

Meeting link :

<https://learn.zoom.us/j/61854223791?pwd=MG9vUFpqSUozTCs3TXk4OE9yTEszdz09>

Meeting ID: 618 5422 3791

Passcode: Xtreme15.0

Algorithmic Programming Competitions

2021-Jul-11

IEEE Xtreme 15.0 Awareness Session

University of Vocational Technology

Gihan Jayatilaka

Acknowledgement: These slides are derivative from [Suren Sritharan](#)'s slides.

WHAT is algorithmic programming

Algorithmic + Programming

WHAT is algorithmic programming

Algorithmic Programming (Broad meaning)

Solving particular **problem** using an **algorithm** which is implemented through a **computer program**

Algorithmic programming

- Almost every programming task could be identified as algorithmic programming. However, what we are interested in is **algorithm intensive programming**.
- Major considerations are **correctness, efficiency (space and time)** and scalability.
- Examples
 - Decision/planning/scheduling/optimization
 - ML/AI/DL
 - Vision /signal-processing

Algorithmic programming

- Almost every programming task could be identified as algorithmic programming. However, what we are interested in is **algorithm intensive programming**.
- Major considerations are **correctness, efficiency (space and time) and scalability**.
- Examples
 - Decision/planning/scheduling/optimization
 - ML/AI/DL
 - Vision /signal-processing

Algorithmic programming

- Almost ever programming task could be identified as algorithmic programming. However, what we are interested in is **algorithm intensive programming**.
- Major considerations are **correctness, efficiency (space and time)** and scalability.
- Examples
 - Decision/planning/scheduling/optimization
 - ML/AI/DL
 - Vision /signal-processing

Competitive Algorithmic Programming

- Very well defined problems (input/output format)
- Constraints on inputs and outputs.
- Marking scheme.
- Resource limits (CPU time, RAM megabytes)
- Held on a platform (eg: HackerRank, codeforces)
- Rules, regulations and awards.
- Limitations on programming languages and libraries.

Constraints and resource limitations

3-divisible Pairs

Time limit: 1000 ms
Memory limit: 128 MB

You are given an array of N positive integers. Compute the number of pairs of integers in the array that have the sum divisible by 3.

Standard input

The first line contains one integer N .

The second line contains N integers representing the elements of the array.

Standard output

Output a single number representing the number of pairs that have the sum divisible by 3.

Constraints and notes

- $1 \leq N \leq 10^5$
- The elements of the array are integers between 1 and 10^5 .

Input

Output

5
1 4 2 3 3

3

4
3 3 3 6

6

8
1 1 7 2 11 3 6 9

9

Problem

**Resource
limits**

Constraints

Examples

**If you
complete
the task
you get
points**

Competitive Algorithmic Programming

- Very well defined problems (input/output format)
- Constraints on inputs and outputs.
- **Marking scheme.**
- Resource limits (CPU time, RAM megabytes)
- Held on a platform (eg: HackerRank, codeforces)
- Rules, regulations and awards.
- Limitations on programming languages and libraries.

Competitive Algorithmic Programming

- Very well defined problems (input/output format)
- Constraints on inputs and outputs.
- Marking scheme.
- Resource limits (CPU time, RAM megabytes)
- Held on a platform (eg: HackerRank, codeforces)
- Rules, regulations and awards.
- Limitations on programming languages and libraries.

Competitive Algorithmic Programming

- Very well defined problems (input/output format)
- Constraints on inputs and outputs.
- Marking scheme.
- Resource limits (CPU time, RAM megabytes)
- Held on a platform (eg: HackerRank, codeforces)
- Rules, regulations and awards.
- Limitations on programming languages and libraries.

Popular platforms

HackerRank.com

CodeForces.com

CSacademy.com

ProjectEuler.com

CodeChef.com

TopCoder.com



Competitive Algorithmic Programming

- Very well defined problems (input/output format)
- Constraints on inputs and outputs.
- Marking scheme.
- Resource limits (CPU time, RAM megabytes)
- Held on a platform (eg: HackerRank, codeforces)
- Rules, regulations and awards.
- Limitations on programming languages and libraries.

Competitive Algorithmic Programming

- Very well defined problems (input/output format)
- Constraints on inputs and outputs.
- Marking scheme.
- Resource limits (CPU time, RAM megabytes)
- Held on a platform (eg: HackerRank, codeforces)
- Rules, regulations and awards.
- Limitations on programming languages and libraries.

What does it take to
win?

Expectation

Programming 50%

Mathematics 30%

Creativity 20%

What does it take to
win?

REALITY

Programming 25%

Mathematics 15%

Creativity 10%

Practice 50%

What should beginners do?

- Pick any programming language and learn the basics (input, output, if/for/else, functions..)
 - C++, Python, Java are popular choices.
- Practise standard questions on a platform like HackerRank or CSacademy.
- Learn basic data structures and algorithms.

What should beginners do?

- Pick any programming language and learn the basics (input, output, if/for/else, functions..)
 - C++, Python, Java are popular choices.
- Practise standard questions on a platform like HackerRank or CSacademy.
- Learn basic data structures and algorithms.

What should beginners do?

- Pick any programming language and learn the basics (input, output, if/for/else, functions..)
 - C++, Python, Java are popular choices.
- Practise standard questions on a platform like HackerRank or CSacademy.
- Learn basic data structures and algorithms.

What should intermediates do?

- Go to local competitions (IEEEExtreme, ACEScoders, MoraXtreme...)
- When you get a difficult question, try it for at least a few hours.
- Learn advanced algorithms.
- Start developing your own library.
 - You are allowed to take this code to competitions.
- Ask for solutions from competitions you attend to.
- Practise as a team.
 - You need to develop a way to communicate fast.
 - Co-dependent skills.

Local competitions

ACES coders

MoraExtreme



What should intermediates do?

- Go to local competitions (IEEEExtreme, ACEScoders, MoraXtreme...)
- When you get a difficult question, try it for at least a few hours.
- Learn advanced algorithms.
- Start developing your own library.
 - You are allowed to take this code to competitions.
- Ask for solutions from competitions you attend to.
- Practise as a team
 - You need to develop a way to communicate fast.
 - Co-dependent skills.

What should intermediates do?

- Go to local competitions (IEEEExtreme, ACEScoders, MoraXtreme...)
- When you get a difficult question, try it for at least a few hours.
- Learn advanced algorithms.
- Start developing your own library.
 - You are allowed to take this code to competitions.
- Ask for solutions from competitions you attend to.
- Practise as a team
 - You need to develop a way to communicate fast.
 - Co-dependent skills.

What should intermediates do?

- Go to local competitions (IEEEExtreme, ACEScoders, MoraXtreme...)
- When you get a difficult question, try it for at least a few hours.
- Learn advanced algorithms.
- Start developing your own library.
 - You are allowed to take this code to competitions.
- Ask for solutions from competitions you attend to.
- Practise as a **team**.
 - You need to develop a way to communicate fast.
 - Co-dependent skills.

What should experts do?

- Learn C++ to the depth.
- Do weekly competitions on codeforces.
- Take part in global competitions (Google CodeJam, Facebook HackerCup)
- **Upsolve** competition problems after the end of the competition.

What should experts do?

- Learn C++ to the depth.
- Do weekly competitions on codeforces.
- Take part in global competitions (Google CodeJam, Facebook HackerCup)
- **Upsolve** competition problems after the end of the competition.

What should experts do?

- Learn C++ to the depth.
- Do weekly competitions on codeforces.
- Take part in global competitions (Google CodeJam, Facebook HackerCup)
- **Upsolve** competition problems after the end of the competition.

Worldwide competitions

IEEEExtreme

Google codeJam

FaceBook HackerCup

ACM ICPC



What should experts do?

- Learn C++ to the depth.
- Do weekly competitions on codeforces.
- Take part in global competitions (Google CodeJam, Facebook HackerCup)
- **Upsolve** competition problems after the end of the competition.

Why do competitive programming?

- Algorithms are the most beautiful/elegant/pure part of CS.
- Looks very good on your profile
 - Analytical / problem solving / communication skills.
 - because this is a straightforward comparison between participants.
- Algorithm knowledge is tested on interviews by major companies (eg: this is the only thing tested by FAANG).
- You can work on “interesting roles” beyond generic software engineering.

Why do competitive programming?

- Algorithms are the most beautiful/elegant/pure part of CS.
- Looks very good on your profile
 - Analytical / problem solving / communication skills.
 - because this is a straightforward comparison between participants.
- Algorithm knowledge is tested on interviews by major companies (eg: this is the only thing tested by FAANG).
- You can work on “interesting roles” beyond generic software engineering.

Why do competitive programming?

- Algorithms are the most beautiful/elegant/pure part of CS.
- Looks very good on your profile
 - Analytical / problem solving / communication skills.
 - because this is a straightforward comparison between participants.
- Algorithm knowledge is tested on interviews by major companies (eg: this is the only thing tested by FAANG).
- You can work on “interesting roles” beyond generic software engineering.

Why do competitive programming?

- Algorithms are the most beautiful/elegant/pure part of CS.
- Looks very good on your profile
 - Analytical / problem solving / communication skills.
 - because this is a straightforward comparison between participants.
- Algorithm knowledge is tested on interviews by major companies (eg: this is the only thing tested by FAANG).
- You can work on “interesting roles” beyond generic software engineering.

What can I do next with the knowledge?

- Get a software engineering job in a top company.
- Go into algorithm intensive fields.
 - Highly scalable system design.
 - OS design, compiler design.
 - Graphics/vision/signal processing.
 - ML/AI
- Go into algorithms research by going for MS/PhD.

What can I do next with the knowledge?

- Get a software engineering job in a top company.
- Go into algorithm intensive fields.
 - Highly scalable system design.
 - OS design, compiler design.
 - Graphics/vision/signal processing.
 - ML/AI
- Go into algorithms research by going for MS/PhD.

What can I do next with the knowledge?

- Get a software engineering job in a top company.
- Go into algorithm intensive fields.
 - Highly scalable system design.
 - OS design, compiler design.
 - Graphics/vision/signal processing.
 - ML/AI
- Go into algorithms research by going for MS/PhD.

Some issues

- Some **mid range** companies doesn't value algorithmic skills. They seek people who are good in different technologies/frameworks.
- Even though you get into a **top company** through algorithms knowledge and start working as a software engineer, your work will not be primarily on algorithm development.
- Reaching the world top in this field is extremely competitive (however, this is true for any field)

Some issues

- Some **mid range** companies doesn't value algorithmic skills. They seek people who are good in different technologies/frameworks.
- Even though you get into a **top company** through algorithms knowledge and start working as a software engineer, your work will not be primarily on algorithm development.
- Reaching the world top in this field is extremely competitive (however, this is true for any field)

Some issues

- Some **mid range** companies doesn't value algorithmic skills. They seek people who are good in different technologies/frameworks.
- Even though you get into a **top company** through algorithms knowledge and start working as a software engineer, your work will not be primarily on algorithm development.
- Reaching the world top in this field is extremely competitive (however, this is true for any field)

IEEEXtreme

- 3 person group.
- 24 questions per 24 hours. Usually 5:30am - 5:30 am. **[some tips]**
- Held on CSacademy.com
 - Create a profile and familiarize with the platform now.
- You need a mentor (typically this an academic staff member).
 - Student teams can reach out to individual mentors or the student branch will collect the team information and find mentors for them.
- All 3 students need to have IEEE student memberships.

IEEEXtreme

- 3 person group.
- 24 questions per 24 hours. Usually 5:30am - 5:30 am.
- Held on CSacademy.com
 - Create a profile and familiarize with the platform now.
- You need a mentor (typically this an academic staff member).
 - Student teams can reach out to individual mentors or the student branch will collect the team information and find mentors for them.
- All 3 students need to have IEEE student memberships.

IEEEXtreme

- 3 person group.
- 24 questions per 24 hours. Usually 5:30am - 5:30 am.
- Held on CSacademy.com
 - Create a profile and familiarize with the platform now.
- You need a mentor (typically this an academic staff member).
 - Student teams can reach out to individual mentors or the student branch will collect the team information and find mentors for them.
- All 3 students need to have IEEE student memberships.

IEEEXtreme

- 3 person group.
- 24 questions per 24 hours. Usually 5:30am - 5:30 am.
- Held on CSacademy.com
 - Create a profile and familiarize with the platform now.
- You need a mentor (typically this an academic staff member).
 - Student teams can reach out to individual mentors or the student branch will collect the team information and find mentors for them.
- All 3 students need to have IEEE student memberships.

IEEE Student membership

- Costs 27 USD per year. (In my opinion, this is on the expensive end).
- Some tips to overcome this issue.
 - Register late for the year (close to the competition). You get one of these two advantages (ask the student branch about this).
 - Half rate 13.50 USD registration fee for the remaining few months of the year.
 - Two years registration for 27 USD.
 - Finding sponsors, organizing an internal event and giving free memberships to top 1/2/3/5 teams.

IEEE Student membership

- Costs 27 USD per year. (In my opinion, this is on the expensive end).
- Some tips to overcome this issue.
 - Register late for the year (close to the competition). You get one of these two advantages (ask the student branch about this).
 - Half rate 13.50 USD registration fee for the remaining few months of the year.
 - Two years registration for 27 USD.
 - Finding sponsors, organizing an internal event and giving free memberships to top 1/2/3/5 teams.

IEEE Student membership

- Costs 27 USD per year. (In my opinion, this is on the expensive end).
- Some tips to overcome this issue.
 - Register late for the year (close to the competition). You get one of these two advantages (ask the student branch about this).
 - Half rate 13.50 USD registration fee for the remaining few months of the year.
 - Two years registration for 27 USD.
 - Finding sponsors, organizing an internal event and giving free memberships to top 1/2/3/5 teams.

FAQ

- Expert in *programming* ?
 - NO
- Expert in *algorithms and datastructures*?
 - NO
- Advanced *mathematical knowledge*?
 - NO (Preliminary knowledge)
- What do you need?
 - **Problem solving** skills and **practice**

Thanks!

- Indrajith Ekanayake from IEEE, SL for inviting me.
- Heshan Jayasinghe from IEEE, Univotec for inviting me.
- IEEE Univotec for organizing the event.
- All other speakers and moderators.
- All attendees (dean, academic staff and students) for listening.

Apology: to IEEE Univotec if I was not enthusiastic about the whole poster stuff.

Summary | Thanks for listening! Q+A

- Algorithmic/competitive programming.
- Programming+Math+Creativity+Practise.
- Correctness, efficiency and solving in time.
- Resource limitations (RAM,CPU), constraints.
- Team competitions ---> communicating complicated logic.
- Advantages for the future (jobs, research, higher education)
- Some issues
- **UPSOLVE after the competitions!!!!**

Some additional resources: <https://gihan.me/resources>

If you have any questions: <https://gihan.me/contact> (email preferred)