ChatGPT said:

To **win contests on Codeforces** in the **0–1199 (Newbie, Pupil) range**, you need to **master core topics and techniques** that appear frequently in **Div 3 and lower Div 2 problems (A/B level)**. These problems usually **don't require advanced algorithms**, but **solid logic, fast implementation**, and **practice with patterns**.

✅ Core Topics for Rating 0–1199 (Newbie to Pupil)

Topic	Why Important
✅ **Implementation**	Most problems are based on simulating what is described.
✅ **Math (Basic Number Theory)**	GCD, LCM, modulo, primes, divisors.
✅ **Greedy Algorithms**	Simple optimal choices (e.g., sorting + decision making).
✅ **Brute Force**	Try all combinations if constraints are small ($n \leq 1000$).
✅ **Two Pointers**	For string and array subrange problems.
✅ **Prefix Sum / Difference Arrays**	For range sum or difference queries.
✅ **Arrays and Strings**	Core data types used in almost every problem.
✅ **Sorting + Binary Search (Basic)**	Key to many greedy and search problems.
✅ **Maps/Sets (Hashing)**	Counting elements, uniqueness, frequency.
✅ **Basic Recursion / Backtracking**	Subset generation, small dfs-style problems.

📈 Practice These Problem Types

Problem Type	Description
✅ **Max/Min subarray, subsegment**	Array/sorting/prefix sum-based
✅ **Frequency Count**	Use maps, arrays, sets
✅ **Greedy sorting**	Sort + simulate (smallest/largest first)
✅ **String manipulation**	Palindrome, character frequency, reverse
✅ **Brute Force search**	Try all possibilities
✅ **Conditions & Loops**	Nested conditions, logic building
✅ **Small Graph Problems (Optional)**	BFS/DFS on small grids (for 1100-1200)

📚 Topic Checklist to Win Div 3 / Improve from 0–1199

- ☑ Arrays (max, min, frequency, reverse, rotate)
- ☑ Strings (substring, palindrome, count chars)

- ☐ Math (divisibility, modulo, GCD/LCM, primes)
- ☐ Prefix Sum / Difference Array
- ☐ Sorting (custom comparators)
- ☐ Hashing (unordered_map, set)
- ☐ Greedy Thinking (always try sort + logic)
- ☐ Bit Manipulation (odd/even, XOR basics)
- ☐ Simple Recursion (factorial, subsets)
- ☐ Basic I/O + Fast implementation

🚀 Tips to Win a Contest in 0–1199

1. **Practice 800–1200 rated problems regularly** – solve at least 3–5 every day.
2. **Upsolve A, B, and C from Div 3 contests**.
3. **Improve typing and debugging speed** – time matters.
4. **Use templates for input/output** – save seconds.
5. **Avoid overthinking** – most problems are simpler than you think.
6. **Always read the constraints** – they guide the solution approach.
7. **Solve A quickly (within 5–10 min)**, then move fast to B and C.

🔥 Bonus: Topicwise Codeforces Practice Links

- Codeforces Problemset: 800–1199
- A2OJ ladders: <1200 problems (unofficial mirror)
- Greedy Problems
- Math Problems
- Implementation Problems

Would you like a **practice sheet with 100 problems (800–1199)** sorted by topic?

You said:

1200 to1399

ChatGPT said:

To **win contests and improve rating from 1200 to 1399 (Specialist)** on Codeforces, you must **go beyond basic implementation** and develop **problem-solving patterns**, **speed**, and **stamina for 3-4 problem contests**. Problems in this range often combine **multiple concepts**, especially for **Problem C and D** in Div 2 and **Problem B/C** in Div 3.

✅ Must-Know Topics for 1200–1399 (Specialist Range)

Core Topics	Why Important
✅ **Binary Search**	Used in many problems involving "minimum/maximum satisfying condition".
✅ **Two Pointers**	For range/subarray problems (e.g., max length with some condition).
✅ **Prefix Sum / Suffix Sum / Difference Array**	Used in range-based counting and optimizations.
✅ **Greedy + Sorting**	Sorting and choosing based on conditions is a classic pattern.
✅ **DFS/BFS on small graphs or grids**	Needed for basic graph traversal questions.
✅ **Math (Mod, GCD, LCM, Divisors, Primes)**	Appears often in constraints and combinatorics.
✅ **Bit Manipulation**	Subset generation, XOR problems, bit-count.
✅ **Hashing (Maps, Sets)**	Frequency counts, unique tracking, hash-based memoization.
✅ **Sliding Window**	Substring, subarray optimizations.
✅ **Simple Dynamic Programming (1D/2D)**	E.g., LIS, Knapsack (small constraints), LCS base cases.

📈 Practice These Problem Patterns

Problem Pattern	Tags
✅ **Binary Search on Answer**	Monotonic condition (can/can't do in k steps)
✅ **Longest Subarray with Condition**	Sliding Window / Two Pointers
✅ **Greedy + Priority Queue**	Optimize cost/value/selection
✅ **Prefix Sum with Count**	Subarray with given sum, count inversions
✅ **DFS/BFS in Grid/Graph**	Connected components, shortest path
✅ **Math + Observation**	Sum of digits, parity, construct numbers
✅ **Sorting + Difference**	Minimum difference, max pairs
✅ **Small DP**	Memoization or bottom-up ($n \leq 1000$)

🎯 Key Goals to Win in This Range

Target	What to Focus
🏁 **Solve Div 2 A, B within 20-30 mins**	Pure implementation/greedy
🏁 **Start solving Div 2 C, D consistently**	Apply Binary Search, Prefix Sum, Two Pointers
🏁 **Learn basic graph traversal**	BFS/DFS on grid or simple graphs
🏁 **Improve problem reading speed**	Spot constraints and hints fast
🏁 **Don't overfit templates**	Understand logic and brute-force first

---

🔥 Must-Solve Topics by Priority

Level 1 (1200–1299)	Level 2 (1300–1399)
✓ **Brute Force with constraints ≤1e3**	✓ Binary Search on Answer
✓ **Greedy with Sorting**	✓ Two Pointers (nested loop → optimized)
✓ **Hash Maps / Sets usage**	✓ DP on subsequence/string
✓ **Prefix Sum / Difference Arrays**	✓ Grid BFS/DFS (4 or 8 directions)
✓ **Bit tricks (XOR swap, set bits)**	✓ Math logic + Modulo tricks

---

🚀 Tips to Improve 1200 → 1399

1. **Upsolve Div 2 C/D or Div 3 C/D after contest** – focus on where you got stuck.
2. **Read editorials if stuck for 30 min post-contest** – understand the idea, not just code.
3. **Practice problems in 1200–1399 range** – aim 3–5/day.
4. **Speed matters now** – so focus on solving A/B in < 20 min total.
5. **Don't skip math/logic problems** – many problems are about simple formulas or patterns.

□ Practice Resources

- 🔗 Codeforces Problems: 1200–1399
- 🔗 Binary Search Practice
- 🔗 Two Pointers Practice
- 🔗 Greedy Problems
- 🔗 Graph (DFS/BFS)

📚 Example 1200–1399 Problem Patterns

Pattern	Sample Problem
Binary Search min/max	Min days to finish work
Prefix Sum Subarray	Max number of 1s by flipping 0s
Two Pointer String Match	Longest good subsequence
Greedy + Sorting	Select tasks with least penalty
Hash Map Count	Max freq with conditions
Bitmask Tricks	Subsets with OR/XOR properties
Graph BFS	Shortest path in unweighted graph

Would you like a **problem sheet (1200–1399) sorted by pattern**, or a **roadmap to reach 1600+ next**?

You said:

1400 to1500

ChatGPT said:

To reach and **win contests in the 1400–1500 Codeforces range (Expert)**, you need to master **combinational logic**, **pattern recognition**, and **clean code with fast implementation**. Problems now **require deeper thinking**, especially in **Div 2 B/C/D** and **Div 3 C/D/E**.

□ Core Skills to Build (1400–1500 Range)

Concept	Importance
✅ **Binary Search on Answer**	Solve "what's the max/min X that satisfies condition".
✅ **Greedy + Invariants**	Prove choices are optimal using logic, not trial-error.
✅ **Two Pointers / Sliding Window**	Used with arrays and strings under constraints.
✅ **Prefix Sums + Hashing**	Subarray/substring matching, frequency checking.
✅ **Bitmasking**	Subset sums, XOR conditions, on/off states.
✅ **Basic DP (1D/2D, with states)**	LIS, LCS, knapsack, segment sums with conditions.
✅ **Graphs (DFS, BFS, Toposort)**	Grids, dependencies, components, simple trees.
✅ **Combinatorics + Math**	Binomial coefficient, mod inverse, number construction.

🎯 Problem Patterns You Must Master

Pattern	Example
🔍 **Binary Search + Greedy Check**	Max k people can sit, max time to finish X
🎯 **DP on Subarrays / Strings**	Longest subsequence, min operations
🔁 **Greedy + Sorting + Conditions**	Select tasks/people based on priorities
⚙ **Bitmask on Small n**	All subsets of 15–20 elements
🔄 **DFS/BFS on Grid or Adjacency List**	Number of components, reachability
□ **Prefix Sum with Count**	Number of pairs with given sum
📐 **Math/Constructive**	Build number with sum/digits/divisibility
🔄 **Cycle Detection / Graph Coloring**	Bipartite graph check, tree check

---

📚 Key Topics List (1400–1500 Tier)

Category	Topics

Greedy	Interval covering, optimal replacement, cost minimization
Number Theory	Modulo, GCD, Primes, Mod Inverse (for large mod)
Binary Search	Minimum/maximum satisfying condition
Sorting + Logic	Task scheduling, minimizing/maximizing gap
Prefix Sums + Hashing	Count substrings or subarrays
Bitmasking	DP with state compression
Graphs	DFS/BFS, Topological Sort, Components
DP (1D, 2D)	Memoization or Tabulation, with transitions
Constructive Algorithms	Problem-specific building logic

How to Improve Fast in 1400–1500

Strategy	Description
Upsolve C/D from Div 2, D/E from Div 3	They're often 1400–1500 in difficulty
Speedrun 1300–1500 problems	Practice 3–5 every day
Use tags, but hide them while solving	Test your skill before looking
After 30 minutes, read hints or editorial	Don’t waste time stuck forever
Build topic-specific notebooks	For Binary Search, DP, Bitmask, etc.

Practice Links for 1400–1500 Range

- All 1400–1500 Codeforces Problems
- DP Problems (sorted by rating)
- Graphs (BFS/DFS)
- Greedy Problems
- Binary Search Problems
- Bitmask Problems

Contest Strategy for 1400–1500

During Contest	After Contest
Solve A in <5 min, B in <15, aim to solve C fast	Upsolve C/D even if they seem hard
Don’t get stuck on one problem – move forward	Analyze why you missed a problem – logic, speed, or pattern?
Always write brute-force first if unsure	Learn transitions in DP and binary search tricks

---

✅ Example Problem Types (1400–1500)

Problem Type	Sample Description
Binary Search on Max Value	Max number of candies for all kids
Greedy + Sort + Pair	Assign tasks to people with capacity
Bitmask DP	Subsets with OR or XOR limit
DP on Index + State	Min steps to make array sorted
Graph Traversal	Count components, check bipartiteness
Constructive	Make a number divisible by k with constraints
Prefix + Count Map	Count subarrays with k distinct elements

Would you like:

- A **sheet of 100 handpicked 1400–1500 problems**?
- A **next-level roadmap (1500 to 1700+)**?
- Or **topics checklist with difficulty-wise ladder**?

Let me know your choice.

You said:

1600 to 1899

ChatGPT said:

To win **Codeforces contests** and reach **1600–1899 rating (Candidate Master pathway)**, you need to move from **pure implementation → smart algorithms + efficient logic**.

At this level, you're expected to solve:

- ✅ Div 2 C quickly and D with some effort
- ✅ Div 3 D/E/F (within 90–120 min)
- ✅ Div 4 E/F
- ✅ Sometimes try Div 1 A/B if upsolving

□ Key Focus Areas (1600–1899)

Category	Topics to Master
✅ **Advanced Greedy**	With proof: correctness based on invariants
✅ **Binary Search on Functions**	With two-pointer/greedy check inside
✅ **Dynamic Programming (1D/2D/Bitmask)**	Indexed DP, knapsack, LIS, digit DP
✅ **Bitmask + DP/Greedy**	Subset sum, state compression

✅ **Graphs: BFS/DFS + Toposort + Components**	Trees, DAGs, cycles, bridges
✅ **DSU (Disjoint Set Union)**	Union-Find with path compression
✅ **Prefix Sums + Segment Trees/Fenwick Trees**	Range sum, range queries
✅ **Math/Combinatorics + Inclusion-Exclusion**	Modular arithmetic, nCr mod p, totient, primes
✅ **Sliding Window + Monotonic Queues**	Max/min in window, optimal subarray
✅ **Trie / Bit Trie (optional)**	XOR problems, string masks

□ Problem Patterns You'll Face

Type	Example Description
🔍 **Binary Search + Greedy**	Can we satisfy k conditions in m days?
□ **DP with States**	`dp[i][j]` = cost/ways of reaching index with parameter
□ **Graph with Toposort / DP**	Longest path in DAG, detect cycle
□ **DSU for Components**	Minimum edges to connect graph
🎛️ **Bitmask Subsets**	Count subset pairs satisfying a bit condition
⚖️ **Segment Trees**	Range sum, max, or count update/query
🧮 **Math + Observations**	Constructing numbers, divisors, modulo logic
🪟 **Sliding Window + Hashing**	Substring match, character conditions

---

□ Practice Strategy from 1600–1899

Goal	Description
□ **Solve Div 2 A–C in 40 min**	A & B in 5–15 min each, C in 15–30
□ **Focus on Div 2 D/E upsolving**	Try full logic after contest even if TLEs
🔍 **Deep dive into DP & Graphs**	You'll see these 1–2 times every round
⚙️ **Code fast and bug-free**	Read constraints carefully. Clean templates.
□ **Upsolve from recent Div 2/3 contests**	Don't miss editorial explanations — they teach you patterns

---

✅ Essential Problem Topics Ladder

📚 1600–1699:

- Binary Search + Condition check
- DP on sequences (LIS, Prefix DP, etc.)
- BFS/DFS with state tracking
- Bitmask DP for small `n` (≤20)
- Greedy with sorting + prefix checks
- Constructive math logic

- DSU basic (connectivity)

🟩 1700–1799:

- DP with 2D/3D states (like `dp[i][j][k]`)
- Graphs: Toposort, Longest path in DAG
- Math: Divisibility, primes, fast mod inverse
- Greedy with priority queue (heap)
- Sliding window max/min (Monotonic deque)
- Trie for strings/XOR
- Bitset / Set mask operations

🟫 1800–1899:

- DP optimizations (memory/compression)
- DSU with special merges (Rollback DSU)
- Bridges, Articulation points (Tarjan's)
- Segment Tree with Lazy Propagation
- Digit DP
- Bitmask + Graph + DP mix

🔗 Must-Practice Problem Tags

- 🔗 Dynamic Programming – 1600–1900
- 🔗 Graphs – 1600–1900
- 🔗 Binary Search – 1600–1900
- 🔗 Greedy – 1600–1900
- 🔗 Bitmasks – 1600–1900
- 🔗 DSU – 1600–1900

🛠️ Example Topics with Problems

Topic	Sample Problem Name
Binary Search + Greedy	Ehab and the Big Finale
DP on Subsequences	Two Sets II
Bitmasking	Xenia and Bit Operations
DSU + Components	Road Construction
Graph Coloring / Toposort	Fox And Names

💥 Contest Strategy at This Level

Before Contest	During Contest	After Contest
Practice solving C/D under 30–45 min	Focus on problem understanding, not just speed	Analyze why you couldn't solve D/E
Review 2–3 core topics per week	Skip D if C not solved fast; return later	Try writing brute-force before jumping to optimal
Prepare templates for DSU, Segment Tree, etc.	Draw examples, use pen/paper	Compare your logic with editorial & debug why you failed

---

Would you like:

- ✅ A **topic-wise sheet** for 1600–1899 (with handpicked problems)?
- ✅ A **1600–1800 ladder** (100 problems)?
- ✅ A **step-by-step roadmap to reach 1900+ (Candidate Master)?**

Let me know and I'll share it instantly.

You said:

1900 to 2099

ChatGPT said:

To break into the **1900–2099 range (Candidate Master)** on Codeforces, you must consistently solve **Div 2 C/D/E**, and sometimes **Div 1 A/B**. At this level, problems combine **multiple ideas**, and you need strong mastery over:

- □ **Dynamic Programming**
- □ **Number Theory + Combinatorics**
- 📊 **Graphs + Trees (DFS, DP, LCA)**
- 📐 **Binary Search + Monotonic Properties**
- 🎲 **Bitmasking + DP Optimization**
- ⚙️ **Data Structures (Segment Tree, Fenwick Tree, DSU)**

□ Skills Checklist (1900–2099)

Topic	Why It Matters
✅ **Advanced DP (with state, memory compression)**	Most Div 2 D/E problems are DP-heavy
✅ **DP on Trees**	Subtree sums, rerooting, bottom-up + top-down
✅ **Graphs + Toposort + SCC**	Cycles, dependencies, DAGs
✅ **Binary Search on Function + Monotonicity**	Used with DP or Greedy to optimize

✅ **Math (Modular Arithmetic, nCr, Fermat, Totient)**	Needed in construction, counting
✅ **DSU with path compression**	Connected components, Kruskal's algorithm
✅ **Segment Trees / Fenwick Trees**	Range queries, updates, lazy propagation
✅ **Bitmask DP**	For state compression when `n ≤ 20`
✅ **Digit DP / Trie / Bitwise Tricks**	Subset constraints, XOR matching
✅ **Constructive Algorithms**	Build output string/array under logic constraints

---

📚 High-Yield Problem Patterns

Pattern Type	Description
🔁 DP with complex states	2D/3D DP, e.g., `dp[pos][mask][tight]`
🌲 DP on Trees	Subtree problems, count paths, max value in subtree
📏 Binary Search + Greedy	Minimize max cost / steps with greedy check
□ Graph + DSU	Count components, Kruskal, merge small to large
□ Bitmask + DP or Greedy	Min/max value from subset
🎛️ Segment Tree / Lazy Updates	Range sum/assign/increment queries
🎭 Math / Constructive	Build permutations, arrays with constraints
🔁 Cycle Detection + SCC	2-SAT, implication graphs, DAGs

🎯 Key Goal: Be Consistent in Div 2 D/E and Div 1 A

Problem	Strategy
Div 2 C	Fast Greedy / Easy DP — solve in 15–25 mins
Div 2 D	Medium DP / Binary Search — 30–45 mins
Div 2 E	Advanced DP / Graph / Tree — 45–75 mins
Div 1 A	Tough constructive / math — learn to upsolve
Div 1 B	Often advanced DP/graph — study after contests

🔗 Practice Resources (1900–2099)

- 💻 Codeforces 1900–2099 Problems
- 🎹 DP Problems (sorted by rating)
- 🌳 Tree Problems
- 📐 Binary Search + Greedy
- ⚙️ Segment Tree + Lazy
- 🎛️ Bitmasks
- 🔎 Graphs + SCC

🧰 Example Problems You Should Master

Problem Type	Example Problem
Bitmask + DP	CF 1187E - Tree DP
Tree DP	CF 1324F - Maximum White Subtree
Binary Search + DP	CF 1364B - Most Socially Active
Graph + DSU	CF 1133D - Zero XOR Subset
Lazy Segment Tree	CF 652E - Pursuit For Artifacts

□ Topic Breakdown (1900–2099 Ladder)

Category	Topics to Learn
🔁 **DP**	DP on digits, states, subsequences, DP with memo
🌳 **Trees**	Tree DP, rerooting, centroid decomposition
⚙️ **DS**	Segment Tree, BIT, DSU, Ordered Set
📐 **Math**	Combinatorics, mod inverse, CRT (optional)
🧭 **Graphs**	Toposort, Bridges, SCC, Dijkstra, Bellman-Ford
□ **Mix**	DP + Bitmask, DP + Graph, Constructive + Math

🛠️ Roadmap to 2100+

Step	Focus
✅ **Master 1900–2099 problems by topic**	Practice 2–3 problems/day
✅ **Regular Div 2 contests**	Solve A–D during contest
✅ **Upsolve Div 2 D/E and Div 1 A**	Learn from editorial/debug
✅ **Maintain notes for DP, Trees, Binary Search**	Reuse patterns
✅ **Track performance via CF Visualizer**	Analyze weak areas

Would you like:

- ✅ A **100-problem ladder (1900–2099)**?
- ✅ A **1900+ DP-focused sheet**?
- ✅ A **Div 2 D/E and Div 1 A upsolve roadmap**?

Let me know!

You said:

2100 to 2299

ChatGPT said:

To level up in **Codeforces rating 2100–2299** (**Master** tier), you must be a **consistent Div 2 top performer** and often solve **Div 1 A/B/C**.

This range is where problems become **multi-layered** — they require deep thinking, combining multiple algorithms, and efficient implementation.

🎯 Main Goals at 2100–2299

Skill Area	Goal
✅ **Div 2 C–E**	Solve fast and accurately
✅ **Div 1 A–B**	Solve most of the time, at least upsolve
✅ **Editorial-Free Thinking**	Build own logic without reading solutions
✅ **Error-Free Coding**	No WA/TLE in standard logic
✅ **Pattern Recognition**	Recognize graph/DP/bitmask patterns fast
✅ **Complex Reductions**	Reduce problem to known solvable subproblem

---

Advanced Topics You Must Master

Topic Area	Details
🔁 DP Optimization Techniques	Convex Hull Trick, Divide & Conquer DP, Monotonic Queue
🌲 Tree Algorithms	Tree DP, Rerooting DP, HLD (Heavy-Light Decomposition)
⚙️ Advanced Segment Trees	Lazy propagation, Merge SegTree, Persistent ST
🔑 Strong Graph Algorithms	Tarjan's (SCC, Bridges), 2-SAT, Dijkstra + Heap, BFS 0-1
🎯 Bitmask + DP / Graphs	Held-Karp, Bitmask on Graph
📐 Mathematics	Modular Inverse, CRT, Euler's theorem, Number Theoretic Transforms (NTT)
🎛️ Constructive Algorithms	Greedy with strong observations, output-building under constraints
DSU on Tree, MO’s Algorithm	Offline queries, subtree answers
Game Theory Basics	Grundy numbers, Sprague-Grundy Theorem

💻 Common Problem Patterns

Pattern Type	Used In
✅ DP with state compression	Bitmask DP, memory-optimized transitions
✅ Multi-source BFS / BFS 0-1	Weighted shortest path with 0/1 edges
✅ DFS Order Tree Flattening	Used with BIT or Segment Tree

✅ MO's Algorithm on Trees	Offline subtree queries
✅ SCC + DAG	Toposort or Condensed Graph DP
✅ DSU + Rollback	Offline Union operations
✅ Convex Hull / Line Container	Optimize cost transitions in DP
✅ Greedy with interval tricks	Disjoint intervals, min removals
✅ Digit DP	Count numbers with digit constraints

□ Must-Practice Problem Types

Type	What It Looks Like
DP + Bitmask	"Choose minimal cost subset with constraint"
Tree DP	"Max sum path, reroot, include/exclude subtree"
Segment Tree + Lazy	"Update/query in range with conditions"
Binary Search + Monotonic DP	"Min max steps/cost with transitions"
DSU on Tree	"Answer query based on subtree sets"
Math/Combinatorics	"Count ways with mod, inversion, or gcd trick"
Graph + Toposort/SCC	"Process dependencies, reduce graph"
Offline Queries + MO's	"Answer all range/subtree queries offline"

🔗 Practice Sheets and Links

✅ Filtered Codeforces Problems:

- 2100–2299 CF Problems
- DP (1900–2400)
- Graphs (DFS, SCC, 2-SAT)
- Data Structures (Segment Tree, DSU)
- Trees
- Number Theory
- Bitmasks

🚀 Contest Strategy (2100–2299)

Round Type	What to Do
□ Div 2	Solve A–D in < 60 mins. Target E
□ Div 1	Solve A in 15–30 mins. Work toward B upsolve
□ Post-contest	Upsolve Div 1 B/C, focus on logic reconstruction
📝 Maintain notebooks	Write notes for advanced DP/graph patterns
🔁 Daily set	Solve 2–3 problems of 2100+ from CF, AtCoder, or gym

---

📚 Example 2100–2299 Problems (Must Try)

Problem	Concept
CF 1326E - Bombs	Stack + Binary Search + RMQ
CF 1208F - Bits And Pieces	Bitmask + Count Optimization
CF 321E - Ciel and Gondolas	Divide and Conquer DP
CF 321C - Ciel the Commander	Centroid Decomposition
CF 757E - Bash Plays with Numbers	Number theory + DP

🛠️ Tools and Resources

- 🔧 CP-Algorithms.com
- 📚 ACM ICPC Handbook
- 📊 Codeforces Visualizer
- 📖 AtCoder Educational DP Contest

✅ Want Practice Sheets?

Let me know if you'd like:

- □ A **100-problem 2100–2299 ladder**
- □ A **sheet for Tree DP / Bitmask DP / Graph DP**
- 📐 A **math/combinatorics pack (CRT, totient, etc.)**
- 🚀 A **plan to reach 2300+ (International Master)**

Just reply with what you'd like next!

You said:

2300 to2399

ChatGPT said:

To **win contests** and **consistently solve problems rated 2300–2399** on Codeforces (typically **Candidate Master / Master level**), you need a **strong command over advanced algorithmic and mathematical topics**, along with **fast implementation and problem-reading skills**.

Here's a topic list tailored for **2300–2399 rated problems**:

✅ Core Prerequisites

- **Fast implementation** using STL, templates
- **Strong debugging and testing habits**
- **Speed + Accuracy** in reading constraints and handling corner cases

---

📚 Algorithmic Topics

1. Advanced Dynamic Programming

- Divide & Conquer DP
- DP with Bitmasks
- Tree DP (including rerooting techniques)
- Slope Trick (Convex Optimization)
- Knuth Optimization
- Monotonic Queue Optimization
- Aliens Trick

2. Advanced Graph Algorithms

- Strongly Connected Components (Kosaraju/Tarjan)
- Biconnected Components
- 2-SAT Problem
- Bridges and Articulation Points
- Minimum Cost Flow
- Flow with Lower Bounds
- Maximum Matching in Bipartite Graph (Kuhn’s, Hopcroft-Karp)
- Min-Cost Max Flow

3. Data Structures

- Mergeable Heaps (e.g., Leftist Tree, Skew Heap)
- Persistent Segment Trees
- Treap / Splay Tree (Implicit Treap for sequence manipulation)
- Heavy-Light Decomposition (HLD)
- Mo’s Algorithm (with updates)
- Segment Tree Beats

4. Mathematics

- Matrix Exponentiation (for Linear Recurrences)
- Fast Walsh–Hadamard Transform (FWHT)
- Fast Fourier Transform (FFT)
- Number Theoretic Transform (NTT)
- Diophantine Equations
- Gaussian Elimination (Modulo and Floating Point)
- Combinatorics with Generating Functions
- Mobius Inversion
- Burnside’s Lemma and Polya Enumeration

5. String Algorithms

- Suffix Automaton
- Z-function, KMP, Aho-Corasick
- Manacher's Algorithm
- Lyndon Factorization
- Palindromic Tree / Eertree
- Hashing with Collision Control
- Trie with Bitmask Compression

6. Geometry (Advanced)

- Convex Hull Trick (CHT)
- Rotating Calipers
- Line Sweep for Intersections
- Segment Intersection using Order Set
- Point in Polygon / Convex Hull Inclusion
- Circle Intersection and Geometry Tricks

7. Bitmask & Brute Force

- SOS DP (Sum over Subsets)
- Meet in the Middle
- Inclusion-Exclusion over subsets
- Bitwise DP
- Digit DP (Harder variations)

□ Problem Solving Techniques

- Constructive Algorithms (Harder Cases)
- Greedy with Proofs
- Randomized Algorithms (e.g., Treap, Miller-Rabin)
- Game Theory (Grundy Numbers, Sprague-Grundy Theorem)
- Amortized Analysis (Tricky logic with Deques, Heaps)

🔁 Practice Strategy for 2300+

- **Up-solve Div. 1** virtual contests (focus on 2nd and 3rd problem)
- Solve problems by **tag + rating 2300–2399**
- Focus on **timed** virtual contests and post-contest **analysis**
- Learn new algorithms **only when required**, not just theory
- Use resources like cp-algorithms.com and AtCoder Library

Would you like a **list of Codeforces problem links rated 2300–2399 by topic** next?

You said:

2400 to 2599

ChatGPT said:

Here's a focused list of **topics and problem types** you should master to **consistently perform well in Codeforces contests rated 2400–2599**:

🔥 Advanced Topics for 2400–2599 Codeforces Rating

1. Dynamic Programming (Advanced)

- Bitmask DP
- Divide & Conquer DP
- DP with Convex Hull Trick / Line Container
- Knuth Optimization
- Monotonic Queue Optimization
- Tree DP (rerooting, subtree aggregation)
- DP on Subsets / SOS DP

2. Graph Algorithms (Advanced)

- **Max Flow / Min Cut**: Dinic's Algorithm, Edmonds-Karp
- **Strongly Connected Components**: Kosaraju / Tarjan
- **2-SAT** (implication graphs)
- **Bridges, Articulation Points**
- **Centroid Decomposition**
- **Heavy-Light Decomposition (HLD)**
- **Link-Cut Trees / Euler Tour Tree** (rare but appear)
- **Topological Order + DP**

3. Data Structures

- Persistent Segment Tree
- Mergeable Heaps (Treap/Splay)
- Lazy Segment Tree with Complex Updates
- Li Chao Segment Tree (for line intersection problems)
- Sqrt-Decomposition with Offline Queries
- Binary Lifting + LCA with Modifications

4. Mathematics / Number Theory

- Discrete Log (Baby-step Giant-step)
- Euler's Theorem, Fermat's Little Theorem
- Extended Euclidean Algorithm

- Chinese Remainder Theorem (CRT)
- Primitive Roots, Orders
- Combinatorics with Lucas Theorem
- Polynomial Interpolation (Lagrange)

5. Geometry

- Convex Hull Trick (Dynamic Hull)
- Rotating Calipers
- Line Sweep with Event Queue
- Half Plane Intersection
- Voronoi Diagram (rare)
- Geometry + Binary Search / Ternary Search

6. Strings

- Suffix Array + LCP Array
- Suffix Automaton
- Palindromic Tree (Eertree)
- Z-Algorithm, KMP, Aho-Corasick
- Rolling Hash with Double Hashing
- Lyndon Decomposition

7. Constructive Algorithms

- Invariants / Monovariants
- Greedy with Observations
- Bitwise Constructive Proofs
- Interactive Problems

8. Game Theory

- Grundy Numbers / Sprague-Grundy Theorem
- Impartial & Partisan Games
- Nim Game Variants

9. Advanced Search / Optimization

- Ternary Search on Unimodal Functions
- Meet in the Middle (Advanced)
- Divide & Conquer Optimization
- Binary Search + Simulation / Check
- Slope Trick (rare)

□ Contest Tips at 2400+ Level

- Practice virtual contests with difficulty: 2400–2700
- Focus on **speed-solving** first 2 problems quickly (up to ~1900–2000)

- Learn to **debug efficiently** under pressure
- Try **hard constructive** or **DP with observations** problems regularly
- Study **top-rated coders' upsolve strategies**

Would you like a **practice sheet of 50 problems** in this rating range organized by topic?

You said:

2600 to 2899

ChatGPT said:

Here's a **topic-wise roadmap** to help you **win contests** in the **Codeforces rating range 2600 to 2899**. At this level, you're expected to master high-level algorithms, data structures, and advanced problem-solving strategies.

□ Codeforces 2600–2899: Topic List to Win Contests

🏆 **Goal:** Become a consistent candidate master or international master and reach LGM potential.

✅ Advanced Dynamic Programming

- DP with Segment Trees / BIT
- Divide and Conquer DP
- Knuth Optimization
- Convex Hull Trick
- Bitmask DP (on trees or graphs)
- Slope Trick (Advanced)

✅ Advanced Graph Algorithms

- Dynamic Connectivity (Link-Cut Trees, Euler Tour Trees)
- 2-SAT
- Minimum Cost Flow (Cycle Canceling, Primal-Dual)
- Persistent Disjoint Set Union (DSU)
- Heavy-Light Decomposition (HLD)
- Tree Isomorphism
- Centroid Decomposition

✅ Geometry (Hardcore)

- Rotating Calipers
- Minkowski Sum
- Convex Hull in 3D
- Dynamic Convex Hull
- Line Intersection Counting (Sweep Line)
- Geometry with Integer Coordinates (careful precision)

✅ Combinatorics & Counting

- Polya Enumeration Theorem
- Burnside's Lemma
- Stirling Numbers (1st and 2nd kind)
- Inclusion-Exclusion on sets and masks
- Generating Functions (Exponential and Ordinary)
- FFT-based polynomial multiplication

✅ Data Structures (Expert Level)

- Persistent Segment Tree / Trie
- Mergeable Heap
- Link-Cut Tree (Splay Tree or Euler Tour Tree)
- Segment Tree Beats
- Mo's Algorithm on Tree / Queries on Tree with Updates
- Wavelet Trees

✅ Math (Advanced Number Theory)

- Discrete Logarithm (Baby Step Giant Step)
- Chinese Remainder Theorem (non-coprime moduli)
- Lucas Theorem
- Fast Walsh-Hadamard Transform
- Matrix Exponentiation in Rings
- Elliptic Curve Arithmetic (Rare)

✅ Strings (Advanced)

- Suffix Automaton + DP on Automaton
- Suffix Tree with Ukkonen's Algorithm

- Z-algorithm & KMP in combinatorics
- Aho-Corasick Automaton (multi-pattern)
- Palindromic Tree (Eertree)

✅ Problem Solving Strategies

- Constructive + Observations + Binary Search
- Games & Sprague-Grundy Theorem
- Interactive Problems with hidden constraints
- Offline Queries + Event Sorting + DSU/Segment Trees
- Reverse Engineering DP states

🔁 Practice Strategy

Type	How to Practice
CF Div 1/Global	Solve B, C, D, and E problems regularly
Educational Rounds	Focus on C to F (greedy + math + observations)
AtCoder AGC	Excellent source for DP and constructive ideas
POI/CEOI/OI Archive	Focus on dynamic programming and data structures
Virtual Contests	Use gym + upsolve to simulate rating pressure

---

🔥 Example Problems (Codeforces)

- 1416D – Graph and Queries (HLD + DSU)
- 1207H – XOR Permutations (Inclusion-Exclusion + DP)
- 455D – Serega and Fun (Mo's on Trees + Frequency)
- 321E – Ciel and Gondolas (Divide and Conquer DP)
- 833D – Red–Blue Graph (2-SAT)

Would you like a **tracker or practice sheet with direct problem links** for these topics?

You said:

2900+

ChatGPT said:

Codeforces Rating 2900+ (Legendary Grandmaster) is the absolute elite level. To **win contests or perform well consistently at this level**, one must master advanced algorithms, data structures, and problem-solving paradigms that are both rare and very deep.

Here’s a list of **key topics** and ideas that are typically seen at **2900+ level**:

□ Core Requirements

- **Perfect implementation skill**
- **Very fast thinking**
- **Incredible debugging under pressure**
- **Pattern recognition of rare techniques**

🔁 Advanced Dynamic Programming

- **Dynamic Programming on Trees**
- **DP with Segment Trees or BIT**
- **Divide and Conquer DP**
- **Monotonic Queue Optimization in DP**
- **Knuth Optimization**
- **Slope Trick**
- **Aliens trick**
- **Dynamic Convex Hull Trick**

📊 Advanced Data Structures

- **Heavy-Light Decomposition**
- **Link-Cut Trees / Euler Tour Trees**
- **Treaps / Splay Trees / Rope / Cartesian Trees**
- **Persistent Segment Trees / BIT**
- **Dynamic connectivity**
- **Wavelet Trees**
- **Li Chao Tree**

📈 Geometry

- **Convex Hull (Graham scan, Andrew’s Monotone Chain)**
- **Rotating Calipers**
- **Sweep Line Algorithms**
- **Half-plane Intersection**
- **Segment Intersections**
- **Circle-Circle, Circle-Line Intersections**
- **Computational Geometry using complex numbers**

🕸️ Graph Algorithms

- **2-SAT and Strongly Connected Components**
- **Centroid Decomposition**
- **Dynamic Graph Algorithms**
- **Link/Cut Trees for dynamic graphs**
- **Kirchhoff's Theorem (Count spanning trees)**
- **Eulerian Path/Circuit with Hierholzer's Algorithm**
- **Minimum Cost Max Flow**
- **Cycle detection and contraction**
- **Directed Minimum Spanning Trees (Chu-Liu/Edmonds)**

🔢 Number Theory & Math

- **Nimber / Sprague-Grundy Theory (Game Theory)**
- **Continued Fractions**
- **Berlekamp–Massey Algorithm**
- **Gaussian Elimination in modular arithmetic**
- **FFT and NTT (Fast Fourier Transform / Number Theoretic Transform)**
- **Burnside's Lemma / Polya Enumeration**
- **Chinese Remainder Theorem (advanced use cases)**
- **Discrete Logarithm**
- **Modular Inverse via Extended Euclid/Fermat**
- **Mobius Function, Inclusion-Exclusion**

💻 Strings and Suffix Structures

- **Suffix Automaton (SAM)**
- **Suffix Tree**
- **Z Algorithm, KMP**
- **Aho-Corasick Automaton**
- **Manacher's Algorithm**
- **Palindromic Tree / Eertree**
- **Duval's Algorithm (Lyndon factorization)**

□ Combinatorics & Probability

- **Advanced Inclusion-Exclusion**
- **Stirling Numbers / Bell Numbers**
- **Expected Value Computation under Constraints**
- **Multinomial Coefficients**
- **DP + Combinatorics Hybrid Problems**

💪 Bonus Skills

- **Stress-testing code using generators**
- **Writing brute-force for test case validation**
- **Extremely good at understanding constraints and edge cases**
- **Speed: Solve ~3 problems in 1 hour including implementation**

✅ Must Practice

- **Old Educational and Div1+Div2 contests (2000+)**
- **Virtual participation in contests by top-rated setters**
- **Problems solved by top contestants like tourist, Radewoosh, Um_nik, Benq**

Would you like a **practice problem list** for 2900+ level sorted by topic?

Tools