



Servidor de Treino do ONI

Apenas para alunos (e professores) inscritos. Se não for "elegível" e desejar ter uma conta para submissão, por favor contacte a organização.

O servidor está a aceitar submissões em C, C++, Java e Python. Nótem no entanto que os limites de tempo de execução em cada problema são os limites reais usados na prova em causa, refletindo apenas as linguagens de programação que eram permitidas na altura, bem como o servidor de avaliação da altura (ex: Python é um pouco mais lento e poderá ser impossível ter 100 pontos em todos os problemas com os limites de tempo colocados; o atual servidor é também um pouco mais "lento"). Para qualquer dúvida específica basta contactarem a organização.

De momento temos disponíveis os seguintes 76 problemas:

- 12 problemas introdutórios (que acompanham o [guia de iniciação às Olimpíadas](#))
- 35 problemas das qualificações de 2017 a 2024
- 29 problemas das finais de 2016 a 2025

[Fazer login num concurso de treino](#)
[Ver resultados como espectador](#)

Para fazer perguntas sobre os problemas ou sobre o sistema usem o [Discord das ONI](#). Se não o estiverem a conseguir fazer, ou para qualquer coisa mais urgente, podem usar o contacto oni_at_dcc.fc.up.pt.

Olimpíadas Nacionais de Informática

Pedro Ribeiro (DCC/FCUP – Universidade do Porto)



Com o Alto Patrocínio
de Sua Excelência



Pedro Ribeiro - Quem Sou Eu?



Professor na **Universidade do Porto**

- Diretor do **Mestrado em Ciência de Computadores**
- Investigação em **Algoritmos** e em **Ciência de Redes** (*grafos*)



Departamento de Ciência de Computadores

Pedro Ribeiro - Quem Sou Eu?

- Estou há muitos anos ligado a **concursos de programação**

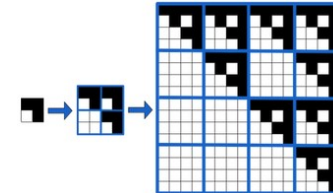
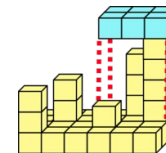
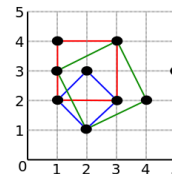
Como **participante** (comecei no 9º ano)
(2º lugar ONI'95, 1º lugar nas ONI 96, 97 e 98)
[representei Portugal nas IOI por 4 vezes]



Como **mentor/treinador e líder** (de delegação)
(lidero a delegação portuguesa nas IOI desde 2025)
[lidero e treino equipas da minha universidade]



Como **organizador**
(final no “meu” DCC desde 2006)
[inúmeros outros concursos para várias faixas etárias]



37	36	35	34	33	32	31
38	17	16	15	14	13	30
39	18	5	4	3	12	29
40	19	6	1	2	11	28
41	20	7	8	9	10	27
42	21	22	23	24	25	26
43	44	45	46	47	48	49 ...

Porquê?



Missão das Olimpíadas de Informática

- Captar e Desenvolver **talento** na Informática



José Santos (IOI'98)
Principal Software Engineer
at **Microsoft**



João Oliveirinha (IOI'03)
Sr Engineering Manager
at **Cloudflare**



André Santos (IOI'06)
Software Engineer
at **Google**



Gonçalo Paredes
(IOI'14,15,16)
Sr. Engineer
at **TripAdvisor**



Pedro Paredes
(IOI'11,12)
Professor na
Univ. Princeton

Pedro Dias (IOI'19,20) U. Cambridge	Rui Whang (IOI'20) U. Oxford	Tiago Marques (IOI'21,22) MIT
--	---------------------------------	----------------------------------

- Promover o gosto pelos **algoritmos** e pela **programação**
- Selecionar e preparar os concorrentes portugueses para as **IOI** (e **WEOI**, **OII** e **EGOI**)
- Uma **comunidade** para quem gosta destes temas

ONI (Olimpíadas Nacionais de Informática)

- ONI já vão na sua **38ª edição**



- Seleccionam para as **IOI (Olimpíadas Internacionais de Informática)**



- uma das grandes Olimpíadas Internacionais de Ciências
- 4ª mais antiga, desde 89 (IMO desde 59, IPhO desde 67, IChO desde 68)

IOI (International Olympiad in Informatics)

- As Olimpíadas são uma **experiência marcante**



Fases das ONI 2026



Fases das ONI 2026

Com patrocínio extra de:

axians



C++, Python



12-18 maio (presencial)

2 a 4 meninas

6 participantes
Potencialmente
+ 2 meninas
C++



Luxemburgo

26-28 junho (presencial)

(aumentar participação feminina!)

Inscrições

Até 15 de março (online)

Qualificação

C/C++, Java, Python

18, 19, 20 de março (local + online)

30 participantes

+ 6 meninas

C/C++, Java, Python

Final Nacional

18 de abril (presencial, DCC/FCUP)

8 a 10 participantes

+ 2 a 4 meninas

C++

Prova de Seleção

final de maio (presencial, local a definir)

4 participantes

C++



UZBEKISTAN

IOI 2026 TASHKENT

9-16 agosto (presencial)

todos da seleção



junho/julho (remoto)

O sistema de submissão usado nas ONI

- Nas ONI usamos o **Mooshak** (que é semelhante a outras plataformas usadas em programação competitiva)



#	Country	Team	Problems											Solved Points
			A	B	C	D	E	F	G	H	I	J	K + L	
1	🇧🇷	UFRJ	100 (1)	100 (1)	100 (3)	100 (2)	100 (7)	100 (5)	100 (2)	100 (3)	100 (4)	100 (1)	100 100	1200
2	🇧🇷	FEUC	100 (1)	100 (1)	100 (1)	100 (1)	60 (5)	100 (2)	100 (1)	100 (1)	100 (1)	100 (1)	100 100	1180
3	🇧🇷	FEUC	100 (3)	100 (2)	100 (2)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 (1)	100 100	1180
4	🇧🇷	FEUC	100 (3)	100 (1)	100 (1)	100 (1)	0 (2)	100 (3)	100 (2)	100 (4)	100 (3)	100 (2)	100 100	1180
5	🇧🇷	FEUC	100 (3)	100 (2)	100 (2)	100 (1)	100 (5)	100 (3)	100 (3)	100 (4)	100 (2)	100 (2)	100 100	1180
6	🇧🇷	FEUC	100 (8)	100 (12)	100 (2)	100 (2)	100 (9)	100 (3)	100 (1)	100 (2)	100 (1)	100 (1)	100 100	1180
7	🇧🇷	FEUC	100 (2)	100 (2)	100 (1)	100 (1)	100 (4)	100 (3)	100 (2)	100 (2)	100 (2)	100 (2)	100 100	1000
8	🇧🇷	FEUC	100 (1)	100 (1)	100 (1)	100 (1)	100 (2)	100 (2)	100 (2)	100 (3)	100 (1)	100 (1)	100 100	900
9	🇧🇷	FEUC	100 (4)	100 (1)	100 (1)	100 (2)	100 (2)	100 (5)	100 (5)	100 (5)	100 (5)	100 (5)	100 100	700
10	🇧🇷	FEUC	100 (2)	100 (1)	100 (3)	100 (3)	100 (1)	100 (3)	100 (3)	100 (3)	100 (3)	100 (3)	100 100	650
11	🇧🇷	FEUC	100 (2)	100 (2)	100 (1)	100 (6)	0 (10)	100 (3)	100 (3)	100 (4)	0 (3)	100 (2)	100 100	610
12	🇧🇷	FEUC	100 (7)	100 (4)	100 (1)	100 (2)	100 (3)	100 (3)	100 (3)	100 (3)	100 (3)	100 (3)	100 100	600
13	🇧🇷	FEUC	100 (3)	100 (1)	100 (1)	100 (2)	0 (3)	100 (10)	100 (5)	100 (5)	100 (5)	100 (5)	100 100	600



- Na sua essência:
 - Submetemos** código-fonte
 - O programa é automaticamente **compilado** e **executado**
 - As respostas dadas pelo vosso programa para um **conjunto de testes** são comparadas com as respostas esperadas
 - A **pontuação** depende dos testes onde “passaram”

O sistema de submissão usado nas ONI

- Vamos agora fazer uma pequena **demonstração interativa** do seu uso, incluindo a obtenção dos seguintes resultados:
 - Accepted (teste aceite)
 - Compile Time Error (o programa não compila)
 - Presentation Error (existem problemas com “*whitespace*”)
 - Wrong Answer (resposta errada)
 - Runtime Error (programa “crasha” durante a execução)
 - Time Limit Exceeded (o problema excedeu o tempo limite)



S887	738:00:54		AE_Arraiolos Diogo Esteves	B	C++	100 Accepted	final
S886	737:56:37		AE_Arraiolos Diogo Esteves	A	C++	100 Accepted	final
S885	737:53:29		AE_Arraiolos Diogo Esteves	A	C++	0 Presentation Error	final
S884	737:49:42		AE_GMachado Prikshit Sahni	B	C	84 Wrong Answer	final
S883	737:48:17		AE_GMachado Prikshit Sahni	B	C	52 Wrong Answer	final
S882	737:47:10		AE_GMachado Prikshit Sahni	B	C	0 Compile Time Error	final
S881	737:32:02		AE_GMachado Miguel Henriques	L	Python	100 Accepted	final
S880	737:30:43		AE_GMachado Miguel Wheeler	D	C++	0 Compile Time Error	final
S879	737:29:41		AE_GMachado Miguel Wheeler	D	C++	0 Compile Time Error	final
S878	737:29:14		AE_GMachado Miguel Henriques	L	Python	40 Time Limit Exceeded	final

Anatomia de um problema de Olimpíadas

- Os problemas são de **caracter algorítmico**
 - Não chega estar “*correto*”, é preciso ser **eficiente** ao nível de **tempo e memória**
- Os problemas podem ter **várias subtarefas**:
 - Restrições diferentes que admitem soluções diferentes (ex: menos eficientes, casos específicos, etc)
- Existem vários tipos de problemas
 - “Batch Problems” (clássicos) – Input / Output
 - Output-Only; Interação; ...

Exemplo de Problema

Problemas anteriores disponíveis no nosso website: <https://oni.dcc.fc.up.pt/problemas>

- Vejamos um **exemplo** de problema (Qualificação'2020)

<https://www.dcc.fc.up.pt/oni/problemas/2020/qualificacao/probA.html>

Problema A - Corridas de Berlindes

Este é problema usa o novo formato de problemas.

Consulta a [página de instruções](#) para informações detalhadas sobre como funciona este novo formato.

Apesar de pouco falado, poucos desportos têm ganho mais espetadores do que as corridas de berlindes (em especial as Olimpíadas da Natação Intensiva). Neste desporto há N berlindes que competem entre si para ver quem é o mais rápido. Como responsável por organizar as corridas das ONI, não podes perder esta oportunidade para mostrar aos novos fãs o melhor que o desporto tem para dar! Para tal, queres organizar as mais diversas e entusiasmantes corridas - os fãs não gostam que as corridas sejam iguais! Neste caso, a diversidade mede-se pelas características dos berlindes: o i -ésimo berlinde tem uma agilidade A_i e uma velocidade V_i , ambos números inteiros.

Parte I

Para manter a competição interessante queres evitar berlindes iguais, ou seja, com a mesma agilidade e a mesma velocidade. Isto garante que as corridas são o mais variadas possível. Como responsável pela organização, queres saber, dado os teus N berlindes, quantos berlindes diferentes tens?

Exemplo

Se $N = 5$ e os berlindes tiverem as seguintes agilidades e velocidades, respetivamente: (1, 2), (3, 4), (1, 2), (1, 4), (3, 2), então há 4 berlindes diferentes porque há dois berlindes com agilidade 1 e velocidade 2.



- **Input:** N berlindes, cada um com uma agilidade A_i e uma velocidade V_i

Exemplo de Problema

Solução foi abordada no webinar – explicação detalhada disponível em:
https://oni.dcc.fc.up.pt/loop/solucoes/2020/qualificacao/prob_a.html

- **Parte I**

Quantos berlindes **diferentes** existem?

- Subtarefa 1: $N \leq 1\,000$
- Subtarefa 2: $N \leq 100\,000$

Solução **bruta** em $O(N^2)$: verificar todos os pares

Solução **eficiente**:

- Em $O(N)$: criar array bidimensional (*restrição chave: $0 \leq A_i, V_i \leq 10$*)
- Em $O(N \log N)$: colocar num **set** (existe equiv. em C++/Java/Python)

Exemplo de Problema

Solução foi abordada no webinar – explicação detalhada disponível em:
https://oni.dcc.fc.up.pt/loop/solucoes/2020/qualificacao/prob_a.html

• Parte II

Se todos combatem contra todos e um berlinde ganha a outro quando tem simultaneamente melhor agilidade e velocidade, **quantas vitórias tem o “campeão”?**

- Subtarefa 1: $N \leq 1\,000$
- Subtarefa 2: $N \leq 100\,000$

Solução **bruta** em $O(N^2)$: verificar todos os pares

Solução **eficiente** em $O(N \times B)$: criar array bidimensional

- onde B é o número de possíveis berlindes (no máximo são... 121)
(*restrição chave novamente é $0 \leq A_i, V_i \leq 10$*)

Exemplo de Problema

Solução foi abordada no webinar – explicação detalhada disponível em:

https://oni.dcc.fc.up.pt/loop/solucoes/2020/qualificacao/prob_c.html

- Vejamos um **outro exemplo** de problema (Qualificação'2020)

<https://www.dcc.fc.up.pt/oni/problemas/2020/qualificacao/probC.html>

Problema C - Rua das Flores

Este é problema usa o **novo formato de problemas**.

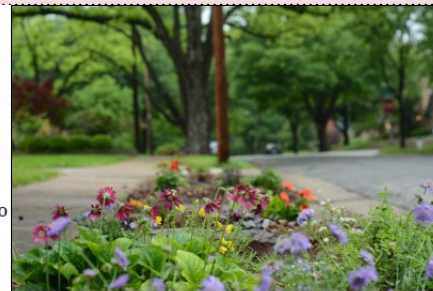
Consulta a [página de instruções](#) para informações detalhadas sobre como funciona este novo formato.

A Rua das Flores é uma rua muito longa (tem N metros de comprimento!) e que há muito tempo está em terríveis condições. No entanto, e felizmente para os condutores que percorrem essa rua frequentemente, o presidente da freguesia prometeu fazer obras nessa rua para ela deixar de ser um problema no quotidiano dos seus transeuntes.

A rua pode ser vista como uma sequência S de N inteiros positivos, onde o i -ésimo, S_i , corresponde ao custo de arranjar o i -ésimo metro da rua. Fazer obras em toda a rua pode exceder o orçamento da junta de freguesia, por isso estão a ser estudadas 3 opções distintas para a obra.

Parte I

Uma das formas de cumprir o plano eleitoral é fazer obras de recuperação num **segmento contíguo** de S de comprimento exatamente K metros. Recordem que um segmento contíguo é um intervalo de elementos seguidos, por exemplo para a sequência $[1, 2, 2, 3]$ a subsequência $[1, 2, 2]$ é uma subsequência contígua (que contém os primeiros três elementos) mas $[1, 2, 3]$ já não (contém o primeiro, segundo e quarto elementos). Entre todos os segmentos contíguos com K metros de comprimento, pretende-se descobrir aquele que tem o custo total mínimo de recuperação.



- No webinar iremos apenas falar da parte I
(e do conceito de **sliding window**)

Analizando a Eficiência

- Para **analisar a eficiência** usamos uma notação (**Big O**) que serve para **caracterizar a complexidade** de um **algoritmo**:
 - Plataforma Loop:
 - <https://oni.dcc.fc.up.pt/loop/guias/inicial/efi/>
 - <https://oni.dcc.fc.up.pt/loop/guias/base/introalg/>
 - Aulas Universitárias:
 - [Slides de uma aula minha](#) (e um vídeo meu)
 - [Big O Cheat Sheet](#)
 - Livros:
 - [Principles of Algorithmic Problem Solving](#) (chapter 5)
 - [Competitive Programmer's Handbook](#) (chapter 2)

Recursos de Treino

- Como me posso então **preparar/treinar**?

Um **misto** entre:

- **Aprender** novas coisas
 - Complexidade, algoritmos, estruturas de dados
 - Livros, videos, blogs
- **Resolver problemas** fora de uma prova
 - Pré-categorizados para cimentar conhecimento
 - Sem saber a categoria (*problem solving*)
- **Participar em provas**
 - Participar em provas com tempo a contar
 - No final ler editoriais/fazer *upsolving*

Não existe “varinha mágica” – **trabalhar, trabalhar!**

Recursos de Treino

- Página online já contém ligações para muitos **recursos de treino**

(quase que existe “*informação a mais*” no universo da Programação Competitiva)

https://oni.dcc.fc.up.pt/plataforma_treino

Plataforma Loop

Em 2016 lançámos o projeto Loop, com o objetivo de auxiliar o treino dos concorrentes das ONI. O Loop oferece artigos sobre várias áreas de informática relevantes para as ONI, categorizados por tópicos e dificuldades e ainda com sugestões de problemas a resolver. Também inclui artigos mais básicos, com temas tão base como aprender a programar ou a configurar um ambiente de programação

- Entrar no Loop
 - Guias Disponíveis no Loop
 - Guia de Iniciação
 - Algumas Soluções para problemas das ONI

Outros Recursos

Para além do Loop e dos seus guias, aqui fica uma lista de ligações interessantes para explorares (conteúdo em inglês):

- *Awesome Competitive Programming* (A curated list of awesome Competitive Programming, Algorithm and Data Structure resources)
- *USACO Guide* | resources (A free collection of curated, high-quality resources to take you from Bronze to Platinum and beyond.)
- **Principais Comunidades/Online Judges de Programação Competitiva**
 - CodeForces
 - AtCoder
 - CSES
 - Kattis
 - DMOJ
- **Principais Livros Recomendados (disponíveis gratuitamente):**
 - Principles of Algorithmic Problem Solving
 - Competitive Programmer's Handbook | PDF
 - Competitive Programming | PDF 2nd Edition
- **Tutoriais/Visualização de Algoritmos:**
 - TopCoder: Competitive Programming Tutorials
 - MAXimal: E-Maxx Algorithms in English
 - PEG: PEG Wiki
 - AlgoWiki: AlgoWiki
 - Geek for Geeks: Fundamentals of Algorithms
 - CodeForces: All the good tutorials found for Competitive Programming
 - Codechef: Data Structures and Algorithms
 - Visualgo: Visualising Data Structures and Algorithms through Animation

Recursos de Treino - Loop

- Conjunto de Guias e Recursos de Treino

<https://oni.dcc.fc.up.pt/loop/>

loop | 

GUIAS CONCURSOS SOBRE SELEÇÃO SOLUÇÕES ONI

Iniciação (Nível 0)

Para quem nunca programou, ou ainda não se sente confortável com os básicos da programação. Nesta secção é possível aprender a preparar um ambiente de programação (compilador, editores de código, ...) em vários sistemas operativos, os básicos de programação assim como alguns conceitos mais avançados de programação.

[Começar a aprender!](#)

Base (Nível 1)

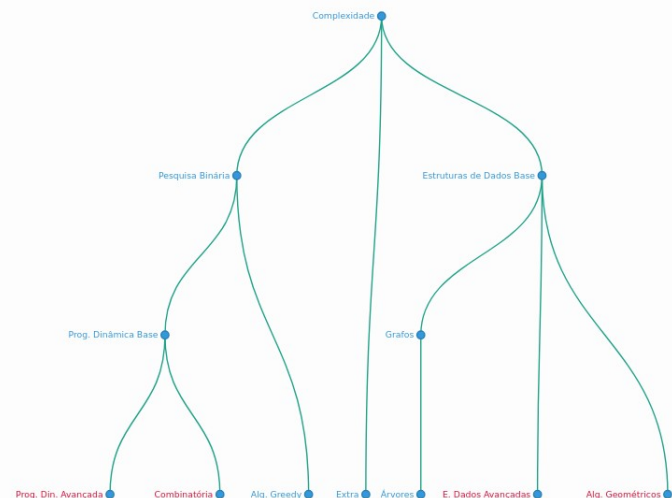
Para quem já sabe programar bem e quer começar a aventurar-se com os conceitos algorítmicos mais simples. Nesta secção atravessa-se um conjunto de temas importantes para construir o leque de ferramentas base de um *problem solver*. Desde ordenação e pesquisa às estruturas de dados mais simples, mas úteis. Quem dominar estes temas não terá dificuldade a passar à fase final das ONI.

[Começar a aprender!](#)

Intermédio (Nível 2)

Para quem já tem os blocos algorítmicos base consolidados e quer aprofundar estes conhecimentos para poder resolver problemas mais difíceis e interessantes. Nesta secção discutem-se temas como grafos ou programação dinâmica, assim como algumas estruturas de dados mais avançadas. Para ter uma hipótese de ser selecionado para a Olimpíada Internacional de Informática é importante dominar estes tópicos.

[Começar a aprender!](#)



Recursos de Treino - Livros

• Principles of Algorithmic Problem Solving

<https://jsannemo.se/latest.pdf>

Contents

Introduction	vii	5 Time Complexity	67	10.2 Extreme Values	156
I Preliminaries	1	5.1 The Complexity of Insertion Sort	67	10.3 Sorting and Exchanges	159
1 Algorithms and Problems	3	5.2 Asymptotic Notation	70	10.4 Intervals	162
1.1 Computational Problems	3	5.3 NP-complete Problems	74	10.5 Constructions	166
1.2 Algorithms	4	5.4 Other Types of Complexities	74	11 Dynamic Programming	173
1.3 Programming Languages	7	5.5 The Importance of Constant Factors	75	11.1 Making Change Revisited	173
1.4 Pseudo Code	8	6 Data Structures	77	11.2 Paths in a DAG	175
1.5 Online Judges	9	6.1 Dynamic Arrays	77	11.3 Standard Techniques	178
2 Programming in C++	11	6.2 Stacks	81	11.4 Standard Problems	193
2.1 Hello World!	11	6.3 Queues	82	12 Divide and Conquer	201
2.2 Variables and Types	14	6.4 Priority Queues	83	12.1 Recursive Constructions	201
2.3 Input and Output	18	6.5 Bitsets	87	12.2 Sequences	205
2.4 Operators	19	6.6 Hash Tables	88	12.3 Binary Search	208
2.5 If Statements	21	7 Recursion	95	12.4 Centroids	213
2.6 For Loops	23	7.1 Recursive Definitions	95	13 Data Structures	225
2.7 While Loops	25	7.2 The Time Complexity of Recursive Functions	98	13.1 Union-Find	225
2.8 Functions	25	7.3 Choice	99	13.2 Range Queries	233
2.9 Structures	28	7.4 Multidimensional Recursion	103	13.3 Sliding Windows	239
2.10 Arrays	31	7.5 Recursion vs. Iteration	104	III Other Topics	241
2.11 Lambdas	33	8 Graph Theory	107	14 Graph Algorithms	243
2.12 The Preprocessor	34	8.1 Graphs	107	14.1 Weighted Shortest Path	243
3 The C++ Standard Library	37	8.2 Representing Graphs	110	14.2 Eulerian Walks	257
3.1 Data Structures	37	8.3 Breadth-First Search	112	14.3 The Depth-First Search	259
3.2 Math	45	8.4 Depth-First Search	120	14.4 Minimum Spanning Trees	271
3.3 Algorithms	45	8.5 Trees	123	15 Maximum Flows	279
3.4 Strings	47	8.6 Topological Sorting	127	15.1 Flow Networks	279
3.5 Input/Output	49			15.2 Edmonds-Karp	281
4 Implementation Problems	53			15.3 Applications of Flows	283
4.1 Structuring your Code	53			16 Game Theory	287
				16.1 Combinatorial Games	287
				16.2 Mathematical Techniques	288
				16.3 Game Graphs	296
				16.4 Cyclic Games	298

Recursos de Treino - Livros

• Competitive Programmer's Handbook

<https://cses.fi/book/book.pdf>

Contents

Preface

ix

I Basic techniques

1

1 Introduction

3

- 1.1 Programming languages 3
- 1.2 Input and output 4
- 1.3 Working with numbers 6
- 1.4 Shortening code 8
- 1.5 Mathematics 10
- 1.6 Contests and resources 15

2 Time complexity

17

- 2.1 Calculation rules 17
- 2.2 Complexity classes 20
- 2.3 Estimating efficiency 21
- 2.4 Maximum subarray sum 21

3 Sorting

25

- 3.1 Sorting theory 25
- 3.2 Sorting in C++ 29
- 3.3 Binary search 31

4 Data structures

35

- 4.1 Dynamic arrays 35
- 4.2 Set structures 37
- 4.3 Map structures 38
- 4.4 Iterators and ranges 39
- 4.5 Other structures 41
- 4.6 Comparison to sorting 44

5 Complete search

47

- 5.1 Generating subsets 47
- 5.2 Generating permutations 49
- 5.3 Backtracking 50
- 5.4 Pruning the search 51
- 5.5 Meet in the middle 54

6 Greedy algorithms

57

- 6.1 Coin problem 57
- 6.2 Scheduling 58
- 6.3 Tasks and deadlines 60
- 6.4 Minimizing sums 61
- 6.5 Data compression 62

7 Dynamic programming

65

- 7.1 Coin problem 65
- 7.2 Longest increasing subsequence 70
- 7.3 Paths in a grid 71
- 7.4 Knapsack problems 72
- 7.5 Edit distance 74
- 7.6 Counting tilings 75

8 Amortized analysis

77

- 8.1 Two pointers method 77
- 8.2 Nearest smaller elements 79
- 8.3 Sliding window minimum 81

9 Range queries

83

- 9.1 Static array queries 84
- 9.2 Binary indexed tree 86
- 9.3 Segment tree 89
- 9.4 Additional techniques 93

10 Bit manipulation

95

- 10.1 Bit representation 95
- 10.2 Bit operations 96
- 10.3 Representing sets 98
- 10.4 Bit optimizations 100
- 10.5 Dynamic programming 102

II Graph algorithms

107

11 Basics of graphs

109

- 11.1 Graph terminology 109
- 11.2 Graph representation 113

12 Graph traversal

117

- 12.1 Depth-first search 117
- 12.2 Breadth-first search 119
- 12.3 Applications 121

13 Shortest paths

123

- 13.1 Bellman-Ford algorithm 123
- 13.2 Dijkstra's algorithm 126
- 13.3 Floyd-Warshall algorithm 129

14 Tree algorithms

133

- 14.1 Tree traversal 134
- 14.2 Diameter 135
- 14.3 All longest paths 137
- 14.4 Binary trees 139

15 Spanning trees

141

- 15.1 Kruskal's algorithm 142
- 15.2 Union-find structure 145
- 15.3 Prim's algorithm 147

16 Directed graphs

149

- 16.1 Topological sorting 149
- 16.2 Dynamic programming 151
- 16.3 Successor paths 154
- 16.4 Cycle detection 155

17 Strong connectivity

157

- 17.1 Kosaraju's algorithm 158
- 17.2 2SAT problem 160

18 Tree queries

163

- 18.1 Finding ancestors 163
- 18.2 Subtrees and paths 164
- 18.3 Lowest common ancestor 167
- 18.4 Offline algorithms 170

19 Paths and circuits

173

- 19.1 Eulerian paths 173
- 19.2 Hamiltonian paths 177
- 19.3 De Bruijn sequences 178
- 19.4 Knight's tours 179

20 Flows and cuts

181

- 20.1 Ford-Fulkerson algorithm 182
- 20.2 Disjoint paths 186
- 20.3 Maximum matchings 187
- 20.4 Path covers 190

III Advanced topics

195

21 Number theory

197

- 21.1 Primes and factors 197
- 21.2 Modular arithmetic 201
- 21.3 Solving equations 204
- 21.4 Other results 205

22 Combinatorics

207

- 22.1 Binomial coefficients 208
- 22.2 Catalan numbers 210
- 22.3 Inclusion-exclusion 212
- 22.4 Burnside's lemma 214
- 22.5 Cayley's formula 215

23 Matrices

217

- 23.1 Operations 217
- 23.2 Linear recurrences 220
- 23.3 Graphs and matrices 222

24 Probability

225

- 24.1 Calculation 225
- 24.2 Events 226
- 24.3 Random variables 228
- 24.4 Markov chains 230
- 24.5 Randomized algorithms 231

25 Game theory

235

- 25.1 Game states 235
- 25.2 Nim game 237
- 25.3 Sprague-Grundy theorem 238

26 String algorithms

243

- 26.1 String terminology 243
- 26.2 Trie structure 244
- 26.3 String hashing 245
- 26.4 Z-algorithm 247

27 Square root algorithms

251

- 27.1 Combining algorithms 252
- 27.2 Integer partitions 254
- 27.3 Mo's algorithm 255

28 Segment trees revisited

257

- 28.1 Lazy propagation 258
- 28.2 Dynamic trees 261
- 28.3 Data structures 263
- 28.4 Two-dimensionality 264

Recursos de Treino - Livros

• Competitive Programming

https://www.comp.nus.edu.sg/~stevenha/myteaching/competitive_programming/cp2.pdf

Contents

Foreword	v
Preface	vi
Authors' Profiles and Copyright	xi
Convention and Problem Categorization	xii
List of Abbreviations	xiii
List of Tables	xiv
List of Figures	xv
1 Introduction	1
1.1 Competitive Programming	1
1.2 Tips to be Competitive	2
1.2.1 Tip 2: Quickly Identify Problem Types	4
1.2.2 Tip 3: Do Algorithm Analysis	5
1.2.3 Tip 4: Master Programming Languages	8
1.2.4 Tip 5: Master the Art of Testing Code	10
1.2.5 Tip 6: Practice and More Practice	12
1.2.6 Tip 7: Team Work (ICPC Only)	12
1.3 Getting Started: The Ad Hoc Problems	13
1.4 Chapter Notes	19
2 Data Structures and Libraries	21
2.1 Overview and Motivation	21
2.2 Data Structures with Built-in Libraries	22
2.2.1 Linear Data Structures	22
2.2.2 Non-Linear Data Structures	26
2.3 Data Structures with Our-Own Libraries	29
2.3.1 Graph	29
2.3.2 Union-Find Disjoint Sets	30
2.3.3 Segment Tree	32
2.3.4 Fenwick Tree	35
2.4 Chapter Notes	38
3 Problem Solving Paradigms	39
3.1 Overview and Motivation	39
3.2 Complete Search	39
3.2.1 Examples	40
3.2.2 Tips	41
3.3 Divide and Conquer	47

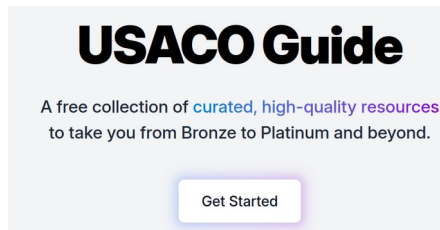
3.3.1 Interesting Usages of Binary Search	47
3.4 Greedy	51
3.4.1 Examples	51
3.5 Dynamic Programming	55
3.5.1 DP Illustration	55
3.5.2 Classical Examples	61
3.5.3 Non Classical Examples	66
3.6 Chapter Notes	70
4 Graph	71
4.1 Overview and Motivation	71
4.2 Graph Traversal	71
4.2.1 Depth First Search (DFS)	71
4.2.2 Breadth First Search (BFS)	72
4.2.3 Finding Connected Components (in an Undirected Graph)	73
4.2.4 Flood Fill - Labeling/Coloring the Connected Components	74
4.2.5 Topological Sort (of a Directed Acyclic Graph)	75
4.2.6 Bipartite Graph Check	76
4.2.7 Graph Edges Property Check via DFS Spanning Tree	76
4.2.8 Finding Articulation Points and Bridges (in an Undirected Graph)	77
4.2.9 Finding Strongly Connected Components (in a Directed Graph)	80
4.3 Minimum Spanning Tree	84
4.3.1 Overview and Motivation	84
4.3.2 Kruskal's Algorithm	84
4.3.3 Prim's Algorithm	85
4.3.4 Other Applications	86
4.4 Single-Source Shortest Paths	90
4.4.1 Overview and Motivation	90
4.4.2 SSSP on Unweighted Graph	90
4.4.3 SSSP on Weighted Graph	91
4.4.4 SSSP on Graph with Negative Weight Cycle	93
4.5 All-Pairs Shortest Paths	96
4.5.1 Overview and Motivation	96
4.5.2 Explanation of Floyd Warshall's DP Solution	96
4.5.3 Other Applications	98
4.6 Maximum Flow	101
4.6.1 Overview and Motivation	101
4.6.2 Ford Fulkerson's Method	101
4.6.3 Edmonds Karp's	102
4.6.4 Other Applications	104
4.7 Special Graphs	107
4.7.1 Directed Acyclic Graph	107
4.7.2 Tree	112
4.7.3 Eulerian Graph	113
4.7.4 Bipartite Graph	114
4.8 Chapter Notes	119
5 Mathematics	121
5.1 Overview and Motivation	121
5.2 Ad Hoc Mathematics Problems	121
5.3 Java BigInteger Class	125
5.3.1 Basic Features	125
5.3.2 Bonus Features	126

5.4 Combinatorics	129
5.4.1 Fibonacci Numbers	129
5.4.2 Binomial Coefficients	130
5.4.3 Catalan Numbers	131
5.4.4 Other Combinatorics	131
5.5 Number Theory	133
5.5.1 Prime Numbers	133
5.5.2 Greatest Common Divisor (GCD) & Least Common Multiple (LCM)	135
5.5.3 Factorial	136
5.5.4 Finding Prime Factors with Optimized Trial Divisions	136
5.5.5 Working with Prime Factors	137
5.5.6 Functions Involving Prime Factors	138
5.5.7 Modulo Arithmetic	140
5.5.8 Extended Euclid: Solving Linear Diophantine Equation	141
5.5.9 Other Number Theoretic Problems	142
5.6 Probability Theory	142
5.7 Cycle-Finding	143
5.7.1 Solution using Efficient Data Structure	143
5.7.2 Floyd's Cycle-Finding Algorithm	143
5.8 Game Theory	145
5.8.1 Decision Tree	145
5.8.2 Mathematical Insights to Speed-up the Solution	146
5.8.3 Nim Game	146
5.9 Powers of a (Square) Matrix	147
5.9.1 The Idea of Efficient Exponentiation	147
5.9.2 Square Matrix Exponentiation	148
5.10 Chapter Notes	149
6 String Processing	151
6.1 Overview and Motivation	151
6.2 Basic String Processing Skills	151
6.3 Ad Hoc String Processing Problems	153
6.4 String Matching	156
6.4.1 Library Solution	156
6.4.2 Knuth-Morris-Pratt (KMP) Algorithm	156
6.4.3 String Matching in a 2D Grid	158
6.5 String Processing with Dynamic Programming	160
6.5.1 String Alignment (Edit Distance)	160
6.5.2 Longest Common Subsequence	161
6.5.3 Palindrome	162
6.6 Suffix Trie/Tree/Array	163
6.6.1 Suffix Trie and Applications	163
6.6.2 Suffix Tree	163
6.6.3 Applications of Suffix Tree	164
6.6.4 Suffix Array	166
6.6.5 Applications of Suffix Array	170
6.7 Chapter Notes	174
7 (Computational) Geometry	175
7.1 Overview and Motivation	175
7.2 Basic Geometry Objects with Libraries	176
7.2.1 0D Objects: Points	176
7.2.2 1D Objects: Lines	177

Recursos de Treino - Wikis/Tutoriais/Visualização

- <https://usaco.guide/>

programa de treino integrado americano
(altamente recomendado)



- <https://cp-algorithms.com/>

repositório de implementações (C++)
de algoritmos com explicação

Algorithms for Competitive Programming

contributors 424 pull requests 27 open pull requests 1.2k closed build passing translation progress 85.2%

The goal of this project is to translate the wonderful resource <https://e-maxx.ru/algo> which provides descriptions of many algorithms and data structures especially popular in field of competitive programming. Moreover we want to improve the collected knowledge by extending the articles and adding new articles to the collection.

We're an ad-free, volunteer-run website that's free for everyone. Users can contribute articles or help sponsor bounties on articles for greater algorithmic coverage. Your help is greatly appreciated.

Compiled pages are published at <https://cp-algorithms.com/>.

- **Wikis/Tutorials:**

- <https://www.geeksforgeeks.org/dsa/dsa-tutorial-learn-data-structures-and-algorithms/>
- <https://wiki.algo.is/>
- <https://wcipeg.com/wiki/Special:AllPages>
- <https://codeforces.com/blog/entry/57282>

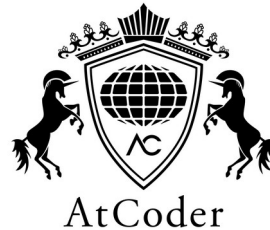
Recursos de Treino – Concursos/Problemsets

- **Codeforces:** <https://codeforces.com/>

A maior comunidade de programação competitiva com concursos regulares (pode-se fazer concursos anteriores e ver editoriais)



- **AtCoder:** <https://atcoder.jp/>



- **CSES:** <https://cses.fi/problemset/>



- **DM:OJ:** <https://dmoj.ca/problems/?search=IOI>



Recursos de Treino - Quero saber mais?

https://oni.dcc.fc.up.pt/plataforma_treino

- <https://github.com/Inishan/awesome-competitive-programming>
A curated list of awesome Competitive Programming, Algorithm and Data Structure resources.
- **IOI Syllabus:** <https://algo.sk/ioi-syllabus/>

The International Olympiad in Informatics Syllabus

The Syllabus aims to achieve these goals by providing a classification of topics and concepts from mathematics and computer science. More precisely, this Syllabus classifies each topic into one of six categories. Ordered by topic suitability, these are:

1 Version and status information

IOI 2024 is using the current official Syllabus version (unchanged since March 2022). This is a draft of the Syllabus version intended for IOI 2025. At the moment no changes other than those already included in this document are expected. At the end of IOI 2024 this document should become the official Syllabus.

- ✓ Included, unlimited
- ✓📄 Included, to be defined
- ✓📄 Included, not for task description
- ? Outside of focus
- ✗? Excluded, but open to discussion
- ✗ Explicitly excluded

Final da Apresentação

- Obrigado pela atenção! :)
- **Contactos:**
 - Website: <https://oni.dcc.fc.up.pt/>
 - Email: oni@dcc.fc.up.pt
 - Discord: <https://discord.com/invite/c8B7WBEQeF>

