Competitive Programming
ICPC SWERC Training

Jill-Jênn Vie

First class

This course is about algorithmic problem solving

- You don’t know an algorithm unless you’ve implemented it (without any bugs).
- Combining simple techniques to solve bigger problems

ICPC SWERC, 27–28 January 2024, Sorbonne Université, Paris

- 10 problems
- 5 hours
- 3 people
- 1 keyboard

`swerc.eu`

Probably 3 teams per university/school.

Judges

Input

```
9 10
##########
.....#...#
####.###.#
#..#.#...#
#..#.#.###
###..#.#.#
#.#.####.#
#........#
########.#
```

Output

```
##########
XXXXX#...#
####X###.#
#..#X#...#
#..#X#.###
###XX#.#X#
#X#X####X#
#XXXXXXXX#
########X#
```

```
python laby.py < laby.in > laby.out  # Python

make laby
./laby < laby.in > laby.out  # C++
```

Pathfinding in graphs

```
todo = SomeDataStructure()
Put start in todo
While todo is not empty
    Get node from todo
    For each neighbor of node
        Add neighbor to todo if not visited yet
```

According to the data structure, the complexity and algorithm can be different

- Stack $\rightarrow$ what?
- Queue $\rightarrow$ what?
- Heap $\rightarrow$ what?
- ? $\rightarrow$ graph with edges 0 and 1

Actually, when we mark nodes can have an impact on the complexity too

Schedule

- Lessons are 14:00-17:00 on Thursdays
- November: Team selection and SWERC registration deadline
- 27–28 January 2024: SWERC

Outline

1. Intro
2. Shortest paths
3. DP: Dynamic Programming
4. Matching & flows
5. Text algorithms (suffix arrays)
6. Advanced DP
7. Maths: Arithmetics, Combinatorics and Linear algebra
8. Dynamic data structures (segment trees)
9. Geometry & sweep line
10. Ad-hoc problems
11. Final tricks
12. Team selection

Advice

- It is a team competition
  - You should learn to debug each other's code
- Identify asap the easy problems
- Avoid presentation errors (missing spaces, etc.)
- Think about extreme cases (empty graph)
- Think about out-of-bounds (sometimes it is better to allocate more memory)
  - E.g. integer bounds: you may need an `unsigned long long int` (`%lld`)
- Evaluate the complexity before implementing it
  - Sometimes it is good to code the naive solution just to debug a better one
- If there are several instances, make sure everything is cleared, notably global variables

More advice

- Highlight the important points of the statement (bounds).
  Is it a DP? A graph problem?
- Think about corner cases / edge cases for the rest of your team
- Learn to solve problems on paper
- It is a team competition
  - If a submission fails, print your code and debug it by hand in order to free the keyboard for someone else

Objectives for today

- Set up an account on Kattis and tell me your username
- Configure VSCode/VSCodium
- Read and solve a few problems using X notebook