香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Introduction to Competitive Programming

Ethen Yuen {ethening}

2026-02-14

What is Competitive Programming?

"Competitions where contestants write computer programs to solve a set of well-defined problems within a limited amount of time. The judging is based on correctness and time spent."

What is Competitive Programming?

"Competitions where contestants **write computer programs** to solve a set of well-defined problems within a limited amount of time. The judging is based on correctness and time spent."

Common misconceptions: Competitive programming is all about **writing code**.

What is Competitive Programming?

> "Competitions where contestants write computer programs to **solve a set of well-defined problems** within a limited amount of time. The judging is based on correctness and time spent."

Actually, competitive programming focuses mainly on **problem solving**, and to represent your solution in a formal way (that is, by writing code).

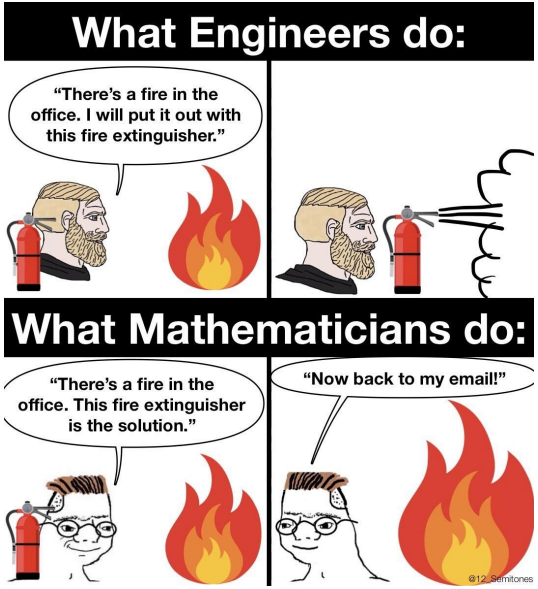
- In HKOI Training, we will teach you both **1. problem solving** and **2. coding effectively**.

Competitive Programming VS Math Olympiad

"... solve a set of well-defined problems ..."

Competitive Programming VS Math Olympiad

While they do share a lot of common problem solving techniques, they care about different parts of the problem generally.



https://www.reddit.com/r/mathmemes/comments/p2u4xh/who_wants_to_be_applicable/

Competitive Programming VS Math Olympiad

Mathematics Olympiad

- Prove the existence of ...
- Prove the characteristics of ...
- Find the greatest k such that ...

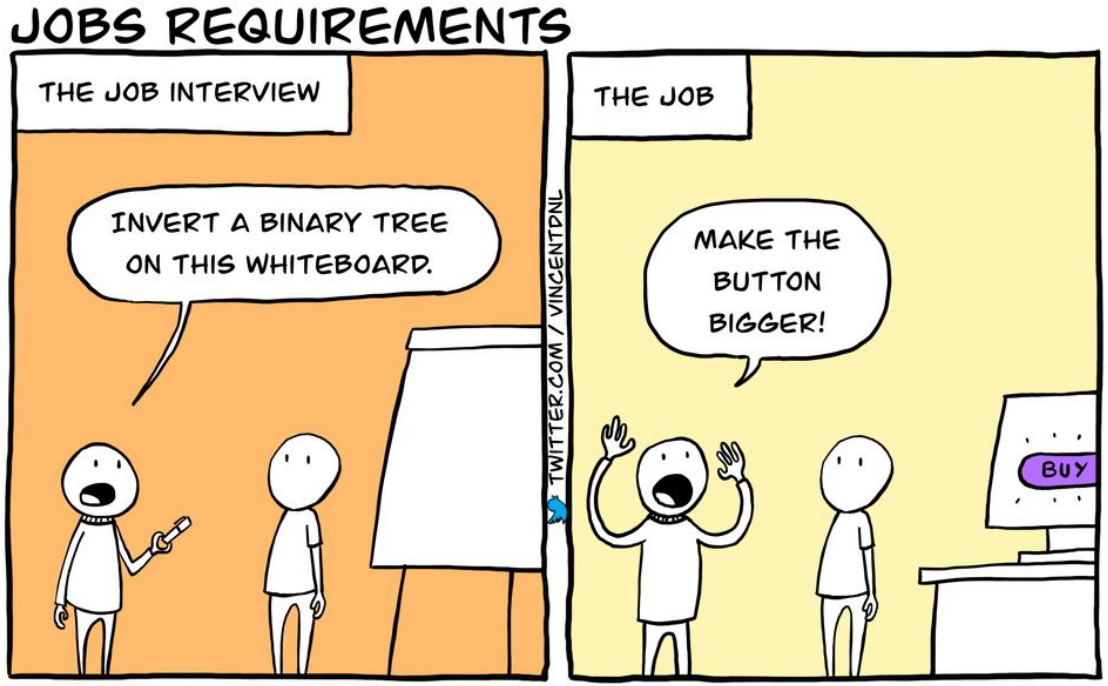
Competitive Programming

- Find the number of ... after n days, where $1 \le n \le 10^9$. The program should **terminated within 1.00 second**.
- Given input files of 2D points, connect the points using the least number of line segment, scoring is determined by **how many lines you use**.

We also can have solution in competitive programming *(that doesn't make sense in Math Olympiad)* like:

- Let's randomly pick 100 candidates and see if they are the answer, else return impossible

Competitive Programming VS Real-world Programming



https://www.reddit.com/r/ProgrammerHumor/comments/i49h96/jobs_requirements/

Competitive Programming VS Real-world Programming

"solve a set of well-defined problems"

- **Languages**
  mostly C++ (because it's fast!)
- **Applications**
  not really for real-life use, and you won't have $N=10^5$ friends
- **Knowledge**
  data structures, algorithms, maybe some maths, basic coding
- **What we care**
  time efficiency (99.99% of the time), space efficiency, ...

"well-defined problems" (?)

- **Languages**
  Any (Python, JavaScript, Java, C, C++, Assembly, ...)
- **Applications**
  websites, softwares, mobiles apps, robots, OSs, bots, ...
- **Knowledge**
  domain-based, e.g. web/app frameworks, game engines, ...
- **What we care**
  code maintainability, readability, sometimes efficiency, ...

Why Competitive Programming?

- Strengthen your problem solving / logical thinking skills
- Maybe strengthen your coding skills
- Learn more stuffs related to Computer Science

- **Learn how to be determined and strong to aim for the top**

Why Competitive Programming (Sport Programming)?

- **Learn how to be determined and strong to aim for the top**

- Similar to sports activity
  - Always trying to be beat your personal best
  - Learn from your peers and compete with them
  - Constantly hold yourself to a high standard
  - Learn to cope with failure
  - Become the **best of the best** to be a **HK representative**

Why Competitive Programming (Sport Programming)?

Great blog to read: [My winning theory in IOI 2018 & 2019 — Why I won 2 golds in IOI]

> ***"The competition results will not be determined until the last minutes, the last second, and even the last 0.1 second. The person who never gives up until the last moment will win."***

E869120

(The author do 10-hours long virtual contest to prepare for 5-hours IOI and do marathon *(actual running)* to train physical and mental strength.)

3 years ago, # ^ | ☆ +79

Around 7 hours per day. This is an example of my schedule (weekends) around 2 month before IOI.

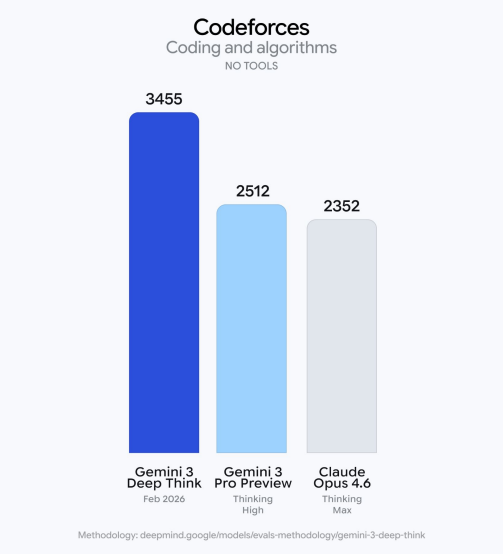
```
04:45 Wake up
05:00 Breakfast / warm up (jogging etc.)
06:00 - 11:00 IOI-like Virtual Day1 (5 hours)
11:10 - 11:40 Lunch
11:45 - 12:45 Review of IOI-like Virtual Day1
13:00 - 18:00 IOI-like Virtual Day2 (5 hours)
18:00 - 19:00 Review of IOI-like Virtual Day2
19:00 - 20:00 Dinner
20:00 - 22:00 Codeforces Virtual (Div.1)
22:15 Go to bed
```

This is an example of contest-based days. There are also some review-based days. In review-based day, I will review some of virtual contest that I did before.

→ Reply

Why Competitive Programming (Sport Programming)?

- "Is it still worth learning coding and competitive programming?"



- Even putting all the tangible benefits aside. It's still quite fun to challenge yourself intellectually!
- I hope you can find the enjoyment within competitive programming.

Starting from HKOI

What's next after HKOI Final Event?

Trainings & Competitions!

More details on https://hkoi.org/.

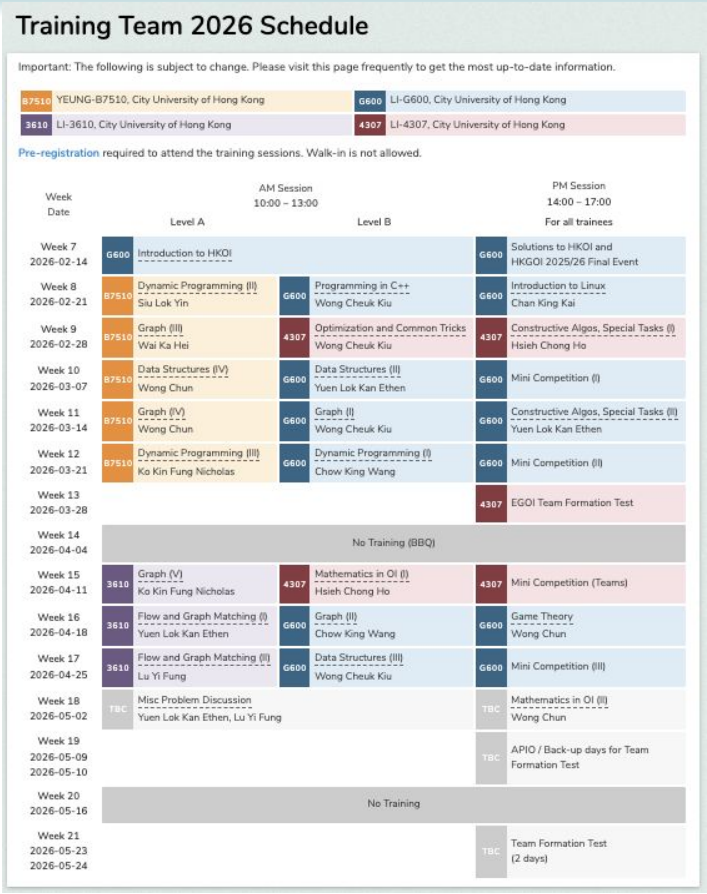
Regular Training

Lecture

- Topics on different algorithmic skills (and more)
  - Data Structures
  - Graph Theory
  - Dynamic Programming
  - Maths
  - Ad-hoc Tricks
  - Computer Science related knowledge

Minicomp

- 4 Mini Competitions to get you prepared for the Team Formation Test.



Training Team 2026 Schedule

Important: The following is subject to change. Please visit this page frequently to get the most up-to-date information.

B7510 YEUNG-B7510, City University of Hong Kong
G600 LI-G600, City University of Hong Kong
3610 LI-3610, City University of Hong Kong
4307 LI-4307, City University of Hong Kong

Pre-registration required to attend the training sessions. Walk-in is not allowed.

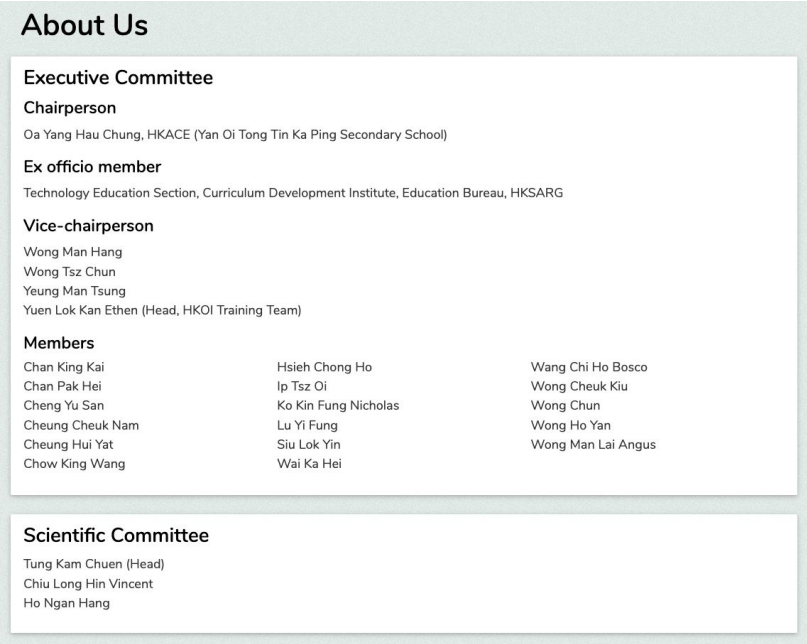
Week / Date	AM Session 10:00 – 13:00 Level A	AM Session 10:00 – 13:00 Level B	PM Session 14:00 – 17:00 For all trainees
Week 7 2026-02-14	G600 Introduction to HKOI		G600 Solutions to HKOI and HKGOI 2025/26 Final Event
Week 8 2026-02-21	B7510 Dynamic Programming (II) Siu Lok Yin	G600 Programming in C++ Wong Cheuk Kiu	G600 Introduction to Linux Chan King Kai
Week 9 2026-02-28	B7510 Graph (III) Wai Ka Hei	4307 Optimization and Common Tricks Wong Cheuk Kiu	4307 Constructive Algos, Special Tasks (I) Hsieh Chong Ho
Week 10 2026-03-07	B7510 Data Structures (IV) Wong Chun	G600 Data Structures (II) Yuen Lok Kan Ethen	G600 Mini Competition (I)
Week 11 2026-03-14	B7510 Graph (IV) Wong Chun	G600 Graph (I) Wong Cheuk Kiu	G600 Constructive Algos, Special Tasks (II) Yuen Lok Kan Ethen
Week 12 2026-03-21	B7510 Dynamic Programming (III) Ko Kin Fung Nicholas	G600 Dynamic Programming (I) Chow King Wang	G600 Mini Competition (II)
Week 13 2026-03-28			4307 EGOI Team Formation Test
Week 14 2026-04-04	No Training (BBQ)		
Week 15 2026-04-11	3610 Graph (V) Ko Kin Fung Nicholas	4307 Mathematics in OI (I) Hsieh Chong Ho	4307 Mini Competition (Teams)
Week 16 2026-04-18	3610 Flow and Graph Matching (I) Yuen Lok Kan Ethen	G600 Graph (II) Chow King Wang	G600 Game Theory Wong Chun
Week 17 2026-04-25	3610 Flow and Graph Matching (II) Lu Yi Fung	G600 Data Structures (III) Wong Cheuk Kiu	G600 Mini Competition (III)
Week 18 2026-05-02	TBC Misc Problem Discussion Yuen Lok Kan Ethen, Lu Yi Fung		TBC Mathematics in OI (II) Wong Chun
Week 19 2026-05-09 2026-05-10			TBC APIO / Back-up days for Team Formation Test
Week 20 2026-05-16	No Training		
Week 21 2026-05-23 2026-05-24			TBC Team Formation Test (2 days)

Regular Training

Lecturer Lineup

- **Trainers** who are previously HKOI trainees!
  - Undergraduates, postgrads and software engineers in different fields
  - Ex-HK representatives
  - Still actively competing in contests

- Feel free to ask us any questions about the lectures or tasks or get general tips on competitive programming.



About Us

Executive Committee

Chairperson

Oa Yang Hau Chung, HKACE (Yan Oi Tong Tin Ka Ping Secondary School)

Ex officio member

Technology Education Section, Curriculum Development Institute, Education Bureau, HKSARG

Vice-chairperson

Wong Man Hang
Wong Tsz Chun
Yeung Man Tsung
Yuen Lok Kan Ethen (Head, HKOI Training Team)

Members

Chan King Kai
Chan Pak Hei
Cheng Yu San
Cheung Cheuk Nam
Cheung Hui Yat
Chow King Wang
Hsieh Chong Ho
Ip Tsz Oi
Ko Kin Fung Nicholas
Lu Yi Fung
Siu Lok Yin
Wai Ka Hei
Wang Chi Ho Bosco
Wong Cheuk Kiu
Wong Chun
Wong Ho Yan
Wong Man Lai Angus

Scientific Committee

Tung Kam Chuen (Head)
Chiu Long Hin Vincent
Ho Ngan Hang

https://hkoi.org/en/about-us/

Regular Training

Structure

- Topics are classified into 4 progressive levels
- Practice tasks for each level for self-learning and lecture-use.

Level A Training

Level B Training

Level C HKOI

Level D HKDSE

Regular Training

Structure

- Foundation topics (mostly Level C & D) will be taught in video-lectures format.
  - Recursion, Divide and Conquer Part 1 / HKOI
  - Searching and Sorting Part 1 / HKOI
  - Data Structures (I) Part 1 / HKOI
  - Greedy Algorithms Part 1 / HKOI
  - More to come...
  - For new training team members and non-trainees, you are strongly suggested to learn these topics on your own.

Training Team 2026 Schedule

Important: The following is subject to change. Please visit this page frequently to get the most up-to-date information.

B7510 YEUNG-B7510, City University of Hong Kong
G600 LI-G600, City University of Hong Kong
3610 LI-3610, City University of Hong Kong
4307 LI-4307, City University of Hong Kong

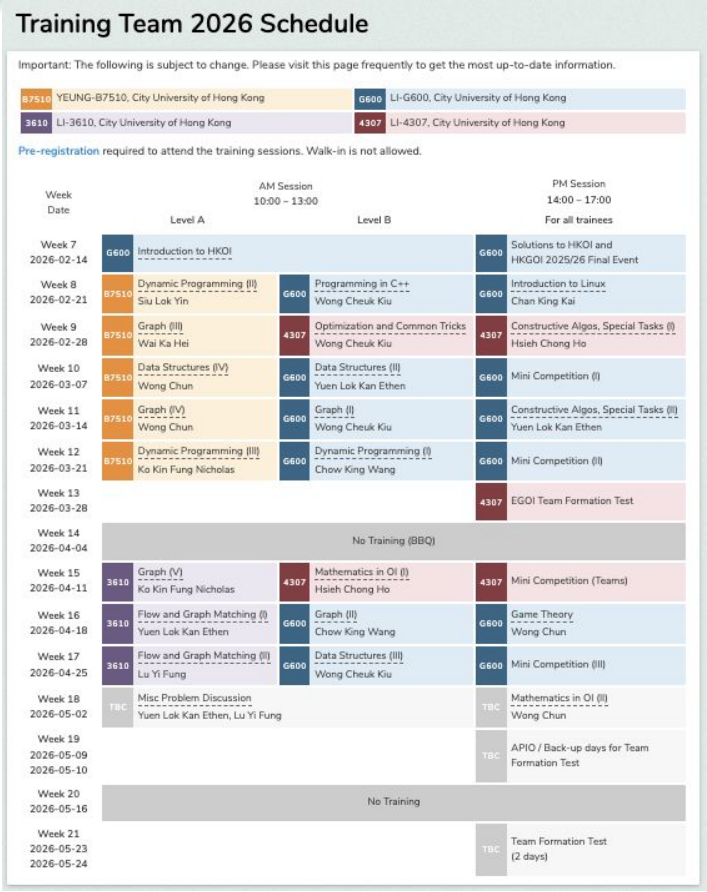
Pre-registration required to attend the training sessions. Walk-in is not allowed.

Week / Date	AM Session 10:00 – 13:00 Level A	AM Session 10:00 – 13:00 Level B	PM Session 14:00 – 17:00 For all trainees
Week 7 2026-02-14	G600 Introduction to HKOI		G600 Solutions to HKOI and HKGOI 2025/26 Final Event
Week 8 2026-02-21	B7510 Dynamic Programming (II) Siu Lok Yin	G600 Programming in C++ Wong Cheuk Kiu	G600 Introduction to Linux Chan King Kai
Week 9 2026-02-28	B7510 Graph (III) Wai Ka Hei	4307 Optimization and Common Tricks Wong Cheuk Kiu	4307 Constructive Algos, Special Tasks (I) Hsieh Chong Ho
Week 10 2026-03-07	B7510 Data Structures (IV) Wong Chun	G600 Data Structures (II) Yuen Lok Kan Ethen	G600 Mini Competition (I)
Week 11 2026-03-14	B7510 Graph (IV) Wong Chun	G600 Graph (I) Wong Cheuk Kiu	G600 Constructive Algos, Special Tasks (II) Yuen Lok Kan Ethen
Week 12 2026-03-21	B7510 Dynamic Programming (III) Ko Kin Fung Nicholas	G600 Dynamic Programming (I) Chow King Wang	G600 Mini Competition (II)
Week 13 2026-03-28			4307 EGOI Team Formation Test
Week 14 2026-04-04	No Training (BBQ)		
Week 15 2026-04-11	3610 Graph (V) Ko Kin Fung Nicholas	4307 Mathematics in OI (I) Hsieh Chong Ho	4307 Mini Competition (Teams)
Week 16 2026-04-18	3610 Flow and Graph Matching (I) Yuen Lok Kan Ethen	G600 Graph (II) Chow King Wang	G600 Game Theory Wong Chun
Week 17 2026-04-25	3610 Flow and Graph Matching (II) Lu Yi Fung	G600 Data Structures (III) Wong Cheuk Kiu	G600 Mini Competition (III)
Week 18 2026-05-02	TBC Misc Problem Discussion Yuen Lok Kan Ethen, Lu Yi Fung		TBC Mathematics in OI (II) Wong Chun
Week 19 2026-05-09 2026-05-10			TBC APIO / Back-up days for Team Formation Test
Week 20 2026-05-16	No Training		
Week 21 2026-05-23 2026-05-24			TBC Team Formation Test (2 days)

Regular Training

Structure

- AM Session: divided into level A & level B
- **Level B**
  - For first-year training team students
  - Fundamental algorithms
- **Level A**
  - For experienced students
  - Advanced topics
- **Entry Criteria** for Level A (any one)
  - Received HKOI Training Team 2025 Certificate of Attendance
  - Participated in HKOI Training Camp 2025.
  - Received Gold Medal award in HKOI 2025/26.
  - Solved 200 or more tasks on HKOI Online Judge.



Training Team 2026 Schedule

Important: The following is subject to change. Please visit this page frequently to get the most up-to-date information.

B7510 YEUNG-B7510, City University of Hong Kong
G600 LI-G600, City University of Hong Kong
3610 LI-3610, City University of Hong Kong
4307 LI-4307, City University of Hong Kong

Pre-registration required to attend the training sessions. Walk-in is not allowed.

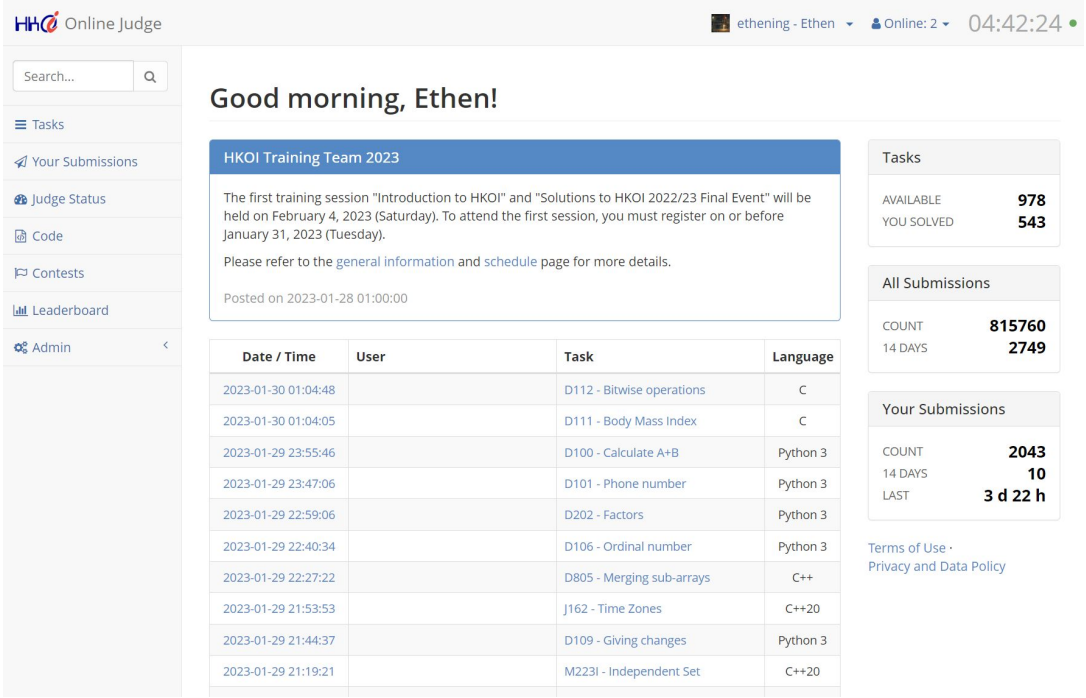
Week / Date	AM Session 10:00 – 13:00 Level A	AM Session 10:00 – 13:00 Level B	PM Session 14:00 – 17:00 For all trainees
Week 7 2026-02-14	G600 Introduction to HKOI		G600 Solutions to HKOI and HKGOI 2025/26 Final Event
Week 8 2026-02-21	B7510 Dynamic Programming (II) Siu Lok Yin	G600 Programming in C++ Wong Cheuk Kiu	G600 Introduction to Linux Chan King Kai
Week 9 2026-02-28	B7510 Graph (III) Wai Ka Hei	4307 Optimization and Common Tricks Wong Cheuk Kiu	4307 Constructive Algos, Special Tasks (I) Hsieh Chong Ho
Week 10 2026-03-07	B7510 Data Structures (IV) Wong Chun	G600 Data Structures (II) Yuen Lok Kan Ethen	G600 Mini Competition (I)
Week 11 2026-03-14	B7510 Graph (IV) Wong Chun	G600 Graph (I) Wong Cheuk Kiu	G600 Constructive Algos, Special Tasks (II) Yuen Lok Kan Ethen
Week 12 2026-03-21	B7510 Dynamic Programming (III) Ko Kin Fung Nicholas	G600 Dynamic Programming (I) Chow King Wang	G600 Mini Competition (II)
Week 13 2026-03-28			4307 EGOI Team Formation Test
Week 14 2026-04-04	No Training (BBQ)		
Week 15 2026-04-11	3610 Graph (V) Ko Kin Fung Nicholas	4307 Mathematics in OI (I) Hsieh Chong Ho	4307 Mini Competition (Teams)
Week 16 2026-04-18	3610 Flow and Graph Matching (I) Yuen Lok Kan Ethen	G600 Graph (II) Chow King Wang	G600 Game Theory Wong Chun
Week 17 2026-04-25	3610 Flow and Graph Matching (II) Lu Yi Fung	G600 Data Structures (III) Wong Cheuk Kiu	G600 Mini Competition (III)
Week 18 2026-05-02	TBC Misc Problem Discussion Yuen Lok Kan Ethen, Lu Yi Fung		TBC Mathematics in OI (II) Wong Chun
Week 19 2026-05-09 2026-05-10			TBC APIO / Back-up days for Team Formation Test
Week 20 2026-05-16	No Training		
Week 21 2026-05-23 2026-05-24			TBC Team Formation Test (2 days)

Attendance Policy

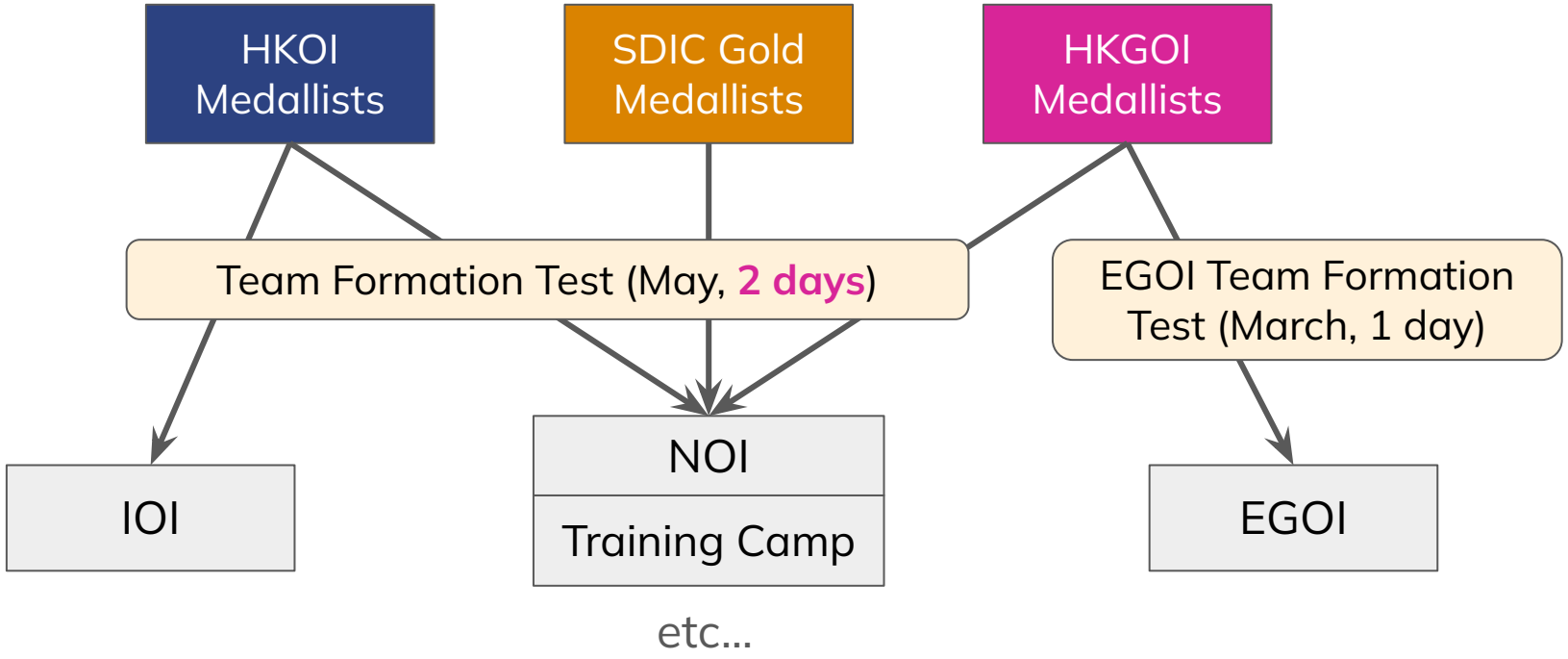
- No session is compulsory, but we do take attendance.
- Attendance is taken separately for AM and PM sessions.
- You must take attendance if you are present.
- You may be considered absent if you arrive late / leave early.
- Training Team members who attend at least 60% of the sessions (each day of Team Formation Test and APIO each counts as 2 sessions) will receive a certificate.

Online Judge

- Mainly, we use the HKOI Online Judge for training purpose.
- Tasks for previous HKOI events, other local events, minicomps, and team formation tests are available.

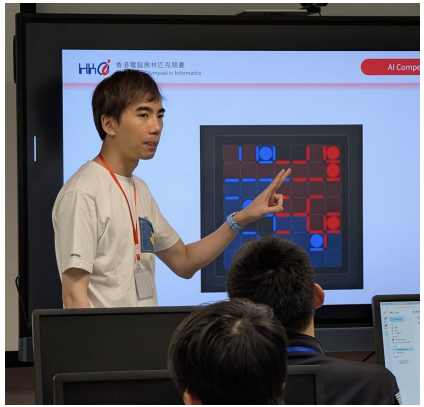


Team Formation Test



HKOI Training Camp

*The camp is mandatory for all HK representatives.



Training Camp 2025

ACM-HK Contest / ICPC Hong Kong Regional Contest

ACM-HK Programming Contest 2025

ICPC Hong Kong Regional Contest 2025

National Olympiad in Informatics

全国青少年信息学奥林匹克竞赛

NOI 2025, Shaoxing

International Olympiad in Informatics

IOI 2025, Sucre, Bolivia

Other External Competitions

- APIO - Asia Pacific Informatics Olympiad
- IOM - International Olympiad of Metropolises

IOM 2019, Moscow

IOM 2021, Online

What awaits you in front…

UZBEKISTAN
IOI 2026 TASHKENT

"Uzbekistan, officially known as the Republic of Uzbekistan, is a landlocked country in Central Asia."

IOI 2026, Tashkent, Uzbekistan

https://ioi2026.uz/

What awaits you in front…

EGOI 2026, Cesenatico, Italy

https://egoi2026.it/

Important Notice

- Make sure your travel documents have **at least 6 month validity** beyond the date of departure from the Hong Kong.
- The registration deadline of external competitions is usually close to the selection test.

- You **should renew** your Mainland Travel Permit & Passport **now**, regardless of how likely you would become a HK representative.
- We will select the next contestant in the ranking if there are troubles with the documents' validity.

The World of Competitive Programming

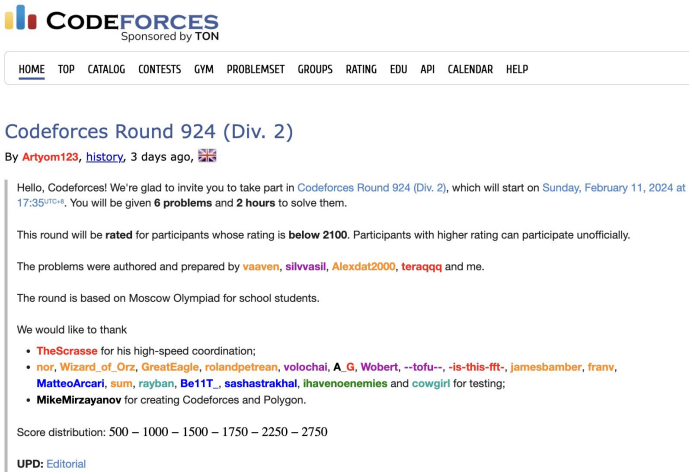
ICPC - International Collegiate Programming Contest

- Team-based contest for college students
  - Regional
  - Regional Final
  - World Final
- Each solve earns you 1 balloon

The World of Competitive Programming

Codeforces: https://codeforces.com/ (recommended)

- Regular contests of different divisions, compete with worldwide top players
- Most recognized Elo rating system (if you heard about someone showing off their “color”)

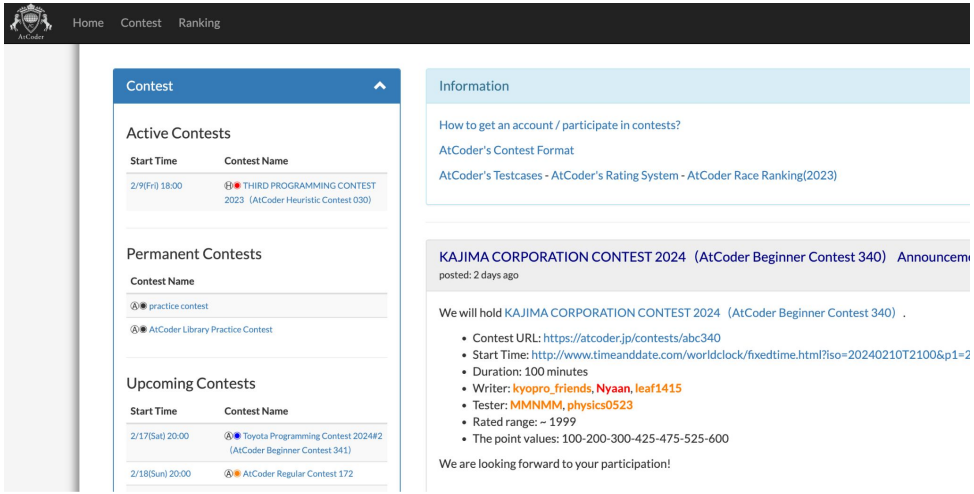


Rating range	Title	Division
3000 — 3999+	Legendary Grandmaster	1
2600 — 2999	International Grandmaster	1
2400 — 2599	Grandmaster	1
2300 — 2399	International Master	1
2100 — 2299	Master	1
1900 — 2099	Candidate Master	1/2
1600 — 1899	Expert	2
1400 — 1599	Specialist	2/3
1200 — 1399	Pupil	2/3/4
≤ 1199	Newbie	2/3/4

The World of Competitive Programming

AtCoder: https://atcoder.jp/ (recommended)

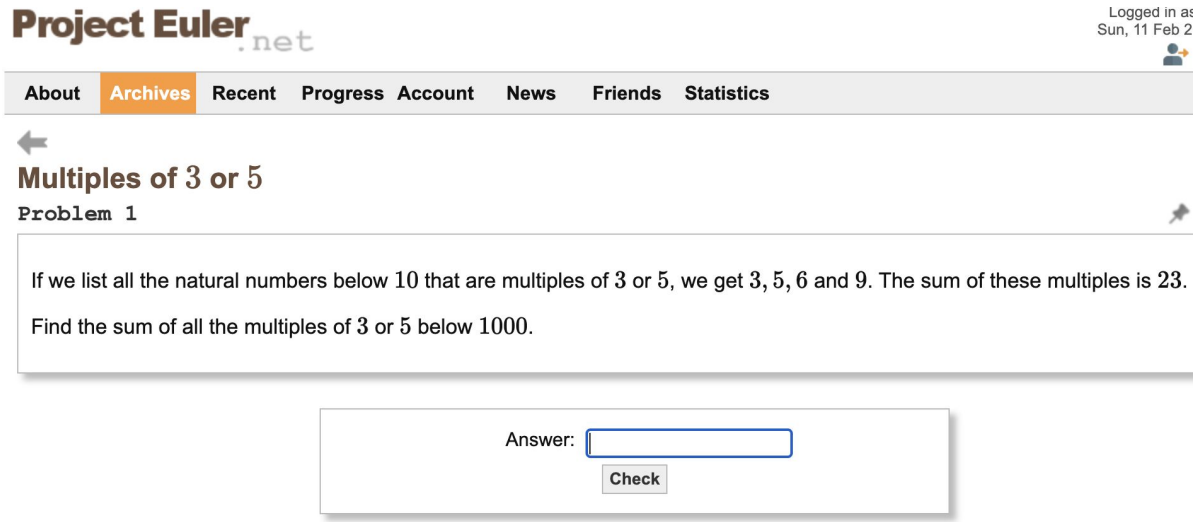
- Japan-based contest site (more observation-based tasks)
- Algorithmic contest & Heuristic contest



The World of Competitive Programming

Project Euler: https://projecteuler.net/about

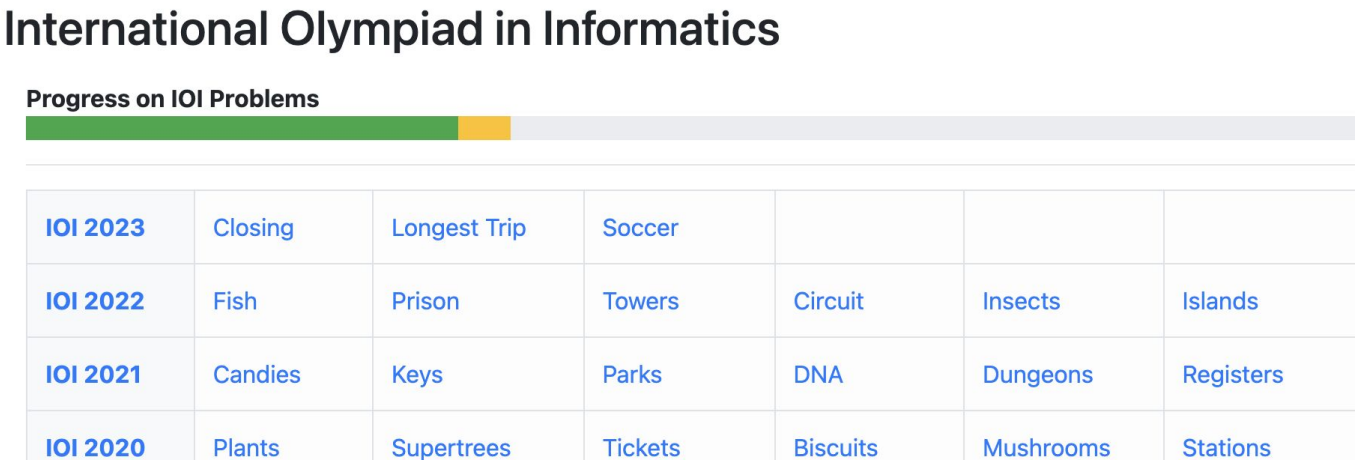
- Judge that only accepted numerical answer, more mathy tasks



The World of Competitive Programming

OI Checklist: https://oichecklist.pythonanywhere.com/ (strongly recommended)

- Checklist for hard problems from past OIs of different countries



Common Contest Rules

IOI Style

- 3 to 5 tasks, 100 points each (usually)
- Each task will be divided into subtasks (or cases)
- Rank based on points you get

- Solving speed is less important
- Common in high school contests
  - HKOI Final Event, HK TFT (Team Formation Test)

Rank	First Name	Last Name	Team	cand…	keys	parks	IOI 202…	dna	dun…	regi…	IOI 202…	Global
1	Mingyang	Deng	CHN3	100	100	100	300	100	100	100	300	600
2	Yi	Qian	CHN2	67	100	70	237	100	100	100	300	537
3	Chenxin	Dai	CHN4	100	100	70	270	100	89	46	235	505
4	Haoxiang	Yu	CHN1	100	37	100	237	100	89	71	260	497
5	Rain	Jiang	USA2	100	100	45	245	100	100	34	234	479
6	Ryomei	Sugai	JPN2	100	67	100	267	100	62	46	208	475
7	Ditbul	Ban	KOR1	100	67	70	237	100	62	75	237	474
8	Ashley Aragorn	Khoo	SGP1	100	37	100	237	100	89	46	235	472
9	Zixiang	Zhou	CAN3	100	37	100	237	100	100	33	233	470
10	Harris	Leung	HKG2	100	100	70	270	100	0	75	175	445
11	Matej	Urban	SVK1	100	37	70	207	100	36	100	236	443
12	Siyong	Huang	USA1	100	37	70	207	100	62	71	233	440
13	Marco	Meijer	NLD1	100	37	55	192	100	100	47	247	439
14	Dorijan	Lendvaj	HRV2	100	0	70	170	100	62	100	262	432
15	Jan	Strzeszyński	POL2	38	100	70	208	100	62	46	208	416
16	Patrick	Pavić	HRV1	100	37	70	207	100	50	58	208	415
17	Timothy	Feng	USA4	100	37	30	167	100	89	58	247	414
17	Ioan	Popescu	ROU4	11	100	70	181	100	100	33	233	414
19	Mikhail	Budnikov	RUS3	67	67	70	204	100	62	46	208	412
19	Egor	Lifar	RUS1	67	67	70	204	100	62	46	208	412
21	Alireza	Keshavarz Hedayati	IRN2	67	100	15	182	100	100	21	221	403
22	Joël	Huber	CHE1	100	37	70	207	100	62	21	183	390
22	Shen	Xing Yang	SGP2	100	37	70	207	100	62	21	183	390
24	Timofei	Fedoseev	RUS2	11	100	70	181	100	50	58	208	389
25	Oleh	Naver	UKR1	67	37	70	174	100	62	46	208	382
26	Jiří	Kalvoda	CZE2	100	37	35	172	100	62	46	208	380
27	Ping-Hsuan	Lin	TWN1	38	37	70	145	100	100	33	233	378
28	Pikatan Arya	Bramajati	IDN2	100	100	55	255	100	11	10	121	376
28	Daiki	Kodama	JPN3	38	37	30	105	100	100	71	271	376
30	Tan	Si Jie	SGP3	67	37	100	204	100	36	33	169	373

Harris Leung (HKG)

IOI 2021 Scoreboard
https://ranking.ioi2021.sg/

Common Contest Rules

ICPC Style

- 8 to 12 tasks (usually)
- No subtasks
- Rank based on tasks solved, then by time penalty

- Solving speed is important
- Common in uni contests
  - ICPC, HKOI Team Minicomp

ICPC World Finals 2019 Scoreboard
https://icpc.global/scoreboard/

RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K
1	Northern Eurasia Moscow State University	10 1531	42 1 try	142 1 try		56 1 try	40 2 tries	279 4 tries	114 1 try	92 2 tries	245 1 try	72 1 try	249 6 tries
2	North America Massachusetts Institute of Technology	9 1191	27 1 try	90 1 try		107 1 try	56 1 try	168 1 try	119 2 tries	63 2 tries	278 1 try		243 1 try
3	Asia Pacific The University of Tokyo	9 1386	40 1 try	204 2 tries		62 1 try	31 1 try	230 4 tries	128 3 tries	57 1 try		157 1 try	297 4 tries
4	Europe University of Warsaw	8 891	49 1 try	126 2 tries	11 tries	32 1 try	14 1 try		55 1 try	32 1 try		111 1 try	292 9 tries
5	National Taiwan University	8 1179	27 1 try	165 1 try		38 1 try	142 1 try		130 1 try	208 1 try	278 1 try	191 1 try	
6	University of Wroclaw	8 1200	29 1 try	277 4 tries		28 1 try	57 1 try		212 2 tries	91 2 tries	263 2 tries	103 2 tries	1 try
7	Seoul National University	7 783	74 1 try	103 1 try		31 2 tries	69 3 tries	3 tries	146 3 tries	82 1 try		118 4 tries	
8	Asia East KimChaek University of Technology	7 803	32 1 try	132 2 tries		78 2 tries	43 1 try		97 1 try	188 1 try		193 1 try	
9	Asia West Sharif University of Technology	7 923	23 1 try	170 2 tries		75 1 try	46 1 try	1 try	148 2 tries	133 1 try		288 1 try	
10	Moscow Institute of Physics & Technology	7 954	47 1 try	155 1 try	1 try	140 1 try	78 2 tries		145 1 try	113 1 try		236 2 tries	
11	National Research University Higher School of Economics	7 990	50 1 try	199 3 tries		76 2 tries	51 2 tries		137 2 tries	104 1 try		273 1 try	
12	The Chinese University of Hong Kong	7 1057	90 1 try	239 4 tries		42 1 try	59 1 try	2 tries	217 2 tries	127 1 try		203 1 try	
13	Peking University	7 1106	34 1 try	245 1 try		119 2 tries	163 2 tries		143 1 try	114 1 try		228 2 tries	
14	Fudan University	7 1179	24 2 tries	284 6 tries		27 1 try	197 7 tries		152 1 try	131 1 try	1 try	124 1 try	
15	Nanjing University	7 1184	21 1 try	185 4 tries		86 2 tries	58 1 try		262 4 tries	159 5 tries		173 2 tries	
16	University of Oxford	7 1185	54 1 try	295 6 tries		30 1 try	85 2 tries		235 3 tries	178 1 try		148 1 try	
17	St. Petersburg ITMO University	7 1296	34 1 try	178 5 tries		105 1 try	160 4 tries		291 4 tries	63 2 tries		245 1 try	
18	University of Cambridge	7 1359	161 3 tries	139 1 try		81 1 try	153 1 try		202 4 tries	242 1 try		261 2 tries	
19	Shanghai Jiao Tong University	7 1427	47 1 try	298 12 tries		97 1 try	42 1 try		257 2 tries	166 3 tries		220 2 tries	
20	Tsinghua University	7 1634	235 5 tries	180 3 tries		95 1 try	150 5 tries		194 2 tries	178 1 try		282 6 tries	1 try
21	Universitat Politècnica de Catalunya	6 636	62 1 try	185 1 try		52 1 try	31 1 try		181 1 try	125 1 try			
22	Belarusian State University	6 773	86 1 try	219 3 tries		27 1 try	46 1 try		129 1 try	186 3 tries			
23	Universitas Indonesia	6 815	115 4 tries	223 6 tries		92 1 try	64 1 try	5 tries	128 1 try	33 1 try			
24	KAIST	6 869	40 2 tries	290 7 tries		66 1 try	23 2 tries	5 tries	162 4 tries	68 1 try			
25	Technische Universität München	6 946	38 1 try	260 4 tries		61 1 try	17 1 try		212 3 tries	198 4 tries		1 try	1 try
26	Harvard University	6 965	159 3 tries	4 tries		82 1 try	115 2 tries		232 2 tries	35 1 try		182 5 tries	
27	National Chiao Tung University	6 987	75 1 try	289 1 try		182 1 try	32 1 try		223 3 tries	146 1 try			
28	The University of Texas at Austin	6 989	81 2 tries	284 3 tries		138 1 try	42 1 try			105 1 try		239 3 tries	
29	Zhongshan (Sun Yat-sen) University	6 1017	39 1 try			66 1 try	154 4 tries		273 5 tries	80 1 try		205 4 tries	
30	Stanford University	6 1073	91 2 tries	3 tries		99 1 try	78 3 tries		196 1 try	253 4 tries		236 1 try	

Ho Ngan Hang,
Poon Lik Hang,
Yik Wai Pan (CUHK)

Other Contest Rules

Codeforces Round #698 (Div. 1)

Final standings

You may double click into cells (or ctrl+click) to view the submissions history or hack the solution

Standings

#	Who	=	*	A 500	B 1000	C 1500	D 2250	E 2750	E 4000
1	maroonrk	5882		496 00:02	902 00:12	1386 00:19	1800 00:50	1298 02:12	
2	panole	5843		492 00:04	932 00:17	1284 00:36	1683 01:03	1452 01:58	
3	tourist	5533		470 00:15	892 00:27	1172 00:38	1323 01:43	1676 01:24	
4	Miracle03	5530		494 00:03	932 00:17	1300 00:25	1701 01:01	1103 02:07	
5	boboniu	4900		494 00:03	948 00:13	1362 00:23			2096 01:59
6	Um_nik	4736		496 00:02	952 00:12	1398 00:17	1890 00:40		-4
7	Benq	4725		498 00:01	936 00:16	1446 00:09	1845 00:45	-2	
8	Radewoosh	4709		494 00:03	948 00:13	1404 00:16	1863 00:43	-1	
9	dorijanlendvaj	4496	-1	494 00:03	940 00:15	1380 00:20		1732 01:28	
10	atomicenergy	4460		490 00:05	940 00:15	1320 00:30	1710 01:00		
11	Golovanov399	4452		494 00:03	936 00:16	1380 00:20	1642 01:02		
12	yhx.12243	4374		442 00:04	944 00:14	1314 00:11	1674 01:04		
13	aid	4371		486 00:07	912 00:22	1308 00:32	1665 01:05		
14	jcvb	4365		488 00:06	936 00:16	1362 00:23	1579 01:09	-3	
15	Petr	4357		490 00:05	920 00:20	1350 00:25	1597 01:07		
16	jiangly	4333		494 00:03	956 00:11	1344 00:26	1539 01:19		
17	Swistakk	4301		494 00:03	672 01:22	1434 00:11	1701 01:01		
18	semiexp	4284		496 00:02	844 00:39	1360 00:15	1584 01:14		
19	Marcin_smu	4279		446 00:27	908 00:23	1296 00:34	1629 01:09	-1	
20	300iq	4230		424 00:13	920 00:20	1338 00:27	1548 01:18	-1	

Virtual World Finals 2020

Show round overview

The winner of this round will be our Code Jam 2020 World Champion.

		Pack the Slopes	Adjacent and Cons...	Hexacoin Jam	Musical Cords	Replace All
		Open problem	Open problem	Open problem	Open problem	Open problem
	Everyone	10 22	10 32	10 10 22	15 27	15 27
1	Gennady.Korotkevich	3:11:02 +0 ✓ ✓	2:12:41 +0 ✓ ✓	2:58:14 +0 ✓ ✓ ✓	3:10:46 +0 ✓ —	3:10:52 +0 ✓ ✓
2	ksun48	0:48:50 +2 ✓ ✓	2:45:24 +0 ✓ ✓		3:22:36 +0 ✓ —	1:55:41 +0 ✓ ✓
3	ecnerwala	0:16:34 +0 ✓ ✓	2:37:31 +0 ✓ ✓	1 attempt — —	3:55:58 +0 ✓ —	1:14:01 +0 ✓ ✓
4	scottwu	3:16:50 +0 ✓ ✓	2:12:06 +0 ✓ ✓	1 attempt — —	3:53:58 +5 ✓ —	2:51:51 +0 ✓ ✓
5	eatmore	0:58:15 +0 ✓ ✓	3:54:39 +0 ✓ ✓			2:10:36 +0 ✓ ✓
6	Benq	0:13:12 +0 ✓ ✓	2:03:17 +3 ✓ —	2:46:10 +0 ✓ — —	3:31:30 +0 ✓ —	3:13:56 +0 ✓ ✓
7	KalininN	0:17:02 +0 ✓ ✓	2:19:18 +1 ✓ —	2 attempts — —	3:26:54 +2 ✓ —	1:31:04 +2 ✓ ✓
8	ainta	0:32:38 +3 ✓ ✓	3:40:30 +5 ✓ —		1:06:09 +0 ✓ —	2:28:46 +1 ✓ ✓
9	ikatanic	0:45:19 +0 ✓ ✓	2:10:07 +2 ✓ ✓			3:25:22 +3 ✓ —
10	yutaka1999	0:40:10 +0 ✓ ✓	3:53:44 +0 ✓ —	14 attempts — — —		1:20:40 +0 ✓ ✓
11	sam.rei	1:43:40 +0 ✓ ✓	2:54:23 +1 ✓ —			3:49:53 +0 ✓ ✓
12	koosaga	0:16:10 +0 ✓ ✓	0:50:57 +1 ✓ —	3:51:39 +1 ✓ — —	3:16:18 +1 ✓ —	1:49:12 +1 ✓ —

Standings

per page: 10 **20** 50 100 1000

1 2 4 8 16 26

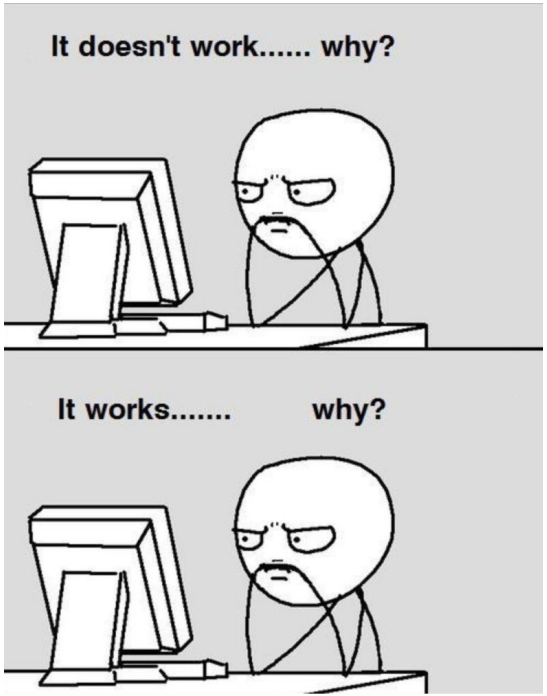
Customize

Rank	User	Score	A	B	C	D	E	F
1	semiexp	6800 (1) 233:00	500 6:17	800 23:15	1300 62:23	1800 106:55	-	2400 (1) 228:00
2	yutaka1999	6800 (1) 254:41	500 6:29	800 141:02	1300 (1) 86:22	1800 124:11	-	2400 249:41
3	Petr	6400 (6) 296:43	500 6:34	800 13:59	1300 (2) 90:00	1800 46:50	2000 (4) 266:43	-
4	Benq	5100 (3) 252:29	500 4:38	800 34:27	(4)	1800 121:17	2000 (3) 237:29	-
5	Um_nik	5100 (3) 262:05	500 7:03	800 18:08	-	1800 133:49	2000 (3) 247:05	-
6	yokozuna57	5000 (3) 278:32	500 13:23	800 28:52	1300 (2) 71:57	-	-	2400 (1) 263:32
7	ecnerwala	4400 111:35	500 15:41	800 9:07	1300 70:30	1800 111:35	(4)	-
8	Stonefeang	4400 (1) 138:31	500 (1) 28:37	800 8:16	1300 94:51	1800 133:31	-	-
9	Merkurev	4400 (3) 146:28	500 (1) 16:12	800 28:33	1300 (2) 83:04	1800 131:28	-	-
10	antontrygubO_o	4400 (2) 151:31	500 29:49	800 (1) 22:00	1300 (1) 76:26	1800 141:31	-	(3)
11	tourist	4400 (3) 159:13	500 9:06	800 21:07	1300 (2) 144:13	1800 (1) 126:31	-	(4)
12	Lagoon	4400 (6) 166:35	500 (1) 12:14	800 (3) 45:59	1300 (1) 95:08	1800 (1) 136:35	-	-
13	aid	4400 (2) 167:01	500 9:58	800 51:35	1300 (2) 110:47	1800 157:01	-	-
14	ugly2333	4400 (1) 167:24	500 (1) 162:24	800 154:22	1300 154:20	1800 154:19	-	-
15	jiangly	4400 (4) 176:52	500 5:49	800 (1) 37:58	1300 (3) 156:52	1800 80:36	-	-

Competitive Programming in Real Life

Language	Result
C++11	Wrong Answer
C++11	Wrong Answer
C++11	Wrong Answer
Pascal	Runtime Error
C++11	Compilation Error
Python 3	Wrong Answer
C	Wrong Answer
C++11	Compilation Error
C++11	Time Limit Exceeded
Python 3	Wrong Answer
C	Wrong Answer
Java 8	Wrong Answer
C++11	Time Limit Exceeded
C	Wrong Answer
C++11	Wrong Answer
Pascal	Compilation Error
Pascal	Wrong Answer
C++11	Time Limit Exceeded
Pascal	Compilation Error
Pascal	Wrong Answer
C++11	Wrong Answer
C++11	Wrong Answer
C++11	Wrong Answer
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
Python 3	Wrong Answer
C	Wrong Answer
Java 8	Wrong Answer
C++11	Time Limit Exceeded
C	Wrong Answer
C++11	Wrong Answer
Pascal	Compilation Error
Pascal	Wrong Answer
C++11	Time Limit Exceeded
Pascal	Compilation Error
Pascal	Wrong Answer
C++11	Wrong Answer
C++11	Wrong Answer
C++11	Wrong Answer
C++11	Time Limit Exceeded

Competitive Programming in Real Life



Competitive Programming in Real Life

- When you just start learning programming, the hardest part is to implement your thoughts into actual code (which should be able to compile and run)
  - Similar to you trying to speak out your thoughts in a foreign language

- But when you get better, you will find out the tasks ask for more than **a solution**. It wants an **efficient** solution.

Competitive Programming in Real Life

- Let's say you need to calculate 1 + 2 + 3 + ... + N.

- **Solution 1**: A for loop looping from 1 to N, adding them each to a sum.
- **Solution 2**:
  - Note that you can duplicate all numbers, and pair them up (1, N), (2, N - 1), (3, N - 2), ..., (N - 1, 2), (N, 1).
  - Every pair sums to N + 1. There are N pairs in total.
  - Hence, the sum is **N * (N + 1) / 2**.

- You should probably notice that Solution 2 is faster. But how do we say that **formally?** We usually use something called **Big-O notation**.

What is Big-O?

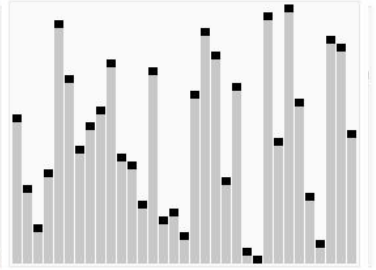
Motivation

- A way to measure the run-time of our program / algorithm
- Commonly used among competitive programming world

Time Complexity

- Best-case performance
- Average performance
- Worst-case performance

Quicksort



Animated visualization of the quicksort algorithm. The horizontal lines are pivot values.

Class	Sorting algorithm
Worst-case performance	$O(n^2)$
Best-case performance	$O(n \log n)$ (simple partition) or $O(n)$ (three-way partition and equal keys)
Average performance	$O(n \log n)$

Quicksort - Wikipedia
https://en.wikipedia.org/wiki/Quicksort

Time Complexity

What we really care:

- Worst-case performance (99% of the time)
- Average performance (1% of the time)
- Best-case performance (nearly never)

About hacking Java's `Array.sort()`: https://codeforces.com/blog/entry/64109

Time Complexity

- Try to express the number of (basic) operations our program will execute
- As a function in terms of inputs

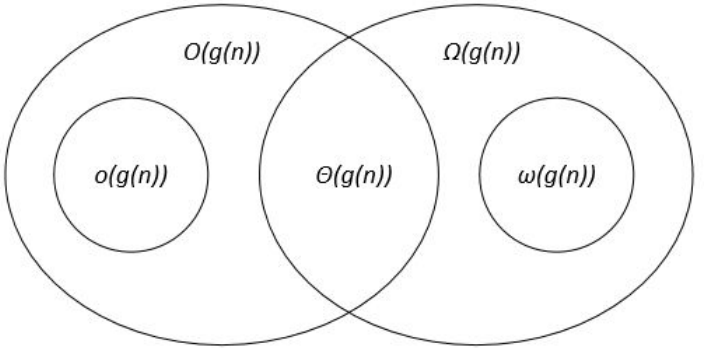
Example:

```
cin >> N >> K;                  // 1 operation
for (int i = 1; i <= N; i++)    // N operations
  if (i % K == 0)               // 1 operation that happens N times
    ans++;                      // 1 operation that happens N/K times
cout << ans;                    // 1 operation

// total 2N+N/K+2 operations
```

Big-O Notation

- Big-O notation helps us simplify the time complexity function
- In theory, we are interested in when the inputs could be infinitely large



Other types of bounds

[advanced] Formal definition:

$f(n) = O(g(n))$ if there exists n_0, $c > 0$ such that

$$f(n) \leq cg(n) \text{ for all } n \geq n_0$$

So, formally $3N^2$ is also $O(N^N)$ but $O(N^2)$ is a tighter known valid bound, thus we would say it's $O(N^2)$

Big-O Notation

Theoretically, we do not care coefficients and constant terms

- $3N+5 \rightarrow O(N)$
- $5N^3+26Q \rightarrow O(N^3+Q)$
- $10000KN^{0.5} \rightarrow O(KN^{0.5})$
- $3N^{2K} \rightarrow O(N^{2K})$
- $500+7000 \rightarrow O(1)$

Big-O Notation

We only care terms that dominate the function

- $N^2+N+Q \rightarrow O(N^2+Q)$
- $N^K+N^{0.5K}+K^N \rightarrow O(N^K+K^N)$
- $2^N+N^9 \rightarrow O(2^N)$
- $N^N+N! \rightarrow O(N^N)$
- $NlogN \rightarrow O(NlogN)$
- $logN+N \rightarrow O(N)$
- $log_3N \rightarrow O(logN)$

Examples

```
cin >> N >> K;                  // 1 operation
for (int i = 1; i <= N; i++)    // N operations
  if (i % K == 0)               // 1 operation that happens N times
    ans++;                      // 1 operation that happens N/K times
cout << ans;                    // 1 operation

// total 2N+N/K+2 operations → O(N)   note: assume K never falls in (-1,1)
```

Examples

Count number of pairs (`i`, `j`) where $1 \le$ `i`, `j` $\le$ `N` and `j` is a multiple of `i`

```
cin >> N;
for (int i = 1; i <= N; i++)     // N operations
  for (int j = 1; j <= N; j++)  // N operations that happen N times
    if (j % i == 0)
      ans++;
cout << ans;

// total time complexity: O(N^2)
```

Examples

- Linear search → $O(N)$
- Binary search → $O(\log N)$

- Bubble sort → $O(N^2)$
- Insertion sort → $O(N^2)$
- Merge sort → $O(N \log N)$

Examples

```
cin >> N;
for (int i = 1; i <= N; i++)
  for (int j = 1; j <= i * i; j++)
    ans++;   // how many times?
cout << ans;

// 1^2+2^2+3^2+4^2+...+N^2 = ?
//                  = N(N+1)(2N+1)/6
// total time complexity: O(N^3)
```

Examples

```
cin >> N;
for (int i = 1; i < N; i++)
  for (int j = i * i; j < (i+1) * (i+1); j++)
    ans++;   // how many times?
cout << ans;

// (2²-1²)+(3²-2²)+(4²-3²)+...+(N²-(N-1)²) = N²-1²
// total time complexity: O(N²)
```

Examples

Count number of pairs (i, j) where $1 \le i, j \le N$ and j is a multiple of i

Previous code

```
cin >> N;
for (int i = 1; i <= N; i++)
  for (int j = 1; j <= N; j++)
    if (j % i == 0)
      ans++;
cout << ans;

// total time complexity: O(N^2)
```

New code

```
cin >> N;
for (int i = 1; i <= N; i++)
  for (int j = i; j <= N; j += i)
    ans++; // j % i == 0 always
cout << ans;


// time complexity is better than O(N^2)  :D
// why?

// N/1+N/2+N/3+N/4+...+N/N = ?
// It's NlgN    [check out Harmonic Series]

// Therefore, time complexity is O(NlgN)
```

Comparisons

Is O(N) always worse than O(1)?
Is O(NlgN) always worse than O(N)?

- Yes and no
- Yes: in computer science theory, we care about when N is infinitely large
- No: in real-life, it depends on your program, constant terms still matter

Estimations

A modern CPU can execute more than 10^9 instructions per second
When $N=10^9$, O(N) program always finish in 1 second?

Some factors to think about:

- Type of instructions (e.g. bitwise operations are faster than modulus and divisions)
- Type of functions/containers (e.g. `std::sort` v.s. `std::set`)
- Data type and judging machine (e.g. 64-bit computations on 32-bit machine)

Estimations

For TL=1s,

- O(N) → $N \leq 10^7$
- O(NlgN) → $N \leq 10^6$
- $O(N^2)$ → $N \leq 5000$
- $O(N^3)$ → $N \leq 300$
- $O(N^4)$ → $N \leq 100$
- $O(2^N)$ → $N \leq 20$
- O(N!) → $N \leq 10$

Estimations

For TL=1s,

- $N \le 10^7$
  → maybe O(N)?
- $N \le 10^5$, 5×10^5, 10^6
  → maybe O(NlgN)? or O(N), $O(N\lg^2 N)$
- $N \le 1000$, 5000
  → maybe $O(N^2)$? or $O(N^2 \lg N)$
- Large coefficient & constant terms would affect this decision.

For TL=1s,

- O(N) → $N \le 10^7$
- O(NlgN) → $N \le 10^6$
- $O(N^2)$ → $N \le 5000$
- $O(N^3)$ → $N \le 300$
- $O(N^4)$ → $N \le 100$
- $O(2^N)$ → $N \le 20$
- O(N!) → $N \le 10$

Amortized Complexity

Sometime, the time complexity is more accurate when analyze as a whole, but not as individual parts.

Consider implementing a data structure that supports the following:

- push(x) - push x into the stack
- pop() - pop the top element in the the stack (if it exists)
- multipop(n) - pop the top n elements in the stack, or until the stack is empty

What is the time complexity of the program **handling N operations** on this data structure?

Amortized Complexity

If analyze the time complexity of each operations individually

- push(x) - O(1)
- pop() - O(1)
- multipop(n) - O(N)

What is the time complexity of the program **handling N operations** on this data structure? $O(N^2)$

Does this make sense?

Amortized Complexity

If analyze the time complexity of each operations individually

- push(x) - O(1)
- pop() - O(1)
- multipop(n) - O(N)

What is the time complexity of the program **handling N operations** on this data structure? $O(N^2)$

- **multipop(n)** is only costly if there are many **push(x)** operations before.
  How do we formulate a better time complexity bound by considering both the costly operations and cheap operations?

Amortized Complexity

- push(x) - push x into the stack
- pop() - pop the top element in the the stack (if it exists)
- multipop(n) - pop the top n elements in the stack, or until the stack is empty

You may observe that the time cost is actually bounded by the number of elements that have been push into the stack, that is **O(N) in total**.

- By distributing this time cost onto each operation, we can say that the operations are **amortized O(1)**.

Amortized Complexity

There are more formal ways of analysing the algorithm's amortized complexity. From wiki, there are

- **Aggregate method** - calculate amortized complexity by T(n) / n
- **Accounting method** - define a cost for each operation
- **Potential method** - define a *potential function* for the state of the data structure, and analysis on how the operation change the potential

They are not really useful during contest time (usually you should just reason it by intuition), but you may encounter them down the road for advanced data structure.

Amortized Complexity

Amortized complexity also comes into play for commonly used C++ standard container like std::vector.

```cpp
const int N = 10'000'000;

int main() {
  vector<int> v;
  for (int i = 0; i < N; i++) {
    v.push_back(i);  // what's the time complexity? O(1)?
  }
}

// In fact, it's only amortized O(1)
```

http://cplusplus.com/reference/vector/vector/push_back/
https://assets.hkoi.org/training2019/adv-cpp.pdf

Amortized Complexity

```cpp
#include <bits/stdc++.h>
using namespace std;
using namespace chrono;

const int N = 10'000'000;

int main() {
  vector<int> t, v;
  for (int i = 0; i < N; i++) {
    auto t1 = high_resolution_clock::now();
    v.push_back(i);
    auto t2 = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(t2 - t1).count();
    t.push_back(duration);
  }

  int total = accumulate(t.begin(), t.end(), 0, std::plus<int>());

  sort(v.begin(), v.end(), [&t](const int& lhs, const int& rhs) {
    return t[lhs] > t[rhs];
  });

  printf("total time = %6dus\n", total);
  printf("    median = %6dus\n", t[v[N / 2]]);
  printf("\n");

  for (int i = 0; i < 8; i++) {
    printf("t[%7d] = %6dus [%7.4f%%]\n", v[i], t[v[i]], t[v[i]] *
100. / total);
  }
}
```

```
total time = 179725us
    median =      0us

t[8388608] =  27744us [15.4369%]
t[4194304] =  14108us [ 7.8498%]
t[2097152] =   6905us [ 3.8420%]
t[1048576] =   3847us [ 2.1405%]
t[ 524288] =   1935us [ 1.0766%]
t[ 262144] =   1010us [ 0.5620%]
t[ 131072] =    498us [ 0.2771%]
t[  65536] =    264us [ 0.1469%]
```

~~Accepted~~

No more TLE!