ACM Lab – HS 2015

Prof. Dr. Angelika Steger

September 17th, 2015

Tutorial

- Tutorial: Thursday 14 – 16 (CAB G52)
- Revise and introduce algorithmic concepts
- Discuss exercises
- Answer questions

Tutorial

- Tutorial: Thursday 14 – 16 (CAB G52)
- Revise and introduce algorithmic concepts
- Discuss exercises
- Answer questions

Tutorial

- Tutorial: Thursday 14 – 16 (CAB G52)
- Revise and introduce algorithmic concepts
- Discuss exercises
- Answer questions

Tutorial

- Tutorial: Thursday 14 – 16 (CAB G52)
- Revise and introduce algorithmic concepts
- Discuss exercises
- Answer questions

Exercises

- Every week, we post ca. 3 problems.
- We have automated feedback by an online judge.
- The exercises will usually be discussed in the tutorials.
- You can discuss the exercises with your colleagues in the forum (no spoilers!).
- You are not required to solve the exercises (but it is very much recommended).

Exercises

- Every week, we post ca. 3 problems.
- We have automated feedback by an online judge.
- The exercises will usually be discussed in the tutorials.
- You can discuss the exercises with your colleagues in the forum (no spoilers!).
- You are not required to solve the exercises (but it is very much recommended).

Exercises

- Every week, we post ca. 3 problems.
- We have automated feedback by an online judge.
- The exercises will usually be discussed in the tutorials.
- You can discuss the exercises with your colleagues in the forum (no spoilers!).
- You are not required to solve the exercises (but it is very much recommended).

Exercises

- Every week, we post ca. 3 problems.
- We have automated feedback by an online judge.
- The exercises will usually be discussed in the tutorials.
- You can discuss the exercises with your colleagues in the forum (no spoilers!).
- You are not required to solve the exercises (but it is very much recommended).

Exercises

- Every week, we post ca. 3 problems.
- We have automated feedback by an online judge.
- The exercises will usually be discussed in the tutorials.
- You can discuss the exercises with your colleagues in the forum (no spoilers!).
- You are not required to solve the exercises (but it is very much recommended).

Exam / Grades

The grade is based only on the final exam:

- Length: 6 hours
- Takes place in a computer room in HG
- Submission/judging of programs is as in the semester, but there can be hidden testsets which you don't see.
- The exam is closed book, but you have access to the STL documentation.

Exam / Grades

The grade is based only on the final exam:

- Length: 6 hours
- Takes place in a computer room in HG
- Submission/judging of programs is as in the semester, but there can be hidden testsets which you don't see.
- The exam is closed book, but you have access to the STL documentation.

Exam / Grades

The grade is based only on the final exam:

- Length: 6 hours
- Takes place in a computer room in HG
- Submission/judging of programs is as in the semester, but there can be hidden testsets which you don't see.
- The exam is closed book, but you have access to the STL documentation.

Exam / Grades

The grade is based only on the final exam:

- Length: 6 hours
- Takes place in a computer room in HG
- Submission/judging of programs is as in the semester, but there can be hidden testsets which you don't see.
- The exam is closed book, but you have access to the STL documentation.

Course Web Site

- `https://moodle-app2.let.ethz.ch`
- Log in with your NETHZ-Account.
- Course material: exercises, slides, same documentation as in exam
- Submit solutions to judge and check the scoreboard
- Forum: discuss problems, get news

Course Web Site

- `https://moodle-app2.let.ethz.ch`
- Log in with your NETHZ-Account.
- Course material: exercises, slides, same documentation as in exam
- Submit solutions to judge and check the scoreboard
- Forum: discuss problems, get news

Course Web Site

- `https://moodle-app2.let.ethz.ch`
- Log in with your NETHZ-Account.
- Course material: exercises, slides, same documentation as in exam
- Submit solutions to judge and check the scoreboard
- Forum: discuss problems, get news

Course Web Site

- `https://moodle-app2.let.ethz.ch`
- Log in with your NETHZ-Account.
- Course material: exercises, slides, same documentation as in exam
- Submit solutions to judge and check the scoreboard
- Forum: discuss problems, get news

Course Web Site

- `https://moodle-app2.let.ethz.ch`
- Log in with your NETHZ-Account.
- Course material: exercises, slides, same documentation as in exam
- Submit solutions to judge and check the scoreboard
- Forum: discuss problems, get news

How to get help

If you cannot solve an exercise

- first, ask your colleagues in the forum (or in private),
- alternatively, wait until the next tutorial.

How to get help

If you cannot solve an exercise

- first, ask your colleagues in the forum (or in private),
- alternatively, wait until the next tutorial.

How to get help

If you cannot solve an exercise

- first, ask your colleagues in the forum (or in private),
- alternatively, wait until the next tutorial.

Useful algorithmic knowledge

You will need to know/learn basic algorithmic techniques:

- complete search/backtracking,
- greedy optimization,
- divide and conquer,
- dynamic programming,

and sometimes specific algorithms and data structures (MST, bridge finding, shortest paths, . . .).

- Will also be repeated in the tutorials.

Useful algorithmic knowledge

You will need to know/learn basic algorithmic techniques:

- complete search/backtracking,
- greedy optimization,
- divide and conquer,
- dynamic programming,

and sometimes specific algorithms and data structures (MST, bridge finding, shortest paths, . . .).

- Will also be repeated in the tutorials.

Useful algorithmic knowledge

You will need to know/learn basic algorithmic techniques:

- complete search/backtracking,
- greedy optimization,
- divide and conquer,
- dynamic programming,

and sometimes specific algorithms and data structures (MST, bridge finding, shortest paths, . . .).

- Will also be repeated in the tutorials.

Useful C++ stuff

You won't need the fancy features of C++. Look up how to

- do IO using `cin`/`cout`
- use `vector` (C++ dynamic arrays) from STL
- sort vectors (of structs) according to a given predicate
- (maybe) use stacks/priority queues from STL

and you will be fine.

Useful C++ stuff

General remarks:

- in g++ an `int` stores numbers up to $2^{31} - 1$. Use `long long` if you need larger numbers (up to $2^{63} - 1$).
- Use `double` instead of `float` for higher precision.
- Pass large objects (e.g. vectors) by reference:

```
// this copies the whole vector (!)
int f(vector<int> v) { ... }

// much faster
int f(const vector<int> &v) { ... }
```

- If you only use cin/cout (and only then), speed up IO by setting `std::ios::sync_with_stdio(false);`.

Useful C++ stuff

General remarks:

- in g++ an `int` stores numbers up to $2^{31} - 1$. Use `long long` if you need larger numbers (up to $2^{63} - 1$).
- Use `double` instead of `float` for higher precision.
- Pass large objects (e.g. vectors) by reference:

```
// this copies the whole vector (!)
int f(vector<int> v) { ... }

// much faster
int f(const vector<int> &v) { ... }
```

- If you only use cin/cout (and only then), speed up IO by setting `std::ios::sync_with_stdio(false);`.

Useful C++ stuff

General remarks:

- in g++ an `int` stores numbers up to $2^{31} - 1$. Use `long long` if you need larger numbers (up to $2^{63} - 1$).
- Use `double` instead of `float` for higher precision.
- Pass large objects (e.g. vectors) by reference:

```
// this copies the whole vector (!)
int f(vector<int> v) { ... }

// much faster
int f(const vector<int> &v) { ... }
```

- If you only use cin/cout (and only then), speed up IO by setting `std::ios::sync_with_stdio(false);`.

Useful C++ stuff

General remarks:

- in g++ an `int` stores numbers up to $2^{31} - 1$. Use `long long` if you need larger numbers (up to $2^{63} - 1$).
- Use `double` instead of `float` for higher precision.
- Pass large objects (e.g. vectors) by reference:

```cpp
// this copies the whole vector (!)
int f(vector<int> v) { ... }

// much faster
int f(const vector<int> &v) { ... }
```

- If you only use `cin/cout` (and only then), speed up IO by setting `std::ios::sync_with_stdio(false);`.

ACM Local Competition

If you like the type of problems presented in this lab, then sign up for the local ACM competition!

More info: `http://acm.vis.ethz.ch`, `acm@vis.ethz.ch`

Sample Problem: Hello World

Workflow how to solve exercises:

- Go to moodle and download exercise.
- Choose your favourite text editor and write a solution.
- Compile and test your solution.
  - `g++ solution.cpp -o solution`
- Submit the solution to the judge (Enrolment key: *acmlab2015*).

Sample Problem: Hello World

Workflow how to solve exercises:

- Go to moodle and download exercise.
- Choose your favourite text editor and write a solution.
- Compile and test your solution.
  - `g++ solution.cpp -o solution`
- Submit the solution to the judge (Enrolment key: *acmlab2015*).

Sample Problem: Hello World

Workflow how to solve exercises:

- Go to moodle and download exercise.
- Choose your favourite text editor and write a solution.
- Compile and test your solution.
  - `g++ solution.cpp -o solution`
- Submit the solution to the judge (Enrolment key: *acmlab2015*).

Sample Problem: Hello World

Workflow how to solve exercises:

- Go to moodle and download exercise.
- Choose your favourite text editor and write a solution.
- Compile and test your solution.
  - `g++ solution.cpp -o solution`
- Submit the solution to the judge (Enrolment key: *acmlab2015*).

Sample Problem: Hello World

Workflow how to solve exercises:

- Go to moodle and download exercise.
- Choose your favourite text editor and write a solution.
- Compile and test your solution.
  - `g++ solution.cpp -o solution`
- Submit the solution to the judge (Enrolment key: *acmlab2015*).

Judge Results

- correct: your solution is correct :)
- timelimit: solution is too slow
- wrong-answer: the program outputs a wrong answer
- assertion-failure SIGABRT: memory screwup or assertion failure
- segmentation-fault SIGSEGV: memory screwup
- run-error: nonzero exit status (main should return 0)
- forbidden: bad syscall or other safety