introToCP.cpp

imagine something cool ^_^

Unlocking Secrets of 'Competitive Programming' {

[KickStart Your CP Journey]

<Frameworks are temporary, Algorithms are eternal>

}

Presenters - Abhinav, Harasees, Ankur

What is ‘Competitive Programming’;

Competitive Programming is a mind sport in which participants try to solve programming problems within given time constraints.

The competitions, usually called ‘Coding Contests’ involve a set of logical or mathematical problems and contestants are required to write computer programs which can solve these problems.

Contestants solving most problems in the least amount of time are rewarded with higher rating.

Requirements for ‘CP’ {

01 LOGIC

< CP requires application of logical abilities with good understanding of programming syntax and data structures >

02 ACCURACY

< CP tests analysis of all possible cases for a problem and rewards participants for better accuracy and fewer mistakes >

03 SPEED

< CP needs solving problems in as less time as possible and checks how quickly participants can build and implement their solutions >

}

01 {

[Benefits of CP]

< Reasons why it is worth the effort >

}

Improves Problem Solving < /1 > {

< CP provides us a bunch quality problems on a regular basis. Solving and practicing them improves our logical, mathematical and coding skills. >

}

Helps in Time Management < /2 > {

< CP exposes us to time constraints. We are required to solve problems within the contest time and faster than others. This improves our speed and time management. >

}

New Concepts & Algorithms < /3 > {

< In every contest, we get to solve problems requiring unique algorithms and logic. This expands our concepts and knowledge of various algorithms. >

}

Exposure to Pressure < /4 > {

< CP provides a competitive environment where we battle against contestants from all over the world and face the pressure of live competition and rankings. >

}

Interview Preparation < /5 > {

< The problems in CP contests are as good as any company’s coding tests and can help us practice our programming skills to crack any online assessment or interview. >

}

As an Achievement < /6 > {

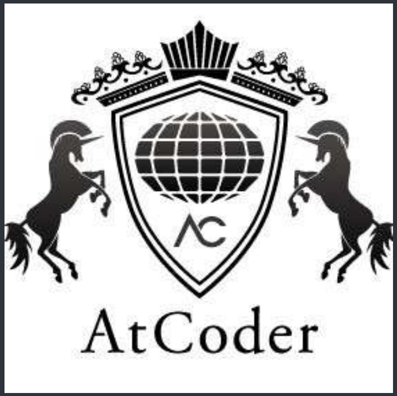
< Good CP ratings on reputed platforms make your CV all the more impressive and significantly improve chances of selection. Also, high CP ratings give something to brag about in front of your friends! >

}

02 {

PLATFORMS FOR CP

}



Atcoder Beginner Contest
happen every Saturday.
Must give.

Almost 3 contests per week.
So you will never be out of
Practice.

How to Practice ?

Contests </1>

- Try to give at least 2 contests per week.
- Contests help in improving speed and keep you in competitive shape.
- If you can't give a live contest for some reason , try appearing in it's virtual contest

Stuck on a problem ? </2 >

- Don't try a problem for more than 45-50 minutes if you're completely blank on what approach to use.
- Abandon it, try other problems , discuss with others and then return to it.

Identify your Weakness </3 >

{

- You won’t gain much even if you solve 100 problems of topics that are easy or those that you’re comfortable with
- Try to identify a topic whose problems consistently trouble you , work on those topics

}

Check Editorials </4>

- Contain ideal approach to solve problems.
- Read editorials even for which questions that you were able to solve might contain better approach

}

Discuss </5 >

- Create Communities and try to find like minded people who have different skill sets so that you can learn from each other
- Exposure to multiple approaches to a specific problem.

Don’t Give Up </6 >

- If you’re struggling initially don’t give up, hang in there, the pleasure and satisfaction you’ll get once you find your groove is unmatched

03 {

COMMON MISTAKES TO AVOID IN CP AS A BEGINNER

}

All practising, No learning </1>

- Becoming better in CP is not only about solving problems.
- Unless you keep learning CP related topics, you’re bound to get stuck

All Learning , No practising </2>

- Knowledge of a particular topic is invaluable unless you can code it during a live contest or interview.
- Must know when and where to apply an algorithm.

Not Participating in Contests </3>

- Contests help understand our position in real world
- No way to evaluate your growth without participating in a contest

Plagiarism/Cheating </4>

- Don’t run after ratings, work on your skills and you’ll see improved results.
- Will never know your actual level if you copy solutions

NOT UPSOLVING </5>

- Upsolving means solving problems of the contest just after its over
- Important for self growth else you'll be stuck at the same level

Inconsistency </6>

- Inconsistency is one of the biggest mistakes that beginners make
- Various consequences like forgetting concepts, lack of motivation etc

04 {

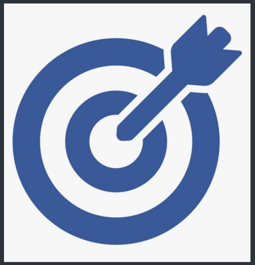
[IMPORTANCE AND VALUE OF CONTESTS]

}

Tests Your Endurance < /1>

< Most CP contests last for over 2 hours testing your mental endurance to code for a long period of time under pressure. >

Helps in Improving Strategy < /2>



< Giving more CP contests can help in improving one’s contest attempting strategy like an effective strategy may be to attempt the most solved problem first rather than getting stuck on a hard problem initially. >

Tests Your Mental Strength < /3>

< The key remains to always stay calm and not panic if you’re unable to solve a problem, remember that the goal is not always to solve all the problems. >

Provides Motivation < /4>

< CP gives us an indication of where we stand among the top coders, and can motivate a person to keep working hard to eventually reach that level. >

Modern Problems < /5>

< The problems asked in contests don’t generally repeat and mostly represent the modern problem set which can help in your interview preparation as well. >

Speed And Efficiency < /6>

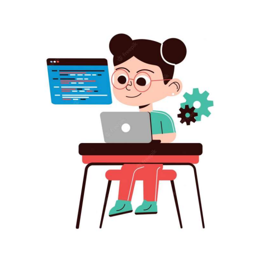
< CP can also help in improving one’s speed in solving questions as the person who has solved the same question in less time and more efficiently will achieve a higher rating. >

05 {

[WHAT TO DO AFTER A CONTEST?]

}

Upsolving < /1>



< Upsolving means solving all the problems you couldn't solve during the contest. If you don't work on these problems after the end of the contest, you're basically in the same place and have gained nothing from the contest. >

Analyse Your Performance < /2>

© CanStockPhoto.com

< Your main motive should be to learn and to keep getting better therefore analysis of your performance and identifying weak areas remains key. >

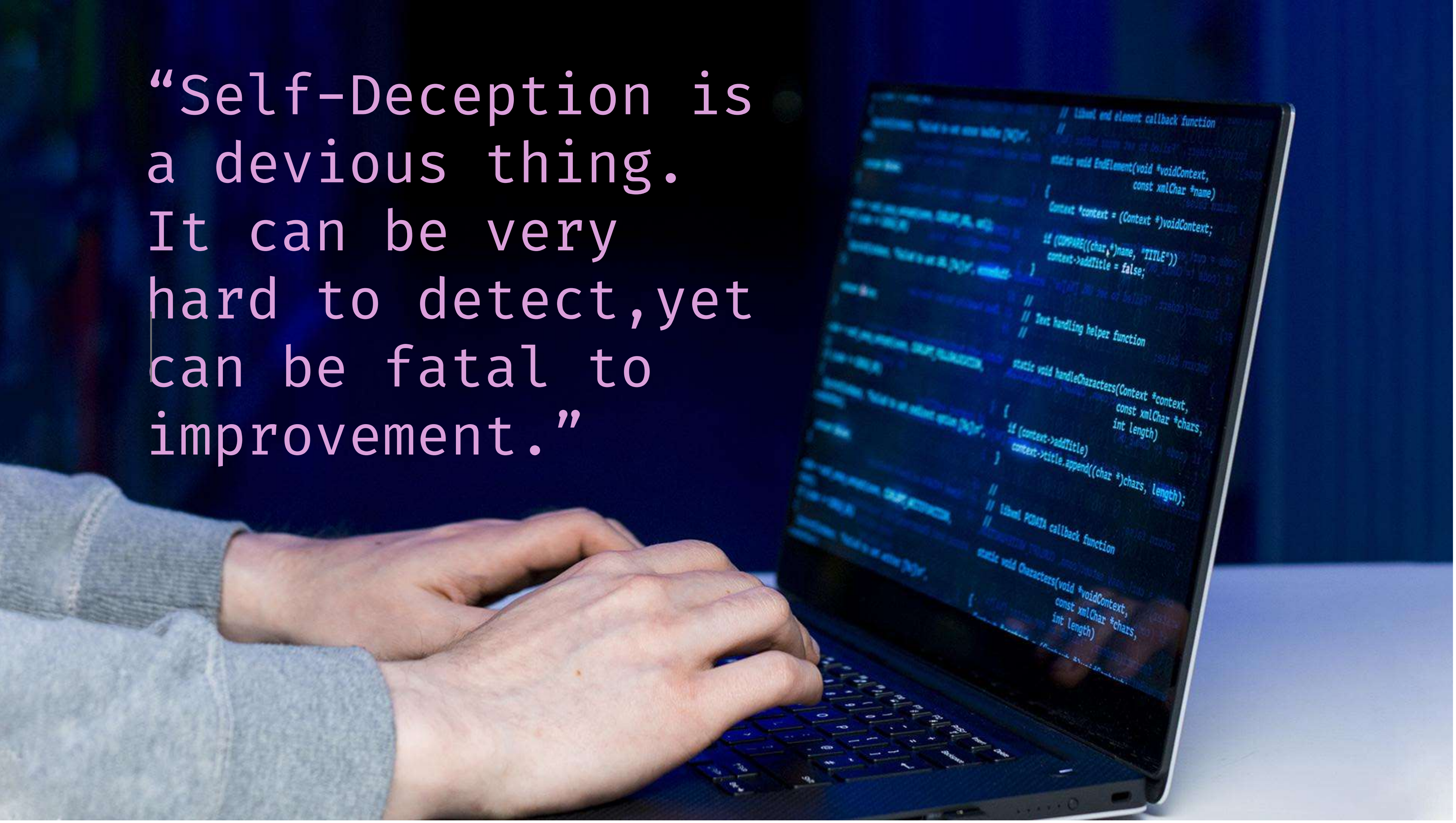
Work on Weak Subjects < /3>

< Try to solve a series of problems related to the topics you were unable to solve during the contest so you don’t get stuck on a similar problem again. >

Learning from Failure < /4>

< Don’t get upset if you achieve negative rating in a contest and keep giving contests consistently. >

“Self-Deception is
a devious thing.
It can be very
hard to detect,yet
can be fatal to
improvement.”


// Libxml end element callback function
static void EndElement(void *voidContext,
const xmlChar *name)
Context *context = (Context *)voidContext;
if (COMPARE((char *)name, "TITLE"))
context->addTitle = false;
// Text handling helper function
static void handleCharacters(Context *context,
const xmlChar *chars,
int length)
if (context->addTitle)
context->title.append((char *)chars, length);
// Libxml PCDATA callback function
static void Characters(void *voidContext,
const xmlChar *chars,
int length)

Top CPers: {

Gennady Korotkevich (tourist)

< Undisputedly the best competitive programmer out there />

Benjamin Qi (Benq)

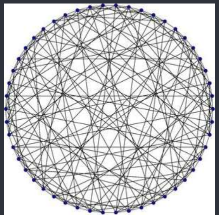
< Currently highest rated user on codeforces, writes clean and easy to understand code />

Lingyu Jiang (jiangly)

< Writes clean code and generally implements solution cleverly to make code short />

}

Some more top-coders : {



Don’t know real name (Peti1234)

< writes clean C++ code />

Alex Danilyuk (Um_nik)

< uploads contest faceCam on his youtube channel with explanation of code />

Pranav (mePranav)

< regularly uploads contest editorials on his youtube channel />

}

Good Youtube channels{

Priyansh Aggarwal

<Guidance , editorials and classes on some selected topics >

Utkarsh Gupta

<Guidance and atCoder DP editorials >

A Code Daily

<Regular editorials and theoretical videos (also has a discord server for doubts)>

Colin Galen

<Topic streams , guidance and theoretical videos >

}

Some other youtube channels {

* Errichto : Contest face-cams
* Um_nik : Regular Contest face-cams
* Vivek Gupta : DP playlist , guidance
* Rachit Jain : Guidance and lectures
* CodeNCode : Topic playlists
* Luv : Dedicated CP playlist
* Pavel Mavrin : Educator from CF-edu

}

How to find Resources ?{

< Let’s say , you encounter an algorithm while studying from an editorial/video and you never even heard of that algorithm >

Now, What next ?

< You should simply study that algorithm ⛳ >

}

Where to study from ? {

- If you like reading blogs , then you can simply search `topic_name codeforces` on Google.
- Another resource can be CP-Algo and USACO guide.
- However, if you are not familiar that way , you can find if the earlier mentioned youtube channels have taught that topic.
- Another resource for video lectures is Codeforces Edu Section , which also contains problems alongside lectures.
- Finally, to test out your understanding of the topic, you can again search `topic_name codeforces problems` and will most likely find a corresponding blog

}

Some excellent resources : {

For theory and practice:

- <USACO guide> → completely detailed and guided resource for CP (contains problems also)

For problem practice:

- <CSES problem set> → best problem set to prepare topic-wise ; highly educational problems; however , no official editorials; (must try) ;

For tracking upcoming contests:

- <CLIST> → website to track upcoming contests over different platforms ; extremely helpful ;

}

Why C++ ? {

C++ is mostly preferred over Java/Python for CP.

C++ is a simple language and is much easier to write; C++ is compiled to binaries, so it runs immediately and therefore faster.

C++ has biggest peer group in CP, so it would be easier for you to understand/stalk other’s solutions.

C++ also has in-built STL , which provides a lot of pre-written code which saves a lot of time while coding the solution.

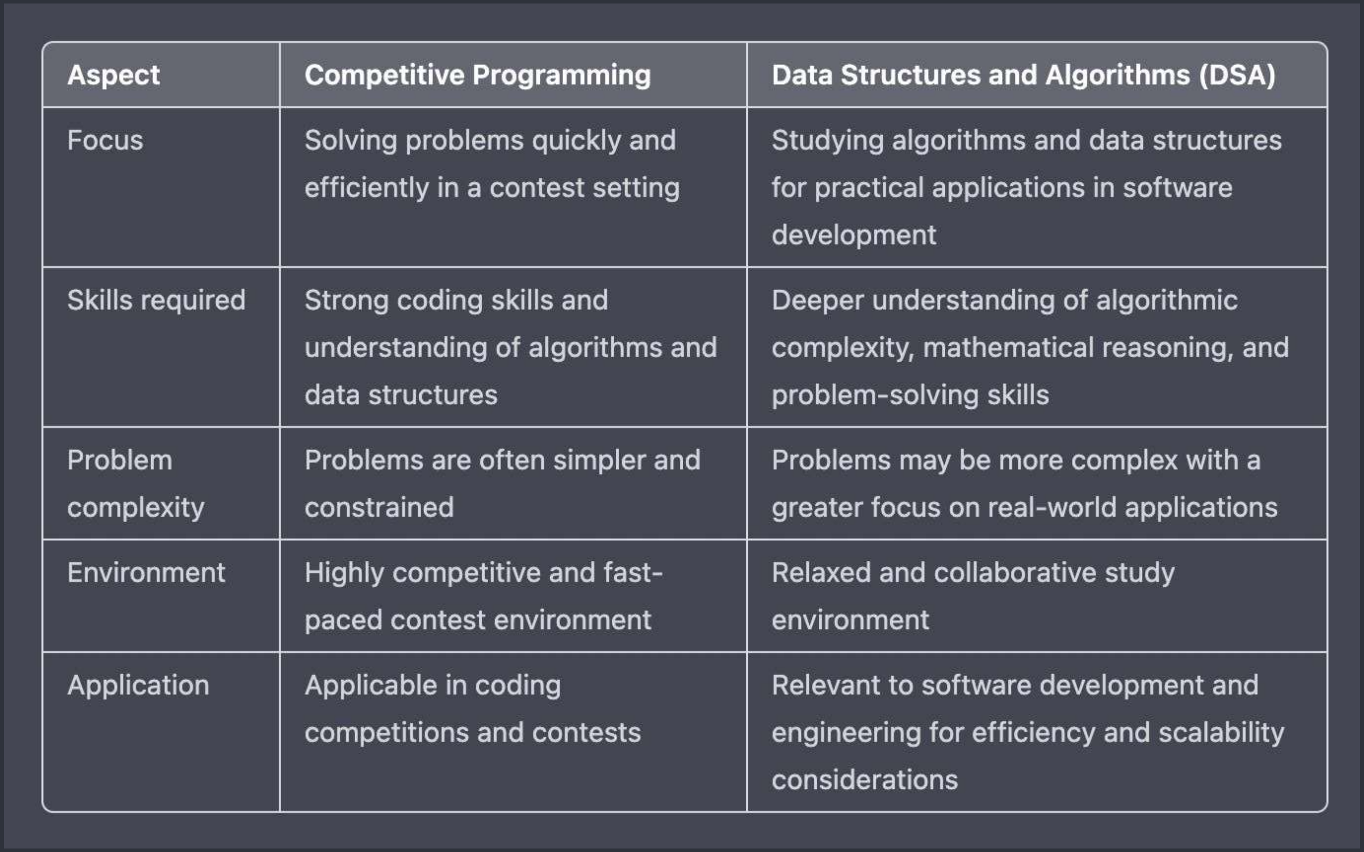
}

03 {

[DSA VS CP]

What’s the difference?

}



Aspect	Competitive Programming	Data Structures and Algorithms (DSA)
Focus	Solving problems quickly and efficiently in a contest setting	Studying algorithms and data structures for practical applications in software development
Skills required	Strong coding skills and understanding of algorithms and data structures	Deeper understanding of algorithmic complexity, mathematical reasoning, and problem-solving skills
Problem complexity	Problems are often simpler and constrained	Problems may be more complex with a greater focus on real-world applications
Environment	Highly competitive and fast-paced contest environment	Relaxed and collaborative study environment
Application	Applicable in coding competitions and contests	Relevant to software development and engineering for efficiency and scalability considerations

04 {

[MYTHS ABOUT CP]

}

forbeginners.cpp

workshop.py

Competitive Programming is Compulsory for Cracking the Tech-Giants.

Competitive Programming can Only be Done in “X” Language.

You need to be great at Maths to excel at Competitive Programming.

Competitive Programming is only for Computer Science majors.

Competitive programming is a solitary activity.

Programming Language