**Data Structures and Algorithms for Online Programming Contest**

**1. Sorting**

i. [Merge Sort](algorithms/Merge\_Sort.cpp)

ii. [Quick Sort](algorithms/Quick\_Sort.cpp)

iii. [Heap Sort](algorithms/heap\_sort.cpp)

iv. [Bubble Sort](algorithms/Bubble\_Sort.cpp)

v. [Insertion Sort](algorithms/Insertion\_Sort.cpp)

vi [Selection Sort](algorithms/Selection\_Sort.cpp)

vii. [Bucket Sort](algorithms/Bucket\_Sort.cpp)

viii. [Count Sort](algorithms/Count\_Sort.cpp)

ix. [Radix Sort](algorithms/Radix\_Sort.cpp)

x. [Pancake Sort](algorithms/Pancake\_Sort.cpp)

**2. Data Structure**

i. [Singly Linked List](algorithms/singly\_linked\_list.cpp)

ii. [Doubly Linked List](algorithms/doubly\_linked\_list.cpp)

iii. [Vector](algorithms/vector.cpp)

iv. [Stack](algorithms/stack.cpp)

v. [Queue](algorithms/queue.cpp)

vi. [List](algorithms/list.cpp)

vii. [Hashtable](algorithms/hashtable.cpp)

viii. [HashMap](algorithms/hashmap.cpp)

ix. [HashSet](algorithms/hashset.cpp)

x. [Union Find(Disjoint Set)](algorithms/union\_find.cpp)

xi. [Binary Search Tree](algorithms/BST.cpp)

xii. [Segment Tree](algorithms/segment\_tree.cpp)

xiii. [Segment Tree (Lazy Propagation)](algorithms/segment\_tree\_lazy.cpp)

xiv. [2D Segment Tree (Quad tree)](algorithms/2D\_segment\_tree.cpp)

xv. [Binary Indexed Tree](algorithms/BIT.cpp)

xvi. [2D Binary Indexed Tree](algorithms/2D\_BIT.cpp)

xvii. [(AVL Tree) Self Balanced BST](algorithms/AVL.cpp)

xviii. [(Splay Tree) Self Balanced BST](algorithms/splay.cpp)

xix. [Ternary Search Tree](algorithms/ternary\_search\_tree.cpp)

xx. [Heap (Min)](algorithms/heap.cpp)

**3. Graph Theory**

i. [Floyd Warshall’s](algorithms/floyd\_warshall.cpp)

ii. [Loop Detection](algorithms/topsort.cpp)

iii. [Topological Sort](algorithms/topsort.cpp)

iv. [Strongly Connected Component (Kosaraju)](algorithms/scc.cpp)

v. [Lowest Common Ancestor(sparse table)](algorithms/lca.cpp)

vi. [Articulation Point](algorithms/articulation\_point.cpp)

vii. [Bridge](algorithms/bridge.cpp)

viii. [Breadth First Search](algorithms/bfs.cpp)

ix. [Dijkstra](algorithms/dijkstra.cpp)

x. [Bellman Ford's](algorithms/bellman\_fords.cpp)

xi. [Kruskal Minimum Spanning Tree](algorithms/kruskal.cpp)

xii. [Minimum Vertex Cover](algorithms/min\_vertex\_cover.cpp)

xiii. [Maximum Flow (Edmonds Karp’s) I](algorithms/max\_flow.cpp)

xiv. [Maximum Flow (Edmonds Karp’s) II](algorithms/Maximum\_Flow\_Problem\_I\_Edmond\_Karp.cpp)

xv. [Maximum Bipartite Matching](algorithms/maximum\_bipartite\_matching.cpp)

xvi. [Stable Marriage Problem](algorithms/stable\_marriage\_problem.cpp)

xvii. [Heavy Light Decomposition](algorithms/HLD.cpp)

**4. Game Theory**

i. [Game Tree(Memorization)](algorithms/game\_tree.cpp)

ii. [Nim](algorithms/nim.cpp)

iii. [Misère Nim](algorithms/nim.cpp)

iv. [Nimble Nim](algorithms/nim.cpp)

v. [Poker Nim](algorithms/nim.cpp)

vi. [Prime Power Nim](algorithms/nim.cpp)

vii. [Spagrue Grundy Problem](algorithms/grundy.cpp)

viii. [Grundy Variant: Zero Nim Game](algorithms/grundy.cpp)

ix. [Grundy Variant: Coins on Chessboard](algorithms/grundy.cpp)

x. [Green HackenBush(Colon Principle)](algorithms/hackenbush.cpp)

**5. Dynamic Programming**

i. [Coin Change and variants](algorithms/coin\_change.cpp)

ii. [Knapsack Problem and variants](algorithms/knapsack.cpp)

iii. [Matrix Chain Multiplication](algorithms/mcm.cpp)

iv. [Longest Increasing Subsequence( O(n^2) )](algorithms/lis.cpp)

v. [Longest Increasing Subsequence( O(nlogn) )](algorithms/lis.cpp)

vi. [Travelling Salesman Problem](algorithms/tsp.cpp)

vii. [Maximum Sum Subarray( O(n^4) and O(n^3) )](algorithms/maximum\_sum\_subarray.cpp)

viii. [Kadane Algorithm](algorithms/kadane.cpp)

ix. [Maximum Sum Subarray using Kadane( O(n^3) )](algorithms/kadane.cpp)

x. [Optimal Binary Search Tree](algorithms/optimal\_search\_tree.cpp)

xi. [Subset Sum](algorithms/subset\_sum.cpp)

xii. [Catalan Number](algorithms/CatalanNumber.cpp)

xiii. [DAG Minimum Path](algorithms/DAG\_min\_path.cpp)

xiv. [Minimum Cost Path](algorithms/min\_cost\_path.cpp)

xv. [Digit Dp I](algorithms/Digit\_dp\_I.cpp)

xvi. [Digit Dp II](algorithms/Digit\_dp\_II.cpp)

xvii. [Digit Dp III](algorithms/Digit\_dp\_III.cpp)

xviii. [Digit Dp IV](algorithms/Digit\_dp\_IV.cpp)

**6. Backtracking**

i. [Permutation Generator](algorithms/permutation\_generator.cpp)

ii. [N-Queen](algorithms/nqueen.cpp)

iii. [Prime Ring](algorithms/prime\_ring.cpp)

**7. Greedy Algorithm**

i. [Huffman Coding](algorithms/Huffman\_coding.cpp)

**8. Binary Search**

i. [Binary Search](algorithms/binary\_search.cpp)

ii. [Lower Bound](algorithms/binary\_search.cpp)

iii. [Upper Bound](algorithms/binary\_search.cpp)

iv. [Equal Range](algorithms/binary\_search.cpp)

**9. Hashing**

i. [Double Hashing](algorithms/double\_hashing.cpp)

ii. [String Hashing by Map](algorithms/map\_hashing.cpp)

iii. [Berenstain String Hashing](algorithms/berenstain\_hashing.cpp)

iv. [Rolling Hash](algorithms/rabin\_karp.cpp)

**10. Combinotorics**

i. [Factorial](algorithms/factorial.cpp)

ii. [nCr](algorithms/ncr.cpp)

iii. [De-arrangement](algorithms/dearrangement.cpp)

**11. String Algorithm**

i. [Aho-Corasick Algorithm](algorithms/Aho\_Corasick.cpp)

ii. [Knuth-Morris-Pratt’s Algorithm](algorithms/kmp.cpp)

iii. [Rabin Karp Pattern Searching](algorithms/rabin\_karp.cpp)

iv. [Z Algorithm](algorithms/z.cpp)

v. [Finite Automata Pattern Searching](algorithms/Finite\_Automata\_Pattern\_Searching.cpp)

vi. [Trie (Prefix/Radix Tree)](algorithms/Trie.cpp)

vii. [Longest Common Subsequence](algorithms/lcs.cpp)

viii. [Edit Distance](algorithms/edit\_distance.cpp)

ix. [Longest Palindromic Subsequence](algorithms/lps.cpp)

x. [Suffix Array](algorithms/suffix\_lcp.cpp)

xi. [Longest Common Prefix](algorithms/suffix\_lcp.cpp)

xii. [Minimum Expression](algorithms/Minimum\_expression.cpp)

xiii. [Suffix Automata](algorithms/suffix\_automata.cpp)

**12. Mathematics**

i. [Power Function(Big mod)](algorithms/Power.cpp)

ii. [Modular Mutiplicative Inverse(using Big mod)](algorithms/Power.cpp)

iii. [Prime(Sieve of Erathonesis)](algorithms/prime.cpp)

iv. [Segmented Sieve of Erathonesis](algorithms/prime.cpp)

v. [Prime factorization(using Sieve)](algorithms/prime.cpp)

vi. [Prime factorization](algorithms/prime.cpp)

vii. [Primality Test(School method)](algorithms/primality\_test.cpp)

viii. [Miller–Rabin Primality Test](algorithms/primality\_test.cpp)

ix. [Euler Totient (Phi Function)](algorithms/phi.cpp)

x. [Extended Euclid](algorithms/extended\_euclid.cpp)

xi. [Linear Diophatine Equation](algorithms/extended\_euclid.cpp)

xii. [Modular Mutiplicative Inverse(using Extended Euclid)](algorithms/extended\_euclid.cpp)

xiii. [Matrix Exponentiation](algorithms/matrix\_exp.xpp)

xiv. [Floyd Cycle Finding Algorithm](algorithms/floyd\_cycle\_finding.cpp)

xv. [Big Integer](algorithms/big\_integer.cpp)

xvi. [Josephus Recurrence](algorithms/Josephus\_Recurrence.cpp)

xvii. [Fast Fourier Transform](algorithms/FFT.cpp)

**13. Computational Geometry**

i. [Computational Geometry Template](algorithms/computational\_geometry\_template.cpp)

ii. [Convex Hull (Jarvis’s Algorithm or Wrapping)](algorithms/convex\_hull\_jarvis.cpp)

iii. [Convex Hull (Graham Scan)](algorithms/convex\_hull\_graham\_scan.cpp)

**14. Miscellaneous**

i. [Template](template.cpp)