

```
takesfw@takes$ g++ -Wall -O2 mysolution.cpp
takesfw@takes$ ./a.out < in.txt
8 4 2 1
42 21 64 32 16 8 4 2 1
15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
takesfw@takes$
```

Testing (2)

```
takesfw@takes$ g++ -Wall -O2 mysolution.cpp
takesfw@takes$ ./a.out < in.txt
8 4 2 1
42 21 64 32 16 8 4 2 1
15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
takesfw@takes$

takesfw@takes$ ./a.out < in.txt > myout.txt
takesfw@takes$ diff myout.txt out.txt
takesfw@takes$
# (no output = no difference = correct on testcase)
```

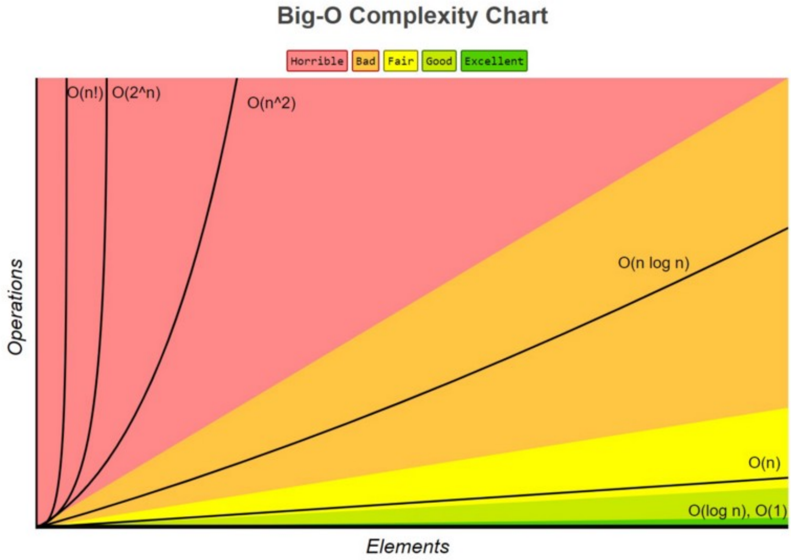
Understanding contraints

- Input size is given

  e.g., a puzzle on an array of length n where the goal is to find some element.

  What can you do if:

  - $n = 8$
  - $n = 100$
  - $n = 100,000$
  - $n = 10,000,000$



Source: `http://bigocheatsheet.com`

Sorting algorithms

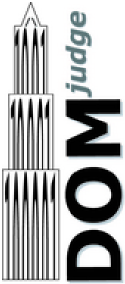
`https://www.youtube.com/watch?v=ZZuD6iUe3Pc`

Problem types

- Straightforward
- Simulation
- Greedy
- Brute-force
- Divide and conquer
- Searching
- Sorting
- Graph, network flow
- Dynamic programming
- String processing
- Geometry
- Mathematics

DOMjudge

- DOMjudge: software for running a programming contest
- Users are members of teams
- Teams can compete in contests
- Contests have an associated problemset
- A problemset contains multiple problems
- Each problem is of the form as discussed before

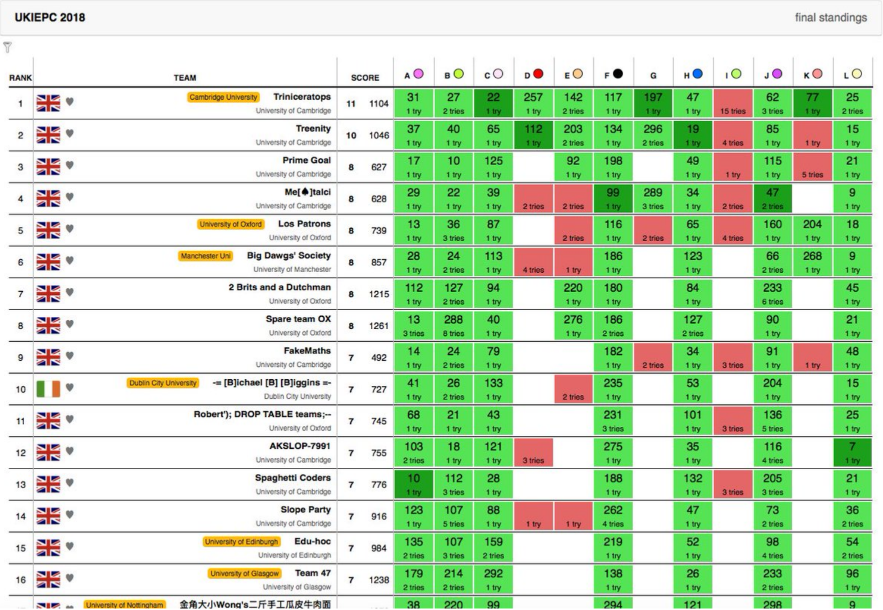


Possible results

- CORRECT: the submission passed all tests, problem solved!
- COMPILER-ERROR: you can catch this before submitting
- TIMELIMIT: use a less complex approach, check for infinite loops
- RUN-ERROR: seg-faults, divide by 0, tried to allocate too much memory, no “`return 0;`”” at the end, etc.
- NO-OUTPUT: your program did not generate any output or did not use the standard input/output
- OUTPUT-LIMIT: your program generated more output than the allowed limit (and was thus wrong)
- WRONG-ANSWER: go find the bug in your code . . .
- TOO-LATE: you submitted when contest had ended

Contest element

- Fixed amount of time; 5 hours
- Work in teams; 2 or 3 people
- Solve as many of the ca. 12 problems in the problemset as possible
- Work in teams, on one computer
- More problems solved is better
- Ties are determined by sum of time to CORRECT over all solved problems; penalty for WRONG ANSWER
- Nice: be the first to solve a problem
- Scoreboard of all teams



UKIEPC 2018 final standings

RANK	TEAM	SCORE		A	B	C	D	E	F	G	H	I	J	K	L
1	Cambridge University **Triniceratops** University of Cambridge	11	1104	31 1 try	27 2 tries	22 1 try	257 1 try	142 2 tries	117 1 try	197 1 try	47 1 try	15 tries	62 3 tries	77 1 try	25 2 tries
2	**Treenity** University of Cambridge	10	1046	37 1 try	40 1 try	65 1 try	112 1 try	203 2 tries	134 1 try	296 2 tries	19 1 try	4 tries	85 1 try	1 try	15 1 try
3	**Prime Goal** University of Cambridge	8	627	17 1 try	10 1 try	125 1 try		92 1 try	198 1 try		49 1 try	1 try	115 1 try	5 tries	21 1 try
4	**Me[♠]talci** University of Cambridge	8	628	29 1 try	22 1 try	39 1 try	2 tries	2 tries	99 1 try	289 3 tries	34 1 try	2 tries	47 2 tries		9 1 try
5	University of Oxford **Los Patrons** University of Oxford	8	739	13 1 try	36 3 tries	87 1 try		2 tries	116 1 try	2 tries	65 1 try	4 tries	160 1 try	204 1 try	18 1 try
6	Manchester Uni **Big Dawgs' Society** University of Manchester	8	857	28 1 try	24 2 tries	113 1 try	4 tries	1 try	186 1 try		123 1 try		66 2 tries	268 1 try	9 1 try
7	**2 Brits and a Dutchman** University of Oxford	8	1215	112 1 try	127 2 tries	94 1 try		220 1 try	180 1 try		84 1 try		233 6 tries		45 1 try
8	**Spare team OX** University of Oxford	8	1261	13 3 tries	288 8 tries	40 1 try		276 1 try	186 2 tries		127 2 tries		90 1 try		21 1 try
9	**FakeMaths** University of Cambridge	7	492	14 1 try	24 2 tries	79 1 try			182 1 try	2 tries	34 1 try	3 tries	91 1 try	1 try	48 1 try
10	Dublin City University **-= [B]ichael [B] [B]iggins =-** Dublin City University	7	727	41 1 try	26 2 tries	133 1 try		2 tries	235 1 try		53 1 try		204 1 try		15 1 try
11	**Robert'); DROP TABLE teams;--** University of Oxford	7	745	68 1 try	21 1 try	43 1 try			231 3 tries		101 1 try	3 tries	136 5 tries		25 1 try
12	**AKSLOP-7991** University of Cambridge	7	755	103 2 tries	18 1 try	121 1 try	3 tries		275 1 try		35 1 try		116 4 tries		7 1 try
13	**Spaghetti Coders** University of Cambridge	7	776	10 1 try	112 3 tries	28 1 try			188 1 try		132 1 try	3 tries	205 3 tries		21 1 try
14	**Slope Party** University of Cambridge	7	916	123 1 try	107 5 tries	88 1 try	1 try	1 try	262 4 tries		47 1 try		73 2 tries		36 2 tries
15	University of Edinburgh **Edu-hoc** University of Edinburgh	7	984	135 2 tries	107 3 tries	159 2 tries			219 1 try		52 1 try		98 4 tries		54 2 tries
16	University of Glasgow **Team 47** University of Glasgow	7	1238	179 2 tries	214 2 tries	292 1 try			138 1 try		26 1 try		233 2 tries		96 1 try
	University of Nottingham **金角大小Wong's二斤手工瓜皮牛肉面**			38	220	99			294		121		298		9

Programming contests

- Leids Kampioenschap Programmeren (LKP)
- Benelux Algorithm Programming Contest (BAPC)
- North-Western European Regional Contest (NWERC)
- International Collegiate Programming Contest (ICPC)
- Online: Topcoder, HackerRank, Codeforces, AtCoder, CodeChef, USACO, ICPC Live Archives . . .

Plexus Informatie Trefpunt voor Studenten

39
Sun
PELTOR
Fanta
Northwestern European Programming Contest
KTH - Royal Institute of Technology
Stockholm, 18-19 November 2006
acm International Collegiate Programming Contest
event sponsor
IBM

Lab session: Domjudge introduction

- From 10.15 to 11:00 in Snellius room 302/304
- Navigate to `http://domjudge.liacs.nl`
- Register an account
- Familiarize yourself with the domjudge team manual at `https://www.domjudge.org/docs/team-manual.pdf`
- Submit solutions to the three (toy example) assignments in C++
- Try to at least once get WRONG ANSWER and TIMELIMIT
- Finish by submitting a CORRECT solution to all three assignments (at least before next week's lecture)
- Sign up and play around with real problems at “ICPC Live Archive”: `https://icpcarchive.ecs.baylor.edu`

Credits

This course, in particular these slides, are largely based on:

- Antti Laaksonen, *Guide to Competitive Programming*, Springer, 2017.
- Steven Halim and Felix Halim, *Competitive Programming 3*, Lulu.com, 2013.

Where applicable, full credit for text, images, examples, etc. goes to the authors of these books.