IOI-ს არქივი https://ioi.contest.codeforces.com/group/32KGsXgiKA/blog

ვექტორი

381A

490A

600B

1511C

გროვა

377B

140C

სიგანეში ძებნა

POJ 3170 [Knights of Ni](http://poj.org/problem?id=3170)

1063/B 543/B

520B [330D](https://codeforces.com/contest/330/problem/D) 1651/D

930A 1037D 769C

სტეკი

Poj 3044

<https://www.spoj.com/problems/HISTOGRA/>

427A 490A

343B 320D 117C?

დეკი

487B 1179A

დინამიური პროგრამირება

|  |  |
| --- | --- |
| POJ 2385 [Apple Catching](http://poj.org/problem?id=2385)  POJ 3176 [Cow Bowling](http://poj.org/problem?id=3176)  189A  313B  166E  118E  4D, 702A - უდიდესი ზრდადი ქვემიმდევრობა |  |

დეიქსტრას ალგორითმი

POJ 2457 PART ACQUISITION

POJ 2394 CHECKING AN ALIBI

20C

144D

449B

მინიმალური დამფარავი ხეები

POJ BAD COWTRACTORS

605B

160D

609E

TRIE-ხეები

455B

566A

ხის დიამეტრი

120F 911F

მოს ალგორითმი

620F

სეგმენტ–ხე

codeforces :  
[Xenia and Bit Operations](https://codeforces.com/problemset/problem/339/D)  
[Knight Tournament](https://codeforces.com/contest/356/problem/A)  
[Bash and a Tough Math Puzzle](https://codeforces.com/problemset/problem/914/D)  
[Pashmak and Parmida's problem](https://codeforces.com/contest/459/problem/D)  
[Enemy is weak](https://codeforces.com/contest/61/problem/E)  
[Circular RMQ](https://codeforces.com/contest/52/problem/C)  
[REQ](https://codeforces.com/contest/594/problem/D)  
[Lucky Queries](https://codeforces.com/contest/145/problem/E)  
[XOR on Segment](https://codeforces.com/problemset/problem/242/E)  
[Sereja and Brackets](https://codeforces.com/problemset/problem/380/C)  
[Ant Colony](https://codeforces.com/problemset/problem/474/F)  
[Babaei and Birthday Cake](https://codeforces.com/contest/629/problem/D)  
[A Simple Task](https://codeforces.com/contest/558/problem/E)  
[Frogs and mosquitoes](https://codeforces.com/contest/609/problem/F)  
[Copying Data](https://codeforces.com/contest/292/problem/E)  
[Lucky Array](https://codeforces.com/contest/121/problem/E)  
[Vika and Segments](https://codeforces.com/contest/610/problem/D)  
[Misha and Permutations Summation](https://codeforces.com/contest/501/problem/D)  
[Little Elephant and Inversions](https://codeforces.com/contest/220/problem/E)  
[One Occurrence](https://codeforces.com/contest/1000/problem/F)  
[Optimize!](https://codeforces.com/contest/338/problem/E)  
[Yaroslav and Divisors](https://codeforces.com/contest/301/problem/D)  
[TorCoder](https://codeforces.com/contest/240/problem/F)  
[New Year Domino](https://codeforces.com/contest/500/problem/E)  
[Pillars](https://codeforces.com/contest/474/problem/E)  
[On Changing Tree](https://codeforces.com/contest/396/problem/C)  
[Infinite Inversions](https://codeforces.com/contest/540/problem/E)  
[Interesting Array](https://codeforces.com/contest/482/problem/B)  
[Alphabet Permutations](https://codeforces.com/contest/610/problem/E)  
[DZY Loves Fibonacci Numbers](https://codeforces.com/contest/446/problem/C)  
[Points](https://codeforces.com/contest/19/problem/D)  
[Function](https://codeforces.com/contest/455/problem/E)  
[Eyes Closed](https://codeforces.com/problemset/problem/895/E)  
[Tree and Queries](https://codeforces.com/contest/375/problem/D)  
[Valera and Queries](https://codeforces.com/contest/369/problem/E)  
[Nikita and stack](https://codeforces.com/contest/760/problem/E)  
[Linear Kingdom Races](https://codeforces.com/contest/115/problem/E)  
[Camping Groups](https://codeforces.com/contest/173/problem/E)  
[Little Girl and Problem on Trees](https://codeforces.com/contest/276/problem/E)  
[Domino Principle](https://codeforces.com/contest/56/problem/E)  
[Water Tree](https://codeforces.com/contest/343/problem/D)  
[Propagating tree](https://codeforces.com/contest/383/problem/C)  
[Kefa and Watch](https://codeforces.com/contest/580/problem/E)  
[Jeff and Removing Periods](https://codeforces.com/contest/351/problem/D)  
[Drazil and Morning Exercise](https://codeforces.com/contest/516/problem/D)  
[Subsequences](https://codeforces.com/contest/597/problem/C)  
[SUM and REPLACE](https://codeforces.com/contest/920/problem/F)  
[Tree or not Tree](https://codeforces.com/contest/117/problem/E)  
[The Child and Sequence](https://codeforces.com/contest/438/problem/D)  
[Encryption (hard)](https://codeforces.com/contest/958/problem/C3)  
[Hot is Cold](https://codeforces.com/problemset/problem/1146/E)  
[Developing Game](https://codeforces.com/contest/377/problem/D)  
[Drazil and Park](https://codeforces.com/contest/515/problem/E)  
[Board Game](https://codeforces.com/contest/605/problem/D)  
[The Untended Antiquity](https://codeforces.com/problemset/problem/869/E)  
[Danil and a Part-time Job](https://codeforces.com/contest/877/problem/E)  
[More Queries to Array...](https://codeforces.com/problemset/problem/266/E)  
[Cow Tennis Tournament](https://codeforces.com/contest/283/problem/E)  
[Coins Exhibition](https://codeforces.com/contest/930/problem/E)  
[Till I Collapse](https://codeforces.com/problemset/problem/786/C)  
[Editor](https://codeforces.com/contest/1263/problem/E)  
[Serge and Dining Room](https://codeforces.com/contest/1180/problem/E)  
[Subarray Sorting](https://codeforces.com/contest/1187/problem/D)  
[Physical Education Lessons](https://codeforces.com/contest/915/problem/E)  
[Alyona and towers](https://codeforces.com/contest/739/problem/C)  
[Sasha and Array](https://codeforces.com/contest/718/problem/C)  
[New Year and Old Subsequence](https://codeforces.com/contest/750/problem/E)  
[Timofey and our friends animals](https://codeforces.com/contest/763/problem/E)  
[Souvenirs](https://codeforces.com/contest/765/problem/F)  
[Yash And Trees](https://codeforces.com/contest/633/problem/G)  
[Legacy](https://codeforces.com/contest/786/problem/B)  
[Periodic RMQ Problem](https://codeforces.com/contest/803/problem/G)  
[Leha and security system](https://codeforces.com/contest/794/problem/F)  
[Vladik and Entertaining Flags](https://codeforces.com/contest/811/problem/E)  
[MEX Queries](https://codeforces.com/contest/817/problem/F)  
[The Bakery](https://codeforces.com/contest/834/problem/D)  
[Nauuo and Bug](https://codeforces.com/contest/1172/problem/F)  
[Duff in the Army](https://codeforces.com/problemset/problem/587/C)  
[Fools and Roads](https://codeforces.com/problemset/problem/191/C)

codeforces gym :  
[Queries](https://codeforces.com/gym/100739/problem/A)  
[Letter Array](https://codeforces.com/gym/100482/problem/C)  
[Hacker Cups and Balls](https://codeforces.com/problemset/gymProblem/101234/A)  
[Reflection](https://codeforces.com/gym/101992/problem/L)  
[Rectangles](https://codeforces.com/gym/101982)  
[Colonial Mansions](https://codeforces.com/gym/101962/problem/I)  
[Colors Overflow](https://codeforces.com/gym/101801)  
[Running a penitentiary](https://codeforces.com/gym/101879/problem/G)  
[Subsequence Sum Queries](https://codeforces.com/gym/101741/problem/J)

spoj :

[BGSHOOT — Shoot and kill](http://www.spoj.com/problems/BGSHOOT/)  
[HORRIBLE — Horrible Queries](http://www.spoj.com/problems/HORRIBLE/)  
[GSS1 — Can you answer these queries I](http://www.spoj.com/problems/GSS1/)  
[GSS3 — Can you answer these queries IV](http://www.spoj.com/problems/GSS3/)  
[GSS5 — Can you answer these queries V](http://www.spoj.com/problems/GSS5/)  
[KGSS — Maximum Sum](http://www.spoj.com/problems/KGSS/)  
[POSTERS — Election Posters](http://www.spoj.com/problems/POSTERS/)  
[BRCKTS — Brackets](http://www.spoj.com/problems/BRCKTS/)  
[LITE — Light Switching](http://www.spoj.com/problems/LITE/)  
[KQUERYO — K-Query Online](http://www.spoj.com/problems/KQUERYO/)  
[CNTPRIME — Counting Primes](http://www.spoj.com/problems/CNTPRIME/)  
[MULTQ3](http://www.spoj.com/problems/MULTQ3/)  
[IOPC1207 — GM plants](http://www.spoj.com/problems/IOPC1207/)  
[QUERYIT — SLIS](https://www.spoj.com/problems/QUERYIT/)  
[ADACABAA — Ada and Species](https://www.spoj.com/problems/ADACABAA/)  
[DQUERY — D-query](https://www.spoj.com/problems/DQUERY/)  
[PATULJCI — Snow White and the N dwarfs](https://www.spoj.com/problems/PATULJCI/)  
[ADAGF — Ada and Greenflies](https://www.spoj.com/problems/ADAGF/)  
[ADATREE — Ada and Trees](https://www.spoj.com/problems/ADATREE/)  
[Mass Change Queries](https://codeforces.com/contest/911/problem/G)  
[PRMQUER — Prime queries](https://www.spoj.com/problems/PRMQUER/)  
[NAJ0001 — Divisible Number Sum](https://www.spoj.com/problems/NAJ0001/)  
[DCEPC11I — Impossible Boss](https://www.spoj.com/problems/DCEPC11I/)  
[PERMPATT — Check 1324](https://www.spoj.com/problems/PERMPATT/)  
[THRBL — Catapult that ball](https://www.spoj.com/problems/THRBL/)  
[MON2012 — Monkey and apples](https://www.spoj.com/problems/MON2012/)  
[GOODE — Good Debugging](https://www.spoj.com/problems/GOODE/) [SBO — MAXIMUM RARITY](https://www.spoj.com/problems/SBO/)

lightoj :

[1082 — Array Queries](http://www.lightoj.com/volume_showproblem.php?problem=1082)  
[1080 — Binary Simulation](http://www.lightoj.com/volume_showproblem.php?problem=1080)  
[1112 — Curious Robin Hood](http://www.lightoj.com/volume_showproblem.php?problem=1112)

e-olymp :  
[In the Country of Unlearned Lessons](https://www.e-olymp.com/en/problems/4481)  
[In a country of Unlearned Lessons 2](https://www.e-olymp.com/en/problems/4482)  
[The kid who learned to count](https://www.e-olymp.com/en/problems/4483)  
[Investigation by koloboks](https://www.e-olymp.com/en/problems/4484)  
[Цветик-Семицветик](https://www.e-olymp.com/en/problems/4485)  
[Cheburashka and Crocodile Gena](https://www.e-olymp.com/en/problems/4486)  
[Винни-Пух](https://www.e-olymp.com/en/problems/4487)  
[Винни-Пух 2](https://www.e-olymp.com/en/problems/4488)  
[Карлсон, который живет на крыше](https://www.e-olymp.com/en/problems/4489)  
[Ну, погоди!](https://www.e-olymp.com/en/problems/4490)  
[Three from Prostokvashino](https://www.e-olymp.com/en/problems/4491)  
[Трое из Простоквашино 2](https://www.e-olymp.com/en/problems/4492)  
[Трое из Простоквашино 3](https://www.e-olymp.com/en/problems/4493)  
[Золотой ключик или приключения Буратино](https://www.e-olymp.com/en/problems/4494)  
[Приключения кота Леопольда](https://www.e-olymp.com/en/problems/4495)  
[Adventures of Dunno and His Friends](https://www.e-olymp.com/en/problems/4496)  
[38 попугаев](https://www.e-olymp.com/en/problems/4497)  
[Возвращение блудного попугая](https://www.e-olymp.com/en/problems/4498)  
[Сказка о попе и работнике его Балде](https://www.e-olymp.com/en/problems/4499)  
[Чипполино](https://www.e-olymp.com/en/problems/4500)  
you can use google translate for the problems written in russian.

POJ 3264 [Balanced Lineup](http://poj.org/problem?id=3264)

ლინკი <http://bitdevu.blogspot.com/>

Classic :  
[339D - Xenia and Bit Operations](https://codeforces.com/contest/339/problem/D)  
[356A - Knight Tournament](https://codeforces.com/contest/356/problem/A)  
[459D - Pashmak and Parmida's problem](https://codeforces.com/contest/459/problem/D)  
[61E - Enemy is weak](https://codeforces.com/contest/61/problem/E)  
[380C - Sereja and Brackets](https://codeforces.com/contest/380/problem/C)  
[474F - Ant colony](https://codeforces.com/contest/474/problem/F)  
[292E - Copying Data](https://codeforces.com/contest/292/problem/E)  
[501D - Misha and Permutations Summation](https://codeforces.com/contest/501/problem/D)  
[220E - Little Elephant and Inversions](https://codeforces.com/contest/220/problem/E)  
[338E - Optimize!](https://codeforces.com/contest/338/problem/E)  
[19D - Points](https://codeforces.com/contest/19/problem/D)  
[351D - Jeff and Removing Periods](https://codeforces.com/contest/351/problem/D)  
[515E - Drazil and Park](https://codeforces.com/contest/515/problem/E)  
[540E - Infinite Inversions](https://codeforces.com/contest/540/problem/E)  
[609F - Frogs and mosquitoes](https://codeforces.com/contest/609/problem/F)  
[594D - REQ](https://codeforces.com/contest/594/problem/D)  
[455E - Function](https://codeforces.com/contest/455/problem/E)

Lazy Propagating:  
[52C - Circular RMQ](https://codeforces.com/contest/52/problem/C)  
[145E - Lucky Queries](https://codeforces.com/contest/145/problem/E)  
[558E - A Simple Task](https://codeforces.com/contest/558/problem/E)  
[240F - TorCoder](https://codeforces.com/contest/240/problem/F)  
[446C - DZY Loves Fibonacci Numbers](https://codeforces.com/contest/446/problem/C)  
[115E - Linear Kingdom Races](https://codeforces.com/contest/115/problem/E)  
[438D - The Child and Sequence](https://codeforces.com/contest/438/problem/D)  
[121E - Lucky Array](https://codeforces.com/contest/121/problem/E)  
[610E - Alphabet Permutations](https://codeforces.com/contest/610/problem/E)  
[580E - Kefa and Watch](https://codeforces.com/contest/580/problem/E)

Segment tree with Vector:  
[369E - Valera and Queries](https://codeforces.com/contest/369/problem/E)  
[610D - Vika and Segments](https://codeforces.com/contest/610/problem/D)

Offline Query:  
[301D - Yaroslav and Divisors](https://codeforces.com/contest/301/problem/D)  
[500E - New Year Domino](https://codeforces.com/contest/500/problem/E)

Segment Tree & Dp:  
[474E - Pillars](https://codeforces.com/contest/474/problem/E)  
[597C - Subsequences](https://codeforces.com/contest/597/problem/C)  
[56E - Domino Principle](https://codeforces.com/contest/56/problem/E)

Segment Tree & Bits:  
[482B - Interesting Array](https://codeforces.com/contest/482/problem/B)  
[242E - XOR on Segment](https://codeforces.com/contest/242/problem/E)

Segment Tree & Tree:  
[383C - Propagating tree](https://codeforces.com/contest/383/problem/C)  
[343D - Water Tree](https://codeforces.com/contest/343/problem/D)  
[173E - Camping Groups](https://codeforces.com/contest/173/problem/E)  
[276E - Little Girl and Problem on Trees](https://codeforces.com/contest/276/problem/E)  
[396C - On Changing Tree](https://codeforces.com/contest/396/problem/C)  
[516D - Drazil and Morning Exercise](https://codeforces.com/contest/516/problem/D)  
[375D - Tree and Queries](https://codeforces.com/contest/375/problem/D)

DSU

Codeforces [277A](http://codeforces.com/problemset/problem/277/A) , [356A](http://codeforces.com/problemset/problem/356/A) , [292D](http://codeforces.com/problemset/problem/292/D) , 371D

ფენვიკი

## timus 1316. Electronic Auction

## codeforces 61E 652D (მონაკვეთები)

## codeforces 816B (კარენი და ყავა)

## codeforces 597C (10 ფენვიკი)

## codeforces 1234D

## codeforces 1430E 74D 1674F

ტოპოლოგიური სორტირება

Codeforces 510 C

RMQ

514D

LCA

733F

[208E - Братья по крови](https://codeforces.com/contest/208/problem/E)  
[191C - Дураки и дороги](https://codeforces.com/contest/191/problem/C)  
[519E - A и B и аудитории](https://codeforces.com/contest/519/problem/E)  
[587C - Duff в армии](https://codeforces.com/contest/587/problem/C)  
[609E - Минимальное остовное дерево по каждому ребру](https://codeforces.com/contest/609/problem/E)  
[178B3 - Жадные торговцы](https://codeforces.com/contest/178/problem/B3)  
[176E - Археология](https://codeforces.com/contest/176/problem/E)  
[466E - Граф информаций](https://codeforces.com/contest/466/problem/E)

[A Secret Mission — LOJ](http://www.lightoj.com/volume_showproblem.php?problem=1101) , [Min Max Roads — LOJ](http://www.lightoj.com/volume_showproblem.php?problem=1162) , [LCA — SPOJ](http://www.spoj.com/problems/LCA/) , [Kth Ancestor — HackerRank](https://www.hackerrank.com/challenges/kth-ancestor)

[504E - Миша и LCP на дереве](https://codeforces.com/contest/504/problem/E) LCA + hash

[Qtree spoj](https://codeforces.com/blog/entry/spoj.com/problems/QTREE) : LCA + Heavy-light Decomposition + segment tree

[372D - Весело выбирать поддерево](https://codeforces.com/contest/372/problem/D) : LCA + sorting + dfs(starting time calculating) + two\_pointer (or Heavy-light Decomposition)

[342E - Ксюша и дерево](https://codeforces.com/contest/342/problem/E) : LCA + Sqrt Decomposition

[342E — Xenia and Tree](https://codeforces.com/contest/342/problem/E) can be solved by both -[decomposition](https://codeforces.com/blog/entry/7383) and [centroid decomposition](https://threads-iiith.quora.com/Centroid-Decomposition-of-a-Tree).

[629E - Фома Дор и дороги](https://codeforces.com/contest/629/problem/E)

[916E - Джейми и дерево](https://codeforces.com/contest/916/problem/E)

[986E - Задача Принца](https://codeforces.com/contest/986/problem/E)

<http://www.spoj.com/problems/DISQUERY/> this is problem based only on LCA. this problem is from the SPOJ. update: this problem is not based only on lca, sorry for mistake.

<https://codeforces.com/contest/1062/problem/E> Nice LCA problem ^\_^

**Centroid Decomposition**

* [321C - Ciel the Commander](https://codeforces.com/contest/321/problem/C)
* [766E - Mahmoud and a xor trip](https://codeforces.com/contest/766/problem/E)
* [716E - Digit Tree](https://codeforces.com/contest/716/problem/E)
* [161D - Distance in Tree](https://codeforces.com/contest/161/problem/D)
* [776F - Sherlock's bet to Moriarty](https://codeforces.com/contest/776/problem/F)
* [379F - New Year Tree](https://codeforces.com/contest/379/problem/F)
* [342E - Xenia and Tree](https://codeforces.com/contest/342/problem/E)
* [293E - Close Vertices](https://codeforces.com/contest/293/problem/E)
* [150E - Freezing with Style](https://codeforces.com/contest/150/problem/E)
* [348E - Pilgrims](https://codeforces.com/contest/348/problem/E)

მატრიცების გადამრავლება

<https://www.spoj.com/problems/LISA/>

ფლოიდ–ვორშელი

29E

295B

ერატოსთენეს საცერი

154B 26A 17A 546D

პრეფიქს-ჯამები

433B

SET

520A (პანგრამა), 220B, 527C (მინის დაჭრა), 855A

390A, 992A, 780A (წინდები), 787A (რიკი და მორტი)

371D (ჭურჭლები), 356A (რაინდთა ტურნირი), 899E,

224B, 962D (ერთნაირების შერწყმა), 1234D (26 SET)

675D (BST), 368B, 900C (რეკორდი) , 343D, 920E 830B 468B

MAP

4C, 702B, 1100B (ორმაგი მეპი) 43A

757A 220B 519D 69E(სეტი+მეპი)

567C (პროგრესია) 733D

670C 977E 652D 1679B 1722C

LCA

[208E - Братья по крови](https://codeforces.com/group/LvpSmvtoB3/contest/208/problem/E)  
[191C - Дураки и дороги](https://codeforces.com/group/LvpSmvtoB3/contest/191/problem/C)  
[519E - A и B и аудитории](https://codeforces.com/group/LvpSmvtoB3/contest/519/problem/E)  
[587C - Duff в армии](https://codeforces.com/group/LvpSmvtoB3/contest/587/problem/C)  
[609E - Минимальное остовное дерево по каждому ребру](https://codeforces.com/group/LvpSmvtoB3/contest/609/problem/E)  
[178B3 - Жадные торговцы](https://codeforces.com/group/LvpSmvtoB3/contest/178/problem/B3)  
[176E - Археология](https://codeforces.com/group/LvpSmvtoB3/contest/176/problem/E)  
[466E - Граф информаций](https://codeforces.com/group/LvpSmvtoB3/contest/466/problem/E)

ძებნის ორობითი ხე

675D

ხის შემოვლა

http://codeforces.com/problemset/problem/29/D

ჯამები ინტერვალზე

276c

next\_permutation

Balanced Teams - usaco 2013-14 january

სტატიკური აჯამვა

706B

სიღრმეში ძებნა DFS

117C

107A

SQRT დეკომპოზიცია

455D

იდეა

622C

რეალიზაცია

686A 678A

255A

545D

581B

[1004E - Sonya and Ice Cream](https://codeforces.com/contest/1004/problem/E) (rating: 2400)

Solution

Implementation by [nor](https://codeforces.com/profile/nor) (C++): [151009669](https://codeforces.com/contest/1004/submission/151009669)

[633F - The Chocolate Spree](https://codeforces.com/contest/633/problem/F) (rating: 2600)

Solution

Implementation by [nor](https://codeforces.com/profile/nor) (C++): [151018941](https://codeforces.com/contest/633/submission/151018941)

[1434D - Roads and Ramen](https://codeforces.com/contest/1434/problem/D) (rating: 2800)

Solution

Implementation by [nor](https://codeforces.com/profile/nor) (C++): [151024814](https://codeforces.com/contest/1434/submission/151024814)

Other problems

[CSES — Tree Distances I](https://cses.fi/problemset/task/1132) (to check your implementation) (suggested by [nor](https://codeforces.com/profile/nor))  
[102694B - Dynamic Diameter](https://codeforces.com/gym/102694/problem/B)  
[1404B - Tree Tag](https://codeforces.com/contest/1404/problem/B) ([**flashmt**](https://codeforces.com/profile/flashmt))  
[734E - Anton and Tree](https://codeforces.com/contest/734/problem/E) ([**preet\_25**](https://codeforces.com/profile/preet_25))  
[456E - Civilization](https://codeforces.com/contest/456/problem/E) ([**RetiringNextCentury**](https://codeforces.com/profile/RetiringNextCentury))  
[Code Jam 2022 — Interesting Outing](https://codingcompetitions.withgoogle.com/codejamio/round/00000000009d9870/0000000000a33bc7) ([**srishtik\_16**](https://codeforces.com/profile/srishtik_16))  
[abc221\_f](https://atcoder.jp/contests/abc221/tasks/abc221_f)  
[IOI 2013 — Dreaming](https://oj.uz/problem/view/IOI13_dreaming) ([**timreizin**](https://codeforces.com/profile/timreizin))  
[agc033\_c](https://atcoder.jp/contests/agc033/tasks/agc033_c) ([**antontrygubO\_o**](https://codeforces.com/profile/antontrygubO_o))  
[arc117\_d](https://atcoder.jp/contests/arc117/tasks/arc117_d) ([**flashmt**](https://codeforces.com/profile/flashmt))  
[USACO 2018 — New Barns](http://www.usaco.org/index.php?page=viewproblem2&cpid=817) ([**Olympia**](https://codeforces.com/profile/Olympia))  
[1617E - Christmas Chocolates](https://codeforces.com/contest/1617/problem/E) ([**feecIe6418**](https://codeforces.com/profile/feecIe6418))  
[arc108\_f](https://atcoder.jp/contests/arc108/tasks/arc108_f) ([**feecIe6418**](https://codeforces.com/profile/feecIe6418))  
[IOI 2015 — Towns](https://oj.uz/problem/view/IOI15_towns) ([**defnotmee**](https://codeforces.com/profile/defnotmee))  
[1214H - Tiles Placement](https://codeforces.com/contest/1214/problem/H) ([**lrvideckis**](https://codeforces.com/profile/lrvideckis))  
[CEOI 2019 — Dynamic Diameter](https://oj.uz/problem/view/CEOI19_diameter) (hard) ([**nor**](https://codeforces.com/profile/nor))  
[RMI 2021 — Paths](https://oj.uz/problem/view/RMI21_paths) (hard) ([**valeriu**](https://codeforces.com/profile/valeriu))

Eolymp

<https://www.eolymp.com/en/contests/18181> C intro

https://www.eolymp.com/ru/contests/21578 C intro1

https://www.eolymp.com/ru/contests/21579 C intro2

27401, 27402 WHILE ციკლი

27222, 27223 FOR ციკლი

27660, 27661, 27946, 28164 მასივები

<https://www.eolymp.com/en/contests/16683> ორგანზომილებიანი მასივები

<https://www.eolymp.com/en/contests/12396> bit operation <https://www.eolymp.com/ru/contests/21176> [Sort & Search](https://www.eolymp.com/ru/contests/21176)

<https://www.eolymp.com/en/contests/6686> რეკურსია

<https://www.eolymp.com/ru/contests/21168> [STL Intro](https://www.eolymp.com/ru/contests/21168)

16639 set-multiset intro

<https://www.eolymp.com/en/contests/31878> სტეკი

<https://www.eolymp.com/en/contests/32107> რიგი

27015 ფენვიკი

<https://www.eolymp.com/ru/contests/21480> greedy

30008 [Bipartite graphs](https://www.eolymp.com/ru/contests/30008)

26983 დეიქსტრა

27209 ბელმან-ფორდი

# Graph Introduction 992 (Cities and roads), 993 (Traffic lights), 994(Coloured rain), 2470 (Checking for an undirected graph), 2471 (From adjacency matrix to the list of edges), 2472 (Operations on graph), 3981 (From adjacency matrix to adjacency list), 3982 (From adjacency list to adjacency matrix), 3986 (Sources and sinks), 3987 (Complete graph), 4761 (Loops)

შეტანა-გამოტანა

https://www.eolymp.com/en/contests/26845

https://www.eolymp.com/en/contests/26846

IF ოპერატორი

<https://www.eolymp.com/en/contests/27001>

<https://www.eolymp.com/en/contests/27002>

FOR

<https://www.eolymp.com/en/contests/27222>

<https://www.eolymp.com/en/contests/27223>

WHILE

<https://www.eolymp.com/en/contests/27401>

<https://www.eolymp.com/en/contests/27402>

სტრინგები, char-მასივები

<https://www.eolymp.com/en/contests/27946>

https://www.eolymp.com/en/contests/28164

მასივები

<https://www.eolymp.com/en/contests/27660>

https://www.eolymp.com/en/contests/27661

aho

<http://www.spoj.com/problems/AHOCUR/> 5 //Aho-Corassic + DP

bfs

11312 UVA (3)

11392 UVA (4)

10968 UVA (3) //EASY + NICE (bfs withot <=2 nodes)

10888 UVA (4) //VERY NICE — but not main technique ... ++ DP /or/ MCMF

<http://www.spoj.com/problems/DIGOKEYS/> (4) //Easy [Nice problem — weird statement]

<http://www.spoj.com/problems/SPIKES/> (3) //Easy bfs (# of 's' \* 2)

<http://www.spoj.com/problems/MULTII/> (4) //VERY NICE: BFS over numbers (K\*10+d)%N

bfs-grid

10977 UVA (3)

928 UVA (3)

13116 UVA (4)

314 UVA (3)

11487 UVA (4)

5622 LA (7)

11931 UVA (5)

<http://www.spoj.com/problems/KNMOVE/> 3 //simple knights

<http://www.spoj.com/problems/SERGRID/> 3 //almost classical

<http://www.spoj.com/problems/NAKANJ/> 3 //Classical chess — KNIGHT

<http://www.spoj.com/problems/PUCMM223/> (4) //NICE (but not many languages) — 2 moving [x][y]

<http://www.spoj.com/problems/SPIRALGR/> (4) //NICE (not typical) [SIEVE]

<http://www.spoj.com/problems/DCEPC706/> (4) //NICE — travelling outside

big

11645 UVA 4

11375 UVA 3

<http://www.spoj.com/problems/MINNUM/> 3 // BIG/9+!!(BIG%9)

10844 UVA 4 //Bell numbers + big (might be slightly slow!)

<http://www.spoj.com/problems/NITT2/> 2 //Divisibility by two constants

binary\_search

<http://codeforces.com/contest/729/problem/C> 3

<http://codeforces.com/contest/714/problem/D> 8

13150 (UVA) 4

<http://codeforces.com/contest/749/problem/D> 5

11692 (UVA) 4

11516 (UVA) 3

<http://codeforces.com/contest/760/problem/B> 3

<http://codeforces.com/contest/675/problem/D> 4 //

<http://www.spoj.com/problems/NDS/> 4 //BS over LIS

<http://www.spoj.com/problems/VECTAR4/> 3

<http://codeforces.com/contest/767/problem/D> 4 //NICE

<http://codeforces.com/contest/627/problem/D> (7) //with dp — NICE

<http://codeforces.com/contest/779/problem/D> (3) //NICE + EASY

<http://www.spoj.com/problems/CNTINDX/> (4) //Map+BS === OK

13177 UVA (3) //BS over answer == OK

<http://codeforces.com/contest/801/problem/C> (3) //BS + SUM -EASY

<http://codeforces.com/contest/803/problem/D> (3) //BS by answer

<http://codeforces.com/contest/807/problem/C> (3) //Or math

<http://codeforces.com/contest/818/problem/F> (4) //NICE — Live VS Clique

<http://codeforces.com/contest/845/problem/E> (5) //VERY NICE — min(X,Y) .. add time, repeat

<http://www.spoj.com/problems/MATHLOVE/> (2) //BS + Gaus (or otter ways)

<http://www.spoj.com/problems/SABBIRGAME/> (3) //Binary search over answer ::max(0,ANS)

<http://codeforces.com/contest/846/problem/D> (4) //BS+Precalculation OR 2D-RMQ

bits

11659 UVA (4)

11535 UVA (4)

<http://codeforces.com/contest/779/problem/E> (5) //NICE + Parsing

<http://www.spoj.com/problems/EC_CONB/> (1) //reverse numbers

<http://codeforces.com/contest/769/problem/D> (4) //freq + brute-force

<http://www.spoj.com/problems/HAP01/> (2) //builtin\_popcount

bitset

<http://codeforces.com/contest/754/problem/E> 6

<http://www.spoj.com/problems/UCBINTC/> 5 //polymul with bitset

blossom

11439

bridges

<http://codeforces.com/contest/732/problem/F> 7

<http://codeforces.com/contest/700/problem/C> 7

<http://www.spoj.com/problems/EC_P/> (3) //bridges ONLY

<http://www.spoj.com/problems/SUBMERGE/> (3) //Direct articulation

<http://www.spoj.com/problems/GRAFFDEF/> (5) //Bridge tree

brute-force

UVA 12169 (2)

<http://codeforces.com/contest/725/problem/C> 4

<http://codeforces.com/contest/725/problem/E> 6

<http://codeforces.com/contest/724/problem/B> 3

11961 UVA (2)

11898 UVA (4)

11659 UVA (4)

<http://codeforces.com/contest/753/problem/C> 7

11699 UVA (4)

11548 UVA (3)

11471 UVA (5) //With dynamic programming

<http://codeforces.com/contest/698/problem/D> 8 //with geometry

11206 UVA (6) //4^20 (but somehow passes)

11214 UVA (6) //Úvaha + pruning

11127 UVA (4) //Simple dfs [just realize you can do so]

<http://www.spoj.com/problems/BOKAM143SOU/> (3) //just implement for-cycles

<http://www.spoj.com/problems/BLOPER/> (4) dfs with little pruning

13173 UVA (3) //just brute-force + branching

<http://codeforces.com/contest/799/problem/D> (4) //VERY NICE [only top 34 needed] — trick with 2 [~20]

10890 UVA (4) //Simple brute-force times out, but with simple pruning AC (answer detection

<http://codeforces.com/contest/813/problem/B> (3) //All\*All (BF) care for overflow! NICE & EASY

<http://codeforces.com/contest/817/problem/C> (3) //Check S+Constant (NICE!)

10732 UVA (2) //Brute-force passes .. just don't trust them O(N^2)

10748 UVA (5) //VERY Nice (knights have K^2 moves not 8^K)

<http://codeforces.com/contest/818/problem/D> (4) //NICE for each 'A' check all remaining (max SQRT)

<http://codeforces.com/contest/834/problem/E> (5) //NICE — hard to imple: all 11122...999 OK

<http://codeforces.com/contest/839/problem/E> (5) //NICE! Max-Clique

<http://www.spoj.com/problems/JHAGIRLS/> (4) //Efficient — or store output in array

<http://codeforces.com/contest/846/problem/B> (3) //Brute-force

<http://www.spoj.com/problems/ALONE/> (4) //Generate everything <10^15 [NICE]

centroid

<http://codeforces.com/contest/715/problem/C> 9

<http://codeforces.com/contest/741/problem/D> 8

13164 UVA (7)

<http://codeforces.com/contest/752/problem/F> 5

<http://codeforces.com/contest/766/problem/E> 6

<http://codeforces.com/contest/833/problem/D> 7 //Very nice — hard (thinking + imple) + FW

coloring

<http://codeforces.com/contest/741/problem/C> 6

11331 UVA (4)

<http://codeforces.com/contest/664/problem/D> 4

combinatorics

12001 UVA (3)

12034 UVA (4)

11719 UVA (5)

11798 UVA (5)

11282 UVA (4) //dearrangement

11174 UVA (4)

<http://codeforces.com/contest/666/problem/C> 7

<http://www.spoj.com/problems/JOKER1/> 3 prod(Ai-i)

<http://www.spoj.com/problems/ANTP/> //4

<http://codeforces.com/contest/645/problem/E> (5) //formula: A[i]=Sum(A)+1

<http://www.spoj.com/problems/SPCE/> (5) // N^{K-2}\*Prod(comp\_size)

<http://codeforces.com/contest/785/problem/D> (5) // F'(' C"(+)-1","("

13184 UVA (3)

<http://codeforces.com/contest/816/problem/D> (5) // Observation

13214 (4) //OEIS? : C(N+M-2,N-1)

<http://codeforces.com/contest/844/problem/B> (2) //Easy — pro prvaky

<http://www.spoj.com/problems/JOSWAP/> (3) //Frequence array

<http://www.spoj.com/problems/UCV2013E/> (4) //NICE&EASY: Choose steps to direction

<http://www.spoj.com/problems/PARCARD1/> (4) //Partition function (raw)

<http://www.spoj.com/problems/GOODB/> (2) //Easy (NICE): Choose [order]

<http://www.spoj.com/problems/LOOPEXP/> (4) //A000254/N!

<http://www.spoj.com/problems/DTPOLY/> (5) //DP might work too

<http://www.spoj.com/problems/DTPOLY2/> (7) //Harder version of above (NICE but hell)

<http://www.spoj.com/problems/HC12/> (3) //NICE — Sort + C(i,K-1)\*A[i]

<http://www.spoj.com/problems/STONE2/> (4) //NICE — Mostly DP [INVERSION][FACTORIAL]

constructive

<http://codeforces.com/contest/802/problem/H> (6) //We can do "N+k" by adding a letter p+k\*x+u+xx

dfs

12186 UVA (3)

<http://codeforces.com/contest/734/problem/E> (5)

<http://codeforces.com/contest/727/problem/A> (3)

<http://codeforces.com/contest/723/problem/E> (6)

<http://codeforces.com/contest/709/problem/E> (6)

<http://codeforces.com/contest/710/problem/E> (4)

<http://codeforces.com/contest/758/problem/E> (8)

11323 UVA (5)

<http://codeforces.com/contest/760/problem/B> (3)

<http://codeforces.com/contest/761/problem/E> (6)

<http://codeforces.com/contest/638/problem/B> (3) //connect cons. letters

<http://codeforces.com/contest/638/problem/C> (4) //greedy idea — easy

<http://codeforces.com/contest/638/problem/D> (5) //spec-DAG articulatin

<http://codeforces.com/contest/767/problem/C> (4)

<http://codeforces.com/contest/781/problem/C> (5)

<http://codeforces.com/contest/794/problem/D> (5) //NICE! Right done dfs

<http://codeforces.com/contest/802/problem/K> (5) //Slightly DP-like (NICE) TREE

<http://codeforces.com/contest/813/problem/C> (3) //Simply 2 DFS: NICE + EASY

<http://codeforces.com/contest/841/problem/D> (4) //DFS while tracking "next"

<http://codeforces.com/contest/845/problem/G> (5) //Keep track of cycles

<http://codeforces.com/contest/844/problem/E> (5) //Post-Order → line, Connect i → N-2: star

<http://www.spoj.com/problems/CAC/> (5) //VERY NICE! — Find all cycles in cactus

<http://codeforces.com/contest/849/problem/C> (3) //State search by gauss

<http://codeforces.com/contest/846/problem/E> (5) //NICE: DFS + some overflow logic

<http://www.spoj.com/problems/KOZE/> (3) //NICE: Floods

<http://www.spoj.com/problems/RIOI_2_3/> (4) //DFS /OR/ BFS /OR/ DSU [NICE][EASY][BF]

<http://www.spoj.com/problems/MAKEMAZE/> (3) //EASY — Simple dfs on grid

digits

<http://www.spoj.com/problems/PR003004/> (4) //Simple digits count

<http://codeforces.com/contest/770/problem/B> (3) //max num max digsum

dijkstra

<http://codeforces.com/contest/716/problem/D> 7

12047 UVA 4

11514 UVA 4

<http://codeforces.com/contest/757/problem/F> 7

11338 UVA (4)

11374 UVA (4)

11097 UVA (4) //Divide to N\*1000 nodes and go!

13172 UVA (5) //6\*DJ per query + permutations

10816 UVA (4) //Easy Linear-Search by answer + DJ with path

<http://codeforces.com/contest/827/problem/F> 7 //Very nice — Even&Odd

divide\_conquer

<http://codeforces.com/contest/817/problem/D> (5) //Very nice NlogN

dp

11552 UVA (3)

12172 UVA (3)

4507 LA (5)

4510 LA (5) [+ geometry]

12181 UVA (6)

<http://codeforces.com/contest/729/problem/F> 6

<http://codeforces.com/contest/735/problem/E> 9

<http://codeforces.com/contest/731/problem/E> 5

12030 UVA (4)

<http://codeforces.com/contest/721/problem/E> 7

<http://codeforces.com/contest/742/problem/D> 4

12040 UVA (5)

<http://codeforces.com/contest/712/problem/D> 5

13162 UVA (6)

<http://codeforces.com/contest/743/problem/E> 6

11908 UVA (3)

11932 UVA (4)

<http://codeforces.com/contest/745/problem/E> (7)

11806 UVA (4)

<http://codeforces.com/contest/747/problem/F> (5)

11843 UVA (4)

<http://codeforces.com/contest/752/problem/E> (5)

<http://codeforces.com/contest/703/problem/E> (7)

11753 UVA (4)

11725 UVA (5)

<http://codeforces.com/contest/722/problem/E> (9)

<http://codeforces.com/contest/760/problem/F> (8)

11795 UVA (3)

11654 UVA (4)

11523 UVA (5)

11404 UVA (4)

11432 UVA (4)

11451 UVA (4) //C==20 mistake in statement

11301 UVA (4)

<http://codeforces.com/contest/762/problem/D> 5

11361 UVA (4) //digit DP

11365 UVA (7)

11391 UVA (4) //easy+implementation

11394 UVA (3)

11218 UVA (2)

11125 UVA (4) //slightly implementation

11076 UVA (3)

11081 UVA (4) //3 string subsequences (beware of fail)

<http://codeforces.com/contest/678/problem/E> (5) //bitset dp + probability

<http://codeforces.com/contest/766/problem/C> (4)

<http://codeforces.com/contest/667/problem/C> (3)

<http://www.spoj.com/problems/MOVIFAN/> (3)

<http://www.spoj.com/problems/ORDSUM23/> (3)

<http://www.spoj.com/problems/DIVSEQ/> (4) //N^3 (but better...) works fine

<http://codeforces.com/contest/633/problem/F> (7) //Tree dp

<http://www.spoj.com/problems/ADJDUCKS/> (4) sort + pick 2-3 continous O(N)

<http://www.spoj.com/problems/JLNT/> (4) //pick 0 or 2 | 1e3\*5e3

<http://www.spoj.com/problems/TPCPALIN/> (5) //500^3 works (3rd countable)

<http://www.spoj.com/problems/COLORSEG/> (4) //50^4==OK 50^4log(N)=TLE NICE

<http://www.spoj.com/problems/POWERCAR/> (3) //1e3\*1e3\*2 — follow rules

<http://www.spoj.com/problems/INGRED/> (5) //TSP-like [reduce + go]

<http://www.spoj.com/problems/BADXOR/> (4) //classical subsets

<http://www.spoj.com/problems/SPCO/> (5) //64\*64\*2 DP {OPT: prime O(1) + clear only half}

<http://www.spoj.com/problems/WAYHOME/> (5) //NICE: 1) 1\*1 b)12,1,\*\*,2

<http://www.spoj.com/problems/NFURY/> (2) //Minimal sum of squares

<http://www.spoj.com/problems/GDIL/> (3) //combinatorics

<http://codeforces.com/contest/791/problem/D> (5) //Tree

<http://codeforces.com/contest/791/problem/E> (6) //V,K,X — pick any

<http://codeforces.com/contest/789/problem/C> (3)

13176 (4) //N^6

13179 (5) //NICE [Ath][Bth][TimeDiff]

<http://codeforces.com/contest/796/problem/E> (6) //NICE: N\*P\*K\*K (WC can't happen!)

<http://codeforces.com/contest/797/problem/E> (4) //NICE: Almost BF-able (but care of low K)

<http://codeforces.com/contest/793/problem/D> (3) //NICE & EASY: begin/end/actual/USED

<http://codeforces.com/contest/803/problem/E> (4) //State search — many IF's (EASY)

<http://codeforces.com/contest/805/problem/F> (7) //NICE: DP on tree + fast BF + hack

<http://codeforces.com/contest/808/problem/E> (5) //NICE!

<http://codeforces.com/contest/811/problem/C> (4) //Precalculate + DP (greedy thinking)

10817 UVA 4 //Easy but slightly implementation

10859 UVA 4 //Nice — on tree .. but for a reason small constrains

10898 UVA 4 //Hash is lesser than 1e6 .. try everything

<http://codeforces.com/contest/812/problem/B> (3) //Not only DP, yet imho easiest ..many spec cases

<http://codeforces.com/contest/813/problem/D> (5) //VERY VERY NICE — N\*N (none picked between a/b)

<http://codeforces.com/contest/814/problem/E> 5 //VERY NICE — Harder imple: Combinatorics

<http://codeforces.com/problemset/problem/816/E> (6) //NICE — Tree (hard 2C complexity)

<http://codeforces.com/contest/837/problem/D> (5) //NICE — yet kinda pain [must be iterative]

<http://www.spoj.com/problems/AUT/> (4) //NICE — K is interesting ~ at most 1600

<http://www.spoj.com/problems/GNYR04C/> (3) //Easy — Nice idea [Big→ Low approach]

<http://www.spoj.com/problems/TIEROPE/> (4) //Process 2\*L ~ otherwise pick BIG

<http://www.spoj.com/problems/IITKWPCE/> (4) //Palindromes [efficiency!] — NICE!

IITKWPCD SPOJ (4) //+Slightly geometry

<http://www.spoj.com/problems/LKS/> (3) //Classical knapsack

<http://www.spoj.com/problems/UOFTAE/> (3) //Easy & Sympatic DP

<http://www.spoj.com/problems/DCOWS/> (4) //Very NICE (sort + GO)

<http://www.spoj.com/problems/FARIDA/> (3) //Easy & Sympatic ((u+1) | Price+(u+2))

<http://www.spoj.com/problems/AU7_5/> (2) //EASY: dyn(n-1)+dyn(n-k-1)

<http://www.spoj.com/problems/NAIVELOK/> (4) //NICE [depalindromisation]

<http://codeforces.com/contest/846/problem/C> (4) //With print

<http://www.spoj.com/problems/CNT_LUCK/> (4) //Number (binary) dp [NICE] {ull care 0-1}

<http://www.spoj.com/problems/MAY99_4/> (3) //Almost combinatoric Sub and 0/1,1/0

<http://www.spoj.com/problems/GEEKOUNT/> (4) //Number dp

<http://www.spoj.com/problems/MUTDNA/> (4) // N\*2 (turned?) [not sure if grd poss.]

<http://www.spoj.com/problems/RIOI_3_2/> (5) //VERY NICE (easy imple — Number Theory thinking)

<http://www.spoj.com/problems/MAXWOODS/> (3) //NICE [EASY][GRID]

<http://www.spoj.com/problems/DIEHARD/> (3) //Easy — prolly solvable by greedy (but dp is easier)

<http://www.spoj.com/problems/DCEPC810/> (4) //VERY VERY NICE — Subsequence 2pointers+2bools

<http://www.spoj.com/problems/EQ2/> (4) //NICE: Digit + Carry (from back) — iff-party

dsu

<http://codeforces.com/contest/723/problem/F> 7

13153 UVA (5)

13169 UVA (3)

11987 UVA (3)

11474 UVA (4)

<http://codeforces.com/contest/687/problem/D> 6

<http://codeforces.com/contest/680/problem/E> 7 //+precalculation/brute force

<http://codeforces.com/contest/766/problem/D> 5

<http://www.spoj.com/problems/LEXSTR/> (3) //Nice na stringu

<http://codeforces.com/contest/805/problem/C> 3 //NICE (dijkstra like :P)

<http://www.spoj.com/problems/IITKWPCI/> (3) //VERY NICE

<http://www.spoj.com/problems/FRNDCIRC> (3) //Classical DSU (NICE for practice)

<http://www.spoj.com/problems/FOXLINGS/> (3) Easy — just renumbering

<http://www.spoj.com/problems/NITTROAD/> (4) //Process from back

<http://www.spoj.com/problems/SHAHBG/> (2) //DSU not needes (simulated by array)

euler\_function

<http://www.spoj.com/problems/NAJPWG/> 4 //Gauss-Euler

<http://www.spoj.com/problems/DCEPC12G/> (5) //Do what is written there

<http://www.spoj.com/problems/INVPHI/> (5) //Inverse euler

euler\_tour

<http://codeforces.com/contest/789/problem/D> //Adj EG + Self/everything

events

<http://codeforces.com/contest/747/problem/C> 3

11776 UVA (3)

11134 UVA (3)

factorization

12005 UVA (7)

12062 UVA (6)

11960 UVA (3)

<http://www.spoj.com/problems/FACTCG2/> (3)

<http://www.spoj.com/problems/FACT0/> (4)

<http://codeforces.com/contest/546/problem/D> 5

<http://codeforces.com/contest/222/problem/C> 6

<http://www.spoj.com/problems/COMDIV/> 3

<http://www.spoj.com/problems/SINEGGS/> 3

<http://www.spoj.com/problems/BDOI16B/> 4

<http://www.spoj.com/problems/HG/> 4 //Map == OK

11099 UVA (3) //factor + recursion

13194 UVA (3) //factorize+generate /or just check

13191 UVA (6) //Pollard-Rho

<http://codeforces.com/contest/818/problem/E> (4) // Efficient + Two Pointers

<http://codeforces.com/contest/831/problem/F> (6) //MAGIC

<http://codeforces.com/contest/839/problem/D> (4) // Combinatorics + IE

<http://www.spoj.com/problems/SAS002/> (5) //Find all divisors (big numbers)

<http://www.spoj.com/problems/GCDS/> (4) //Lowest unused prime

<http://www.spoj.com/problems/IITKWPCF/> (4) //Nonprime divisors of N/2

<http://codeforces.com/contest/851/problem/D> (4) //Properties of GCD + factor: NICE

<http://www.spoj.com/problems/PTIME/> (3) //Low bounds — check each prime independently

fenwick

<http://codeforces.com/contest/707/problem/E> 7 [2D]

<http://codeforces.com/contest/749/problem/E> 8

<http://codeforces.com/problemset/gymProblem/101055/D> 5 [2D]

11240 UVA (4)

<http://codeforces.com/contest/669/problem/E> 5 //fenwicks — sparse

<http://codeforces.com/contest/777/problem/E> 4 //MAXIMUM

<http://www.spoj.com/problems/TULIPNUM/> 4 //inc — 1 nor+num|sum(A[B],A[E])

<http://codeforces.com/contest/799/problem/C> 3 //MAX FW (but possibly easier?)

<http://codeforces.com/contest/831/problem/E> 4 //MAP to get ORDER — FW == LIST

<http://www.spoj.com/problems/SAS001/> (4) //Nice — number of inversions + 2P

<http://www.spoj.com/problems/TPGA/> (4) //NICE — Lesser\*(N-i-1)!

<http://www.spoj.com/problems/SGIFT/> (4) //BS works too

<http://www.spoj.com/problems/SUMSUM/> (5) //Bit-by-Bit cnt 0/1

<http://www.spoj.com/problems/AKVQLD03/> (3) //Classical fenwick — easy

<http://www.spoj.com/problems/ZIGZAG2/> (6) //Very nice — FW + BS + DP

<http://codeforces.com/contest/849/problem/E> (7) //2D Fenwick / ST+TP [NICE]

<http://www.spoj.com/problems/CRAYON/> (5) //VERY NICE [2\*FW — begin + end]

<http://www.spoj.com/problems/NITT8/> (4) //Norm. + Store indices in MAX-Fenwick [REVERSE] [VERY NICE]

<http://www.spoj.com/problems/DCEPC705/> (4) //NICE! Sort + Fenwick

fft

<http://www.spoj.com/problems/TSUM/> 5

13182 UVA 5 //ACTG hamming

<http://codeforces.com/contest/827/problem/E> (8) //MAGIC

flow

<http://www.spoj.com/problems/FASTFLOW/> 3 //simple flow

<http://codeforces.com/contest/808/problem/F> 6 //NICE — nontrivial (max matching with bigger flows)

floyd-warshall

13211 UVA (5) //NICE — FW adding states

<http://www.spoj.com/problems/ROHAAN/> (3) //Classical

game\_theory

11859 UVA 4

11863 UVA 4

11892 UVA 3 //Probably solved by many

11534 UVA 5

<http://www.spoj.com/problems/VECTAR11/> 4 //Nsqrt(N) passes [with break]

<http://codeforces.com/contest/768/problem/E> (4) //NICE — Grundy

<http://www.spoj.com/problems/SYNC13C/> (4) //2\*DP {maybe not seeing sth}

<http://codeforces.com/contest/787/problem/C> (4)

<http://codeforces.com/contest/794/problem/C> (3) //Find optimal strategy: choose back/front

<http://codeforces.com/contest/794/problem/E> (5) //NICE Find stategy: Even/Odd

<http://codeforces.com/contest/812/problem/E> (7) //Advanced NIM strategy

<http://www.spoj.com/problems/GAMEMVS/> (4) //Nimbers (Ai^X)<=Ai

<http://www.spoj.com/problems/PLAYGAME/> (3) //Check pattern

<http://www.spoj.com/problems/CHAOS_CC/> (4) //VERY NICE [nimbers]

<http://codeforces.com/contest/851/problem/E> (5) //Very nice [nimbers] [bitset]

<http://www.spoj.com/problems/CHGROOM/> (4) //+Factorisation [NICE & Easy]: Win unless 2 prime factors

geometry

12173 UVA 3

12194 UVA 4

11894 UVA 3

11769 UVA 7

11665 UVA 5

11509 UVA 4

11355 UVA 5

11265 UVA 6 //Nice one | polygon — cut/pt-check/area

11123 UVA 4 //Counting trapezoids

11177 UVA 6 //BS+Polygon/Circle intersection

11186 UVA 3

11008 UVA 5 //with DP → #intersected triangles

11012 UVA 5 //Nejvzdálenější body (Manhatton 3D)

11072 UVA 4 //Body v poly gonu

<http://codeforces.com/problemset/problem/682/E> 6 (biggest triangle)

<http://codeforces.com/contest/672/problem/C> 4 //easy — just think it up

<http://codeforces.com/contest/667/problem/A> 2 //vzorecky

<http://codeforces.com/contest/793/problem/C> 5 //EASY but beware of epsilons (NICE)

<http://codeforces.com/contest/794/problem/B> 2 //Can be done with BS

<http://codeforces.com/contest/814/problem/D> 5 //+DP on trees (NICE — but low geom.)

10750 UVA 3 //Closest points — try all pairs

<http://codeforces.com/contest/820/problem/B> 3 //Polygon angle find!

13213 UVA 5 //VERY NICE — Voronoi diagram (low constraints so not actually needed)

13215 UVA 3 //EASY — Sum areas and find side lengths

<http://www.spoj.com/problems/IITKWPCC/> (5) //VERY VERY NICE — Nqrt(N)log(N)

<http://www.spoj.com/problems/NNS/> (5) Closest points query [fake geometry] {\_\_128}

<http://codeforces.com/contest/849/problem/B> (3) //X-Product — side

<http://www.spoj.com/problems/AMR12C/> (5) //Point closest to all other points (with speed)

graph

<http://codeforces.com/contest/27/problem/D> (5)

11387 (UVA) 4

<http://www.spoj.com/problems/VFRIEND2/> (5) //Graph possible check

greedy

<http://codeforces.com/contest/729/problem/D> 3

<http://codeforces.com/contest/729/problem/E> 4

<http://codeforces.com/contest/725/problem/D> 4

<http://codeforces.com/contest/725/problem/F> 9

<http://codeforces.com/contest/732/problem/E> 5

<http://codeforces.com/contest/727/problem/F> 6

<http://codeforces.com/contest/724/problem/D> 5

<http://codeforces.com/contest/723/problem/C> 4

<http://codeforces.com/contest/719/problem/B> 2

<http://codeforces.com/contest/712/problem/C> 3

13152 UVA (4)

<http://codeforces.com/contest/746/problem/E> 5

<http://codeforces.com/contest/746/problem/D> 3

<http://codeforces.com/contest/749/problem/C> 3

11737 UVA (3)

11786 UVA (4)

11630 UVA (5)

11563 UVA (4)

11491 UVA (4)

11330 UVA (3)

11089 UVA (2)

<http://www.spoj.com/problems/SQRMINSUM/> 3 //solve-able in O(N+M)-arrayqueue

<http://www.spoj.com/problems/MSCHED/> 3 //sweep from back

<http://www.spoj.com/status/ns=18780683> 4 //all perm + A<B<C works

<http://www.spoj.com/problems/NINJA7/> (3) //sort by diff

<http://www.spoj.com/problems/NINJA2/> (4) //try all possib. (26)

<http://codeforces.com/contest/767/problem/E> (6)

<http://codeforces.com/contest/637/problem/B> (3) //NICE pro prvaky

<http://codeforces.com/contest/777/problem/B> (3) // -||-

<http://codeforces.com/contest/777/problem/D> (3) //just go from end

<http://codeforces.com/contest/779/problem/C> (3) //NICE pro prváky

<http://www.spoj.com/problems/SPCU/> (2) //Easy — zamysleni (max int = index)

<http://www.spoj.com/problems/LOPOV/> (4) //sort + queue (or just queue) NICE

<http://codeforces.com/contest/792/problem/E> (5) //T%S<=T/S + check proper

<http://codeforces.com/contest/807/problem/E> (5) //NICE — put asice P2 / rest — greedy from small

<http://codeforces.com/contest/799/problem/E> (5) //Many queues — but NICE

<http://codeforces.com/contest/808/problem/C> (3) //EASY

<http://codeforces.com/contest/802/problem/B> (4) //Priority by "next"

10850 UVA (4) //Queue a brute-force

<http://codeforces.com/contest/813/problem/A> (1) //Zahrivacka pro prvaky

10716 UVA (4) //NICE — always find closest pair

<http://codeforces.com/contest/816/problem/C> (3) //NICE — greater<lesser side

<http://codeforces.com/contest/820/problem/D> (5) //VERY NICE — O(N) -~- 5 events per number

<http://codeforces.com/contest/818/problem/B> (2) //Zahrivacka pro prvaky

<http://codeforces.com/contest/822/problem/C> (4) //Almost classical Sort+Queue

<http://codeforces.com/contest/825/problem/C> (2) //Nice & Easy

<http://codeforces.com/contest/825/problem/D> (3) //Update by modulo

<http://codeforces.com/contest/835/problem/B> (2) // Zahhrivacka pro prvaky

<http://codeforces.com/contest/839/problem/B> (3) //Nasty iffs — yet nice excersize

<http://www.spoj.com/problems/PCPC12I/> (4) //Swipe MINIMUM from left/right [10^6-A[i] trick]

<http://www.spoj.com/problems/AMR12I/> (3) //NICE a) MAX\_SEG>=K b) (SEG\_SIZE-1)/K+1

<http://www.spoj.com/problems/BUSYMAN/> (2) //NICE&EASY — Sort + keep minimum

hash

12012 UVA 4

<http://codeforces.com/contest/727/problem/E> 7

<http://codeforces.com/contest/718/problem/D> 8

11855 UVA 4

<http://codeforces.com/contest/752/problem/D> 5

<http://codeforces.com/contest/825/problem/F> 5 //String + Periods

<http://codeforces.com/contest/835/problem/D> 4 //Palindromes

hull

chess

11852 UVA (6)

<http://www.spoj.com/problems/KLUG1/> (2) //Jumps of horse

implementation

<http://codeforces.com/contest/719/problem/C> 3

<http://codeforces.com/contest/747/problem/E> 4

<http://codeforces.com/contest/754/problem/C> 5

11482 UVA (4)

11291 UVA (3)

11070 UVA (5) //evaluation of expression

11074 UVA (2)

<http://codeforces.com/contest/678/problem/B> 2 //calendar days

<http://codeforces.com/contest/643/problem/A> 3 //easy — just simulate

<http://codeforces.com/contest/770/problem/D> 5 //easy — but pain — draw

<http://codeforces.com/contest/789/problem/B> 3 //simulate (mby twice)

13171 UVA (1)

10800 UVA (3)

<http://codeforces.com/contest/828/problem/B> 2 //EASY & NICE — just analysis

<http://codeforces.com/contest/825/problem/B> 2 //EASY & NICE — Piskvorky — pro prvaky

<http://codeforces.com/contest/837/problem/B> 2 //Just implementation

<http://codeforces.com/contest/837/problem/C> 2 //Some nasty iffs

<http://codeforces.com/contest/845/problem/B> 2 //Easy pro prvaky

<http://codeforces.com/contest/845/problem/D> 3 //Iffs — folow the rules

<http://www.spoj.com/problems/UNIHW/> 4 //NICE (but many iffs)

inclusion-exclusion

<http://www.spoj.com/problems/KPRIMESB/> (4)

<http://www.spoj.com/problems/IITKWPCH/> (4) //NICE — on bits

<http://www.spoj.com/problems/SUBSET/> (5) //VERY NICE — 3^10 (^2 but not exactly) (+ sorting)

interactive

<http://codeforces.com/contest/727/problem/C> (2)

<http://codeforces.com/contest/810/problem/D> (4) //BS \* 3 (same)

<http://codeforces.com/contest/811/problem/D> (4) //BFS — easy .. some ifs

<http://codeforces.com/contest/835/problem/E> (4) //NICE! Bitsets + Detect + XOR

<http://codeforces.com/contest/844/problem/D> (5) //NICE! Randomized algo

josephus

11351 UVA (2)

<http://www.spoj.com/problems/CLSLDR/> (4) //Muchas queries — go DP

KMP

<http://www.spoj.com/problems/NAJPF/> (4) //classical kmp — all patterns

<http://codeforces.com/contest/808/problem/G> (6) //with DP -harder

lca

<http://codeforces.com/contest/733/problem/F> 7

11354 UVA (4)

<http://www.spoj.com/problems/POLICEMEN/> (3) //simple + small graph

<http://www.spoj.com/problems/QTREE2/> (5) //very easy if bin. understrood

<http://codeforces.com/contest/828/problem/F> 7 // Differently MST / Outside

<http://codeforces.com/contest/832/problem/D> (5) //Classical + Depth /OR/ HLD +ST

<http://www.spoj.com/problems/DRTREE/> (5) //NICE [finding ancestor + depths]

<http://codeforces.com/problemset/problem/838/B> (6) //VERY NICE [HLD + ET + ST]

<http://www.spoj.com/problems/NTICKETS/> (4) //Maximum on path

<http://www.spoj.com/problems/GRASSPLA/> (5) //HLD

lcs\_subsequence

10949 UVA (6) — Hunt or Bit

<http://www.spoj.com/problems/MC/> (3) //Classical

<http://www.spoj.com/status/ns=20097091> (4) //Permutation

<http://www.spoj.com/problems/LCS0/> (7) //Bit

lis

<http://www.spoj.com/problems/ALTSEQ/> 3 //solvable by FW in Nlog(N)

<http://www.spoj.com/problems/VISIBLEBOX/> (4) //with multiset

<http://www.spoj.com/problems/DOSA/> (5)

<http://www.spoj.com/problems/CODERE3/> 3 //Low bounds LIS/LDS

<http://www.spoj.com/problems/BRDGHRD/> 4 //lis (nondecreasing)

matching

<http://codeforces.com/contest/727/problem/D> 4

11985 UVA (5)

<http://www.spoj.com/problems/AMR12A/> (5) //VERY NICE [goophers + bonus](http://codeforces.com/blog/entry/BS)

<http://www.spoj.com/problems/NITT4/> (4) //VERY NICE [Chessboard matching]

matrix

12045 UVA (4)

matrix\_exponentiation

11551 UVA (4)

11486 UVA (5)

10743 UVA (5) //A001169 [easy multi / hard to come with recurence]

<http://codeforces.com/contest/821/problem/E> (6) //Not trivial to come-up with matrix

<http://www.spoj.com/problems/DCEPCA06/> (4) //NICE — 10x10 matrix

<http://www.spoj.com/problems/GSWORDS/> (3) //NICE&EASY — 3-states "OO,OX,XO"

mcmf

11613 UVA (6)

<http://codeforces.com/contest/802/problem/C> (8) //Nice but hard to see + negative

<http://codeforces.com/contest/802/problem/N> (5) //Easy but faster MCMF needed

<http://codeforces.com/contest/818/problem/G> (6) //NICE + MUCH Faster MCMF

<http://www.spoj.com/problems/BNMT/> (7) //VERY NICE (some optimalisations needed + weird data set)

median

<http://codeforces.com/contest/713/problem/C> 7

<http://www.spoj.com/problems/RMID2/> 4

<http://www.spoj.com/problems/RMID/> 3 (as above just not so dynamic)

<http://www.spoj.com/problems/EC_ESTA/> 4 //classical dynamic median

<http://www.spoj.com/problems/DCEPCA09/> (6) //VERY NICE [MO +++ MEDIAN, MEAN, FREQ]

meet\_in\_middle

11851 UVA (5)

11465 UVA (5)

13207 UVA (4) //Straight-forward MIM

<http://www.spoj.com/problems/COLOR_CC/> (4) //VERY NICE — div by partity (take lesser) → 8^6

MO

<http://www.spoj.com/problems/COT/> (7) //ON TREE [but very tight TLE]

<http://www.spoj.com/problems/GOT/> (5) //ON TREE

<http://www.spoj.com/problems/CPAIR2/> (4) //MO + Fenwick [VERY NICE]

number\_rectangle

12192 UVA 5

<http://codeforces.com/contest/729/problem/B> 2

<http://codeforces.com/contest/710/problem/C> 4

11871 UVA 6

11617 UVA (3)

11573 UVA (4)

11499 UVA (5)

11230 UVA (4)

number\_theory

<http://codeforces.com/contest/731/problem/F> 4

12031 UVA (8)

<http://codeforces.com/contest/722/problem/F> (8)

<http://codeforces.com/contest/716/problem/C> 4

<http://codeforces.com/contest/711/problem/E> (8)

<http://codeforces.com/contest/710/problem/D> (6)

13154 (UVA) 7

13166 (UVA) 5

11962 (UVA) 2

11718 UVA 3

11510 UVA (5)

11538 UVA (3) //good one — just math

11556 UVA (1)

<http://codeforces.com/contest/757/problem/E> (8)

<http://codeforces.com/contest/758/problem/F> (7)

11481 UVA (4)

11237 UVA (4) //Nice — seems like knapsbag but it it not

11155 UVA (4) //Almost as previous problem

11038 UVA (3) //counting digits on interval

<http://codeforces.com/contest/763/problem/C> (7)

11087 UVA (4) //Sums of two numbers divisible with <=500 (10^5)

<http://codeforces.com/contest/678/problem/C> 2 //LCM

<http://codeforces.com/problemset/problem/665/F> (8) //p^3 | p\*q

<http://www.spoj.com/problems/LCMSUM/> //Vzorec v knihovničce

<http://www.spoj.com/problems/FRNDZND/> (2) // (size 1 == 1, else 0)

<http://www.spoj.com/problems/EXPOR/> //bit-by-bit (+ formula)

<http://www.spoj.com/problems/FACTDIV/> (5) //dyn-update of ans/factors GOOD

<http://www.spoj.com/problems/PAIRDIV/> (6) //cyka möbius -\_-

<http://www.spoj.com/problems/FCDC/> (4) //keep factorized factorial

<http://www.spoj.com/problems/NTHPRIME/> (7) // BS + NumPrime GOOD!!

<http://www.spoj.com/problems/DIVFACT3/> (7) // Sieve 10^8 + sqrt search

<http://www.spoj.com/problems/DIVFACT4/> (8) // Prime count

<http://codeforces.com/contest/776/problem/C> (4) //segments div. by number

<http://codeforces.com/contest/776/problem/E> (6) //vypsat cisla: f(N)=Phi(N),g(N)=N

<http://www.spoj.com/problems/PHT/> (2) //easy BS (NN+2N) mby math?

<http://www.spoj.com/problems/GCDEX/> (7) //OEIS A006579 — enough [well imp]

<http://www.spoj.com/problems/APS/> (3) //just sieve + generate

<http://www.spoj.com/problems/WPC5I/> (3)//fc: C[a]!=C[b]:F[a]^max(C[a],C[b])

<http://www.spoj.com/problems/SPEC_SET/> N→N/k→N/k/k

<http://www.spoj.com/problems/DCEPC11B/> (5) //Wilson't theorem!

<http://www.spoj.com/problems/FACTMULN/> (5) //each f[i]/c[i] separately

<http://www.spoj.com/problems/SPCM/> (4) //just factorisation + prime check (10^12)

<http://www.spoj.com/problems/TWOGAME/> (5) //gcd == Power of 3 => YES

<http://www.spoj.com/problems/MKEQUAL/> (2) //Chceck if sum is divisible by N

<http://www.spoj.com/problems/TIPTOP/> (3) //sqrt(N)==N? NICE!!

<http://www.spoj.com/problems/PSYCHON/> (4) // fast factorisation

<http://www.spoj.com/problems/ENIGMATH/> (1) // swap and div by gcd

<http://www.spoj.com/problems/SNGPG/> (3) // prime gen + check

<http://codeforces.com/contest/795/problem/A> (2) //brute-force

<http://codeforces.com/contest/801/problem/E> (6) //NICE! — Clique-DAG + inversion

<http://codeforces.com/contest/798/problem/C> (4) //GCD == at the beginning OR 2

<http://codeforces.com/contest/803/problem/C> (3) //Only "low" K and just divisors

10830 (4) //simple add 2→ sqrt(N) + their mirrors

<http://codeforces.com/contest/817/problem/A> (2) //check division + parity

13209 UVA (3) //Simple simulation of division (+states rememberance)

<http://codeforces.com/contest/834/problem/C> (4) //Has cube-root + both num divisible by cuberoot

<http://codeforces.com/contest/837/problem/E> (5) //Factorisation + GCD attributes

<http://www.spoj.com/problems/SUMMATION/> (3) //Number contribution: 2^(N-1)

<http://www.spoj.com/problems/SECTORS/> (4) //Odd len OR sum of odd indices == sum of even

<http://www.spoj.com/problems/IITKWPCM/> (6) //VERY NICE — Gauss's generalisation of Wilsons Theorem

<http://www.spoj.com/problems/UCV2013A/> (4) //N\*(N^L-1)/(N-1)

<http://www.spoj.com/problems/KIMO1/> (4) //NICE — Adding/Subing by modulus

<http://www.spoj.com/problems/AFS2/> (4) //Sum of divisort (sqrt(N)) — (+128int)

<http://www.spoj.com/problems/FUNNUMS/> (4) //Permutations (get ith + guess ith)

<http://www.spoj.com/problems/MAY99_3/> (3) //GCD

<http://www.spoj.com/problems/PUCMM334/> (3) //Classical hats problem

<http://www.spoj.com/problems/LCPC11B/> (4) //Factorize + count all subsets

<http://www.spoj.com/problems/THREENUMBERS/> (2) //EASY & NICE [lcm]

<http://www.spoj.com/problems/GAMES/> (2) //EASY&NICE — Go discrete (by 10^k) → 10^k/GCD

<http://www.spoj.com/problems/SUBSHARD/> (4) //dig\*10^sufix\*(choose sufix)\*^Prefix [VERY NICE]

<http://www.spoj.com/problems/INVDIV/> (6) //Sum of divisors function + factorisation [NICE]

<http://www.spoj.com/problems/JNEXT/> (2) //EASY — Zahřívačka pro prváky

oeis

12004 UVA 2

11273 UVA 5 //https://oeis.org/A001088

11077 UVA 3 //https://oeis.org/A094638

<http://www.spoj.com/problems/VECTAR5/> 3 //https://oeis.org/A038721

<http://www.spoj.com/problems/ESYRCRTN/> 2 //generate and see

<http://www.spoj.com/problems/IITWPC4B/> 3 //http://oeis.org/A005044

<http://www.spoj.com/problems/POLCONST/> (4) //A003401+Fermat Number (Prime)

<http://www.spoj.com/problems/CUTCAKE/> (3) // pattern [1 22 333 4444 55555]

10872 UVA 3 //Alcuin's Sequence

<http://www.spoj.com/problems/LOVINGPW/> (3) //A000788

<http://www.spoj.com/problems/CBANK/> (3) //A000292 Tetrahedral numbers

<http://www.spoj.com/problems/GUMATH2/> (4) //A000240 Modulo by MOD\*2

<http://www.spoj.com/problems/MATHII/> (4) //A006218 (Two formulas => sqrt(N))

offline

11266 UVA (6) //slightly knapsack || moc hezka

<http://codeforces.com/contest/761/problem/F> (7)

<http://www.spoj.com/problems/UPDATEIT/> (2) //basic method

13189 UVA (4) //simulation + sort queries

palindromes

<http://www.spoj.com/problems/MSUBSTR/> (4) //Simple manacher (or other)

PAST\_CONTESTS

1) Big Integer: <https://a2oj.com/standings?ID=29173>

2) Sieve: <https://a2oj.com/standings?ID=29311>

3) Factorisation: <https://a2oj.com/standings?ID=29497>

4) Power: <https://a2oj.com/standings?ID=29722>

5) Inversion: <http://pastebin.com/Fk1PBMQ2>

6) Matrix Exponentiation: <https://a2oj.com/contest?ID=29975>

7) Primality Testing: <https://a2oj.com/contest?ID=30152>

8) XOR TRIE: <http://pastebin.com/w9Xfwf0h>

9) Point in Polygon: <https://a2oj.com/contest?ID=30414>

10)Bridges & Articulations: <https://a2oj.com/contest?ID=31087>

11)Dijkstra: <https://a2oj.com/contest?ID=31537>

12)Belman-Ford: <https://a2oj.com/contest?ID=31786>

13)FW: <https://a2oj.com/contest?ID=31812>

14)Kruskal: <https://a2oj.com/contest?ID=32579>

15)LCA: <https://a2oj.com/contest?ID=32584>

16)Fenwick: <https://a2oj.com/edit?ID=32669>

17)DFS: <https://a2oj.com/contest?ID=32968>

18)Segment Tree: <https://a2oj.com/edit?ID=33052>

patter-matching

11019 UVA (5)

permutations

<http://codeforces.com/contest/844/problem/C> 3 //NICE Permutations in array

persistent\_segment\_tree

<http://codeforces.com/contest/813/problem/E> (6) //Easy but hard data structure

phi

<http://www.spoj.com/problems/ETF/> 3

<http://www.spoj.com/problems/TIP1/> (4) //Phi + perms

<http://www.spoj.com/problems/DCEPCA03/> (3) //Phi + Reduce cycles: P\*PrevixP\*2-P^2

pollard-rho

<http://www.spoj.com/problems/PUCMMT02/> (7) //wrong bounds- but ll OK

power

11029 UVA (3)

preprocess

<http://codeforces.com/contest/777/problem/C> (4) //NICE

<http://codeforces.com/contest/818/problem/C> (4) //Prefix Sum

<http://codeforces.com/contest/834/problem/B> (2) //26 queries — NICE rozehrivacka pro prvaky

<http://www.spoj.com/problems/RANGESUM/> (4) //NICE: Offline (delta) + Prefix Sum

<http://www.spoj.com/problems/RANDG/> (3) //NICE [but too low bounds] [PrefixSum] [Try all indexes]

<http://www.spoj.com/problems/HARSHAD/> (3) //Sieve + simple function

<http://www.spoj.com/problems/PUCMM210/> (3) //Number theory (thinking not necessary)

prime-testing

<http://www.spoj.com/problems/ABA12A/> (3)

10871 UVA (3) //Easy — fermat not necessary

<http://www.spoj.com/problems/POP1/> (4) //Fast primality testing (or somehow)

<http://www.spoj.com/problems/POP2/> (5) //NICE — same as above (yet with ll)

<http://www.spoj.com/problems/POP3/> (6) //same as above (yet with big)

priority\_queue

13190 UVA (2) //just priority queue

probability

11762 UVA (5)

11427 UVA (5)

11348 UVA (2)

<http://codeforces.com/contest/768/problem/D> (4) //With DP

<http://www.spoj.com/problems/IITWPC4J/> (4) //with DP

10828 UVA (5) //Nice problem but bad statemend: Expected value of visits MC

10777 UVA (4) //NICE — yet solvable with DP

<http://codeforces.com/contest/839/problem/C> (3) //NICE & Easy => Tree

<http://www.spoj.com/problems/ZCR/> (3) //Easy (+DP)

<http://www.spoj.com/problems/IITKWPCN/> (2) //Easy — Odd/Eve (black balls)

<http://codeforces.com/contest/846/problem/F> (5) // Expected number of unique elements

recursion

<http://www.spoj.com/problems/GOC11A/> 4

13170 UVA (7) //heavy implementation — but NICE!

10854 UVA (3) //if/else

RMQ

<http://codeforces.com/contest/713/problem/D> 6

<http://codeforces.com/contest/675/problem/E> 5

<http://www.spoj.com/problems/POSTERIN/> 5 //VERY NICE — Delete all minimas

<http://www.spoj.com/problems/RPLN/> (3) //RMQ only

<http://www.spoj.com/problems/CITY2/> (4) //RMQ + MAP [NICE][VAGUE STATEMENT]

rope

<http://www.spoj.com/problems/AROPE/> 4

<http://www.spoj.com/problems/AROPE2/> 5 //same as above (+time)

scc

<http://www.spoj.com/problems/TFRIENDS/> (4) //just scc size

segment\_tree

<http://codeforces.com/contest/739/problem/C> (8)

<http://codeforces.com/contest/718/problem/C> (8)

<http://codeforces.com/contest/750/problem/E> (7)

<http://codeforces.com/contest/759/problem/C> (7)

11165 UVA (5)

<http://codeforces.com/contest/763/problem/E> (8) //VERY NICE — [non-trivial]

<http://www.spoj.com/problems/BGSHOOT/> (5) //normalize — then easy

<http://www.spoj.com/problems/KGSS/> (4)

<http://codeforces.com/contest/765/problem/F> (7) //VERY NICE — CASCADE

<http://www.spoj.com/problems/GSS1/> (5) //Idea — then easy

<http://www.spoj.com/problems/KQUERYO/> (5) //Seg-tree of vectors

<http://codeforces.com/contest/633/problem/G> (8) //EulerTree+Seg+Bitset

<http://www.spoj.com/problems/NAJ0001/> (7) //10^8 int — memory (and worked)

<http://www.spoj.com/problems/PRMQUER/> (5) //2 segment trees + sieve

<http://www.spoj.com/problems/EC_DIVS/> (5) //dunno if intended

<http://www.spoj.com/problems/DCEPC11I/> (5) //NICE — 1,2,3,4,5,.. inc

<http://www.spoj.com/problems/QUE2/> (4) //kth number

<http://codeforces.com/contest/785/problem/E> (6) //Seg+Treap [and faster]

<http://codeforces.com/contest/786/problem/B> (6) //+Dijkstra

13183 UVA (6) //Merge-Sort-Tree [dunno]

<http://codeforces.com/contest/803/problem/G> (5) //VERY NICE!! — ST 10^9 + ST/RMQ 10^5

<http://codeforces.com/contest/794/problem/F> (7) //Digit by digit! (N\*log(N)\*100 )

<http://codeforces.com/contest/811/problem/E> (6) //VERY NICE — DSU (easier Timofey + animals)

<http://codeforces.com/contest/817/problem/F> (7) //10^18 + MEX ~~ NICE yet problematic

<http://codeforces.com/contest/816/problem/B> (3) //Or offline trick makes it easier

<http://codeforces.com/contest/834/problem/D> (5) //+Dynamic Programming | NICE

<http://www.spoj.com/problems/SBO/> (5) //preLast→ last (-1), last→ now (+1) — VERY NICE

<http://www.spoj.com/problems/GOODE/> (5) //NICE: Inversion + L-Mex

<http://www.spoj.com/problems/CNTPRIME/> (3) //ST+Sieve (short range)

<http://www.spoj.com/problems/SEGSQRSS/> (4) //NICE {weak data} ~~ SQRT works too

sequences

11885 UVA 7 //Previous problem requested for statement

11522 UVA 3 //Trick — low numbers only :P

sieve

11610 UVA (5)

11353 UVA (3)

<http://www.spoj.com/problems/TDPRIMES/> (4)

<http://www.spoj.com/problems/VECTAR8/> (3)

<http://www.spoj.com/problems/NFACTOR/> (4)

<http://www.spoj.com/problems/HS08PAUL/> (4) //simply generate

<http://codeforces.com/contest/776/problem/B> (3) //Easy — trict: PM-1/ELSE-2

<http://www.spoj.com/problems/GGD/> (4) // N/lowestDiv\*(lowestDiv-1)

<http://codeforces.com/contest/822/problem/D> (4) //DP + Lowest factor

<http://www.spoj.com/problems/NGIRL/> (4) //Squares — Primes + BS == Easiest

<http://www.spoj.com/problems/PTRI/> (5) //Very fast sieve necessary:/

<http://www.spoj.com/problems/AFS/> (3) //Sum of divisort + DP

<http://www.spoj.com/problems/BSPRIME/> (4) //Very fast sieve needed

simulation

12187 UVA (2)

<http://codeforces.com/contest/724/problem/C> 5

<http://codeforces.com/contest/746/problem/C> 3

11093 UVA (2)

<http://codeforces.com/contest/768/problem/C> (4)

<http://www.spoj.com/problems/WRONG/> (5) //VERY NICE — precalculate from back, then go from front

sorting

12189 UVA (3)

12196 UVA (4)

<http://codeforces.com/contest/731/problem/D> 7

11925 UVA (4)

11979 UVA (3)

<http://codeforces.com/contest/747/problem/D> (4)

11890 UVA (4)

<http://www.spoj.com/problems/KAOS/> (4) //pocet inverzí — GOOD problem!!!!

<http://www.spoj.com/problems/KSMALL/> (5) //fast sort /or/ quick-select

<http://www.spoj.com/problems/RKS/> (3) //use map

<http://www.spoj.com/problems/SPCJ/> (4) //reverse + go from back

<http://codeforces.com/contest/785/problem/B> (2) //last-first + vice versa

<http://codeforces.com/contest/798/problem/D> (4) //Take 1st then take best B of every pair (sort by A)

<http://codeforces.com/contest/810/problem/B> (2)

<http://codeforces.com/contest/810/problem/C> (3) //+Math

<http://codeforces.com/contest/814/problem/A> (1) //Pro prváky — but nice observation

<http://codeforces.com/contest/817/problem/B> (3) //Frequency of TOP 3

10769 UVA (3) //Sadly N^4 passes

13208 UVA (4) //Sort + Prefix Sum

13212 UVA (3) //Number of inversions

<http://codeforces.com/contest/831/problem/C> (3) //NICE ~ Check all "add" against first

<http://codeforces.com/contest/831/problem/D> (4) //Can be solved with BS+Max-Match

<http://codeforces.com/contest/841/problem/C> (3) //NICE — match greatest to lowest

<http://codeforces.com/contest/845/problem/C> (2) //EASY — pro prvaky

<http://www.spoj.com/problems/HSHW/> (4) //Test every big/low pair + big/big low/low on +/-

spanning\_tree

BLINNET SPOJ (3)

11183 UVA (4) //Directed [need to know algo!]

<http://www.spoj.com/problems/ULM09/> (3) //Sum-Kruskal

<http://www.spoj.com/problems/IITKWPCG/> (4) //VERY NICE [log instead of price]

spfa

11478 UVA (5)

sqrt

12003 UVA 7

11990 UVA (5)

<http://www.spoj.com/problems/GIVEAWAY/> (7) //SQRT + BS > [or Seg+Trie]

<http://codeforces.com/contest/786/problem/C> (5) //Nsqrn (bg) + sqrSegs (end)

<http://codeforces.com/contest/840/problem/D> (5) //NICE — Either frequent OR brute-force

stl

<http://codeforces.com/contest/799/problem/B> (2) //EASY — MAP

<http://codeforces.com/contest/808/problem/D> (3) //MAP

10887 (2) //string + map

10730 (3) //Easy with hash-map

<http://codeforces.com/contest/821/problem/C> (3) //STACK (vector) Nice+Easy

<http://www.spoj.com/problems/SOLVEIT/> (3) //Set + lower\_bound

<http://www.spoj.com/problems/IITKWPCA/> (2) //Set + getline

<http://codeforces.com/contest/849/problem/D> (5) //Queue

<http://www.spoj.com/problems/CRAN02/> (4) //Map (+Math)

<http://www.spoj.com/problems/MAX_NUM/> (4) //Queue (possibly multiple ways)

<http://www.spoj.com/problems/SID/> (5) //Sort + Vector (or similar) [strict TLE]

strings

<http://codeforces.com/contest/762/problem/C> 5

<http://www.spoj.com/problems/LCS0/> 10 //LCS

<http://www.spoj.com/problems/IITWPC4H/> 2 //Frequence array

<http://www.spoj.com/problems/ANAGR/> 2 //frequency + palindromes

13186 UVA (6) //Bitset + Trie ~ NICE [6-7 mby?]

<http://codeforces.com/contest/798/problem/B> (2) //Brute-force .. pro prváky

10745 UVA (4) //Frequency (N^2 possible if efficient!!)

<http://codeforces.com/contest/822/problem/B> (2) //Easy pro prvaky (slightly imple.)

<http://codeforces.com/contest/828/problem/C> (4) //+Sorting (process only necessary!)

<http://codeforces.com/contest/832/problem/B> (3) //Naive compare back+front [+freq]

<http://www.spoj.com/problems/STC04/> (5) //Next + pairs O(N\*26) [frist look O(26^2\*N)]

<http://www.spoj.com/problems/IITKWPCJ/> (4) //GCD or HASHING

<http://www.spoj.com/problems/SUBSN/> (4) //Next (NICE — bad input):

<http://www.spoj.com/problems/AMR12D/> (1) //Palindrome check //Zahrivacka pro prvaky

<http://www.spoj.com/problems/BOGGLE/> (2) //EASY [MAP][STREAM]

suffix\_array

12191 UVA 5

SARRAY SPOJ 3

4513 LA 6

<http://www.spoj.com/problems/LCS2/> 7 // must be linear (SA+LCP+MQ)

<http://codeforces.com/contest/802/problem/I> 7 //NICE! SA+LCP+(Segmentree/queue)

sweep

12048 UVA (5)

ternary\_search

12197 UVA (4)

TMP

topo

<http://codeforces.com/contest/765/problem/E> 5

<http://codeforces.com/contest/770/problem/C> 4 //reduce + toposort

<http://codeforces.com/contest/825/problem/E> 4 //Toposort from biggest/backward

treap

<http://codeforces.com/contest/762/problem/E> 6

<http://www.spoj.com/problems/COUNT1IT/> 5

<http://www.spoj.com/problems/IITWPC4D/> 4 //From end — pick i-th + del i-th

<http://www.spoj.com/problems/ALLIN1/> 4 //Typical treap operations

tree

<http://codeforces.com/contest/746/problem/G> 5

<http://codeforces.com/contest/750/problem/F> 9

<http://www.spoj.com/problems/RTREE/> 3 //longest path tree — query

13175 UVA (2) //something like preorder build

<http://codeforces.com/contest/796/problem/C> (3) //Just counting — inc by at most 2

<http://codeforces.com/contest/797/problem/D> (4) //VERY NICE — sort + D&C all

<http://codeforces.com/contest/805/problem/E> (4) //NICE — Treewidth coloring (greedy)

<http://codeforces.com/contest/828/problem/D> (3) //Star construction

<http://www.spoj.com/problems/TREEDEGREE/> (3) //Degree from euler tree

<http://www.spoj.com/problems/UCV2013J/> (3) //Find what is leaf in Binary Tree

trie\_bit

<http://codeforces.com/contest/714/problem/C> 5

<http://www.spoj.com/problems/SUBXOR/> (4)

<http://codeforces.com/contest/817/problem/E> (5) //Classical — remember sum! NICE!

trie\_string

11732 UVA (5)

11539 UVA (5)

11488 UVA (4)

<http://www.spoj.com/problems/TRYCOMP/> (4)

10860 UVA (4) //DP + Trie [nice — slightly generic]

TSP

10937 UVA (4) //find '!' / BFS / TSP — NICE!

10944 UVA (4)

10818 UVA (5) //Easy — but not-easy implementation: ++Dijkstra [LEX!]

<http://www.spoj.com/problems/A_W_S_N/> (4) //BFS + TSP (path) — NICE

two-pointers

<http://codeforces.com/contest/746/problem/F> 6

11436 UVA (5)

<http://codeforces.com/contest/760/problem/D> 4

11386 UVA (4)

<http://www.spoj.com/problems/WOWSUBSTR2/> 3

<http://www.spoj.com/problems/ARRAYSUB/> 4

<http://www.spoj.com/problems/CODFURY/> 3 //easy — ukazkove

<http://codeforces.com/contest/769/problem/B> 3 //sort + TP

<http://codeforces.com/contest/814/problem/C> 4 //NICE — maybe some DP +/-

<http://www.spoj.com/problems/CRAN04/> 4 //NICE — (more or less) 3 pointers

<http://www.spoj.com/problems/OPCPIZZA/> 3 //NICE [EASY] [AGAINS EACH OTHER]

z-function

<http://www.spoj.com/problems/SUFEQPRE/> 4

100000

12174 UVA (4)

<http://www.spoj.com/problems/LCPC12F/> (2) //EASY — Little number theory

2SAT

11930 UVA (4)

<http://codeforces.com/contest/776/problem/D> (5)

MINE problems [SPOJ] — I cn also mke some outline in case of interest:

long long compute\_hash(string const& s, bool upper) {

const int p = 31;

if (upper) p = 53;

const int m = 1e9 + 9;

long long hash\_value = 0;

long long p\_pow = 1;

for (char c : s) {

hash\_value = (hash\_value + (c - 'a' + 1) \* p\_pow) % m;

p\_pow = (p\_pow \* p) % m;

}

return hash\_value;

}

http://codeforces.com/problemset/problem/158/A

http://codeforces.com/problemset/problem/231/A

http://codeforces.com/problemset/problem/282/A

http://codeforces.com/problemset/problem/281/A

<http://codeforces.com/problemset/problem/977/A>

ბიტური ოპერაციები

http://codeforces.com/problemset/problem/399/B

#include <stdio.h>

int n;

char s[55];

long long res;

int main() {

scanf("%d%s", &n, s);

for(int i = 0; i < n; i++) if(s[i] == 'B') res |= 1ll << i;

printf("%I64d\n", res);

return 0;

}

<http://codeforces.com/gym/224476>

The second semester starts next week! As usual, the live lectures will be on [Twitch](https://twitch.tv/pmavrin) on Fridays at 18:00 MSK, and the recorded videos will be on [Youtube](https://www.youtube.com/pavelmavrin).

Topics of the second semester:

* Segment Trees and similar data structures
  + Segment Tree
  + Fenwick Tree
  + Sparse Table
  + 2D Trees
* Binary Search Trees
  + AVL Tree
  + Treap
  + Splay Tree
* Data Structures for Trees
  + Binary Lifting
  + LCA and LA problems
  + Heavy-Light Decomposition
  + Link-Cut Tree
  + Centroid Decomposition
* Plus something more :)

**1. Segment Tree:**  
  
To Read :  
<http://www.topcoder.com/tc?d1=tutorials&d2=lowestCommonAncestor&module=Static>  
<http://ronzii.wordpress.com/2011/07/08/segment-tree-tutorial/>  
<http://se7so.blogspot.in/2012/12/segment-trees-and-lazy-propagation.html>  
<http://olympiad.cs.uct.ac.za/presentations/camp3_2007/interval_trees.pdf>  
<http://codeforces.com/blog/entry/6281>  
<http://apps.topcoder.com/forums/?module=Thread&threadID=651820&start=0&mc=2#1146133>  
<http://www.algorithmist.com/index.php/Segmented_Trees>  
<http://letuskode.blogspot.in/2013/01/segtrees.html>  
<http://wcipeg.com/wiki/Heavy-light_decomposition>  
<http://discuss.codechef.com/questions/5960/rnestescape-from-the-mines>  
<http://ideone.com/dPS5N> (Heavy Light implementation).  
<https://sites.google.com/site/indy256/algo/heavy_light> (Heavy Light implementation).  
  
Problems:  
  
<http://www.spoj.com/problems/GSS1>  
<http://www.spoj.com/problems/GSS2>  
<http://www.spoj.com/problems/GSS3>  
<http://www.spoj.com/problems/GSS4>  
<http://www.spoj.com/problems/GSS5>  
<http://www.spoj.com/problems/GSS6>  
<http://www.spoj.com/problems/GSS7>  
<http://www.spoj.com/problems/ANDROUND/>  
<http://www.spoj.com/problems/BRCKTS/>  
<http://www.spoj.com/problems/DQUERY/>  
<http://www.spoj.com/problems/FREQUENT/>  
<http://www.spoj.com/problems/HEAPULM/>  
<http://www.spoj.com/problems/HELPR2D2/>  
<http://www.spoj.com/problems/KGSS/>  
<http://www.spoj.com/problems/MKTHNUM/>  
<http://www.spoj.com/problems/NICEDAY/>  
<http://www.spoj.com/problems/YODANESS/>  
<http://www.spoj.pl/problems/INCSEQ/>  
<http://www.spoj.pl/problems/INCDSEQ/>  
<http://www.spoj.pl/problems/KQUERY/>  
<http://www.spoj.pl/problems/QTREE/>  
<http://www.spoj.pl/problems/QTREE2/>  
<http://www.spoj.pl/problems/QTREE3/>  
<http://www.spoj.com/problems/QTREE4/>  
<http://www.spoj.com/problems/QTREE5/>  
<http://www.spoj.pl/problems/CTRICK/>  
<http://www.spoj.pl/problems/MATSUM/>  
<http://www.spoj.pl/problems/RATING/>  
<http://www.spoj.pl/problems/RRSCHED/>  
<http://www.spoj.pl/problems/SUPPER/>  
<http://www.spoj.pl/problems/ORDERS/>  
<http://www.spoj.com/problems/MULTQ3/>  
<http://www.spoj.com/problems/RPAR/>  
<http://www.spoj.com/problems/PATULJCI/>  
<http://www.spoj.com/problems/DISUBSTR/>  
<http://www.spoj.com/problems/HORRIBLE>  
<http://www.spoj.pl/problems/IOPC1207/>  
<http://www.spoj.com/problems/SEGSQRSS/>  
<http://www.spoj.com/problems/ORDERSET/>  
<http://www.spoj.com/problems/HELPR2D2/>  
<http://www.spoj.com/problems/TEMPLEQ>  
<http://www.codechef.com/problems/QTREE>  
<http://www.codechef.com/problems/LEBOBBLE>  
<http://www.codechef.com/problems/DGCD>  
<http://www.codechef.com/problems/QUERY>  
<http://codeforces.com/problemset/problem/280/D>  
<http://codeforces.com/problemset/problem/117/E>  
<http://codeforces.com/problemset/problem/167/D>  
<http://codeforces.com/problemset/problem/266/E>  
<http://codeforces.com/problemset/problem/145/E>  
<http://codeforces.com/problemset/problem/226/E>  
<http://codeforces.com/problemset/problem/311/C>  
<http://codeforces.com/problemset/problem/276/E>  
<http://codeforces.com/problemset/problem/221/D>  
<http://codeforces.com/problemset/problem/174/C>  
<http://codeforces.com/problemset/problem/301/D>  
<http://codeforces.com/problemset/problem/61/E>  
<http://codeforces.com/problemset/problem/103/D>  
<http://codeforces.com/problemset/problem/165/D>  
<http://codeforces.com/problemset/problem/52/C>  
<http://codeforces.com/problemset/problem/85/D>  
<http://codeforces.com/problemset/problem/242/E>  
<http://codeforces.com/problemset/problem/111/B>  
<http://codeforces.com/problemset/problem/220/B>  
<http://codeforces.com/problemset/problem/195/E>  
<http://codeforces.com/problemset/problem/219/E>  
<http://codeforces.com/problemset/problem/281/D>  
<http://codeforces.com/problemset/problem/121/E>  
<http://codeforces.com/problemset/problem/86/D>  
<http://codeforces.com/problemset/problem/182/C>  
<http://codeforces.com/problemset/problem/19/D>  
<http://codeforces.com/problemset/problem/258/E>  
<http://codeforces.com/problemset/problem/190/E>  
<http://codeforces.com/problemset/problem/295/E>  
<http://codeforces.com/problemset/problem/160/E>  
<http://codeforces.com/problemset/problem/163/E>  
<http://codeforces.com/problemset/problem/192/E>  
<http://codeforces.com/problemset/problem/316/E3>  
<http://codeforces.com/problemset/problem/280/E>  
<http://codeforces.com/problemset/problem/238/D>  
  
SRM 310 -> [Floating Median](http://www.topcoder.com/stat?c=problem_statement&pm=6551&rd=9990)  
[http://acm.pku.edu.cn/JudgeOnline/problem?id=1986](http://www.topcoder.com/tc?module=LinkTracking&link=http://acm.pku.edu.cn/JudgeOnline/problem?id=1986&refer=)  
[http://acm.pku.edu.cn/JudgeOnline/problem?id=2374](http://www.topcoder.com/tc?module=LinkTracking&link=http://acm.pku.edu.cn/JudgeOnline/problem?id=2374&refer=)  
[http://acmicpc-live-archive.uva.es/nuevoportal/data/problem.php?p=2045](http://www.topcoder.com/tc?module=LinkTracking&link=http://acmicpc-live-archive.uva.es/nuevoportal/data/problem.php?p=2045&refer=)  
[http://acm.pku.edu.cn/JudgeOnline/problem?id=2763](http://www.topcoder.com/tc?module=LinkTracking&link=http://acm.pku.edu.cn/JudgeOnline/problem?id=2763&refer=)  
[http://www.spoj.pl/problems/QTREE2/](http://www.topcoder.com/tc?module=LinkTracking&link=http://www.spoj.pl/problems/QTREE2/&refer=)  
[http://acm.uva.es/p/v109/10938.html](http://www.topcoder.com/tc?module=LinkTracking&link=http://acm.uva.es/p/v109/10938.html&refer=)  
<http://acm.sgu.ru/problem.php?contest=0&problem=155>  
  
**2. BIT (also called Fenwick Tree).**  
Mostly problems of BIT can also be solved by Segment Tree. But it is shorted and faster to code. Hence it is sometimes very easy.  
  
To Read:  
  
<http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=binaryIndexedTrees>  
<http://p--np.blogspot.in/2011/04/binary-indexed-tree.html>  
<http://www.algorithmist.com/index.php/Fenwick_tree>  
<http://en.wikipedia.org/wiki/Fenwick_tree>  
<http://pavelsimo.blogspot.in/2012/09/counting-inversions-in-array-using-BIT.html>  
<http://petr-mitrichev.blogspot.in/2013/05/fenwick-tree-range-updates.html>  
  
Problems:  
<http://community.topcoder.com/stat?c=problem_statement&pm=10976>  
And try some earlier problems which are solvable by this data structure to practice more.  
Try mobile problem of IOI. 2D BIT.  
  
3. Dynamic Programming (DP):  
  
To Read:  
  
<http://en.wikipedia.org/wiki/Edit_distance>  
<http://www.codechef.com/wiki/tutorial-dynamic-programming>  
  
Problems:  
  
SGU Problems :  [269](http://acm.sgu.ru/problem.php?contest=0&problem=269), [273](http://acm.sgu.ru/problem.php?contest=0&problem=273), [304](http://acm.sgu.ru/problem.php?contest=0&problem=304), [317](http://acm.sgu.ru/problem.php?contest=0&problem=317), [356](http://acm.sgu.ru/problem.php?contest=0&problem=356), [396](http://acm.sgu.ru/problem.php?contest=0&problem=396), [445](http://acm.sgu.ru/problem.php?contest=0&problem=445), [447](http://acm.sgu.ru/problem.php?contest=0&problem=447), [458](http://acm.sgu.ru/problem.php?contest=0&problem=458), [489](http://acm.sgu.ru/problem.php?contest=0&problem=489), [494](http://acm.sgu.ru/problem.php?contest=0&problem=494)  
<http://www.spoj.com/problems/SAMER08D/>  
<http://acm.sgu.ru/problem.php?contest=0&problem=199>  
<http://www.spoj.com/problems/MDOLLS/>  
<http://www.spoj.com/problems/MSTICK/>  
<http://www.spoj.com/problems/MCARDS/>  
<http://www.spoj.com/problems/MIXTURES/>  
<http://www.spoj.com/problems/SCUBADIV/>  
<http://z-trening.com/tasks.php?show_task=5000000355>  
<http://z-trening.com/tasks.php?show_task=5000000286>  
<http://z-trening.com/tasks.php?show_task=5000000465>  
<http://z-trening.com/tasks.php?show_task=5000000310>  
<http://z-trening.com/tasks.php?show_task=5000000778>  
<http://z-trening.com/tasks.php?show_task=5000000363>  
<http://z-trening.com/tasks.php?show_task=5000001024>  
<http://www.spoj.com/problems/VOCV/>  
<http://www.spoj.com/problems/PT07F/>  
<http://www.spoj.com/problems/PT07X/>  
<http://z-trening.com/tasks.php?show_task=5000000070>  
<http://z-trening.com/tasks.php?show_task=5000000569>  
<http://z-trening.com/tasks.php?show_task=5000000441>  
<http://z-trening.com/tasks.php?show_task=5000000050>  
<http://www.spoj.com/problems/RENT/>  
<http://www.spoj.com/problems/INCSEQ/>  
<http://www.spoj.com/problems/INCDSEQ/>  
<http://z-trening.com/tasks.php?show_task=5000000624>  
<http://z-trening.com/tasks.php?show_task=5000000742>  
<http://z-trening.com/tasks.php?show_task=5000000749>  
<http://z-trening.com/tasks.php?show_task=5000001044>  
<http://www.spoj.com/problems/SEQ/>  
<http://www.spoj.com/problems/SPP/>  
<http://z-trening.com/tasks.php?show_task=5000000078>  
<http://z-trening.com/tasks.php?show_task=5000000543>  
<http://z-trening.com/tasks.php?show_task=5000000718>  
<http://z-trening.com/tasks.php?show_task=5000000237>  
<http://z-trening.com/tasks.php?show_task=5000000311>  
<http://www.spoj.com/problems/MORSE/> (dp + trie) Very Hard.  
<http://www.spoj.com/problems/MPOLY/>  
<http://www.spoj.com/problems/CVXPOLY/>  
<http://www.spoj.com/problems/MTRIAREA/>  
<http://acm.zju.edu.cn/onlinejudge/showContestProblem.do?problemId=2222>  
<http://www.lightoj.com/login_main.php?url=volume_showproblem.php?problem=1122>  
<http://www.lightoj.com/login_main.php?url=volume_showproblem.php?problem=1122>  
<http://www.lightoj.com/login_main.php?url=volume_showproblem.php?problem=1125>  
Game (IOI 2008, Practice session)  
<http://www.topcoder.com/tc?d1=tutorials&d2=dynProg&module=Static>  
  
  
  
Graph Theory:  
<http://community.topcoder.com/stat?c=problem_statement&pm=10736>  
<http://www.spoj.com/problems/TRAFFICN/>  
<http://www.spoj.com/problems/PA06ANT/>  
<http://www.spoj.com/problems/PT07Z/>  
<http://www.spoj.com/problems/EXPLOSN/>  
<http://www.spoj.com/problems/BUGLIFE/>  
<http://www.spoj.com/problems/SSORT/>  
<http://www.spoj.com/problems/ARBITRAG/>  
<http://www.spoj.com/problems/CODE/>  
<http://www.spoj.com/problems/FROGGER/>  
<http://www.spoj.com/problems/GCPC11C/>  
<http://www.spoj.com/problems/GCPC11J/>  
<http://www.spoj.com/problems/GHOSTS/>  
<http://www.spoj.com/problems/MAKETREE/>  
<http://www.spoj.com/problems/PARADOX/>  
<http://www.spoj.com/problems/QTREE/>  
<http://www.spoj.com/problems/QTREE2/>  
<http://www.spoj.com/problems/QUEEN/>  
<http://www.spoj.com/problems/ROBOTGRI/>  
<http://www.spoj.com/problems/ELEVTRBL/>  
<http://www.spoj.com/problems/TRIPINV/>  
<http://www.spoj.com/problems/CAPCITY/>  
<http://www.spoj.com/problems/KOICOST/>

Editorials: (these editorials are merely hints, You need to work things yourself for final answers)  
Hope you will like them. Leave a comment if something is unclear or needs to be modified, Also leave your opinion about the contest.

1. **Gopu and Palindromes: (**[**http://www.spoj.com/problems/SPCS/**](http://www.spoj.com/problems/SPCS/)**)**

     Consider the string s. Let it has some consecutive characters of length L, Then you can directly replace L by just 1 character. After such replacements just check if the string is palindrome or not?  
     eg. s = "ababbbaaa".  
           s = "ababa". Replace bbb with b and aaa with a.  
     You need to do this in O(n).  
  
2.**Gopu and Validity of Arrangement:** (<http://www.spoj.com/problems/SPCU/>)  
   You can easily prove that if answer exists then it is unique, Only condition you need to check that any ith indexed person does not say that ≥≥ i persons are before him. You can relate this problem with permutations.  
  
3. **Gopu and Digits Divisibility: (**<http://www.spoj.com/problems/SPCU/>)  
   You can easily do a brute force. Think about it why this works, Hint: for checking divisibility by 9, sum of digits should be divisible by 9. (Though hint is not of great use in the proof).  
  
4. **Gopu and Function: (**<http://www.spoj.com/problems/SPCM/>**)**  
    Value of n will be atleast halved in each computation of f. As Take the best cases when n = 2\*p where p is prime, Sum of divisors is 2 + p which is 2 + n / 2. In other cases, sum would be even less than n/2. For finding prime divisors of n, use simple sqrt algorithm. Final complexity would be O(n−−√2n2 logn).  
  
5. **Gopu and Create Collections Part two (**<http://www.spoj.com/problems/SPCJ/>)  
   You can use dp over the tree for solving this. You can use a greedy algo also. Sort the numbers in increasing order, go from right to left and for each number K if you find a number K/2 which is not yet taken then take it and add it to answer.  
   You can also use greedy algorithm by constructing the binary tree explicitly and then going from leaves to the root of the tree.  
  
6. **Gopu and Combinatorics on Graphs: (**<http://www.spoj.com/problems/SPCE/>)  
  This is standard problem. You should know <http://en.wikipedia.org/wiki/Cayley%27s_formula>. Then figuring out things wont be tough. I dont want to give formulla as I want you to work it out.  
  
7. **Gopu and Counting bitwise prime numbers: (**<http://www.spoj.com/problems/SPCO/>**)**  
   This problem had one slow solution:  
     use dp(i, tight): where i denotes the position in the binary representation of the number (i goes from most significant digit to lower), tight denotes wheteher the current number has overshoot the number n or n. See my slow code <http://ideone.com/numnIE>.  
    You can optimize it slightly and get it passed. Basically think in terms of combinatorics, You will get the idea.

[[](http://cfrp.azurewebsites.net/)](http://cfrp.azurewebsites.net/)

[In English](http://cfrp.azurewebsites.net/blog/entry/55274?locale=en) [По-русски](http://cfrp.azurewebsites.net/blog/entry/55274?locale=ru)

[Enter](http://cfrp.azurewebsites.net/enter?back=%2Fblog%2Fentry%2F55274%3Flocale%3Den) | [Register](http://cfrp.azurewebsites.net/register)

* [HOME](http://cfrp.azurewebsites.net/)
* [TOP](http://cfrp.azurewebsites.net/top)
* [CATALOG](http://cfrp.azurewebsites.net/catalog)
* [CONTESTS](http://cfrp.azurewebsites.net/contests)
* [GYM](http://cfrp.azurewebsites.net/gyms)
* [PROBLEMSET](http://cfrp.azurewebsites.net/problemset)
* [GROUPS](http://cfrp.azurewebsites.net/groups)
* [RATING](http://cfrp.azurewebsites.net/ratings)
* [EDU](http://cfrp.azurewebsites.net/edu/courses)
* [API](http://cfrp.azurewebsites.net/apiHelp)
* [CALENDAR](http://cfrp.azurewebsites.net/calendar)
* [HELP](http://cfrp.azurewebsites.net/help)
* [DYTECHLAB CUP 2022 [](http://cfrp.azurewebsites.net/dytechlab)](http://cfrp.azurewebsites.net/dytechlab)

Top of Form

Bottom of Form

aho

<http://www.spoj.com/problems/ADAJOBS/>

URI 2226 (5) //[NICE][NUMBERS][DP]

<http://www.spoj.com/problems/SUB_PROB/en/>

http://codeforces.com/contest/696/problem/D 8

<http://www.spoj.com/problems/AHOCUR/> 5 //Aho-Corassic + DP

<https://www.codechef.com/problems/LYRC> (5) //Sample aho-brute-force

http://codeforces.com/problemset/problem/346/B //Proposed by [**bradyawn**](http://cfrp.azurewebsites.net/profile/bradyawn)

automat

6861 [LA] //CYK

UVA 10679 //Suffix Automat

<http://www.spoj.com/problems/STRMATCH/> //Suffix Automat — trie might do too

<http://www.spoj.com/problems/NSUBSTR2/> //Suffix Automaton

belman-ford

UVA 12519

<http://www.spoj.com/problems/ARBITRAG/> (4) //Or Floyd-Warshall

bfs

<http://www.spoj.com/problems/ADACYCLE/> [FW]

UVA 13295 (6) //[NICE][EFFICIENCY][IMPLEMENTATION]

http://codeforces.com/gym/101992/problem/H (4) //[VERY NICE][MAX]

http://codeforces.com/gym/100112 [H](http://cfrp.azurewebsites.net/blog/entry/3)

6151 — Beehives (4) //[NICE] //Search for shortest cycle

<https://devskill.com/CodingProblems/ViewProblem/60>

<https://devskill.com/CodingProblems/ViewProblem/150>

11312 UVA (3)

11392 UVA (4)

http://codeforces.com/contest/653/problem/E (6)

http://codeforces.com/contest/769/problem/C 5 //FL:ODD/\*\*\*\* | bfs+greed NICE

10968 UVA (3) //EASY + NICE (bfs withot <=2 nodes)

http://codeforces.com/contest/796/problem/D (3) //NICE+EASY ... print visited in bfs (not par)

10888 UVA (4) //VERY NICE — but not main technique ... ++ DP /or/ MCMF

http://codeforces.com/contest/821/problem/D (5) //VERY NICE — Consider only points not GRID

<http://www.spoj.com/problems/DIGOKEYS/> (4) //Easy [Nice problem — weird statement]

<http://www.spoj.com/problems/SPIKES/> (3) //Easy bfs (# of 's' \* 2)

<http://www.spoj.com/problems/MULTII/> (4) //VERY NICE: BFS over numbers (K\*10+d)%N

<http://www.spoj.com/problems/ADV04F1/> (5) //VERY NICE: [imple] ~ N^4\*BigConstant

<http://www.spoj.com/problems/INVESORT/> (5) //Big limit (really usefull :P)

http://codeforces.com/contest/59/problem/E (5) //[NICE][DOUBLE-STATES][SET]

http://codeforces.com/contest/877/problem/D (4) //[NICE] Add vector to # of states

bfs-grid

4291 — Sculpture (6) //[NICE][PREPROCESS][NORMALIZE]

10977 UVA (3)

928 UVA (3)

13116 UVA (4)

314 UVA (3)

11487 UVA (4)

5622 LA (7)

11931 UVA (5)

<http://www.spoj.com/problems/KNMOVE/> 3 //simple knights

<http://www.spoj.com/problems/SERGRID/> 3 //almost classical

<http://www.spoj.com/problems/NAKANJ/> 3 //Classical chess — KNIGHT

<http://www.spoj.com/problems/PUCMM223/> (4) //NICE (but not many languages) — 2 moving [x][y]

<http://www.spoj.com/problems/SPIRALGR/> (4) //NICE (not typical) [SIEVE]

<http://www.spoj.com/problems/DCEPC706/> (4) //NICE — travelling outside

http://codeforces.com/contest/35/problem/C (3) //No obstacles [multiple starts]

big

<https://devskill.com/CodingProblems/ViewProblem/394> (2) //Multiplication (or math)

http://codeforces.com/contest/66/problem/A (2) //Big + iffs + implementation

UVA — 10183

10106 — Product [UVA]

10523 — Very Easy !!! [UVA]

787 — Maximum Sub-sequence Product [UVA]

2871 — Rhyme Schemes [LA][BELL]

UVA — 10497

<http://www.spoj.com/problems/MUL/en/>

<http://www.spoj.com/problems/ITRIX_E/>

11115 — Uncle Jack

11448 — Who said crisis? [UVA]

<http://www.spoj.com/problems/GCD2/>

10083 — Division [UVA]

11830 — Contract Revision [UVA]

1230 — MODEX [UVA]

<http://www.spoj.com/problems/NUMPLAY/>

10519 — UVA

7651 — Pascal's Hyper-Pyramids [LA]

11344 — The Huge One [UVA]

10303 — How Many Trees? [UVA]

<http://www.spoj.com/problems/FAST2/>

495 — Fibonacci Freeze [UVA]

10023 — Square root [UVA]

<http://www.spoj.com/problems/SKYLINE/>

<http://www.spoj.com/problems/NITT2/>

11879 — Multiple of 17

<http://www.spoj.com/problems/MINNUM/>

10494 — If We Were a Child Again [UVA]

10013 — Super long sums [UVA]

10925 — Krakovia [UVA]

10814 — Simplifying Fractions [UVA]

619 — Numerically Speaking [UVA]

713 — Adding Reversed Numbers [UVA]

1226 — Numerical surprises [UVA]

623 — 500! [UVA]

http://codeforces.com/problemset/problem/18/D

<http://www.spoj.com/problems/NUMTSN/>

<https://www.codechef.com/problems/FRJUMP>

10220 — I Love Big Numbers ! [UVA]

<https://www.hackerrank.com/contests/projecteuler/challenges/euler025>

<https://www.hackerrank.com/contests/projecteuler/challenges/euler020>

11645 UVA 4

Gym — 100866A [ACM ICPC 2005–2006 NEERC Moscow Subregional Contest]

CSQUARE [SPOJ]

<http://www.spoj.com/problems/PARCARD1/>

10070 — Leap Year or Not Leap Year and .. [UVA]

<http://www.spoj.com/problems/SOLDIERS/>

12333 — Revenge of Fibonacci [UVA]

<http://www.spoj.com/problems/NDIVPHI/>

<http://www.spoj.com/problems/IWGBS/> [UVA]

<http://www.spoj.com/problems/POP3/> [Prime-Test]

<http://www.spoj.com/problems/VGCD/>

<http://www.spoj.com/problems/NDIVPHI2/>

12924 — Immortal Rabbits [UVA]

Count the Trees [UVA][10007]

10198 — Counting [UVA]

11375 UVA 3

<http://www.spoj.com/problems/MINNUM/> 3 // BIG/9+!!(BIG%9)

10844 UVA 4 //Bell numbers + big (might be slightly slow!)

<http://www.spoj.com/problems/NITT2/> 2 //Divisibility by two constants

<http://www.spoj.com/problems/NUMPLAY/> (3) //With DP

<http://www.spoj.com/problems/IWGBS/> (3) //Fibonacci 10^4

<http://www.spoj.com/problems/PUCMM025/> (2) //Divisibility by 1 → 9

<http://www.spoj.com/problems/CSQUARE/> (3) //Converse + Power

http://codeforces.com/contest/17/problem/D (5) //B^(N-1)\*(B-1)%C [B/N are big]

binary\_search

<https://codeforces.com/contest/1173/problem/C> (4) //[NICE]

<https://codeforces.com/contest/1201/problem/C> (4) //[NICE][SIMPLE]

<https://codeforces.com/contest/1183/problem/C> (3) //[SIMPLE][NICE]

http://codeforces.com/gym/101628/problem/J (4) //[NICE][EASY]

http://codeforces.com/gym/101962 [F](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][FW]

http://codeforces.com/contest/920/problem/G (5) //[NICE][MATH][IE]

http://codeforces.com/contest/140/problem/C (4) //[NICE][GREEDY]

http://codeforces.com/contest/898/problem/E (4) //[NICE][SIMPLE][PREPROCESS]

http://codeforces.com/contest/888/problem/C (3) //Can be done without BS

http://codeforces.com/contest/68/problem/B (3) //[EASY][DOUBLE]

http://codeforces.com/contest/42/problem/A (2) //Or simple math

http://codeforces.com/contest/883/problem/I (4) //[NICE][SET][2Pointers]

http://codeforces.com/contest/51/problem/C (4) //[NICE][GREEDY-CHECK]

http://codeforces.com/contest/729/problem/C 3

http://codeforces.com/contest/714/problem/D 8

13150 (UVA) 4

http://codeforces.com/contest/749/problem/D 5

11692 (UVA) 4

11516 (UVA) 3

http://codeforces.com/contest/760/problem/B 3

http://codeforces.com/contest/675/problem/D 4 //dunno — solvable with treap

<http://www.spoj.com/problems/NDS/> 4 //BS over LIS

<http://www.spoj.com/problems/VECTAR4/> 3

http://codeforces.com/contest/767/problem/D 4 //NICE

http://codeforces.com/contest/627/problem/D (7) //with dp — NICE

http://codeforces.com/contest/779/problem/D (3) //NICE + EASY

<http://www.spoj.com/problems/CNTINDX/> (4) //Map+BS === OK

13177 UVA (3) //BS over answer == OK

http://codeforces.com/contest/801/problem/C (3) //BS + SUM -EASY

http://codeforces.com/contest/803/problem/D (3) //BS by answer

http://codeforces.com/contest/807/problem/C (3) //Or math

http://codeforces.com/contest/818/problem/F (4) //NICE — Live VS Clique

http://codeforces.com/contest/845/problem/E (5) //VERY NICE — min(X,Y) .. add time, repeat

<http://www.spoj.com/problems/MATHLOVE/> (2) //BS + Gaus (or otter ways)

<http://www.spoj.com/problems/SABBIRGAME/> (3) //Binary search over answer ::max(0,ANS)

http://codeforces.com/contest/846/problem/D (4) //BS+Precalculation OR 2D-RMQ

<http://www.spoj.com/problems/RPLC/> (3) //Classical

<http://www.spoj.com/problems/TRIGALGE/> (2) //On doubles — simple function given

<http://www.spoj.com/problems/ABA12E/> (4) //VERY NICE — BS on answer + 2Pointers

http://codeforces.com/contest/847/problem/E (4) //NICE: Back+Front OR Front+Back

<http://www.spoj.com/problems/MAIN8_C/> (3) //Classical — simultion over array

<http://www.spoj.com/problems/FUNFACT/> (4) //VERY NICE — Sterling Approximation

http://codeforces.com/contest/16/problem/C (3) //[or math][simple formula check]

http://codeforces.com/contest/21/problem/C (3) //[NICE][prefix-sum+lower\_bound]

http://codeforces.com/contest/24/problem/E (5) //[doubles]

http://codeforces.com/contest/875/problem/E (6) //VERY NICE [BS][Keep possible places]

bits

<https://devskill.com/CodingProblems/ViewProblem/566> (2)

<https://codeforces.com/gym/102006/problem/F> (6) //[VERY NICE][DP]

http://codeforces.com/gym/101908/problem/I (3) //[EASY][XOR]

http://codeforces.com/contest/879/problem/C (3) //[NICE] one of each operation is enough

http://codeforces.com/contest/92/problem/B (2) //Bit addition/shifting (but big number)

http://codeforces.com/contest/907/problem/C (3) //Nice but ugly statement: sets

11659 UVA (4)

11535 UVA (4)

http://codeforces.com/contest/779/problem/E (5) //NICE + Parsing

<http://www.spoj.com/problems/EC_CONB/> (1) //reverse numbers

http://codeforces.com/contest/769/problem/D (4) //freq + brute-force

<http://www.spoj.com/problems/HAP01/> (2) //builtin\_popcount

http://codeforces.com/contest/862/problem/C (3) //VERY NICE — Random works well

<http://www.spoj.com/problems/KOMPICI/> (4) //NICE — Bitmask over digits

bitset

<http://www.spoj.com/problems/ADACOINS/>

<http://www.spoj.com/problems/ADACHERY/>

<http://www.spoj.com/problems/ADAFUROW/>

<https://codeforces.com/contest/1194/problem/E> (5) //[NICE][BRUTE]

<https://www.spoj.com/problems/TTRGRAPH/> (4) //[NICE][GRAPH][CLIQUES]

<https://www.spoj.com/problems/ILD18ACP/> (5) //[NICE][GRAPH][PATH]

http://codeforces.com/contest/914/problem/F (7) //[VERY NICE][POLYMUL-LIKE][BRUTE]

http://codeforces.com/contest/117/problem/C (5) //Finding 3cycles in directed graph [NICE]

http://codeforces.com/contest/754/problem/E 6

<http://www.spoj.com/problems/UCBINTC/> 5 //polymul with bitset

http://codeforces.com/contest/33/problem/D (4) //VERY NICE [LCA works too]

http://codeforces.com/contest/918/problem/C (4) //Brute-force+Bitset (or...)

bridges

http://codeforces.com/contest/118/problem/E (4) //[NICE]//Orientation [+DFS]

315 — Network

796 — Critical Links

UVA 12363

Gym 100114J [2012-2013 ACM-ICPC, NEERC, Central Subregional Contest]

<http://www.spoj.com/problems/ONBRIDGE/> [ONLINE][HARD][NICE][D&C]

http://codeforces.com/contest/732/problem/F 7

http://codeforces.com/contest/700/problem/C 7

<http://www.spoj.com/problems/EC_P/> (3) //bridges ONLY

<http://www.spoj.com/problems/SUBMERGE/> (3) //Direct articulation

<http://www.spoj.com/problems/GRAFFDEF/> (5) //Bridge tree

http://codeforces.com/contest/1000/problem/E // Sugested by [**Warawreh**](http://cfrp.azurewebsites.net/profile/Warawreh)

brute-force

<https://www.spoj.com/problems/YOSEQ/> (3) //[NICE][OBSERVATION]

UVA 13296 (5) //[NICE][RECURSION]

http://codeforces.com/gym/100112 (7) [L] //[PRECISION]

http://codeforces.com/gym/101806/problem/X (6) //[VERY NICE][DFS][IFS][OBSERVATION]

8259 — High Score [LA](http://cfrp.azurewebsites.net/blog/entry/4) //[VERY NICE][TS works too] add only low number to minimum (NOT WORKING?)

http://codeforces.com/contest/922/problem/B (2) //Test all pairs — observe 3rd

http://codeforces.com/contest/919/problem/B (1) //Simply simulate

http://codeforces.com/contest/146/problem/B (2) //Test all bigger numbers

http://codeforces.com/contest/911/problem/B (1)

7692 — Square Deal (4) //Permutations+Swap

http://codeforces.com/contest/907/problem/A (2) //try all triples 0→ 200

http://codeforces.com/contest/124/problem/B (3) //next-permutation

http://codeforces.com/contest/910/problem/C (3) //Next-permutation

http://codeforces.com/contest/898/problem/B (2) //Try all possibilities

6160 — Countdown (5) //[NICE][DFS][EFFICIENT?]

http://codeforces.com/contest/122/problem/C (3) //Just around 2^10 lucky [RECURSION]

7899 — Mr. Panda and Strips (4) //Weak test-cases

7671 What a Beautiful Lake (2) //Try up/down from every node

http://codeforces.com/contest/110/problem/A (1) //4 or 7

http://codeforces.com/contest/106/problem/B (2) //Cycles -\_-

http://codeforces.com/contest/895/problem/A (2) //All segments [in circle]

http://codeforces.com/contest/893/problem/B (2) //Try each divisor

http://codeforces.com/contest/894/problem/A (1) //3-cycles

http://codeforces.com/contest/892/problem/C (3) //Try to find "1" ASAP

http://codeforces.com/contest/102/problem/A (2) //Iterate over all triples

http://codeforces.com/contest/96/problem/B (2) //Check all

http://codeforces.com/contest/94/problem/B (1) //3cycles

http://codeforces.com/contest/887/problem/B (3) //Test all numbers

http://codeforces.com/gym/101597/problem/A (4) //[MATH][MODULO][SIMULATION]

http://codeforces.com/contest/68/problem/C (5) //[VERY NICE][RECURSION][MAX COST MIN FLOW]

http://codeforces.com/contest/68/problem/A (1) //Simple simulation

http://codeforces.com/contest/66/problem/B (2) //Test always whole platform

http://codeforces.com/contest/879/problem/C (3) //[NICE] one of each operation is enough

http://codeforces.com/contest/46/problem/C (2) //[2pointers][N^2 works too]

http://codeforces.com/contest/47/problem/D (4) //[Implementation][DFS]

http://codeforces.com/contest/51/problem/D (4) //Check all/check without 1s/2nd

<http://code-festival-2017-qualc.contest.atcoder.jp/tasks/code_festival_2017_qualc_b> (2)

http://codeforces.com/contest/53/problem/B (3) //at most 60 possibilities

http://codeforces.com/contest/55/problem/B (3) //Try all permutations & possibilities [NICE]

http://codeforces.com/contest/877/problem/B (3) //NICE [N^2][PrefixSum]

LA 6623 — Battle for Silver (3) //4 for-cycles inside ~ K4 search

UVA 12169 (2)

http://codeforces.com/contest/725/problem/C 4

http://codeforces.com/contest/725/problem/E 6

http://codeforces.com/contest/724/problem/B 3

11961 UVA (2)

11898 UVA (4)

11659 UVA (4)

http://codeforces.com/contest/753/problem/C 7

11699 UVA (4)

11548 UVA (3)

11471 UVA (5) //With dynamic programming

http://codeforces.com/contest/698/problem/D 8 //with geometry

http://codeforces.com/gym/101840 [F](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][BS][DISTANCE]

11206 UVA (6) //4^20 (but somehow passes)

11214 UVA (6) //Úvaha + pruning

11127 UVA (4) //Simple dfs [just realize you can do so]

<http://www.spoj.com/problems/BOKAM143SOU/> (3) //just implement for-cycles

<http://www.spoj.com/problems/BLOPER/> (4) dfs with little pruning

13173 UVA (3) //just brute-force + branching

http://codeforces.com/contest/799/problem/D (4) //VERY NICE [only top 34 needed] — trick with 2 [~20]

10890 UVA (4) //Simple brute-force times out, but with simple pruning AC (answer detection

http://codeforces.com/contest/813/problem/B (3) //All\*All (BF) care for overflow! NICE & EASY

http://codeforces.com/contest/817/problem/C (3) //Check S+Constant (NICE!)

10732 UVA (2) //Brute-force passes .. just don't trust them O(N^2)

10748 UVA (5) //VERY Nice (knights have K^2 moves not 8^K)

http://codeforces.com/contest/818/problem/D (4) //NICE for each 'A' check all remaining (max SQRT)

http://codeforces.com/contest/834/problem/E (5) //NICE — hard to imple: all 11122...999 OK

<http://www.spoj.com/problems/JHAGIRLS/> (4) //Efficient — or store output in array

http://codeforces.com/contest/846/problem/B (3) //Brute-force

<http://www.spoj.com/problems/ALONE/> (4) //Generate everything <10^15 [NICE]

http://codeforces.com/contest/861/problem/B (3) //Check all floor-sizes

<http://www.spoj.com/problems/RRANGE/> (3) //Compare all queries agains all updates + GAUSS

http://codeforces.com/contest/598/problem/B (3) //Treap works too ;-)

<http://www.spoj.com/problems/AMR10I/> (4) //Can be solved with brute-force with dp

http://codeforces.com/contest/868/problem/C (4) //Brute-force (fixet at most 6 each bits)

http://codeforces.com/contest/868/problem/D (5) //NICE: Maximal K is low (I assumed 15)

http://codeforces.com/contest/31/problem/C (2) //LOW-Constaints: N^2

http://codeforces.com/contest/32/problem/D (3) //Simply try all possibilities

http://codeforces.com/contest/876/problem/C (3) //Try N and ~100 lower

http://codeforces.com/contest/44/problem/B (2) //N^2 works fine

centroid

<https://codeforces.com/gym/101991/problem/A> (5) //[VERY NICE][FW]

http://codeforces.com/gym/101864 [D](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][SACK][ARPA][HLD]

http://codeforces.com/gym/101840 [E](http://cfrp.azurewebsites.net/blog/entry/5) //NICE

DCP-176: Motku2 [DevSkills]

http://codeforces.com/contest/715/problem/C 9

http://codeforces.com/contest/741/problem/D 8

13164 UVA (7)

http://codeforces.com/contest/752/problem/F 5

http://codeforces.com/contest/766/problem/E 6

http://codeforces.com/contest/833/problem/D 7 //Very nice — hard (thinking + imple) + FW

<http://www.spoj.com/problems/HOLI/> (4) //VERY NICE: 2\*Distances from centroids

coloring

<https://codeforces.com/gym/102006/problem/K> (6) //[VERY NICE][THEOREM][OBSERVATION]

http://codeforces.com/contest/741/problem/C 6

11331 UVA (4)

http://codeforces.com/contest/664/problem/D 4

combinatorics

<http://www.spoj.com/problems/ADATEAMS/>

<http://www.spoj.com/problems/ADASUM/>

<https://codeforces.com/contest/1173/problem/D> (4) //[VERY NICE][DFS][FACTORIAL]

<https://atcoder.jp/contests/abc136/tasks/abc136_f> (5) //[VERY NICE][IE][POWER][FENWICK]

<https://codeforces.com/contest/1178/problem/C> (3) //[NICE][EASY][POWER]

<https://codeforces.com/gym/101972/problem/J> (4) //[NICE][THEOREM][COMB-NUMBERS]

<https://codeforces.com/gym/101972/problem/I> (5) //[NICE][SHORT][COMB-NUMBERS]

http://codeforces.com/gym/101628/problem/G (5) //[NICE][COMB-NUMBERS]

http://codeforces.com/gym/101864 [E](http://cfrp.azurewebsites.net/blog/entry/3) //Combination number, power

http://codeforces.com/gym/101879/problem/E (2) //Power of 2

http://codeforces.com/gym/101873 (5) //Necklace (mby burnside)

http://codeforces.com/gym/101808/problem/F (3) //[EASY]][FACTORIAL]

http://codeforces.com/problemset/problem/364/A (4) //[NICE][STL-or-sorting][IF]

http://codeforces.com/contest/145/problem/D (7) //[VERY NICE][SET][OBSERVATION][BS][BIG]

http://codeforces.com/contest/909/problem/B (3)

http://codeforces.com/contest/131/problem/C (3) //[EASY][BRUTE-FORCE][COMB-NUMBERS]

http://codeforces.com/contest/131/problem/B (2) //[EASY]

http://codeforces.com/contest/129/problem/E (5) //[NICE][COMB-NUMBERS]//Observe

http://codeforces.com/contest/111/problem/D (5) //[COMB-NUMBERS][POWER][VERY NICE][DP]

http://codeforces.com/contest/895/problem/D (4) //[COMB-NUMBERS][OPTI]

http://codeforces.com/contest/893/problem/E (5) //[COMB-NUMBERS][NUMBER-THEORY]

http://codeforces.com/contest/894/problem/B (3) //try without last row

http://codeforces.com/contest/890/problem/E (6) //Permutations

http://codeforces.com/contest/52/problem/B (4) //[NICE][PREPROCESS][ROTATION]

3917 //Grid tiling [fancy approximation fomula]

http://codeforces.com/contest/760/problem/F

<https://devskill.com/CodingProblems/ViewProblem/61>

<https://devskill.com/CodingProblems/ViewProblem/255>

UVA 10918

UVA 12576

UVA 1118 //Parity

<http://www.spoj.com/problems/HLP_RAMS/> //Comb parity

Project Euler #78: Coin partitions //Partition function

<http://www.spoj.com/problems/MAIN75/> //DP #of trees

12001 UVA (3)

12034 UVA (4)

11719 UVA (5)

11798 UVA (5)

11282 UVA (4) //dearrangement

11174 UVA (4)

http://codeforces.com/contest/666/problem/C 7

<http://www.spoj.com/problems/JOKER1/> 3 prod(Ai-i)

<http://www.spoj.com/problems/ANTP/> //4

http://codeforces.com/contest/645/problem/E (5) //formula: A[i]=Sum(A)+1

<http://www.spoj.com/problems/SPCE/> (5) // N^{K-2}\*Prod(comp\_size)

http://codeforces.com/contest/785/problem/D (5) // F'(' C"(+)-1","("

13184 UVA (3)

http://codeforces.com/contest/816/problem/D (5) // Observation

13214 (4) //OEIS? : C(N+M-2,N-1)

http://codeforces.com/contest/844/problem/B (2) //Easy — pro prvaky

<http://www.spoj.com/problems/JOSWAP/> (3) //Frequence array

<http://www.spoj.com/problems/UCV2013E/> (4) //NICE&EASY: Choose steps to direction

<http://www.spoj.com/problems/PARCARD1/> (4) //Partition function (raw)

<http://www.spoj.com/problems/GOODB/> (2) //Easy (NICE): Choose [order]

<http://www.spoj.com/problems/LOOPEXP/> (4) //A000254/N!

<http://www.spoj.com/problems/DTPOLY/> (5) //DP might work too

<http://www.spoj.com/problems/DTPOLY2/> (7) //Harder version of above (NICE but hell)

<http://www.spoj.com/problems/HC12/> (3) //NICE — Sort + C(i,K-1)\*A[i]

<http://www.spoj.com/problems/STONE2/> (4) //NICE — Mostly DP [INVERSION][FACTORIAL]

<http://www.spoj.com/problems/MAIN8_D/> (5) //NICE — Suffixes/Prefixes (same add 2^i)

<http://www.spoj.com/problems/ITRIX_E/> (4) //VERY NICE — #Nonattacking 2-queens

<http://www.spoj.com/problems/MAXSUB/> (3) //NICE — Subsets made of zeroes

<http://www.spoj.com/problems/SKYLINE/> (3) //Catalan numbers [weird modulo — care]

http://codeforces.com/contest/26/problem/D (5) //Division of two combintion numbers [NI:/]

http://codeforces.com/contest/872/problem/E (6) // Prod:Sum(C(DistLines,CompSize))

constructive

<https://codeforces.com/contest/1174/problem/C> (4) //[VERY NICE][SIEVE]

<https://codeforces.com/contest/1173/problem/B> (3) //[VERY NICE][MATH]

<https://codeforces.com/contest/1202/problem/D> (4) //[NICE][MATH][GREEDY]

<https://codeforces.com/contest/1199/problem/E> (5) //[VERY NICE!][OBSERVATION][DFS]

<https://codeforces.com/contest/1180/problem/D> (5) //[VERY NICE][OVSERVATION]

<https://codeforces.com/contest/1178/problem/D> (4) //[VERY NICE][OBSERVATION][NUMBER THEORY]

<https://codeforces.com/contest/1187/problem/C> (4) //[NICE][IMPLEMENTATION]

http://codeforces.com/contest/922/problem/F (6) //[NICE][MATH][GREEDY]

http://codeforces.com/contest/916/problem/C (3) //Graph construction

http://codeforces.com/contest/148/problem/B (3) //[SIMULATION](math or bs)

http://codeforces.com/contest/909/problem/F (6) //[VERY NICE][BITS]

http://codeforces.com/contest/141/problem/C (4) //[NICE][BRUTE-FORCE]

http://codeforces.com/contest/907/problem/D (5) //[VERY NICE][RANDOM]

http://codeforces.com/contest/124/problem/C (3) //[NICE][SIEVE]

http://codeforces.com/contest/125/problem/C (4) //[NICE][C(N,2)]

http://codeforces.com/contest/902/problem/C (4) //[NICE]//Tree isomorphism

http://codeforces.com/contest/112/problem/C (3) //[GREEDY]//Max + 1s

http://codeforces.com/contest/110/problem/B (2) //Easy modulo

http://codeforces.com/contest/894/problem/C (3) //[VERY NICE] //AiAiGCD

http://codeforces.com/contest/97/problem/B (4) //NICE — Walls in middles [D&C]

http://codeforces.com/contest/85/problem/A (3) //MODULO / SHIFT

http://codeforces.com/contest/81/problem/D (3) //NICE — MAX(N/2) — even/odd

http://codeforces.com/contest/63/problem/D (3) //NICE[GO BY LINES][4 WAYS B/D ODD/EVEN]

http://codeforces.com/contest/42/problem/C (4) //Constructive works too but random is fine :)

http://codeforces.com/contest/43/problem/D (3) //NICE — Easy to see [implementation]

http://codeforces.com/contest/53/problem/C (2) //EASY [B/E Alternate]

http://codeforces.com/contest/877/problem/C (3) //NICE 3\*Alternative

http://codeforces.com/contest/802/problem/H (6) //We can do "N+k" by adding a letter p+k\*x+u+xx

http://codeforces.com/contest/12/problem/E (5) //g[i][j]=1+(i+j)%(N-1) [+/-]

http://codeforces.com/contest/22/problem/C (4) //Start and then clique without v (+ random)

http://codeforces.com/contest/26/problem/C (5) //make Even\*Even: do by 2\*2 fields

http://codeforces.com/contest/41/problem/E (4) //[NICE][CN/2,N/2]

http://codeforces.com/contest/78/problem/B (2) //NICE — last 3 and then rest in modulo

http://codeforces.com/contest/109/problem/D (5) //[NICE][BACK-POINTERS][SIMULATION][SORT]

dfs

<http://www.spoj.com/problems/ADASEA/>

<https://codeforces.com/contest/1176/problem/E> (4) //[NICE][CONSTRUCTIVE][TREE][TOPO]

<https://codeforces.com/contest/1186/problem/F> (5) //[NICE][GREEDY][CONSTRUCTIVE]

<https://codeforces.com/contest/1189/problem/D2> (5) //[NICE][CONSTRUCTIVE]

<https://atcoder.jp/contests/agc035/tasks/agc035_b> (4) //[NICE][DIRECTION]

<https://www.spoj.com/problems/CHUNK2/> (3) //[DFS][PRIMES]

<https://codeforces.com/gym/101981> [K](http://cfrp.azurewebsites.net/blog/entry/4) //[VERY NICE][IMPLE][TREE][SIMULATION]

<https://www.urionlinejudge.com.br/judge/en/problems/view/1621> (4) //[TREE][IMPLE]

<https://www.urionlinejudge.com.br/judge/en/problems/view/1610> (3) //[CYCLE]

<https://www.urionlinejudge.com.br/judge/en/problems/view/2911> (4) //[NICE][RULES][SIMPLE]

<https://codeforces.com/gym/102001/problem/K> (4) //[VERY NICE]

<https://codeforces.com/gym/101972/problem/B> (6) //[VERY NICE][TREE][STL]

<https://toph.co/p/incompatible-crops> (2)

http://codeforces.com/gym/100112 [K](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][BS][COLORING]

http://codeforces.com/gym/101962 [D](http://cfrp.azurewebsites.net/blog/entry/6) //[NICE] //Bitset shall work too

4286 — Equilibrium Mobile [LA](http://cfrp.azurewebsites.net/blog/entry/5) //[VERY NICE][OBSERVATION][TREE][DP]

http://codeforces.com/gym/101801 (4) [H] //[TREE][DP] //More dfss

<https://www.urionlinejudge.com.br/judge/en/problems/view/2732> (3) //Easy flood

http://codeforces.com/contest/920/problem/E (5) //[NICE][DFS][SET] //Or clever random

http://codeforces.com/contest/915/problem/D (4) //[VERY NICE][CYCLES DETECTION (ORIENTED)]

<http://www.spoj.com/problems/CTTC/> (3) //[EASY][GRAPH-RECONSTRUCTION]

7951 — Islands (3) //[NICE]Flood-Fill

http://codeforces.com/contest/901/problem/D (7) //Observations / Tree reduction

http://codeforces.com/contest/902/problem/B (3) //No dfs needed

http://codeforces.com/gym/101630 {C}(4) //[NICE][SCC]

http://codeforces.com/gym/101620 {J}(4) //DFS + multiples of divisors

http://codeforces.com/contest/120/problem/F (3) //Width of tree

7606 — Percolation (3) //Dfs on grid [EASY]

http://codeforces.com/contest/116/problem/C (2) //[DEPTH]

8080 — Christmas Tree (3) //[SIMPLE][NICE]

6584 — Escape (8) //[VERY VERY VERY NICE][COMPRESSION][MERGING] //Hard but I recommend this one!!

6590 Digraphs (4) //[VERY NICE][CYCLES][DP][IDEA]

http://codeforces.com/contest/893/problem/C (3) //Minimum from each connected component

http://codeforces.com/contest/884/problem/C (3) //[EASY][PERMUTATIONS][SORTING]

http://codeforces.com/contest/883/problem/G (4) //Greedy picking

http://codeforces.com/contest/60/problem/B (3) //3D Flood-Fill [NICE][EASY]

http://codeforces.com/contest/60/problem/C (4) //[VERY NICE][BF]//Not many real possibilities

<https://devskill.com/CodingProblems/ViewProblem/3>

<https://devskill.com/CodingProblems/ViewProblem/17>

<https://devskill.com/CodingProblems/ViewProblem/118> //Kind-of

657 — The die is cast [UVA]

12186 UVA (3)

http://codeforces.com/contest/734/problem/E (5)

http://codeforces.com/contest/727/problem/A (3)

http://codeforces.com/contest/723/problem/E (6)

http://codeforces.com/contest/709/problem/E (6)

http://codeforces.com/contest/710/problem/E (4)

http://codeforces.com/contest/758/problem/E (8)

11323 UVA (5)

http://codeforces.com/contest/760/problem/B (3)

http://codeforces.com/contest/761/problem/E (6)

http://codeforces.com/contest/638/problem/B (3) //connect cons. letters

http://codeforces.com/contest/638/problem/C (4) //greedy idea — easy

http://codeforces.com/contest/638/problem/D (5) //spec-DAG articulatin

http://codeforces.com/contest/767/problem/C (4)

http://codeforces.com/contest/781/problem/C (5)

http://codeforces.com/contest/794/problem/D (5) //NICE! Right done dfs

http://codeforces.com/contest/802/problem/K (5) //Slightly DP-like (NICE) TREE

http://codeforces.com/contest/813/problem/C (3) //Simply 2 DFS: NICE + EASY

http://codeforces.com/contest/841/problem/D (4) //DFS while tracking "next"

http://codeforces.com/contest/845/problem/G (5) //Keep track of cycles

http://codeforces.com/contest/844/problem/E (5) //Post-Order → line, Connect i → N-2: star

<http://www.spoj.com/problems/CAC/> (5) //VERY NICE! — Find all cycles in cactus

http://codeforces.com/contest/849/problem/C (3) //State search by gauss

http://codeforces.com/contest/846/problem/E (5) //NICE: DFS + some overflow logic

<http://www.spoj.com/problems/KOZE/> (3) //NICE: Floods

<http://www.spoj.com/problems/RIOI_2_3/> (4) //DFS /OR/ BFS /OR/ DSU [NICE][EASY][BF]

<http://www.spoj.com/problems/MAKEMAZE/> (3) //EASY — Simple dfs on grid

http://codeforces.com/contest/861/problem/F (5) //VERY NICE: Modify dfs tree so it remains connected

<http://www.spoj.com/problems/GHOSTS/> (3) //NICE — must remain dag after each QR

<http://www.spoj.com/problems/AMR10J/> (5) //VERY NICE! — DFS+DP [DAG with cycles]

http://codeforces.com/contest/24/problem/A (2)//NICE [DFS-ON-CYCLE]

http://codeforces.com/contest/29/problem/C (3) //Find begining/end of line (graph)

http://codeforces.com/contest/29/problem/D (4) //Tree [implementation][simulation]

digits

<https://devskill.com/CodingProblems/ViewProblem/564> (2) //[OBSERVATION]

<http://www.spoj.com/problems/PR003004/> (4) //Simple digits count

http://codeforces.com/contest/770/problem/B (3) //max num max digsum

dijkstra

<http://www.spoj.com/problems/ADATRIP/>

<https://www.urionlinejudge.com.br/judge/en/problems/view/1655> (4) //[EASY][PROBABILITY]

<https://www.urionlinejudge.com.br/judge/en/problems/view/2910> (4) //[TREE]

http://codeforces.com/gym/101628/problem/f (4)

http://codeforces.com/gym/100112 [E](http://cfrp.azurewebsites.net/blog/entry/6) //[NICE][BS][GEOMETRY]

http://codeforces.com/gym/101845/problem/J (5) //[NICE][OBSERVATION][TWICE]

http://codeforces.com/gym/101801 [K](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][MULTIPLE NODES]

http://codeforces.com/gym/101879/problem/F (8) //[NICE][BRIDGE]

http://codeforces.com/gym/101666 [D](http://cfrp.azurewebsites.net/blog/entry/4) //[NICE]

<http://www.spoj.com/problems/ADRABR/> (3) //Classical dijkstra — bad statement

6583 Subway (5) //[NICE]//Not exactly dijkstra by slightly similar [IMPLEMENTATION]

3850 [LA]

Gym 100625D [2013 Benelux Algorithm Programming Contest (BAPC 13)]

UVA 12950

Gym 100753A [2015 German Collegiate Programming Contest (GCPC 15) + POI 10-T3]

UVA 13030

UVA 1027

UVA 11377

http://codeforces.com/problemset/problem/843/D

11813 UVA

Gym 101242B [2016 ACM-ICPC World Finals] //+DP

Gym 100923B [2015 ACM National Contest Romania — Round 1]

UVA 11833

<http://www.spoj.com/problems/EZDIJKST/en/>

LightOJ 1019

UVA 13010 //+TS

2819 [LA]

UVA 12144

http://codeforces.com/contest/716/problem/D 7

12047 UVA 4

11514 UVA 4

http://codeforces.com/contest/757/problem/F 7

11338 UVA (4)

11374 UVA (4)

11097 UVA (4) //Divide to N\*1000 nodes and go!

13172 UVA (5) //6\*DJ per query + permutations

10816 UVA (4) //Easy Linear-Search by answer + DJ with path

http://codeforces.com/contest/827/problem/F 7 //Very nice — Even&Odd

<http://www.spoj.com/problems/DELIVER/> (5) //Normalize coordinates + Optimalize

<http://www.spoj.com/problems/CCHESS/> (4) //Dijkstra as knight

divide\_conquer

<https://codeforces.com/contest/1175/problem/F> (7) //[VERY NICE][HASH][RANDOMISATION]

<https://codeforces.com/contest/1181/problem/E2> (7) //[VERY NICE][SORTING][ST]

<https://codeforces.com/contest/1181/problem/E1> (5) //[VERY NICE][OBSERVATION][SORTING]

<https://codeforces.com/gym/102058/problem/G> (7) //[VERY NICE][BS][PRIORITY\_Q]

http://codeforces.com/gym/101845/problem/L (6) //[VERY NICE][INDUCTION]

http://codeforces.com/gym/101801 [J](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][TRIE][RMQ]

8257 — Factor-Free Tree [LA](http://cfrp.azurewebsites.net/blog/entry/6) //[VERY VERY NICE][PRIMES][FACTOR]

<https://www.urionlinejudge.com.br/judge/en/problems/view/1683> (4) //RAW histogram

http://codeforces.com/contest/817/problem/D (5) //Very nice NlogN

<http://www.spoj.com/problems/DYNACON2/> (8) //Lesser hell — offline Nlog(N) /or/ NlogN^2

http://codeforces.com/contest/876/problem/F (5) //VERY NICE — Find greatest + next/back functions

divisors

<https://codeforces.com/contest/1176/problem/D> (4) //[NICE][GREEDY][SORTING]

<https://atcoder.jp/contests/abc136/tasks/abc136_e> (5) //[NICE][GREEDY][DIVISORS]

<https://codeforces.com/contest/1183/problem/F> (5) //[VERY NICE][SORTING][OBSERVATION][BF]

<https://www.spoj.com/problems/KPOWERSUM/> (4) //[FACTORISATION]

http://codeforces.com/gym/101992/problem/D (5) //[VERY NICE][IE][NUMBER THEORY]

http://codeforces.com/gym/101982 [B](http://cfrp.azurewebsites.net/blog/entry/5) //[FACTORISATION][GCD][SIEVE]

http://codeforces.com/gym/101864 [G](http://cfrp.azurewebsites.net/blog/entry/5) //[VERY NICE][FACTORISATION][GCD]

http://codeforces.com/gym/101840 [D](http://cfrp.azurewebsites.net/blog/entry/4) //[EVENTS][NICE]

http://codeforces.com/gym/101741/problem/F (6) //[NICE][PROBABILITY][GCD]

<http://www.spoj.com/problems/FRNDAM/> (3) //Simply find sum of all divisors (low constraints)

http://codeforces.com/contest/920/problem/F (5) //[NICE][FAST FALL][SET][DIVISORS]

7726 A Simple Math Problem (4) //TLE if naive (consider only divisors)

http://codeforces.com/contest/112/problem/D (4) //[NICE] Last-position [FACTOR]

http://codeforces.com/contest/75/problem/C (3) //[NICE][BS]

LightOJ 1068

LightOJ 1134

Project Euler #95: Amicable chains

NAJ0001 — Divisible Number Sum [SPOJ]

<http://www.spoj.com/problems/LCMSUM/>

<https://www.hackerrank.com/contests/101hack38/challenges/easy-gcd-1/problem>

UVA 13085

<http://www.spoj.com/problems/CDRSANJ/>

<http://www.spoj.com/problems/DIVSEQ/>

UVA 12154

UVA 13058

<http://www.spoj.com/problems/EC_DIVS/>

<https://www.codechef.com/problems/CHEFKEY>

http://codeforces.com/problemset/problem/671/C

Gym 101411G [2009-2010 ACM-ICPC, NEERC, Western Subregional Contest]

http://codeforces.com/problemset/problem/831/F

http://codeforces.com/problemset/problem/839/D

12934 — Factorial Division [UVA]

UVA 10880

<http://www.spoj.com/problems/PSTR/>

http://codeforces.com/problemset/problem/27/E

[LA] 3014

UVA 12843

<https://www.urionlinejudge.com.br/judge/en/problems/view/1164>

http://codeforces.com/problemset/problem/803/F

10892 — LCM Cardinality

<http://www.spoj.com/problems/GCDEX/>

<http://www.spoj.com/problems/INVDIV/>

13083 — Yet another GCDSUM //ll

<http://www.spoj.com/problems/IITKWPCF/> //ll

UVA 13185

UVA 13194

UVA 11388

<http://www.spoj.com/problems/SAS002/>

12425 — Best Friend

http://codeforces.com/problemset/problem/703/E

<https://www.hackerearth.com/problem/algorithm/harry-gets-into-infy/>

UVA 10830 //SUM

<http://www.spoj.com/problems/DIVSUM/en/> //SUM

<http://www.spoj.com/problems/AFS/> //SUM

UVA 11526 //SUM

dp

<http://www.spoj.com/problems/ADAZOO/>

<http://www.spoj.com/problems/ADASEQEN/>

<http://www.spoj.com/problems/ADAMOLD/>

<http://www.spoj.com/problems/ADASALE/>

<https://codeforces.com/contest/1203/problem/F2> (6) //[VERY NICE][GREEDY][SORTING][CASES]

<https://codeforces.com/contest/1176/problem/F> (5) //[NICE][IMPLE][SORTING]

<https://codeforces.com/contest/1200/problem/F> (6) //[NICE][DFS][GRAPH][LCM][MATH]

<https://codeforces.com/contest/1201/problem/D> (5) //[NICE][DP][IMPLE][CASES]

<https://www.spoj.com/problems/THECODE/> (4) //[BITMASK]

<https://codeforces.com/contest/1199/problem/F> (5) //[NICE][DIMENSIONS]

<https://codeforces.com/contest/1183/problem/H> (4) //[NICE][SUBSEQUENCE][NEXT]

<https://codeforces.com/contest/1183/problem/E> (4) //[NICE][SUBSEQUENCE][NEXT]

<https://codeforces.com/contest/1178/problem/F2> (7) //[VERY NICE][OBSERVATION]

<https://codeforces.com/contest/1178/problem/F1> (5) //[NICE][MATH][REDUCTION]

<https://codeforces.com/contest/1178/problem/B> (3) //[EASY]Not even real DP

<https://codeforces.com/contest/1195/problem/C> (3) //[EASY]

<https://codeforces.com/contest/1189/problem/F> (6) //[VERY NICE][OBSERVATION]

<https://atcoder.jp/contests/agc035/tasks/agc035_d> (5) //[VERY NICE][BIT][IDEA]

<https://www.spoj.com/problems/PLOVER/> (4) //[NICE][DIGITS]

<https://www.spoj.com/problems/NICESEQ/> (4) //[DIGITS][POSSIBLY FASTER]

<https://codeforces.com/gym/101981> [L](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE]

<https://www.spoj.com/problems/RANDMOD/> (4) //[OBSERVATION][SEQUENCE]

<https://www.urionlinejudge.com.br/judge/en/problems/view/1592> (4) //[GRAPH]

<https://www.urionlinejudge.com.br/judge/en/problems/view/2824> (4) //Classical

<https://codeforces.com/gym/102006/problem/D> (4) //[NICE][OBSERVATION]

<https://codeforces.com/gym/102021> [K](http://cfrp.azurewebsites.net/blog/entry/4)

<https://codeforces.com/gym/102058/problem/D> (5) //[NICE][DOMINATION]

UVA 13294 (4) //[BIT]

http://codeforces.com/gym/101992/problem/A (6) //[BITS][OBSERVATION]

http://codeforces.com/gym/101628/problem/H (3) //[NICE][CLASSICAL][DAG]

http://codeforces.com/gym/101628/problem/A (3)

http://codeforces.com/gym/100112 [J](http://cfrp.azurewebsites.net/blog/entry/4) //[NICE][TREE]

http://codeforces.com/gym/101982 [D](http://cfrp.azurewebsites.net/blog/entry/5) //[VERY NICE][BITS] //Window

http://codeforces.com/gym/101982 [C](http://cfrp.azurewebsites.net/blog/entry/4) //Classical combinatorics

http://codeforces.com/gym/101845/problem/E (5)

<https://www.codechef.com/problems/SPECTAC> (5) //[NICE]

http://codeforces.com/gym/101801 [E]

http://codeforces.com/gym/101801 [D](http://cfrp.azurewebsites.net/blog/entry/6) //[VERY NICE][SEGMENT TREE][SORTING][NORMALIZE]

http://codeforces.com/gym/101873 [J](http://cfrp.azurewebsites.net/blog/entry/7) //[NICE][IMPLEMENTATION][STRING][SUBSUBSET]

http://codeforces.com/gym/101889 [E](http://cfrp.azurewebsites.net/blog/entry/4)

http://codeforces.com/gym/101808/problem/D (7) //[NICE][OPTIMISATION][EFFICIENCY]

http://codeforces.com/gym/101666 [I](http://cfrp.azurewebsites.net/blog/entry/4)

http://codeforces.com/gym/101840 [A](http://cfrp.azurewebsites.net/blog/entry/4) //Doubles

DevSkills 475: Bunty's Xor Game (4) //[BITS][GAME THEORY]

<https://www.devskill.com/CodingProblems/ViewProblem/489> (5) //[NICE][FACTORISATION]

DevSkills 534 (4) //[BITSET][COMBINATIONS]

http://codeforces.com/gym/101650 [D](http://cfrp.azurewebsites.net/blog/entry/5) //Not hard — more coding + printing

13286 — Ingredients (4) //[NICE][DAG] Classical

8299 — Bricks (5) //[NICE][COMBINATORICS][SORTING]

http://codeforces.com/contest/934/my (4) //Subsequence

http://codeforces.com/contest/922/problem/E (5) //find max mana

http://codeforces.com/contest/920/problem/D (4) //[NICE][KNAPSACK-MODULO][GREEDY]

http://codeforces.com/contest/919/problem/D (4) //[NICE][DFS][CYCLE][DAG]

http://codeforces.com/contest/914/problem/C (3) //[DIGITS][BINARY]

http://codeforces.com/contest/913/problem/E (5) //[NICE][EXPRESSION]

http://codeforces.com/contest/148/problem/E (5) //[DP][PREFIX SUM][GREED]

<http://www.spoj.com/problems/PALMKR/> (4) //Classical palindrome + print + lexicography

http://codeforces.com/contest/146/problem/E (5) //[NICE][COMBINATORICS][OBSERVATION]

http://codeforces.com/contest/909/problem/C (4) //[NICE][CLASSICAL][EASY]

7785 — m-ary Partitions (4) //Combinatorics

http://codeforces.com/contest/133/problem/E (4) //[IMPLEMENTATION]

http://codeforces.com/contest/126/problem/D (5) //[NICE]//Decomposet DP//Own hash-map

http://codeforces.com/contest/910/problem/B (3) //Or many other ways

6154 RNA Secondary Structure (5) //[NICE] Unoptimal might work too

<https://arc087.contest.atcoder.jp/tasks/arc087_b> (4) //[VERY NICE][OBSERVATION]

http://codeforces.com/contest/903/problem/F (5) //[VERY NICE][BITMASK][BRUTE]

http://codeforces.com/contest/903/problem/A (2) //Easy knapsack-like // Low constraints

http://codeforces.com/contest/900/problem/E (4) //[NICE]//Patter-match: FFT or KMP or Brute-Force

http://codeforces.com/contest/118/problem/D (3) //[COMBINATORICS][LOW-CONSTRAINTS]

http://codeforces.com/contest/115/problem/E (5) //[VERY NICE][SEGMENT TREE]

http://codeforces.com/contest/116/problem/D (3) //CLASSICAL[LOW-CONSTRAINTS]

10128 Queue (uva) (4) //One possibility is bitmask — second combinatorics

http://codeforces.com/contest/110/problem/C (3) //[EASY]// Greedy/math works too

8078 — Bracket Sequence (4) //[VERY NICE][DP-LINKS]

http://codeforces.com/contest/106/problem/C (3) //[NICE][EASY]

http://codeforces.com/contest/895/problem/C (4) //[NICE]

8024 Stack Construction (4) //[NICE] Palindromic style — but not exactly

7708 — Cubes (6) //[MATH]

http://codeforces.com/contest/888/problem/F (6) //[NICE][FLAG]

<http://www.spoj.com/problems/ACQUIRE/> (5) //[NICE][CH-OPT]

http://codeforces.com/contest/319/problem/C (6) //[NICE][CH-OPT]

<http://www.spoj.com/problems/NKLEAVES/> (5) //[NICE][DC]

http://codeforces.com/contest/76/problem/D (4) //[BITS][OVERFLOW]

http://codeforces.com/contest/73/problem/C (4) //[NICE][EASY][TRY-ALL-LETTERS]

http://codeforces.com/contest/67/problem/C (4) //[NICE][LEAVENSTEIN]

http://codeforces.com/contest/67/problem/A (3) //[EASY][PRINTING][OTHER POSSIBLE WAYS]

http://codeforces.com/contest/55/problem/D (5) //[NICE][DIGITS][EFFICIENT]

http://codeforces.com/contest/56/problem/D (4) //String modification + printing [NICE]

http://codeforces.com/contest/58/problem/E (6) //[NICE][IMPLEMENTATION]

UVA 12181

<https://devskill.com/CodingProblems/ViewProblem/21>

<https://devskill.com/CodingProblems/ViewProblem/37>

<https://devskill.com/CodingProblems/ViewProblem/71>

<https://devskill.com/CodingProblems/ViewProblem/103>

<https://devskill.com/CodingProblems/ViewProblem/107>

<https://devskill.com/CodingProblems/ViewProblem/115>

<https://devskill.com/CodingProblems/ViewProblem/126>

<https://devskill.com/CodingProblems/ViewProblem/131>

<https://devskill.com/CodingProblems/ViewProblem/134>

<https://devskill.com/CodingProblems/ViewProblem/174>

<https://devskill.com/CodingProblems/ViewProblem/186>

<https://devskill.com/CodingProblems/ViewProblem/201>

<https://devskill.com/CodingProblems/ViewProblem/338>

<https://devskill.com/CodingProblems/ViewProblem/368>

<https://devskill.com/CodingProblems/ViewProblem/392>

<https://devskill.com/CodingProblems/ViewProblem/399>

<https://www.hackerrank.com/contests/world-codesprint-5/challenges/mining> //Opti

UVA 12915 //Opti

UVA 12524 //Opti

http://codeforces.com/problemset/problem/631/E //CH

<https://devskill.com/CodingProblems/ViewProblem/6>

<https://devskill.com/CodingProblems/ViewProblem/11>

11552 UVA (3)

12172 UVA (3)

4507 LA (5)

4510 LA (5) [+ geometry]

12181 UVA (6)

http://codeforces.com/contest/729/problem/F 6

http://codeforces.com/contest/735/problem/E 9

http://codeforces.com/contest/731/problem/E 5

12030 UVA (4)

http://codeforces.com/contest/721/problem/E 7

http://codeforces.com/contest/742/problem/D 4

12040 UVA (5)

http://codeforces.com/contest/712/problem/D 5

13162 UVA (6)

http://codeforces.com/contest/743/problem/E 6

11908 UVA (3)

11932 UVA (4)

http://codeforces.com/contest/745/problem/E (7)

11806 UVA (4)

http://codeforces.com/contest/747/problem/F (5)

11843 UVA (4)

http://codeforces.com/contest/752/problem/E (5)

http://codeforces.com/contest/703/problem/E (7)

11753 UVA (4)

11725 UVA (5)

http://codeforces.com/contest/722/problem/E (9)

http://codeforces.com/contest/760/problem/F (8)

11795 UVA (3)

11654 UVA (4)

11523 UVA (5)

11404 UVA (4)

11432 UVA (4)

11451 UVA (4) //C==20 mistake in statement

11301 UVA (4)

http://codeforces.com/contest/762/problem/D 5

11361 UVA (4) //digit DP

11365 UVA (7)

11391 UVA (4) //easy+implementation

11394 UVA (3)

11218 UVA (2)

11125 UVA (4) //slightly implementation

11076 UVA (3)

11081 UVA (4) //3 string subsequences (beware of fail)

http://codeforces.com/contest/678/problem/E (5) //bitset dp + probability

http://codeforces.com/contest/766/problem/C (4)

http://codeforces.com/contest/667/problem/C (3)

<http://www.spoj.com/problems/MOVIFAN/> (3)

<http://www.spoj.com/problems/ORDSUM23/> (3)

<http://www.spoj.com/problems/DIVSEQ/> (4) //N^3 (but better...) works fine

http://codeforces.com/contest/633/problem/F (7) //Tree dp

<http://www.spoj.com/problems/ADJDUCKS/> (4) sort + pick 2-3 continous O(N)

<http://www.spoj.com/problems/JLNT/> (4) //pick 0 or 2 | 1e3\*5e3

<http://www.spoj.com/problems/TPCPALIN/> (5) //500^3 works (3rd countable)

<http://www.spoj.com/problems/COLORSEG/> (4) //50^4==OK 50^4log(N)=TLE NICE

<http://www.spoj.com/problems/POWERCAR/> (3) //1e3\*1e3\*2 — follow rules

<http://www.spoj.com/problems/INGRED/> (5) //TSP-like [reduce + go]

<http://www.spoj.com/problems/BADXOR/> (4) //classical subsets

<http://www.spoj.com/problems/SPCO/> (5) //64\*64\*2 DP {OPT: prime O(1) + clear only half}

<http://www.spoj.com/problems/WAYHOME/> (5) //NICE: 1) 1\*1 b)12,1,\*\*,2

<http://www.spoj.com/problems/NFURY/> (2) //Minimal sum of squares

<http://www.spoj.com/problems/GDIL/> (3) //combinatorics

http://codeforces.com/contest/791/problem/D (5) //Tree

http://codeforces.com/contest/791/problem/E (6) //V,K,X — pick any

http://codeforces.com/contest/789/problem/C (3)

13176 (4) //N^6

13179 (5) //NICE [Ath][Bth][TimeDiff]

http://codeforces.com/contest/796/problem/E (6) //NICE: N\*P\*K\*K (WC can't happen!)

http://codeforces.com/contest/797/problem/E (4) //NICE: Almost BF-able (but care of low K)

http://codeforces.com/contest/793/problem/D (3) //NICE & EASY: begin/end/actual/USED

http://codeforces.com/contest/803/problem/E (4) //State search — many IF's (EASY)

http://codeforces.com/contest/805/problem/F (7) //NICE: DP on tree + fast BF + hack

http://codeforces.com/contest/808/problem/E (5) //NICE!

http://codeforces.com/contest/811/problem/C (4) //Precalculate + DP (greedy thinking)

10817 UVA 4 //Easy but slightly implementation

10859 UVA 4 //Nice — on tree .. but for a reason small constrains

10898 UVA 4 //Hash is lesser than 1e6 .. try everything

http://codeforces.com/contest/812/problem/B (3) //Not only DP, yet imho easiest ..many spec cases

http://codeforces.com/contest/813/problem/D (5) //VERY VERY NICE — N\*N (none picked between a/b)

http://codeforces.com/contest/814/problem/E 5 //VERY NICE — Harder imple: Combinatorics

http://codeforces.com/problemset/problem/816/E (6) //NICE — Tree (hard 2C complexity)

http://codeforces.com/contest/837/problem/D (5) //NICE — yet kinda pain [must be iterative]

<http://www.spoj.com/problems/AUT/> (4) //NICE — K is interesting ~ at most 1600

<http://www.spoj.com/problems/GNYR04C/> (3) //Easy — Nice idea [Big→ Low approach]

<http://www.spoj.com/problems/TIEROPE/> (4) //Process 2\*L ~ otherwise pick BIG

<http://www.spoj.com/problems/IITKWPCE/> (4) //Palindromes [efficiency!] — NICE!

IITKWPCD SPOJ (4) //+Slightly geometry

UVA 1496 //[Steiner's Tree] Very Nice (8)

<http://www.spoj.com/problems/LKS/> (3) //Classical knapsack

<http://www.spoj.com/problems/UOFTAE/> (3) //Easy & Sympatic DP

<http://www.spoj.com/problems/DCOWS/> (4) //Very NICE (sort + GO)

<http://www.spoj.com/problems/FARIDA/> (3) //Easy & Sympatic ((u+1) | Price+(u+2))

<http://www.spoj.com/problems/AU7_5/> (2) //EASY: dyn(n-1)+dyn(n-k-1)

<http://www.spoj.com/problems/NAIVELOK/> (4) //NICE [depalindromisation]

http://codeforces.com/contest/846/problem/C (4) //With print

<http://www.spoj.com/problems/CNT_LUCK/> (4) //Number (binary) dp [NICE] {ull care 0-1}

<http://www.spoj.com/problems/MAY99_4/> (3) //Almost combinatoric Sub and 0/1,1/0

<http://www.spoj.com/problems/GEEKOUNT/> (4) //Number dp

<http://www.spoj.com/problems/MUTDNA/> (4) // N\*2 (turned?) [not sure if grd poss.]

<http://www.spoj.com/problems/RIOI_3_2/> (5) //VERY NICE (easy imple — Number Theory thinking)

<http://www.spoj.com/problems/MAXWOODS/> (3) //NICE [EASY][GRID]

<http://www.spoj.com/problems/DIEHARD/> (3) //Easy — prolly solvable by greedy (but dp is easier)

<http://www.spoj.com/problems/DCEPC810/> (4) //VERY VERY NICE — Subsequence 2pointers+2bools

<http://www.spoj.com/problems/EQ2/> (4) //NICE: Digit + Carry (from back) — iff-party

<http://www.spoj.com/problems/DCEPC501/> (3) //NICE & EASY

<http://www.spoj.com/problems/NUMTSN/> (4) //NICE — Thinking or Opti

<http://www.spoj.com/problems/GONE/> (4) //NICE & EASY [digits]

<http://www.spoj.com/problems/RAONE/> (4) //NICE & EASY [digits] — almost similar as above

<http://www.spoj.com/problems/STRSEQ/> (4) //VERY VERY NICE — Next-Function

<http://www.spoj.com/problems/MYQ8/> (4) //VERY NICE — 3x3 tic-tac-toe [implementation]

http://codeforces.com/contest/859/problem/C (3) //Easy+Sympathic [PrefixSumOptional]

http://codeforces.com/contest/859/problem/D (4) //NICE [Probabilities]

<http://www.spoj.com/problems/UNICA/> (4) //VERY NICE [Posibilities][Print][Classical]

<http://www.spoj.com/problems/KOPC12H/> (4) //NICE Digit-DP

<http://www.spoj.com/problems/DRACULA/> (4) //NICE Digit-DP (Both sides) — iterate by sum

<http://www.spoj.com/problems/ABCPATH/> (3) //DP over dfs (maybe without dp works too?)

<http://www.spoj.com/problems/BEHAPPY/> (2) //Easy one — low constraints

<http://www.spoj.com/problems/STRCOUNT/> (4) //No input (over bits)

http://codeforces.com/contest/855/problem/B (2) //prolly not even necessary

http://codeforces.com/contest/855/problem/C (4) //dp on tree

http://codeforces.com/contest/855/problem/E (5) //VERY NICE — Digits & Bitmask & Query (learning!)

<http://www.spoj.com/problems/PAINTWAL/> (6) //VERY NICE — Imho hard (opti could beat)

<http://www.spoj.com/problems/ADFRUITS/> (3) //Very simple (substring == subsequence)

<http://www.spoj.com/problems/MAIN113/> (2) //NICE but somehow too low constraints

<http://www.spoj.com/problems/MAIN112/> (4) //NICE — Bitmask

http://codeforces.com/contest/864/problem/E (5) //VERY NICE — Sort

<http://www.spoj.com/problems/NOVICE63/> (4) //NICE -On digits (binary)

<http://www.spoj.com/problems/TUG/> (3) //NICE + Observation {N>100 == YES}

<http://www.spoj.com/problems/DOMINO1/> (4) //Used map to solve it

<http://www.spoj.com/problems/NY10E/> (2) //Easy dp

<http://www.spoj.com/problems/MAIN72/> (3) //Easy knapsack

<http://www.spoj.com/problems/NOVICE43/> (2) //Unbelievably low constraints

http://codeforces.com/contest/598/problem/E (4) //N^5 strategy works fine [VERY NICE]

<http://www.spoj.com/problems/CHAIR/> (3) //Maybe combinatorics too?

<http://www.spoj.com/problems/ACPC10D/> (3) //NICE — DAG traversal

<http://www.spoj.com/problems/CPCRC1C/> (4) //Digits dp (return pair)

<http://www.spoj.com/problems/BORW/> (3) //Inc+Dec sequence (small array)

http://codeforces.com/problemset/problem/16/E (5) //Bitmask [NICE]

http://codeforces.com/problemset/problem/18/E (5) //VERY NICE {no need for second iteration}

http://codeforces.com/contest/2/problem/B (5) //NICE — 2/5 are in-fact independent

http://codeforces.com/contest/4/problem/D (3) //Classical [FW works too] XY > xy

http://codeforces.com/contest/6/problem/D (4) //NICE (N^4)

http://codeforces.com/contest/321/problem/E (7) //VERY NICE — D&C Trick

http://codeforces.com/contest/868/problem/F (8) //VERY VERY NICE D&C Trick — With MO Principal

http://codeforces.com/contest/8/problem/C (5) //NICE — Masks [N\*2^N]

http://codeforces.com/contest/868/problem/E (8) //VERY NICE — HARD — on tree

http://codeforces.com/contest/10/problem/D (4) //LCIS [NICE]

http://codeforces.com/contest/13/problem/C (5) //NICE [sorting][only elements from array]

http://codeforces.com/contest/17/problem/C (5) //[NICE][iterative-sparse][+idea]

http://codeforces.com/contest/19/problem/B (4) //Knapsack (after good look)

http://codeforces.com/contest/30/problem/C (4) //Probabilities + (slight)GEO

http://codeforces.com/contest/31/problem/E (4) //[NICE]

http://codeforces.com/contest/41/problem/D (4) //With printing

dsu

<http://www.spoj.com/problems/ADABRANC/>

<https://www.spoj.com/problems/CONSEC/> (4) //[NICE][OFFLINE]

<https://codeforces.com/gym/102006/problem/C> (4) //[NICE][BRUTE-FORCE]

http://codeforces.com/gym/101962/problem/J (5) //[VERY NICE][TREE-FAT][SPARSE]

http://codeforces.com/contest/915/problem/F (6) //[VERY NICE][SORTING]

http://codeforces.com/contest/141/problem/E (6) //[NICE][SPANNIG TREE]

7903 — Pandaria (7) //[VERY NICE][DSU][SORTING][MERGE][DFS]

http://codeforces.com/contest/110/problem/E (4) //[NICE][COMBINATORICS][TREE]

http://codeforces.com/contest/90/problem/E (5) //[NICE][DSU-LIKE-LINKS][SIMULATION]

http://codeforces.com/contest/87/problem/D (5) //[VERY NICE][SORTING][COMPRES][DFS]

http://codeforces.com/contest/884/problem/E (5) //[VERY NICE][MEMORY SPARSE]

http://codeforces.com/contest/60/problem/D (6) //[NICE][Pythagorean Triples][Gen over max!]

UVA 10947

UVA 12363

LA 3833

http://codeforces.com/problemset/problem/742/D //+DP

UVA 10178

http://codeforces.com/contest/723/problem/F 7

13153 UVA (5)

13169 UVA (3)

11987 UVA (3)

11474 UVA (4)

<http://www.spoj.com/problems/BTCODE_G/>

http://codeforces.com/problemset/problem/691/D

Gym 101174K [2016-2017 ACM-ICPC Southwestern European Regional Programming Contest (SWERC 2016)]

UVA 10583

LightOJ 1003

http://codeforces.com/problemset/problem/731/C

UVA 793

UVA 11966

<https://www.codechef.com/problems/COZIC>

3939 [LA]

UVA 11503

http://codeforces.com/problemset/problem/755/C

UVA 1395

http://codeforces.com/contest/687/problem/D 6

http://codeforces.com/contest/680/problem/E 7 //+precalculation/brute force

http://codeforces.com/contest/766/problem/D 5

<http://www.spoj.com/problems/LEXSTR/> (3) //Nice na stringu

http://codeforces.com/contest/805/problem/C 3 //NICE (dijkstra like :P)

<http://www.spoj.com/problems/IITKWPCI/> (3) //VERY NICE

<http://www.spoj.com/problems/FRNDCIRC> (3) //Classical DSU (NICE for practice)

<http://www.spoj.com/problems/FOXLINGS/> (3) Easy — just renumbering

<http://www.spoj.com/problems/NITTROAD/> (4) //Process from back

<http://www.spoj.com/problems/SHAHBG/> (2) //DSU not needes (simulated by array)

http://codeforces.com/contest/598/problem/D (3) //Can be solved with DFS too

http://codeforces.com/contest/9/problem/E (4) //Making one big cycle

http://codeforces.com/contest/25/problem/D (4) //Could be done linear too

http://codeforces.com/contest/28/problem/B (4) //NICE [imho bad statement]

http://codeforces.com/contest/876/problem/D (4) //DSU adjacent + visited

http://codeforces.com/contest/875/problem/F (6) //NICE — Maximum cactus-forest [kruskal-like]

euler\_function

http://codeforces.com/gym/101879/problem/C (6) //[VERY NICE][DFS]

http://codeforces.com/contest/907/problem/F (7) //[MAGIC]

UVA 10299

UVA 10990

<https://www.codechef.com/problems/SMPLSUM>

<https://www.codechef.com/problems/COZIE>

<http://www.spoj.com/problems/LCMSUM/>

Gym 100975F [2003-2004 Petrozavodsk Summer Training Camp, Saratov SU Contest]

UVA 13132

<http://www.spoj.com/problems/GCDEX/>

UVA 12995

<http://www.spoj.com/problems/TIP1/>

UVA 11327

LightOJ 1007

<http://www.spoj.com/problems/ETF/>

Project Euler #72: Counting fractions

<http://www.spoj.com/problems/DCEPCA03/>

<http://www.spoj.com/problems/NAJPWG/> 4 //Gauss-Euler

<http://www.spoj.com/problems/DCEPC12G/> (5) //Do what is written there

<http://www.spoj.com/problems/INVPHI/> (5) //Inverse euler

euler\_tour

<https://codeforces.com/gym/102006/problem/H> (4) //[SORTING][OBSERVATION]

http://codeforces.com/gym/101650 [I](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][GRAPHS]// Theory is useful

13246 — Chained Words (4) //[NICE][LEXICOGRAPHICAL]

UVA 10735

http://codeforces.com/contest/789/problem/D //Adj EG + Self/everything

http://codeforces.com/contest/21/problem/D (5) //[NICE][EulerTour+DP]

http://codeforces.com/contest/36/problem/E (6) //VERY NICE [4odd is hardest]

factorization

<http://www.spoj.com/problems/ADAHW/> [RHO][Number Theory]

<http://www.spoj.com/problems/ADADIGIT/> [Permutations]

<https://www.spoj.com/problems/PRIMEP/> (5) //[VERY NICE][SEGMENT][BS][SIEVE]

<https://codeforces.com/gym/101981> [J](http://cfrp.azurewebsites.net/blog/entry/4) //[NICE][DIVIDE]

http://codeforces.com/gym/101801 [B](http://cfrp.azurewebsites.net/blog/entry/3)

UVA 13067

Project Euler #108: Diophantine reciprocals I

<http://www.spoj.com/problems/CHGROOM/>

Gym 101370A [2009-2010 Summer Petrozavodsk Camp, Petr Mitrichev Contest 5]

http://codeforces.com/problemset/problem/837/E

<http://www.spoj.com/problems/PSYCHOT/>

<http://www.spoj.com/problems/FACTDIV/>

<http://www.spoj.com/problems/NOSQ/>

<http://www.spoj.com/problems/FCDC/>

<http://www.spoj.com/problems/NFACTOR/>

<http://www.spoj.com/problems/ABA12D/>

<http://www.spoj.com/problems/PSTR/>

<http://www.spoj.com/problems/AMR11E/>

<http://www.spoj.com/problems/FACT1/>

<http://www.spoj.com/problems/FACT2/>

<https://www.hackerearth.com/problem/algorithm/gold-at-lolympics/>

12005 UVA (7)

12062 UVA (6)

11960 UVA (3)

<http://www.spoj.com/problems/FACTCG2/> (3)

<http://www.spoj.com/problems/FACT0/> (4)

http://codeforces.com/contest/546/problem/D 5

http://codeforces.com/contest/222/problem/C 6

<http://www.spoj.com/problems/COMDIV/> 3

<http://www.spoj.com/problems/SINEGGS/> 3

<http://www.spoj.com/problems/BDOI16B/> 4

<http://www.spoj.com/problems/HG/> 4 //Map == OK

11099 UVA (3) //factor + recursion

13194 UVA (3) //factorize+generate /or just check

13191 UVA (6) //Pollard-Rho

http://codeforces.com/contest/818/problem/E (4) // Efficient + Two Pointers

http://codeforces.com/contest/831/problem/F (6) //MAGIC

http://codeforces.com/contest/839/problem/D (4) // Combinatorics + IE

<http://www.spoj.com/problems/SAS002/> (5) //Find all divisors (big numbers)

<http://www.spoj.com/problems/GCDS/> (4) //Lowest unused prime

<http://www.spoj.com/problems/IITKWPCF/> (4) //Nonprime divisors of N/2

http://codeforces.com/contest/851/problem/D (4) //Properties of GCD + factor: NICE

<http://www.spoj.com/problems/PTIME/> (3) //Low bounds — check each prime independently

<http://www.spoj.com/problems/MAIN12B/> (3) //NICE [Factorization][Sort][Unique]

<http://www.spoj.com/problems/AMR11E/> (2) //2664 is the biggest

<http://www.spoj.com/problems/GCPC11A/> (4) //Very nice — factorize + divide N by powers

<http://www.spoj.com/problems/AMR10C/> (3) //Maximum of factor-powers

fenwick

<http://www.spoj.com/problems/ADABEHIVE/> [2D]

<http://www.spoj.com/problems/ADACABAA/> [2D][Sparse]

<https://codeforces.com/contest/1191/problem/F> (5) //[NICE][SORTING]Or similar DS

<https://codeforces.com/gym/102001/problem/H> (4) //[NICE][GREEDY]

<https://toph.co/p/easy-prime> (4) //[EASY][SIEVE]

<https://toph.co/p/mario-and-princess-peach> (5) //[NICE][DP][SEGMENT TREE][MAX]

http://codeforces.com/gym/101628/problem/B (5) //[NICE][SPARSE][STL]

http://codeforces.com/gym/100112 (4) [B] //[NICE][INVERSION][OBSERVATION]

http://codeforces.com/gym/101982 (7) [I] //[VERY NICE][INVERSION][OBSERVATION]

http://codeforces.com/gym/101908/problem/C (4) //[MATH][INVERSION]

http://codeforces.com/gym/101889 (5) //[NICE] normalize

DevSkills-422: Double Pairs (4) //[NICE] normalize

7591 — Distribution Center (4) //[NICE][SORTING]

http://codeforces.com/contest/903/problem/D (5) //[NICE][BIG]

http://codeforces.com/contest/102/problem/D (4) //[NICE][+DP][NORMALIZE]

http://codeforces.com/gym/101047/problem/J [2D]

<http://www.spoj.com/problems/MATSUM/> [2D]

<https://www.hackerearth.com/practice/data-structures/advanced-data-structures/fenwick-binary-indexed-trees/practice-problems/algorithm/counting-in-byteland/> [3D]

<https://devskill.com/CodingProblems/ViewProblem/300>

http://codeforces.com/contest/707/problem/E 7 [2D]

http://codeforces.com/contest/749/problem/E 8

http://codeforces.com/problemset/gymProblem/101055/D 5 [2D]

11240 UVA (4)

http://codeforces.com/contest/459/problem/D (4) //[NICE][SWEEPING]

http://codeforces.com/contest/61/problem/E (4) //[NICE][CLASSICAL][2\*FW][NORMALIZE]

http://codeforces.com/contest/669/problem/E 5 //fenwicks — sparse

http://codeforces.com/contest/777/problem/E 4 //MAXIMUM

<http://www.spoj.com/problems/TULIPNUM/> 4 //inc — 1 nor+num|sum(A[B],A[E])

http://codeforces.com/contest/799/problem/C 3 //MAX FW (but possibly easier?)

http://codeforces.com/contest/831/problem/E 4 //MAP to get ORDER — FW == LIST

<http://www.spoj.com/problems/SAS001/> (4) //Nice — number of inversions + 2P

<http://www.spoj.com/problems/TPGA/> (4) //NICE — Lesser\*(N-i-1)!

<http://www.spoj.com/problems/SGIFT/> (4) //BS works too

<http://www.spoj.com/problems/SUMSUM/> (5) //Bit-by-Bit cnt 0/1

<http://www.spoj.com/problems/AKVQLD03/> (3) //Classical fenwick — easy

<http://www.spoj.com/problems/ZIGZAG2/> (6) //Very nice — FW + BS + DP

http://codeforces.com/contest/849/problem/E (7) //2D Fenwick / ST+TP [NICE]

<http://www.spoj.com/problems/CRAYON/> (5) //VERY NICE [2\*FW — begin + end]

<http://www.spoj.com/problems/NITT8/> (4) //Norm. + Store indices in MAX-Fenwick [REVERSE] [VERY NICE]

<http://www.spoj.com/problems/DCEPC705/> (4) //NICE! Sort + Fenwick

<http://www.spoj.com/problems/DCEPC206/> (3) //NICE & EASY

<http://www.spoj.com/problems/KOPC12G/> (4) //N Fenwick trees

<http://www.spoj.com/problems/TRIPINV/> (4) //2xFenwick (triples counting)

http://codeforces.com/contest/597/problem/C (4) //[VERY NICE] 11\*Fenwick

http://codeforces.com/contest/12/problem/D (4) //NICE [triplet-comparing][sort]

<https://www.spoj.com/problems/NARHIL/> // Sugested by [**ak07\_**](http://cfrp.azurewebsites.net/profile/ak07_)

fft

<http://www.spoj.com/problems/ADAMATCH/>

<https://toph.co/p/play-the-lottery> (7) //[VERY NICE][D&C][MODULAR]

UVA 12633

6886 — Golf Bot [LA]

<http://www.spoj.com/problems/POLYMUL/en/>

Gym 100960C [2015-2016 Petrozavodsk Winter Training Camp, Nizhny Novgorod SU Contest]

<https://www.codechef.com/problems/APRPS>

<https://www.codechef.com/problems/POLYEVAL>

<http://www.spoj.com/problems/TSUM/> 5

13182 UVA 5 //ACTG hamming

http://codeforces.com/contest/827/problem/E (8) //MAGIC

<http://www.spoj.com/problems/MAXMATCH/> 5 //abc hamming

flow

<https://www.spoj.com/problems/DISGRAPH/> (5) //Stoer-Wagner [global]

<https://codeforces.com/gym/101981> [I](http://cfrp.azurewebsites.net/blog/entry/4) //[NICE]

http://codeforces.com/gym/101982 [E](http://cfrp.azurewebsites.net/blog/entry/5) //[VERY NICE][GRID]

http://codeforces.com/gym/101845/problem/F (4) //Basic

http://codeforces.com/gym/101908/problem/G (4) //[NICE][MATCHING][BS]

<http://www.spoj.com/problems/FASTFLOW/en/> //Raw (no sauce)

http://codeforces.com/contest/78/problem/E (5) //NICE [Matching-like][+BFS]

4322 — Destroying the bus stations (Live Archive)

11380 — Down Went The Titanic (UVA) //Interesting grid problem

6395 — Surely You Congest (LA) //VERY NICE [slightly advanced]

7204 — Blood groups (LA)

http://codeforces.com/gym/100963 (Flame of Nucleus — F)

11167 — Monkeys in the Emei Mountain //Also harder (imho)

http://codeforces.com/problemset/problem/653/D (+BS)

13000 — VIP Treatment (+BS)

1242 — Necklace

<https://www.deadline24.pl/assets/problemsets/dl24.elim.2017.B.en.pdf> (DEADLINE 24 problem — not sure if it can be submited :O)

3487 — Duopoly (Near matching)

http://codeforces.com/problemset/problem/727/D

http://codeforces.com/problemset/problem/704/D [Also advanced]

5418 — A Plug for UNIX (LA)

4957 — Fake scoreboard (LA) //If I remember well, other solutions was also possible

1155 — Power Transmission (LOJ) //(classical)

<https://www.codechef.com/problems/ROBOTDAG> //Ford-Fukherson

10804 — Gopher Strategy (UVA)

11506 — Angry Programmer (UVA) //Nodes division

10511 — Councilling (UVA)

563 — Crimewave

1306 — The K-League (UVA)

1345 — Jamie's Contact Groups

10092 — The Problem with the Problem Setter

Problem B. Roller Coaster Scheduling (GCJ — 2017)

259 — Software Allocation (UVA)

5905 — Pool construction (LA) //Imho harder

10480 — Sabotage

http://codeforces.com/contest/808/problem/F 6 //NICE — nontrivial (max matching with bigger flows)

http://codeforces.com/contest/847/problem/J 6 //Probably not flows — matching-like

flow-matching-like

<http://www.spoj.com/problems/ADAHOSE/> [DUAL-GRAPH]

http://codeforces.com/contest/903/problem/G (6) //[VERY NICE][SEG-TREE][CUT]

3837 [LA] //Stable Marriage

UVA 1175 //Stable Marriage

11594 — All Pairs Maximum Flow (UVA)

<http://www.spoj.com/problems/ADABLOOM/> //Maximum matching in general graph

11439 — Maximizing the ICPC //Maximum matching in general graph

1376 — Animal Run //Max flow on planar graph (Dual == shortest path over edges)

10989 — Bomb, Divide and Conquer //Stoer-Wagner — global cut

floyd-warshall

<https://codeforces.com/contest/1202/problem/B> (3) //[VERY NICE]

http://codeforces.com/gym/101845/problem/C

<https://www.urionlinejudge.com.br/judge/en/problems/view/2676> (3) //simple FW

10724 UVA

UVA 117

http://codeforces.com/problemset/problem/21/D

UVA 1198

LightOJ 1086 //+DP

<http://www.spoj.com/problems/INGRED/> //+DP

UVA 10048

UVA 125

Gym 101223C [2017 Facebook Hacker Cup, Round 1] //+DP

LightOJ 1221

UVA 423

UVA 12179 //+DP

UVA 1416

UVA 1233

UVA 10793

10099 UVA

UVA 869

LightOJ 1174

<http://www.spoj.com/problems/ARBITRAG/> //Other algos shall work too

13211 UVA (5) //NICE — FW adding states

<http://www.spoj.com/problems/ROHAAN/> (3) //Classical

http://codeforces.com/contest/25/problem/C (4) //Adding new edges .. need FW principal

http://codeforces.com/contest/33/problem/B (3) //NICE [dijkstra could work too]

friedvaldAlgorithm

<http://www.spoj.com/problems/ADAPOWER/>

4956 [LA]

game\_theory

<http://www.spoj.com/problems/ADAGAME/> [DP]

<http://www.spoj.com/problems/ADAGAME2/>

<http://www.spoj.com/problems/ADAGAME4/>

<http://www.spoj.com/problems/ADAQUBIC/>

<http://www.spoj.com/problems/ADAFIMBR/>

<http://www.spoj.com/problems/ADAGAME5/>

<http://www.spoj.com/problems/ADAXMAS/>

<https://codeforces.com/contest/1194/problem/D> (4) //[OBSERVATIO][PATTERN]

<https://codeforces.com/contest/1191/problem/E> (4) //[NICE][IF][OBSERVATION]

<https://www.spoj.com/problems/HUSGAME/> (4) //[VERY NICE][OBSERVATION][RECURSION]

<https://www.spoj.com/problems/IBIGAME/> (5) //[VERY NICE][OBSERVATION][DP]

<https://codeforces.com/gym/101981> [A](http://cfrp.azurewebsites.net/blog/entry/4)

<https://codeforces.com/gym/102058/problem/F> (4) //[NICE]

http://codeforces.com/gym/101801 [F](http://cfrp.azurewebsites.net/blog/entry/3) //[EASY][OBSERVATION][NIM]

http://codeforces.com/gym/101908/problem/B (5) //[NICE][NIMBERS][OBSERVATION]

http://codeforces.com/gym/101873 [H](http://cfrp.azurewebsites.net/blog/entry/5) //[VERY NICE][OBSERVATION][TREE] //LUP LUP

http://codeforces.com/gym/101808/problem/I (5) //[DP][OBSERVATION]

http://codeforces.com/contest/919/problem/F (6) //[NICE][HARD][IMPLE][BFS][TOPO][GRAF]

http://codeforces.com/contest/918/problem/D (4) //[NICE][DAG][DP][TREE]

http://codeforces.com/contest/914/problem/B (3) //[NICE][EASY][OBSERVATION][PARITY]

http://codeforces.com/contest/148/problem/D (4) //[EASY][DP][PROBABILITY]

http://codeforces.com/contest/138/problem/D (7) //[VERY NICE][DP][OBSERVATION]

<https://arc087.contest.atcoder.jp/tasks/arc087_c> (5) //[VERY NICE][TRIE][SEQUENCE]

http://codeforces.com/contest/120/problem/E (3) //[NICE][SYMETRY][PARITY]

http://codeforces.com/contest/88/problem/E (5) //[VERY NICE][PREFIX-XOR][NIMBERS][SQRT][MATH]

http://codeforces.com/contest/69/problem/D (4) //[NICE][DP] Reflection can be ignored

http://codeforces.com/contest/55/problem/C (4) //[NICE] Size of piece from border

http://codeforces.com/problemset/problem/255/E --MEX

<https://devskill.com/CodingProblems/ViewProblem/91>

<https://devskill.com/CodingProblems/ViewProblem/364>

Project Euler #96: Su Doku //Sudoku

11859 UVA 4

11863 UVA 4

11892 UVA 3 //Probably solved by many

11534 UVA 5

<http://www.spoj.com/problems/VECTAR11/> 4 //Nsqrt(N) passes [with break]

http://codeforces.com/contest/768/problem/E (4) //NICE — Grundy

<http://www.spoj.com/problems/SYNC13C/> (4) //2\*DP {maybe not seeing sth}

http://codeforces.com/contest/787/problem/C (4)

http://codeforces.com/contest/794/problem/C (3) //Find optimal strategy: choose back/front

http://codeforces.com/contest/794/problem/E (5) //NICE Find stategy: Even/Odd

http://codeforces.com/contest/812/problem/E (7) //Advanced NIM strategy

<http://www.spoj.com/problems/GAMEMVS/> (4) //Nimbers (Ai^X)<=Ai

<http://www.spoj.com/problems/PLAYGAME/> (3) //Check pattern

<http://www.spoj.com/problems/CHAOS_CC/> (4) //VERY NICE [nimbers]

http://codeforces.com/contest/851/problem/E (5) //Very nice [nimbers] [bitset]

<http://www.spoj.com/problems/CHGROOM/> (4) //+Factorisation [NICE & Easy]: Win unless 2 prime factors

<http://www.spoj.com/problems/EALP1/> (4) //NICE ~ Possible Moves of NIM

<http://www.spoj.com/problems/GAME3/> (4) //VERY NICE — pattern watching [A145812]

<http://www.spoj.com/problems/GAME2/> (5) //VERY NICE — <https://community.topcoder.com/tc?module=Static&d1=match_editorials&d2=srm338> (CAKE)

<http://www.spoj.com/problems/CF36D/> (5) //Pattern watching (care for 1)

http://codeforces.com/contest/15/problem/C (4) //VERY NICE [XOR: A,1,A+1,0..repeat]

http://codeforces.com/contest/39/problem/E (4) //Slightly [DP][MATH][ROUNDING]

http://codeforces.com/contest/63/problem/E (5) //[NICE][BITMASK-DP]

gauss

12910 — Snakes and Ladders [UVA]

UVA 10828

http://codeforces.com/gym/100923/problem/C

4963 [LA]

UVA 12849

Gym 100962A [2015-2016 Petrozavodsk Winter Training Camp, Moscow SU Trinity Contest][NICE]

UVA 10109 [NICE][HARD-WORK]

geometry

<http://www.spoj.com/problems/ADAPICK/>

<http://www.spoj.com/problems/ADAKOHL/>

<https://codeforces.com/contest/1199/problem/B> (2) //[NICE][SIMPLE]

<https://codeforces.com/gym/101991/problem/J> (5)

<https://codeforces.com/gym/101991/problem/B> (4)

<https://codeforces.com/gym/102006/problem/I> (4) //[VERY NICE][ITERATIVE]

<https://codeforces.com/gym/102021> [G](http://cfrp.azurewebsites.net/blog/entry/8) //[3D]

<https://codeforces.com/gym/102021> [B](http://cfrp.azurewebsites.net/blog/entry/5) //Path on circle

<https://codeforces.com/gym/102058/problem/K> (3) //[BRUTE-FORCE][FAIL]

<https://codeforces.com/gym/101972/problem/C> (4) //[EASY][POINTS DISTANCE][IMAGE]

http://codeforces.com/gym/101628/problem/C (4) //Circles intersection

http://codeforces.com/gym/100112 [G](http://cfrp.azurewebsites.net/blog/entry/4) //Special cases and so on — imple.

http://codeforces.com/gym/101982 [G](http://cfrp.azurewebsites.net/blog/entry/3) //[DISTANCE]

http://codeforces.com/gym/101845/problem/D (5) //[VERY NICE][POLYGON][MO]

http://codeforces.com/gym/101962/problem/G (5) //Angles

http://codeforces.com/gym/101908/problem/K

http://codeforces.com/gym/101879/problem/A (7)

http://codeforces.com/gym/101873 [G](http://cfrp.azurewebsites.net/blog/entry/4) //Pick's Theorem

http://codeforces.com/gym/101873 [A](http://cfrp.azurewebsites.net/blog/entry/6) //heavy implementation

http://codeforces.com/gym/101808/problem/E (5)

http://codeforces.com/gym/101808/problem/A (2)

http://codeforces.com/gym/101666 [A](http://cfrp.azurewebsites.net/blog/entry/4)

http://codeforces.com/gym/101726/problem/J (5) //[NICE][INTERSECTION][DS]

http://codeforces.com/gym/101650 [F](http://cfrp.azurewebsites.net/blog/entry/6) //Polygons + circles

http://codeforces.com/gym/101650 [H](http://cfrp.azurewebsites.net/blog/entry/2) //A few if's

http://codeforces.com/contest/908/problem/C (3)

http://codeforces.com/contest/140/problem/A (3) //Circles around bigger circle

http://codeforces.com/contest/136/problem/D (4) //+Brute-Force

http://codeforces.com/contest/127/problem/A (1) //Points distance

http://codeforces.com/gym/101597/problem/B (3) //Simply brute-force just the closest to points

http://codeforces.com/contest/70/problem/D (5) //Dynamic Convex Hull

<https://icpc.kattis.com/problems/airport> //Proposed by [**Tutis**](http://cfrp.azurewebsites.net/profile/Tutis)

<https://www.hackerrank.com/contests/world-codesprint-7/challenges/elastic-rope/problem>

<https://devskill.com/CodingProblems/ViewProblem/20> [EASY]

UVA 10321 //Polygon intersection

UVA 11265 //Polygon point +/-

UVA 13112 //Polygon

10907 UVA [Area of polygon from a point]

3378 — Swamp Things [LA] — Maximum points on line

UVA 11768 //Discrete points

2542 [LA] //Arc size [formula]

UVA 1571 //As below [easier]

<https://www.codechef.com/problems/ALLPOLY> //[NICE] Point seeing polygon

<http://www.spoj.com/problems/IITKWPCL/> //Point distance

UVA 11281 //circle~triangle

UVA 12921 //circle reconstruction

UVA 190 //circle from 3 points

UVA 12240 //pts>circle

UVA 438 //circle pt

LightOJ 1018 //Minimum # of lines through all pts [VERY NICE]

UVA 11008 //Similar as above

UVA 12830 //Biggest rectangle without points inside

UVA 11012 //Most distant points

UVA 1683 //Closest points

UVA 12389 //3D MH Closest Points

<http://www.spoj.com/problems/AMR12C/> //Pt closest to all other points

<http://www.spoj.com/problems/CLOPPAIR/> //Closest pair of points

UVA 10678 //Circles intersection

http://codeforces.com/problemset/problem/600/D //Circles intersection

LightOJ 1118 //Circles intersection

<http://www.spoj.com/problems/CERC07C/en/> //Bounding circle

UVA-10005 //Bounding circle

2407 [LA] //Bounding Sphere

LightOJ 1120 //Rectangle's Union

LightOJ 1130 //Circle x Rectangle intersection

UVA 11177 //Circle x Convex Polygon

http://codeforces.com/problemset/problem/610/D //Lines intersections (axes parallel)

6263 [LA] //Pt in areas

LightOJ 1058 //# parallelograma

UVA 12931 //Common area of polygons

UVA 10301 //Intersecting circles

UVA 453 //Circles intersection

http://codeforces.com/problemset/problem/681/E //Circles intersection

UVA 920 //Lines intersecion (etc..)

UVA 12556

http://codeforces.com/problemset/problem/793/C //Intersection ans similar

UVA 11343 //Intersection of segments

UVA 866 //Intersection of segments

Gym 100190I [2011 ACM-ICPC East Central North America (ECNA 2011)] //Segment intersection

http://codeforces.com/gym/100917/problem/K

UVA 11686 //Segment intersection

LightOJ 1388

UVA 833 //Segment intersection

LightOJ 1196 //Points sides

UVA 10167 //Points sides

UVA 12818 //Arc & Point distance

<http://www.spoj.com/problems/SICRANO/> //Point-line distance

http://codeforces.com/problemset/problem/614/C //Point-line distance

UVA 13117 //Point-line distance

UVA 12483 //Point-line distance

UVA 12173 //Point-line distance

UVA 10075 //Point distance on sphere

<https://www.hackerrank.com/contests/booking-hackathon/challenges/nearby-attractions/problem> //Pt sphr

UVA 535 //Point distance on sphere

UVA 10897 //Sphere tavelling

UVA 11817 //Sphere travelling

UVA 10316 //Sphere travelling

UVA 1469 //Fractions distance 3D

11930

12173 UVA 3

12194 UVA 4

11894 UVA 3

11769 UVA 7

11665 UVA 5

11509 UVA 4

11355 UVA 5

11265 UVA 6 //Nice one | polygon — cut/pt-check/area

11123 UVA 4 //Counting trapezoids

11177 UVA 6 //BS+Polygon/Circle intersection

11186 UVA 3

11008 UVA 5 //with DP → #intersected triangles

11012 UVA 5 //Nejvzdálenější body (Manhatton 3D)

11072 UVA 4 //Points v poly gonu

http://codeforces.com/problemset/problem/682/E 6 (biggest triangle)

http://codeforces.com/contest/672/problem/C 4 //easy — just think it up

http://codeforces.com/contest/667/problem/A 2 //vzorecky

http://codeforces.com/contest/793/problem/C 5 //EASY but beware of epsilons (NICE)

http://codeforces.com/contest/794/problem/B 2 //Can be done with BS

http://codeforces.com/contest/814/problem/D 5 //+DP on trees (NICE — but low geom.)

10750 UVA 3 //Closest points — try all pairs

http://codeforces.com/contest/820/problem/B 3 //Polygon angle find!

13213 UVA 5 //VERY NICE — Voronoi diagram (low constraints so not actually needed)

13215 UVA 3 //EASY — Sum areas and find side lengths

<http://www.spoj.com/problems/IITKWPCC/> (5) //VERY VERY NICE — Nqrt(N)log(N)

<http://www.spoj.com/problems/NNS/> (5) Closest points query [fake geometry] {\_\_128}[NICE]

http://codeforces.com/contest/849/problem/B (3) //X-Product — side

<http://www.spoj.com/problems/AMR12C/> (5) //Point closest to all other points (with speed)

<http://www.spoj.com/problems/SICRANO/> (3) //Line-Point distance

<http://www.spoj.com/problems/VCIRCLES/> (5) //Heavy geometrical \*\*\*\*

<http://www.spoj.com/problems/CIRU/> (5) //Same as above [yet bigger constraints]

<http://www.spoj.com/problems/THREETW1/> (4) //Fermat point search

<http://www.spoj.com/problems/CLOPPAIR/> (4) //Closest pair of points

<http://www.spoj.com/problems/MAXLN/> (2) //Some basic (4\*r^2+.25)

<http://www.spoj.com/problems/KOLICA/> (4) //VERY NICE [nasty iff party]

http://codeforces.com/contest/598/problem/C (4) //NICE ~ Precision! [ld]

http://codeforces.com/contest/18/problem/A (2) //EASY&FINE: Pythagoreas

http://codeforces.com/contest/40/problem/A (2) //[EASY][DISTANCE]

graph

<https://codeforces.com/contest/1182/problem/D> (5) //[CENTRUM][OBSERVATION][DFS]

<https://www.urionlinejudge.com.br/judge/en/problems/view/1562> (5) //[VERY NICE][QUEUE][GREEDY]

<https://codeforces.com/gym/102001/problem/G> (4) //[NICE][BS]

<https://codeforces.com/gym/102006/problem/G> (4) //[NICE][TOPOSORT][OBSERVATION]

<https://codeforces.com/gym/102058/problem/B> (6) //[VERY NICE][REVERSE-SIMULATION]

http://codeforces.com/gym/101666 [H](http://cfrp.azurewebsites.net/blog/entry/6) //[VERY NICE][GEOMETRY][DSU][EULER][PLANAR]

2827 [LA] //3Coloring

2243 [LA] //3Coloring

5603 [LA] //Coloring

UVA 1658 //Shortest paths

UVA 12821 //Shortest paths

http://codeforces.com/contest/27/problem/D (5)

11387 (UVA) 4

<http://www.spoj.com/problems/VFRIEND2/> (5) //Graph possible check

http://codeforces.com/contest/859/problem/E (4) //VERY NICE (2 cases: CYCLE [x2] / TREE [x(Size+1)]

http://codeforces.com/contest/847/problem/C (2) //Forest making Easy&Nice

http://codeforces.com/contest/863/problem/C (3) //Cycle in states

greedy

<https://codeforces.com/contest/1175/problem/D> (4) //[NICE][PREPROCESS][SORTING]

<https://codeforces.com/contest/1203/problem/F1> (5) //[VERY NICE][SORTING][CASES]

<https://codeforces.com/contest/1176/problem/C> (3) //[SIMPLE][NICE][IMPLE]

<https://codeforces.com/contest/1176/problem/A> (1)

<https://codeforces.com/contest/1200/problem/B> (3) //[SIMPLE][NICE]

<https://codeforces.com/contest/1202/problem/A> (2)

<https://codeforces.com/contest/1181/problem/B> (4) //[BIG]

<https://atcoder.jp/contests/abc136/tasks/abc136_c> (3) //[REVERSE]

<https://codeforces.com/contest/1185/problem/C2> (4)

<https://codeforces.com/contest/1185/problem/B> (2) //[EASY]

<https://codeforces.com/contest/1180/problem/B> (3) //[IMPLEMENTATION][MATH]

<https://codeforces.com/contest/1186/problem/D> (4) //[IMPLEMENTATION]

<https://codeforces.com/contest/1197/problem/B> (3) //[BITONIC][IMPLE]

<https://www.spoj.com/problems/DIVSTR/> (3) //[CUTE]

<https://codeforces.com/gym/101981> [E](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][NORMALIZE][STACK]

<https://codeforces.com/gym/102001/problem/L> (3) //[BITS]

<https://toph.co/p/passwords> (2)

http://codeforces.com/gym/100112 (3) //[FINE]

http://codeforces.com/gym/101666 (4) //[NICE][QUEUE][DFS][EVENTS]

http://codeforces.com/gym/101806/problem/T (5) //[NICE][SEGMENT TREE][SORTING]

http://codeforces.com/contest/916/problem/B (3) //[BITS] FIRST & LAST

http://codeforces.com/contest/913/problem/C (3) //+[DP]

http://codeforces.com/contest/146/problem/D (4) //[NICE][CONSTRUCTION]

http://codeforces.com/contest/146/problem/C (3) //[EASY][NICE][TWO-CASES]

http://codeforces.com/contest/139/problem/D (4) //Muchas cases

7887 — Back to the Future (4) //[NICE][2\*HEAP]

http://codeforces.com/contest/910/problem/A (2) //DP works too

http://codeforces.com/contest/125/problem/D (4) //Limited possibilities

http://codeforces.com/contest/902/problem/A (2) //Just keep last

http://codeforces.com/contest/898/problem/D (4) //[NICE][FENWICK/OR/PREFIX]

http://codeforces.com/contest/118/problem/C (4) //[NICE] — Try each digit (iterate in waves)

7706 — Pokemons (2) //Sweep and keep maximum

http://codeforces.com/contest/893/problem/D (4) //[NICE][SWEEP][MAXIMUM] — pay max when have to

http://codeforces.com/contest/892/problem/B (2) //Sweep from back

http://codeforces.com/contest/102/problem/C (3) //sort by frequency

<https://devskill.com/CodingProblems/ViewProblem/419> (2) //Sweep from back

http://codeforces.com/contest/890/problem/C (3) //frequence

http://codeforces.com/contest/890/problem/B (2) //[EASY][REVERSE]

http://codeforces.com/contest/888/problem/B (2) //Equalize U/D and L/R

http://codeforces.com/gym/101597/problem/J (4) //[NICE][SWEEP]

http://codeforces.com/contest/76/problem/B (4) //[NICE][MATCHING-LIKE][2PTRS]

http://codeforces.com/contest/73/problem/B (4) //[IMPLEMENTATION][SORTING]

http://codeforces.com/contest/883/problem/K (4) //Nice — Two sweeps

http://codeforces.com/contest/49/problem/D (3) //NICE — 1010101 OR 0101010 [haming]

http://codeforces.com/contest/58/problem/C (4) //NICE — group trees by slopes

http://codeforces.com/contest/729/problem/D 3

http://codeforces.com/contest/729/problem/E 4

http://codeforces.com/contest/725/problem/D 4

http://codeforces.com/contest/725/problem/F 9

http://codeforces.com/contest/732/problem/E 5

http://codeforces.com/contest/727/problem/F 6

http://codeforces.com/contest/724/problem/D 5

http://codeforces.com/contest/723/problem/C 4

http://codeforces.com/contest/719/problem/B 2

http://codeforces.com/contest/712/problem/C 3

13152 UVA (4)

http://codeforces.com/contest/746/problem/E 5

http://codeforces.com/contest/746/problem/D 3

http://codeforces.com/contest/749/problem/C 3

11737 UVA (3)

11786 UVA (4)

11630 UVA (5)

11563 UVA (4)

11491 UVA (4)

11330 UVA (3)

11089 UVA (2)

http://codeforces.com/contest/884/problem/D (4) //PQ or Sort

<http://www.spoj.com/problems/SQRMINSUM/> 3 //solve-able in O(N+M)-arrayqueue

<http://www.spoj.com/problems/MSCHED/> 3 //sweep from back

<http://www.spoj.com/status/ns=18780683> 4 //all perm + A<B<C works

<http://www.spoj.com/problems/NINJA7/> (3) //sort by diff

<http://www.spoj.com/problems/NINJA2/> (4) //try all possib. (26)

http://codeforces.com/contest/767/problem/E (6)

http://codeforces.com/contest/637/problem/B (3) //NICE pro prvaky

http://codeforces.com/contest/777/problem/B (3) // -||-

http://codeforces.com/contest/777/problem/D (3) //just go from end

http://codeforces.com/contest/779/problem/C (3) //NICE pro prváky

<http://www.spoj.com/problems/SPCU/> (2) //Easy — zamysleni (max int = index)

<http://www.spoj.com/problems/LOPOV/> (4) //sort + queue (or just queue) NICE

http://codeforces.com/contest/792/problem/E (5) //T%S<=T/S + check proper

http://codeforces.com/contest/807/problem/E (5) //NICE — put asice P2 / rest — greedy from small

http://codeforces.com/contest/799/problem/E (5) //Many queues — but NICE

http://codeforces.com/contest/808/problem/C (3) //EASY

http://codeforces.com/contest/802/problem/B (4) //Priority by "next"

10850 UVA (4) //Queue a brute-force

http://codeforces.com/contest/813/problem/A (1) //Zahrivacka pro prvaky

10716 UVA (4) //NICE — always find closest pair

http://codeforces.com/contest/816/problem/C (3) //NICE — greater<lesser side

http://codeforces.com/contest/820/problem/D (5) //VERY NICE — O(N) -~- 5 events per number

http://codeforces.com/contest/818/problem/B (2) //Zahrivacka pro prvaky

http://codeforces.com/contest/822/problem/C (4) //Almost classical Sort+Queue

http://codeforces.com/contest/825/problem/C (2) //Nice & Easy

http://codeforces.com/contest/825/problem/D (3) //Update by modulo

http://codeforces.com/contest/835/problem/B (2) // Zahhrivacka pro prvaky

http://codeforces.com/contest/839/problem/B (3) //Nasty iffs — yet nice excersize

<http://www.spoj.com/problems/PCPC12I/> (4) //Swipe MINIMUM from left/right [10^6-A[i] trick]

<http://www.spoj.com/problems/AMR12I/> (3) //NICE a) MAX\_SEG>=K b) (SEG\_SIZE-1)/K+1

<http://www.spoj.com/problems/BUSYMAN/> (2) //NICE&EASY — Sort + keep minimum

http://codeforces.com/contest/861/problem/C (3) //2+ but not same

<http://www.spoj.com/problems/WORKB/> (3) //Simple "min" formula for each neighbor

http://codeforces.com/contest/864/problem/D (4) //VERY NICE — Frequency + unused

<http://www.spoj.com/problems/ROADTRIP/> (4) //VERY NICE — Keeping last lesser

http://codeforces.com/contest/597/problem/B (3) //NICE [Classical]

<http://www.spoj.com/problems/SHLIGHTS/> (4)

<http://www.spoj.com/problems/MLK/> (3) //VERY NICE — Sum all prefix sums

http://codeforces.com/contest/867/problem/C (4) //NICE [IMPLE][2POINTERS][MID+EPS]

http://codeforces.com/contest/867/problem/E (5) //NICE [EASY-IMPLE][HARD-CONS]

http://codeforces.com/contest/18/problem/D (4) //+Big Integer

http://codeforces.com/contest/276/problem/D (4) //NICE — Find first mismatch bit (then 111...111)

http://codeforces.com/contest/3/problem/B (4) //Divide 1/2 [sort][2pointers]

http://codeforces.com/contest/3/problem/D (4) //?==) ..if open < 0: set max A-B to (

http://codeforces.com/contest/26/problem/B (4) // +1 ( | -1 ): -1, erase .. erase sum in the end

http://codeforces.com/contest/33/problem/A (2) //EASY [long-statement]

http://codeforces.com/contest/44/problem/E (2) //Try mins then try maxs

http://codeforces.com/contest/45/problem/D (3) //Priority-queue+'sort'

hash

<http://www.spoj.com/problems/ADACLEAN/>

<https://codeforces.com/contest/1200/problem/E> (5) //[VERY NICE]

http://codeforces.com/gym/101808/problem/B (5) //[NICE][NUMBERS]

http://codeforces.com/gym/101741/problem/K (5) //[NICE][SQRT][PATTERN MATCHING]

7979 — Red Rover (3) //[NICE] //Many other ways to solve

http://codeforces.com/contest/898/problem/F (5) //[VERY NICE]//Hash by 10

http://codeforces.com/contest/114/problem/D (4) //[NICE] //N^2Log(N) might/might-not be OK

<https://www.urionlinejudge.com.br/judge/en/problems/view/1503> (7) //[NICE][BS][OPTI]

Gym 101466E [2017 ACM-ICPC, Universidad Nacional de Colombia Programming Contest][NICE]

12012 UVA 4

http://codeforces.com/contest/727/problem/E 7

http://codeforces.com/contest/718/problem/D 8

11855 UVA 4

http://codeforces.com/contest/752/problem/D 5

http://codeforces.com/contest/825/problem/F 5 //String + Periods

http://codeforces.com/contest/835/problem/D 4 //Palindromes

<http://www.spoj.com/problems/CF25E/> (5) //VERY NICE [IMPLE>CONCEPT]

http://codeforces.com/contest/7/problem/D (4) //Palindromes

http://codeforces.com/contest/19/problem/C (4) //[NICE]: Not a string

hull

13300 — Ghost Hunting (4) //Hull + calipers passes

http://codeforces.com/gym/101982 [M](http://cfrp.azurewebsites.net/blog/entry/7) //[VERY NICE][HULL][TERNARY]

<http://www.spoj.com/problems/GARDENHU/en/>

2453 — Wall [LA]

UVA 13213

UVA 11096

Gym 100792G [2015-2016 ACM-ICPC, NEERC, Moscow Subregional Contest]

<https://www.codechef.com/problems/KTHCON>

http://codeforces.com/problemset/problem/605/C

UVA 218

UVA 11072

11168 UVA

UVA 12307

Gym 100963I [2007-2008 Summer Petrozavodsk Camp, Japanese Contest, 2007-08-29]

UVA 11243

UVA 10256

109 SCUD Busters

UVA 13024 [NICE]

UVA 10002

Gym 100886H [2015-2016 Petrozavodsk Winter Training Camp, Saratov SU Contest]

UVA 1139

UVA 681

UVA 811

<https://www.codechef.com/problems/MGCHGEOM>

UVA 11769 //3D

chess

<http://www.spoj.com/problems/ADACHESS/>

<http://www.spoj.com/problems/ADACHES2/>

<http://www.spoj.com/problems/ADAGAME5/>

<http://www.spoj.com/problems/ADAXMAS/>

URI 1100 (3) //[BFS][KNIGHT]

<https://www.codechef.com/problems/CHQUEENS> (4) //[IMPLEMENTATION]

http://codeforces.com/contest/131/problem/E (4) //[NICE][STL][SORTING][QUEEN]

http://codeforces.com/contest/42/problem/B (4) //NICE — Checkmate check

UVA 10748 //bfs

LightOJ 1143

<https://www.hackerearth.com/practice/algorithms/greedy/basics-of-greedy-algorithms/practice-problems/algorithm/harry-and-ron-play-a-game-of-chess/>

UVA 10196

<http://www.spoj.com/problems/KNMOVE/>

UVA 11352

<http://www.spoj.com/problems/NAKANJ/> //BFS

LightOJ 1010

LightOJ 1171 //MM

2883 LA

UVA 439

<http://www.spoj.com/problems/TRKNIGHT/> //??

2308 LA

<http://www.spoj.com/problems/CCHESS/> //Dijkstra

http://codeforces.com/problemset/problem/630/H //[ROOKS][BIG][COMBINATORICS]

LightOJ 1005 //As above (but with real rooks)

LightOJ 1061 //Placing queens

UVA 11085

UVA 10094 //Queen placing [NICE][PATTERN]

<https://devskill.com/CodingProblems/ViewProblem/383>

11852 UVA (6)

<http://www.spoj.com/problems/KLUG1/> (2) //Jumps of horse

<http://www.spoj.com/problems/CODESPTD/> (5) //VERY NICE — DP [Queens]

http://codeforces.com/contest/3/problem/A (2) //Imple — Shortest path for king

http://codeforces.com/contest/38/problem/B (2) //NICE — Simple possition checking

implementation

<https://codeforces.com/contest/1203/problem/A> (1) //[FOR][IF]

<https://codeforces.com/contest/1173/problem/A> (1) //[IF]

<https://devskill.com/CodingProblems/ViewProblem/584> (2)

<https://codeforces.com/contest/1200/problem/A> (2) //[EASY]

<https://codeforces.com/contest/1182/problem/B> (3) //[PRACTICE][GRID-MOVEMENT]

<https://codeforces.com/contest/1201/problem/B> (2)

<https://codeforces.com/contest/1201/problem/A> (2)

<https://atcoder.jp/contests/abc136/tasks/abc136_a> (1)

<https://codeforces.com/contest/1199/problem/A> (2) //[BF]

<https://codeforces.com/contest/1183/problem/A> (2)

<https://codeforces.com/contest/1186/problem/A> (1)

<https://codeforces.com/contest/1178/problem/A> (2)

<https://codeforces.com/contest/1195/problem/A> (2) //[ARRAY]

<https://codeforces.com/contest/1191/problem/A>

<https://www.spoj.com/problems/MOZHSLM/> (3) //Two sweeps

<https://www.spoj.com/problems/OVGDEL/> (3) //[NICE][EASY][FREQUENCY]

<https://www.spoj.com/problems/VTV1001/> (2)

<https://codeforces.com/gym/102001/problem/D> (3)

<https://codeforces.com/gym/102021> (7) //[VERY NICE]Easy thought but nice imple

<https://codeforces.com/gym/101972/problem/E> (2)

<https://codeforces.com/gym/101972/problem/D> (1) //[IF]

<https://toph.co/p/full-pyramid> (1)

http://codeforces.com/gym/101992/problem/F (2)

http://codeforces.com/gym/101628/problem/D (3)

http://codeforces.com/gym/100112 [I](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][STRINGS][BITMASKS]

http://codeforces.com/gym/100112 [A](http://cfrp.azurewebsites.net/blog/entry/2)

http://codeforces.com/gym/101982 [L](http://cfrp.azurewebsites.net/blog/entry/3)

http://codeforces.com/gym/101982 [J](http://cfrp.azurewebsites.net/blog/entry/2)

http://codeforces.com/gym/101982 [A](http://cfrp.azurewebsites.net/blog/entry/2) //Easy

http://codeforces.com/gym/101845/problem/H (2) //Dates

http://codeforces.com/gym/101801 [L](http://cfrp.azurewebsites.net/blog/entry/1)

http://codeforces.com/gym/101801 [A](http://cfrp.azurewebsites.net/blog/entry/1)

http://codeforces.com/gym/101864 [C](http://cfrp.azurewebsites.net/blog/entry/2) //Easy

http://codeforces.com/gym/101889/my [H](http://cfrp.azurewebsites.net/blog/entry/2)

http://codeforces.com/contest/934/problem/A (2)

http://codeforces.com/contest/922/problem/A (2) //Iff-party

http://codeforces.com/contest/914/problem/A (1)

http://codeforces.com/contest/916/problem/A (2) //Time

http://codeforces.com/contest/915/problem/B (2) //Formula / Iff

http://codeforces.com/contest/915/problem/A (1)

http://codeforces.com/contest/913/problem/A (1)

http://codeforces.com/contest/912/problem/A (2) //Easy [corner-cases]

http://codeforces.com/contest/146/problem/A (1)

<http://www.spoj.com/problems/ESYR/> (1) //Bad one :/

http://codeforces.com/contest/908/problem/A (1)

http://codeforces.com/contest/147/problem/A (2) //Parsing

http://codeforces.com/contest/139/problem/A (1)

http://codeforces.com/contest/137/problem/A (1)

7886 — Assigning Teams (2) //[EASY][SORTING]

http://codeforces.com/contest/133/problem/A (1)

http://codeforces.com/contest/134/problem/A (1)

http://codeforces.com/contest/131/problem/A (1)

http://codeforces.com/contest/127/problem/B (1) [EASY]

http://codeforces.com/contest/899/problem/B (2) //Dates

http://codeforces.com/contest/898/problem/C (3) //No thinking — just implementation

6157 How do spiders walk on water? (4) //boring problem

http://codeforces.com/contest/900/problem/A (1)

7613 Relative atomic mass (1)

http://codeforces.com/contest/122/problem/A (1) //Find all lucky

http://codeforces.com/contest/120/problem/B (1) //Iteration

http://codeforces.com/contest/120/problem/A (1) //Iff/Logic

http://codeforces.com/contest/118/problem/B (2) //[PRINTING]

http://codeforces.com/contest/108/problem/A (1) //[EASY][PRINT][TIME]

http://codeforces.com/contest/106/problem/A (1) //Iff-party

http://codeforces.com/contest/890/problem/A (1) //if or perm

http://codeforces.com/contest/90/problem/B (2) //No idea — just cycles

http://codeforces.com/contest/75/problem/A (1) //conversion

http://codeforces.com/contest/884/problem/B (1) //Simple sum +N-1

http://codeforces.com/contest/48/problem/A (1) //Very easy [fe. perm]

http://codeforces.com/contest/51/my (2) //Lex-lowest-string (|s|=4)

http://codeforces.com/contest/54/problem/A (2) //Single sweep

http://codeforces.com/contest/719/problem/C 3

http://codeforces.com/contest/747/problem/E 4

http://codeforces.com/contest/754/problem/C 5

11482 UVA (4)

11291 UVA (3)

11070 UVA (5) //evaluation of expression

11074 UVA (2)

http://codeforces.com/contest/678/problem/B 2 //calendar days

http://codeforces.com/contest/643/problem/A 3 //easy — just simulate

http://codeforces.com/contest/770/problem/D 5 //easy — but pain — draw

http://codeforces.com/contest/789/problem/B 3 //simulate (mby twice)

13171 UVA (1)

10800 UVA (3)

http://codeforces.com/contest/828/problem/B 2 //EASY & NICE — just analysis

http://codeforces.com/contest/825/problem/B 2 //EASY & NICE — Piskvorky — pro prvaky

http://codeforces.com/contest/837/problem/B 2 //Just implementation

http://codeforces.com/contest/837/problem/C 2 //Some nasty iffs

http://codeforces.com/contest/845/problem/B 2 //Easy pro prvaky

http://codeforces.com/contest/845/problem/D 3 //Iffs — folow the rules

<http://www.spoj.com/problems/UNIHW/> 4 //NICE (but many iffs)

http://codeforces.com/contest/5/problem/A (2) //Parsing

http://codeforces.com/contest/5/problem/B (3) //Output formating

http://codeforces.com/contest/6/problem/B (2) //Simply check by adjancecy vector + set

http://codeforces.com/contest/7/problem/A (1) //Oneinteresting if

http://codeforces.com/contest/8/problem/B (2) //Sample vectors + set/or/array

http://codeforces.com/contest/12/problem/A (1) //Find mirror by middle

http://codeforces.com/contest/16/problem/A (1) //Easy check of chars

http://codeforces.com/contest/21/problem/A (2) //Easy but nasty iff imple

http://codeforces.com/contest/31/problem/B (2) //Boring imple

http://codeforces.com/contest/34/problem/C (3) //Easy string parsing

http://codeforces.com/contest/41/problem/C (2) //String parsing

inclusion-exclusion

<http://www.spoj.com/problems/KPRIMESB/> (4)

<http://www.spoj.com/problems/IITKWPCH/> (4) //NICE — on bits

<http://www.spoj.com/problems/SUBSET/> (5) //VERY NICE — 3^10 (^2 but not exactly) (+ sorting)

<https://www.hackerearth.com/practice/algorithms/dynamic-programming/bit-masking/practice-problems/algorithm/special-pairs-7/description/> (4) //NICE [Brute-force]

interactive

http://codeforces.com/contest/897/problem/D (4) //NICE! Back/Front

http://codeforces.com/contest/727/problem/C (2)

http://codeforces.com/contest/810/problem/D (4) //BS \* 3 (same)

http://codeforces.com/contest/811/problem/D (4) //BFS — easy .. some ifs

http://codeforces.com/contest/835/problem/E (4) //NICE! Bitsets + Detect + XOR

http://codeforces.com/contest/844/problem/D (5) //NICE! Randomized algo

http://codeforces.com/contest/862/problem/D (4) //NICE! Find first + Binary Search

http://codeforces.com/contest/872/problem/D (4) //NICE! [no clue why it passed]

isomorphism

<https://www.urionlinejudge.com.br/judge/en/problems/view/2652> (4) //Tree isomorphism

UVA 12489 //Tree

<https://www.hackerrank.com/contests/hourrank-16/challenges/jenny-subtrees/problem>

<http://www.spoj.com/problems/DSUBTREE/> (5) //Isomorphism on trees (try all subsets)

<http://www.spoj.com/problems/TREEISO/> (4) //Simple isomorphism of trees

josephus

http://codeforces.com/gym/101864 [A](http://cfrp.azurewebsites.net/blog/entry/4) //Observation

UVA 151

UVA 1394

UVA 440

3803 [LA]

<https://www.urionlinejudge.com.br/judge/en/problems/view/1030>

UVA 12912

UVA 13114

11351 UVA (2)

<http://www.spoj.com/problems/CLSLDR/> (4) //Muchas queries — go DP

<http://www.spoj.com/problems/WTK/> (Easy) Suggested by [**codeforrest**](http://cfrp.azurewebsites.net/profile/codeforrest)

<http://www.spoj.com/problems/DANGER/> (Easy) Suggested by [**codeforrest**](http://cfrp.azurewebsites.net/profile/codeforrest)

<http://www.spoj.com/problems/POCRI/> (Medium) Suggested by [**codeforrest**](http://cfrp.azurewebsites.net/profile/codeforrest)

<http://www.spoj.com/problems/NG0FRCTN/> (Medium) Suggested by [**codeforrest**](http://cfrp.azurewebsites.net/profile/codeforrest)

KMP

<http://www.spoj.com/problems/ADAPET/>

UVA 12604

UVA 12467

UVA 11019

<http://www.spoj.com/problems/NAJPF/> (4) //classical kmp — all patterns

http://codeforces.com/contest/808/problem/G (6) //with DP -harder

lca

<http://www.spoj.com/problems/ADAAPPLE/> [HLD][LCT works too]

<http://www.spoj.com/problems/ADAVISIT/> [HLD]

<http://www.spoj.com/problems/ADALICI/> [KTH Ancestor — FAST!]

<http://www.spoj.com/problems/ADAORANG/>

<https://www.spoj.com/problems/COWGATH/> (5) //[VERY NICE][EULER][OFFLINE][SEGMENT]

<https://codeforces.com/gym/102021> [A](http://cfrp.azurewebsites.net/blog/entry/5) //[IMPLEMENTATION][LCA]

http://codeforces.com/gym/101908/problem/L (5) //[NICE][HLD][FW]

http://codeforces.com/gym/101808/problem/K (5) //[NICE][IMPLEMENTATION][CYCLE]

<http://www.spoj.com/problems/LCA/> //Easiest one — low constraints [practice]

<http://www.spoj.com/problems/LCASQ/>

http://codeforces.com/contest/916/problem/E (6) //[VERY NICE][SEG-TREE]//REROOT

http://codeforces.com/contest/911/problem/F (5) //[VERY NICE][LONGEST PATH]

<https://devskill.com/CodingProblems/ViewProblem/141>

UVA 12655

<https://www.codechef.com/problems/PSHTTR>

LightOJ 1162

Gym 100685G [2012-2013 ACM-ICPC, NEERC, Moscow Subregional Contest]

UVA 12533

<https://www.codechef.com/problems/CLOSEFAR> //But kinda more comples [ST]

http://codeforces.com/contest/733/problem/F 7

11354 UVA (4)

<http://www.spoj.com/problems/POLICEMEN/> (3) //simple + small graph

<http://www.spoj.com/problems/QTREE2/> (5) //very easy if bin. understrood

http://codeforces.com/contest/828/problem/F 7 // Differently MST / Outside

http://codeforces.com/contest/832/problem/D (5) //Classical + Depth /OR/ HLD +ST

<http://www.spoj.com/problems/DRTREE/> (5) //NICE [finding ancestor + depths]

http://codeforces.com/problemset/problem/838/B (6) //VERY NICE [HLD + ET + ST]

<http://www.spoj.com/problems/NTICKETS/> (4) //Maximum on path

<http://www.spoj.com/problems/GRASSPLA/> (5) //HLD

http://codeforces.com/contest/855/problem/D (4) //VERY VERY BAD STATEMENT (not so bad problem)

http://codeforces.com/gym/101630 {L}(6) //[NICE][HLD or DSU][XOR] //Parenthessis

lcs\_subsequence

<http://www.spoj.com/problems/ADASEED/> [HUNT]

<http://www.spoj.com/problems/MC/>

UVA 10192

UVA 531

LightOJ 1110

UVA 12747 //Perm

<http://www.spoj.com/problems/XMEN/> //Perm

UVA 10405

UVA 12511 //Increasing

UVA 10635 //Increasing

10949 UVA (6) — Hunt or Bit

<http://www.spoj.com/problems/MC/> (3) //Classical

<http://www.spoj.com/problems/LCS0/> (7) //Bit

lct

<http://www.spoj.com/problems/ADAROADS/> [ONLINE][UPDATES][LCT]

<http://www.spoj.com/problems/DYNALCA/en/> //DYNAMIC LCA

http://codeforces.com/gym/100460/problem/C

http://codeforces.com/gym/100960/problem/H

<http://www.spoj.com/problems/DYNACON1/>

http://codeforces.com/contest/892/problem/E (7) [VERY NICE]//By prices .. LCT

lis

UVA 231

UVA 10131

2931 LA

<http://www.spoj.com/problems/NDS/>

UVA 481

<http://www.spoj.com/problems/ELIS/>

UVA 497

UVA 11790

<http://www.spoj.com/problems/ALTSEQ/> 3 //solvable by FW in Nlog(N)

<http://www.spoj.com/problems/VISIBLEBOX/> (4) //with multiset

<http://www.spoj.com/problems/DOSA/> (5)

<http://www.spoj.com/problems/CODERE3/> 3 //Low bounds LIS/LDS

<http://www.spoj.com/problems/BRDGHRD/> 4 //lis (nondecreasing)

<http://www.spoj.com/problems/GONESORT/> 4 //Permutation-lis + riddle statement

http://codeforces.com/contest/847/problem/B 4 //Multiple Lis's

http://codeforces.com/contest/67/problem/D (4) //[NICE][DOUBLE REMAP]

matching

<http://www.spoj.com/problems/ADAPATH/>

<http://www.spoj.com/problems/ADACITY/>

<http://www.spoj.com/problems/ADAPLNTS/>

<https://codeforces.com/contest/1198/problem/E> (7) //[VERY NICE][HARD][KONIG][OBSERVATION]

URI 1330 (4) //[NICE][SIMPLE]

http://codeforces.com/gym/101873 [F](http://cfrp.azurewebsites.net/blog/entry/4)

http://codeforces.com/gym/101666 [E](http://cfrp.azurewebsites.net/blog/entry/6) //[NICE][GEOMETRY][BINARY SEARCH][PROBABILITY]

<http://www.spoj.com/problems/MATCHING/> //Raw (no sauce)

http://codeforces.com/contest/116/problem/B (3) //Low cons. Can be done without algo

6234 — Tile Cut (LA)

10080 — Gopher II (UVA) //Easy — sympathic

11419 — SAM I AM (UVA) //Grid-Matching

http://codeforces.com/gym/101485 (Elementary Math — E) //Very nice principal [not that hard]

3673 — Black-White Grid (UVA) //Grid

12927 — Points Cover (UVA)

6851 — The Programmers (LA)

6887 — Book Club //NICE

6571 — It Can Be Arranged

12530 — Game of Tiles

http://codeforces.com/gym/100820 (Airport — A) //Nice one

http://codeforces.com/gym/100753 (Bounty Hunterr II — B) //VERY NICE — I refered multiple times to this principal

http://codeforces.com/gym/101408 (Cat vs Dog — C)

1171 — Knights in Chessboard (II) (LOJ) //Classical chess

12644 — Vocabulary

http://codeforces.com/gym/101047/problem/H

http://codeforces.com/problemset/problem/659/E

12972 — Cuban Challenge (UVA)

1201 — Taxi Cab Scheme (UVA)

12963 — Astragalus Nitidiflorus

6525 — Attacking rooks (LA)

3415 — Guardian of Decency (LA)

12159 — Gun Fight (UVA)

<https://www.codechef.com/problems/CHEFYODA> //Imho matching is not the crucial part here

12831 — Bob the Builder

http://codeforces.com/problemset/problem/831/D

11262 — Weird Fence

12549 — Sentry Robots

11138 — Nuts and Bolts

http://codeforces.com/contest/727/problem/D 4

11985 UVA (5)

<http://www.spoj.com/problems/AMR12A/> (5) //VERY NICE [goophers + bonus](http://cfrp.azurewebsites.net/blog/entry/BS)

<http://www.spoj.com/problems/NITT4/> (4) //VERY NICE [Chessboard matching]

<http://www.spoj.com/problems/SCPC11H/> (4)//NICE — Match those which fits inside

matrix

12045 UVA (4)

matrix\_exponentiation

<http://www.spoj.com/problems/ADAMONEY/>

UVA 13298 — A Fibonacci Family Formula (4) //[NICE][CLASSICAL]

http://codeforces.com/gym/101628/problem/I (4) //[MODULAR][NICE][TRANSITION]

http://codeforces.com/gym/101845/problem/A

13284 Macarons (4) //[NICE][BITMASK][DP] Classical

http://codeforces.com/contest/147/problem/B (5) //[NICE][BS][GRAPHS]

http://codeforces.com/contest/107/problem/D (5) //[NICE][DP]

UVA 12653

LightOJ 1131

UVA — 12470

http://codeforces.com/problemset/problem/696/D

<https://www.hackerrank.com/contests/mathemagic-bits/challenges/gp-on-fibonacci-matrix> [accesable?]

<https://www.hackerrank.com/contests/mathemagic-bits/challenges/degree-diameter-on-trees> [accesable?]

LightOJ 1052

<https://www.codechef.com/problems/KBIGNUMB>

LightOJ 1160

LightOJ 1096

UVA 11486

http://codeforces.com/problemset/problem/718/C

LightOJ 1070

LightOJ 1132

http://codeforces.com/problemset/problem/621/E

Project Euler #114: Counting block combinations I

http://codeforces.com/problemset/problem/821/E

UVA 12593

<https://www.codechef.com/AUG16/problems/SHAIKHGN> //Not exactly but similar principle

11551 UVA (4)

11486 UVA (5)

10743 UVA (5) //A001169 [easy multi / hard to come with recurence]

http://codeforces.com/contest/821/problem/E (6) //Not trivial to come-up with matrix

<http://www.spoj.com/problems/DCEPCA06/> (4) //NICE — 10x10 matrix

<http://www.spoj.com/problems/GSWORDS/> (3) //NICE&EASY — 3-states "OO,OX,XO"

<http://www.spoj.com/problems/TETRAHRD/> (4) //Easy + Sum

<http://www.spoj.com/problems/NACCI/> (4) //Classical (N-Bonacci)

<http://www.spoj.com/problems/JZPCIR/> (5) //VERY NICE: A137726

mcmf

<http://www.spoj.com/problems/ADAGROW/>

<http://www.spoj.com/problems/ADAFEAR/>

<https://www.urionlinejudge.com.br/judge/en/problems/view/1617> (4) //[BASIC]

13288 — Cordon Bleu (6) //[NICE][MATCHING]

http://codeforces.com/gym/100800 (Aqueduct Construction — A)

10806 — Dijkstra, Dijkstra.

12944 — Earthquake Disaster

12891 — Risk of Trading

<https://www.codechef.com/problems/HOGON>

http://codeforces.com/contest/863/problem/F (5) //VERY NICE

11613 UVA (6)

http://codeforces.com/contest/802/problem/C (8) //Nice but hard to see + negative

http://codeforces.com/contest/802/problem/N (5) //Easy but faster MCMF needed

http://codeforces.com/contest/818/problem/G (6) //NICE + MUCH Faster MCMF

<http://www.spoj.com/problems/BNMT/> (7) //VERY NICE (some optimalisations needed + weird data set)

http://codeforces.com/contest/884/problem/F (6) //Alphabetical buckets

median

http://codeforces.com/contest/713/problem/C 7

<http://www.spoj.com/problems/RMID2/> 4

<http://www.spoj.com/problems/RMID/> 3 (as above just not so dynamic)

<http://www.spoj.com/problems/EC_ESTA/> 4 //classical dynamic median

<http://www.spoj.com/problems/DCEPCA09/> (6) //VERY NICE [MO +++ MEDIAN, MEAN, FREQ]

meet\_in\_middle

http://codeforces.com/contest/912/problem/E (5) //[VERY NICE][BS][2P][PRIMES]

http://codeforces.com/contest/888/problem/E (4) //NICE[EASY]

http://codeforces.com/contest/51/problem/E (6) //NICE[GRAPH][Cycles of length 5]

<https://devskill.com/CodingProblems/ViewProblem/245>

<https://devskill.com/CodingProblems/ViewProblem/256>

11851 UVA (5)

11465 UVA (5)

13207 UVA (4) //Straight-forward MIM

<http://www.spoj.com/problems/COLOR_CC/> (4) //VERY NICE — div by partity (take lesser) → 8^6

MO

<http://www.spoj.com/problems/ADAUNIQ/> [Updates]

<http://www.spoj.com/problems/ADANUM/>

<http://www.spoj.com/problems/ADAPEAR/>

<http://www.spoj.com/problems/ADAPHONE/>

<https://www.spoj.com/problems/MOZHSLS/> (5) //[VERY NICE]

<http://www.spoj.com/problems/ADAFTBLL/> [Tree][Updates]

http://codeforces.com/gym/101879/problem/H (5) //[NICE][CLASSICAL][FREQUENCIES][FENWICK][BS]

<http://www.spoj.com/problems/FREQUENT/>

http://codeforces.com/contest/86/problem/D (4) //[NICE][CLASSICAL]

http://codeforces.com/contest/877/problem/F (6) //[NICE][NORMALIZE]

http://codeforces.com/problemset/problem/687/D

http://codeforces.com/problemset/problem/617/E

<http://www.spoj.com/problems/DCEPCA09/>

<http://www.spoj.com/problems/COT2/>

<https://www.codechef.com/problems/DISTNUM3> //Tree + Update

<https://toph.ws/p/distinct-dishting> //Not sure if still working??

<https://www.codechef.com/problems/CHEFNUMK>

<https://www.hackerearth.com/problem/algorithm/harry-gets-into-infy-1/description/>

<http://www.spoj.com/problems/COT/> (7) //ON TREE [but very tight TLE]

<http://www.spoj.com/problems/GOT/> (5) //ON TREE

<http://www.spoj.com/problems/CPAIR2/> (4) //MO + Fenwick [VERY NICE]

<http://www.spoj.com/problems/HRSIAM/> (5) //With updates

<http://www.spoj.com/problems/KDOMINO/> (4) //NICE — Frequencies

next

<https://codeforces.com/contest/1187/problem/B> (3) //[NICE]

13257 — License Plates [UVA](http://cfrp.azurewebsites.net/blog/entry/3) //Brute-Force 26^3 queries in O(1)

http://codeforces.com/contest/92/problem/C (3) //[VERY NICE]

LightOJ 1157

<https://www.codechef.com/problems/ASTRING>

<http://www.spoj.com/problems/MAIN8_E/>

http://codeforces.com/problemset/problem/701/C

<http://www.spoj.com/problems/STRSEQ/> //+DP

http://codeforces.com/problemset/problem/762/C

http://codeforces.com/problemset/problem/724/D

<http://www.spoj.com/problems/SUBSN/> //+BS

<https://www.hackerrank.com/contests/world-codesprint-5/challenges/short-palindrome> //+DP

np-hard

<http://www.spoj.com/problems/ADACROW>

<http://www.spoj.com/problems/ADAPARTY/>

<http://www.spoj.com/problems/ADAPARTI/>

http://codeforces.com/gym/101908/problem/J (6) //[VERY NICE][STEINER][GEOMETRY]

http://codeforces.com/gym/101666 [G](http://cfrp.azurewebsites.net/blog/entry/6) //[NICE][BURTE FORCE][OBSERVATION][SUBSETS]

http://codeforces.com/contest/907/problem/E (5) //[VERY NICE][CLIQUE][DP]

http://codeforces.com/contest/114/problem/B (3) //[CLIQUE][BRUTE-FORCE]

<http://www.spoj.com/problems/JOHNNY/> //Not exact — points

<http://www.spoj.com/problems/TAKIN/> (3) //[KNAPSACK][RECURSION]

http://codeforces.com/contest/839/problem/E (5) //NICE! Max-Clique

7743 — Cliques (Similar to ADAPARTI) (8) //Parametrisation

7616 Counting Cliques (4) //[NICE][CLIQUES][ELIMINATING-SETS]

number\_rectangle

<https://codeforces.com/contest/1181/problem/A> (2) //[EASY][FORMULA][MODULO]

<https://www.spoj.com/problems/PRADIPSUM/> (2) //[EASY][FORMULA]

UVA 10667 //Largest rectangle

UVA 10074 //Largest rectangle

UVA 836 //Largest rectangle

UVA 1330 //Largest rectangle

12192 UVA 5

http://codeforces.com/contest/729/problem/B 2

http://codeforces.com/contest/710/problem/C 4

11871 UVA 6

11617 UVA (3)

11573 UVA (4)

11499 UVA (5)

11230 UVA (4)

<http://www.spoj.com/problems/JOCHEF/> (5) //NICE: Lagers rectange 0/1 O(N\*M)

number\_theory

<http://www.spoj.com/problems/ADAGCD/> [+STL]

<http://www.spoj.com/problems/ADAHACK/>

<http://www.spoj.com/problems/ADADUNG/>

<http://www.spoj.com/problems/ADACAROT/>

<http://www.spoj.com/problems/ADACON/>

<http://www.spoj.com/problems/ADATAXES/>

<http://www.spoj.com/problems/ADADIG/>

<http://www.spoj.com/problems/ADASETS/>

<https://codeforces.com/contest/1175/problem/A> (2) //[EASY][SYMPATHIC][DIVISION]

<https://codeforces.com/contest/1203/problem/C> (3) //[NICE][GCD]

<https://devskill.com/CodingProblems/ViewProblem/553> (3) //[VERY NICE][SIMPLE]

<https://devskill.com/CodingProblems/ViewProblem/560> (2)

<https://codeforces.com/contest/1176/problem/B> (2) //[MODULO][GREEDY]

<https://codeforces.com/contest/1200/problem/C> (4) //[VERY NICE][GCD][LCM]

<https://codeforces.com/contest/1182/problem/A> (2) //[EASY][DP]

<https://atcoder.jp/contests/abc136/tasks/abc136_b> (2) //[IMPLEMENTATION][CONVERSION]

<https://codeforces.com/contest/1185/problem/A> (1)

<https://codeforces.com/contest/1198/problem/F> (6) //[VERY NICE][RANDOM][FACTORISATION][PRIME][GCD]

<https://codeforces.com/contest/1183/problem/B> (2) //[EASY][FORMULA][MIN/MAX]

<https://codeforces.com/contest/1187/problem/A> (1) //[EASY][FORMULA]

<https://codeforces.com/contest/1195/problem/B> (2) //[EASY][GAUSS][BRUTE]

<https://codeforces.com/contest/1189/problem/E> (5) //[INVERSION][STL]

<https://www.spoj.com/problems/CANPR/> (5) //[NICE][MOBIUS] Weird TC (weak?)

<https://www.spoj.com/problems/OVISLARSUM/> (3) //[FORMULA][128]

<https://www.urionlinejudge.com.br/judge/en/problems/view/2876> (6) //[VERY NICE][GCD][IE]

<https://codeforces.com/gym/102058/problem/E> (3) //[BRUTE-FORCE][GCD]

<https://codeforces.com/gym/101972/problem/A> (3) //[EASY][SIMULATION]

<https://toph.co/p/pascals-triangle> (2)

<https://toph.co/p/big-factorials> (1)

<https://www.codechef.com/viewsolution/21428053> (4) //[SORTING][ANGLES][MATH][COSINUS]

http://codeforces.com/gym/101962 [C](http://cfrp.azurewebsites.net/blog/entry/4)

http://codeforces.com/gym/101808/problem/G (4) //[GCD][IF][LCM]

http://codeforces.com/gym/101666 [L](http://cfrp.azurewebsites.net/blog/entry/5) //[GREED][DP] Logarithms

http://codeforces.com/gym/101666 [C](http://cfrp.azurewebsites.net/blog/entry/4) //[NICE][GCD]

13287 — Shattered Cake (2) //Rectangle

http://codeforces.com/gym/101650 [G](http://cfrp.azurewebsites.net/blog/entry/3) //Graphs

http://codeforces.com/contest/934/problem/D (4) //Polynomials

8253 — Boss Battle [LA](http://cfrp.azurewebsites.net/blog/entry/2)

http://codeforces.com/contest/919/problem/E (5) //[MODULAR-MATH]

http://codeforces.com/contest/149/problem/B (3) //Base to 10

http://codeforces.com/contest/148/problem/B (3) //[SIMULATION](math or bs)

http://codeforces.com/contest/148/problem/A (2) //[EASY][SIEVE][BRUTE-FOCE][LCM][IE]

<http://www.spoj.com/problems/JGTLE/> (3) //Sum reduction

<http://www.spoj.com/problems/BAPM/> (3) //Simple power / Observe, rare colisions

http://codeforces.com/contest/911/problem/C (2) //[Observation][Iff (or brute-froce)]

http://codeforces.com/contest/133/problem/B (2) //Bases + implementation

http://codeforces.com/contest/129/problem/A (1) //Parity

http://codeforces.com/contest/124/problem/D (4)

http://codeforces.com/contest/125/problem/A (1) //mod/div simple — if

http://codeforces.com/contest/902/problem/D (4) //Polynomials / Fibonacci / Bitset

http://codeforces.com/contest/899/problem/D (4) //X99999 possibilities

http://codeforces.com/contest/899/problem/C (3) //[NICE][GREEDY KNAPSACK]

http://codeforces.com/contest/899/problem/A (1)

http://codeforces.com/contest/898/problem/A (1)

http://codeforces.com/gym/101620 {F}(4) //[IFS][INVERSE][CASES]

http://codeforces.com/contest/900/problem/D (4) //[NICE][DIVISORS][GCD][IE]

http://codeforces.com/contest/900/problem/B (2) //Cycles or Brute-force + Division

http://codeforces.com/contest/119/problem/A (2) //[EASY][GCD]

8014 — Birthday Pizza Party (4) //[NICE][OVERFLOW][LCM WITH SHIFT]

7629 — Powers (3) //One path is factorisation

8037 — Military Service (2) //Sample formula

8085 — Divisors (4) //[NICE][PRIMES/FACTORISATION/SIEVE]

6589 — Captain Obvious and the Rabbit-Man (7) //Sylvester's formula

6582 — Magical GCD (4) //[NICE] — Data structures works too

8000 — FizzBuzz (1)

<https://devskill.com/CodingProblems/ViewProblem/421> (2) //6 buckets

http://codeforces.com/contest/93/problem/E (5) //[VERY NICE][DP][RECURSION]

http://codeforces.com/contest/86/problem/A (3) //[NICE] 50\* is best (unless more digits)

http://codeforces.com/contest/83/problem/D (5) //[VERY NICE][PRIMES][BRUTE-FORE-2]

http://codeforces.com/contest/82/problem/A (2) //Simulation

http://codeforces.com/contest/74/problem/C (4) //gcd

http://codeforces.com/contest/76/problem/E (4) //Divide sumations [BF]

http://codeforces.com/contest/71/problem/C (3) //[EASY][NICE][BRUTE-FORCE][DIVISORS]

http://codeforces.com/contest/70/problem/A (2) //3^(N-1)

http://codeforces.com/contest/43/problem/C (2) //Moduly by 3 {0/2+min(1,2)}

http://codeforces.com/contest/50/problem/A (1) //[EASY]

http://codeforces.com/contest/61/problem/C (4) //Base conversion + roman

UVA 355 //bases conversion

2559 [LA] //Base conversion

<https://www.codechef.com/problems/COPRIME3> //Cyka Möbius

<http://www.spoj.com/problems/PSTR/> //Cyka Möbius

http://codeforces.com/problemset/problem/803/F //Cyka Möbius

<https://devskill.com/CodingProblems/ViewProblem/23>

http://codeforces.com/contest/731/problem/F 4

12031 UVA (8)

http://codeforces.com/contest/722/problem/F (8)

http://codeforces.com/contest/716/problem/C 4

http://codeforces.com/contest/711/problem/E (8)

http://codeforces.com/contest/710/problem/D (6)

13154 (UVA) 7

13166 (UVA) 5

11962 (UVA) 2

11718 UVA 3

11510 UVA (5)

11538 UVA (3) //good one — just math

11556 UVA (1)

http://codeforces.com/contest/757/problem/E (8)

http://codeforces.com/contest/758/problem/F (7)

11481 UVA (4)

11237 UVA (4) //Nice — seems like knapsbag but it it not

11155 UVA (4) //Almost as previous problem

11038 UVA (3) //counting digits on interval

http://codeforces.com/contest/763/problem/C (7)

11087 UVA (4) //Sums of two numbers divisible with <=500 (10^5)

http://codeforces.com/contest/678/problem/C 2 //LCM

http://codeforces.com/problemset/problem/665/F (8) //p^3 | p\*q

<http://www.spoj.com/problems/LCMSUM/> //Vzorec v knihovničce

<http://www.spoj.com/problems/FRNDZND/> (2) // (size 1 == 1, else 0)

<http://www.spoj.com/problems/EXPOR/> //bit-by-bit (+ formula)

<http://www.spoj.com/problems/FACTDIV/> (5) //dyn-update of ans/factors GOOD

<http://www.spoj.com/problems/PAIRDIV/> (6) //cyka möbius -\_-

<http://www.spoj.com/problems/FCDC/> (4) //keep factorized factorial

<http://www.spoj.com/problems/NTHPRIME/> (7) // BS + NumPrime GOOD!!

<http://www.spoj.com/problems/DIVFACT3/> (7) // Sieve 10^8 + sqrt search

<http://www.spoj.com/problems/DIVFACT4/> (8) // Prime count

http://codeforces.com/contest/776/problem/C (4) //segments div. by number

http://codeforces.com/contest/776/problem/E (6) //vypsat cisla: f(N)=Phi(N),g(N)=N

<http://www.spoj.com/problems/PHT/> (2) //easy BS (NN+2N) mby math?

<http://www.spoj.com/problems/GCDEX/> (7) //OEIS A006579 — enough [well imp]

<http://www.spoj.com/problems/APS/> (3) //just sieve + generate

<http://www.spoj.com/problems/WPC5I/> (3)//fc: C[a]!=C[b]:F[a]^max(C[a],C[b])

<http://www.spoj.com/problems/SPEC_SET/> N→N/k→N/k/k

<http://www.spoj.com/problems/DCEPC11B/> (5) //Wilson't theorem!

<http://www.spoj.com/problems/FACTMULN/> (5) //each f[i]/c[i] separately

<http://www.spoj.com/problems/SPCM/> (4) //just factorisation + prime check (10^12)

<http://www.spoj.com/problems/TWOGAME/> (5) //gcd == Power of 3 => YES

<http://www.spoj.com/problems/MKEQUAL/> (2) //Chceck if sum is divisible by N

<http://www.spoj.com/problems/TIPTOP/> (3) //sqrt(N)==N? NICE!!

<http://www.spoj.com/problems/PSYCHON/> (4) // fast factorisation

<http://www.spoj.com/problems/ENIGMATH/> (1) // swap and div by gcd

<http://www.spoj.com/problems/SNGPG/> (3) // prime gen + check

http://codeforces.com/contest/795/problem/A (2) //brute-force

http://codeforces.com/contest/801/problem/E (6) //NICE! — Clique-DAG + inversion

http://codeforces.com/contest/798/problem/C (4) //GCD == at the beginning OR 2

http://codeforces.com/contest/803/problem/C (3) //Only "low" K and just divisors

10830 (4) //simple add 2→ sqrt(N) + their mirrors

http://codeforces.com/contest/817/problem/A (2) //check division + parity

13209 UVA (3) //Simple simulation of division (+states rememberance)

http://codeforces.com/contest/834/problem/C (4) //Has cube-root + both num divisible by cuberoot

http://codeforces.com/contest/837/problem/E (5) //Factorisation + GCD attributes

<http://www.spoj.com/problems/SUMMATION/> (3) //Number contribution: 2^(N-1)

<http://www.spoj.com/problems/SECTORS/> (4) //Odd len OR sum of odd indices == sum of even

<http://www.spoj.com/problems/IITKWPCM/> (6) //VERY NICE — Gauss's generalisation of Wilsons Theorem

<http://www.spoj.com/problems/UCV2013A/> (4) //N\*(N^L-1)/(N-1)

<http://www.spoj.com/problems/KIMO1/> (4) //NICE — Adding/Subing by modulus

<http://www.spoj.com/problems/AFS2/> (4) //Sum of divisort (sqrt(N)) — (+128int)

<http://www.spoj.com/problems/FUNNUMS/> (4) //Permutations (get ith + guess ith)

<http://www.spoj.com/problems/MAY99_3/> (3) //GCD

<http://www.spoj.com/problems/PUCMM334/> (3) //Classical hats problem

<http://www.spoj.com/problems/LCPC11B/> (4) //Factorize + count all subsets

<http://www.spoj.com/problems/THREENUMBERS/> (2) //EASY & NICE [lcm]

<http://www.spoj.com/problems/GAMES/> (2) //EASY&NICE — Go discrete (by 10^k) → 10^k/GCD

<http://www.spoj.com/problems/SUBSHARD/> (4) //dig\*10^sufix\*(choose sufix)\*^Prefix [VERY NICE]

<http://www.spoj.com/problems/INVDIV/> (6) //Sum of divisors function + factorisation [NICE]

<http://www.spoj.com/problems/JNEXT/> (2) //EASY — Zahřívačka pro prváky

<http://www.spoj.com/problems/TSHOW1/> (3) //NICE — Almost as binary repre with 5/6

http://codeforces.com/contest/859/problem/B (2) //Easy by iteration

http://codeforces.com/contest/861/problem/A (2) //5,2 division [EASY]

<http://www.spoj.com/problems/ABA12D/> (4) //NICE — Formula for sum divisors Prod(Sum(fac-powers))

<http://www.spoj.com/problems/MAIN74/> (3) //Fibo's

<http://www.spoj.com/problems/NOSQ/> (3) //Sieve + Digit check

<http://www.spoj.com/problems/SQUAREV1/> (3) //NICE (but dumb code limit)

http://codeforces.com/contest/597/problem/A (1) //Simple if+2\*Divide

<http://www.spoj.com/problems/STREETR/> (3) //Easy [GCD]

<http://www.spoj.com/problems/HPYNOSII/> (3) //<=1000 after first move

<http://www.spoj.com/problems/HPYNOS/> (1) //Same as above — 1 number

<http://www.spoj.com/problems/IITD4/> (3) //Brute-force [sieve-like]

<http://www.spoj.com/problems/GIRLSNBS/> (2) //Some easy formula

<http://www.spoj.com/problems/GUESSTHE/> (2) //NICE [EASY][LCM]

http://codeforces.com/contest/867/problem/B (2) //NICE [observation][1 2]

<http://www.spoj.com/problems/NDIVPHI/> (4) //NICE — Multiple of lowest primes [BIG-INT]

<http://www.spoj.com/problems/NDIVPHI2/> (5) //NICE — Same as above [bigger constraints]

http://codeforces.com/contest/6/problem/A (1) //Triangle check (4 pts)

http://codeforces.com/contest/869/problem/B (1) //Answer is usually 0

http://codeforces.com/contest/9/problem/C (2) //Bases (2→ 10)

http://codeforces.com/contest/20/problem/B (2) //Ax^2+Bx+C=0 (find roots)

http://codeforces.com/contest/27/problem/E (5) //[VERY NICE]: Factorisation+Recursion

http://codeforces.com/contest/32/problem/C (3) //DSU works too

http://codeforces.com/contest/872/problem/C (3) //NICE — Compose of 4 (mby one 6 or 9)

http://codeforces.com/contest/876/problem/B (3) //Group by modulo

observation

<http://www.spoj.com/problems/ADABANKET/> //Observation (but Stoer-Wagner works too)

<https://codeforces.com/contest/1194/problem/A> (1) //[TRIVIAL]

<https://www.spoj.com/problems/ARRTWIST/> (3)

<https://www.spoj.com/problems/OPMODULO/> (3) //[NICE]Brute-force and observe

<https://codeforces.com/gym/102001/problem/I> (2) //[PARITY]

<https://codeforces.com/gym/102001/problem/A> (4) //[VERY NICE][FREQUENCY]

<https://codeforces.com/gym/101991/problem/L> (3) //[NICE][EASY][BITS]

<https://codeforces.com/gym/102021> [F](http://cfrp.azurewebsites.net/blog/entry/3) //[NICE][SEQENCE]

<https://www.codechef.com/problems/BUDDYNIM> (3) //[OBSERVATION][NICE]

http://codeforces.com/gym/101962/problem/H (4) //[SEQUENCES][SIMULATION]

http://codeforces.com/contest/922/problem/C (3) //Note that K is always very low

http://codeforces.com/contest/912/problem/B (2) //Bits/XOR

http://codeforces.com/contest/911/problem/D (3) //[VERY NICE]

http://codeforces.com/contest/902/problem/E (5) //[VERY NICE] //LCT works too

7589 Rearranging a Sequence (3) //Print from back and then real

http://codeforces.com/gym/101630 {A}(6) //[NICE][STL]

http://codeforces.com/contest/122/problem/B (1) //just 4 or 7

http://codeforces.com/contest/116/problem/E (6) //[NICE][COMBINATORICS][FIND ANY]

http://codeforces.com/contest/892/problem/D (4) //NICE [SORTING]

http://codeforces.com/contest/897/problem/E (5) //[NICE][SET] //2 will beat all

7730 To begin or not to begin (3) //Parity

http://codeforces.com/contest/122/problem/D (4) //[NICE] Sweep /OR/ cycle

http://codeforces.com/contest/128/problem/D (6) //Normalize + Sweeps

oeis

<https://codeforces.com/gym/101981> [G](http://cfrp.azurewebsites.net/blog/entry/3) //A000332

<https://codeforces.com/gym/102001/problem/C> //De Bruine

6591 — Bus (2) //2^N-1

12004 UVA 2

11273 UVA 5 //https://oeis.org/A001088

11077 UVA 3 //https://oeis.org/A094638

<http://www.spoj.com/problems/VECTAR5/> 3 //https://oeis.org/A038721

<http://www.spoj.com/problems/ESYRCRTN/> 2 //generate and see

<http://www.spoj.com/problems/IITWPC4B/> 3 //http://oeis.org/A005044

<http://www.spoj.com/problems/POLCONST/> (4) //A003401+Fermat Number (Prime)

<http://www.spoj.com/problems/CUTCAKE/> (3) // pattern [1 22 333 4444 55555]

10872 UVA 3 //Alcuin's Sequence

<http://www.spoj.com/problems/LOVINGPW/> (3) //A000788

<http://www.spoj.com/problems/CBANK/> (3) //A000292 Tetrahedral numbers

<http://www.spoj.com/problems/GUMATH2/> (4) //A000240 Modulo by MOD\*2

<http://www.spoj.com/problems/MATHII/> (4) //A006218 (Two formulas => sqrt(N))

<http://www.spoj.com/problems/KOPC12B/> (4) //A002457 (factorial + factorial modular)

<http://www.spoj.com/problems/YUMMY/> (4) //A006534

<http://www.spoj.com/problems/ADV04B1/> (5) //A001850

<http://www.spoj.com/problems/FLWRS/> (4) //A002464

<http://www.spoj.com/problems/RANJAN02/> (3) //A024023

<http://www.spoj.com/problems/BOMARBLE/> (2) //A000326

http://codeforces.com/contest/57/problem/C (4) //A045992

http://codeforces.com/contest/918/problem/A (1) //Fibo

offline

<http://www.spoj.com/problems/ADARAIN/>

<http://www.spoj.com/problems/ADARAINB/>

<http://www.spoj.com/problems/ADAROBOT/>

<https://www.spoj.com/problems/CLFLARR/> (3) //[NICE][EASY][QUEUE][STL]

<https://codeforces.com/gym/102021> [M](http://cfrp.azurewebsites.net/blog/entry/5) //[VERY NICE][SPANNING]//LCA WORKS TOO

http://codeforces.com/gym/101840 [I](http://cfrp.azurewebsites.net/blog/entry/4) //[NICE][PREPROCESS]

http://codeforces.com/problemset/problem/816/B //Easier ST

11266 UVA (6) //slightly knapsack || [NICE]

http://codeforces.com/contest/761/problem/F (7)

<http://www.spoj.com/problems/UPDATEIT/> (2) //basic method

13189 UVA (4) //simulation + sort queries

<http://www.spoj.com/problems/HAYBALE/> (3) //Basic sum + sort (segment tree would work too)

http://codeforces.com/gym/101597/problem/H (6) //Offline 2D sum + geometry

palindromes

<https://www.spoj.com/problems/JUSTAPAL/> (6) //[VERY NICE][SA][RMQ]

<https://www.spoj.com/problems/DRAGKING/> (4) //[CLASSICAL][DP]

<https://codeforces.com/gym/101981> (5) //[NICE][SEGMENT][MANACHER][ZETA][SA]

<https://codeforces.com/gym/101972/problem/F> (3) //[EASY][OBSERVATION][GREEDY]

http://codeforces.com/gym/101628/problem/E (4) //[NICE][DP] //Easy

http://codeforces.com/gym/101864 [J](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][MANACHER]

http://codeforces.com/gym/101806/problem/Q (4) //[HASHING][PALINDROMIC-TREE]

http://codeforces.com/contest/897/problem/B (2) //Palindromic numbers [EASY][FIRST]

http://codeforces.com/contest/883/problem/H (3) //Make as least odd as possible

<http://code-festival-2017-qualc.contest.atcoder.jp/tasks/code_festival_2017_qualc_c> (3) //Similar to remove-dp, but greedy

<http://code-festival-2017-qualc.contest.atcoder.jp/tasks/code_festival_2017_qualc_d> (4) //DP with greedy thinking

http://codeforces.com/contest/59/problem/C (3) //Implementation [constructive]

UVA 13092 //manacher

<http://www.spoj.com/problems/MSUBSTR/> //manacher

UVA 11888 //manacher

<https://www.hackerrank.com/contests/101hack35/challenges/circular-palindromes> //manacher

UVA 12378 //manacher

<https://www.hackerrank.com/contests/world-codesprint-5/challenges/challenging-palindromes> //SA

<http://www.spoj.com/problems/NUMOFPAL/> //Palindromic Tree

http://codeforces.com/problemset/problem/245/H //# of palindromes [DP]

<http://www.spoj.com/problems/ANAGR/> 2 //frequency + palindromes

<http://www.spoj.com/problems/AMR12D/> (1) //Palindrome check //Zahrivacka pro prvaky

http://codeforces.com/problemset/problem/835/D //Hashing

http://codeforces.com/contest/137/problem/D (4) //[NICE][DP]

patter-matching

11019 UVA (5)

permutations

http://codeforces.com/contest/137/problem/B (1) //Frequency + size

http://codeforces.com/contest/136/problem/A (1) //Permutation cycles

http://codeforces.com/contest/844/problem/C 3 //NICE Permutations in array

http://codeforces.com/contest/48/problem/D (3) //NICE [frequency array]

http://codeforces.com/contest/56/problem/B (2) //Restore from 1 reverse [EASY]

http://codeforces.com/contest/122/problem/D (5) //NICE [BRUTE-FORCE]//ONLY BACK MATTERS

persistent\_segment\_tree

http://codeforces.com/contest/893/problem/F (6) //[VERY NICE][EULER TOUR]

http://codeforces.com/contest/813/problem/E (6) //Easy but hard data structure

<https://codeforces.com/contest/893/problem/F> //Suggested by [**phibrain**](http://cfrp.azurewebsites.net/profile/phibrain)

preprocess

<http://www.spoj.com/problems/ADAPLUS/>

<https://codeforces.com/contest/1175/problem/E> (5) //[VERY NICE][LIFTING][PRIORITY]

<https://codeforces.com/contest/1181/problem/C> (4) //[VERY NICE][RMQ]

<https://atcoder.jp/contests/abc136/tasks/abc136_d> (4) //[NICE][PARITY]

<https://codeforces.com/contest/1194/problem/B> (3) //[IMPLEMENTATION][INDEXING]

<http://www.spoj.com/problems/ADASQR/> [Monotone Queue]

<https://toph.co/p/rio-and-inversion> (5) //[NICE][INVERSIONS][PREFIX SUM][2D]

http://codeforces.com/gym/101801 [I](http://cfrp.azurewebsites.net/blog/entry/3) //[PREFIX SUM]

http://codeforces.com/gym/101889 [L](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][PREFIX SUM][BINARY SEARCH]

http://codeforces.com/contest/120/problem/D (3) //But maybe work naively ?!

http://codeforces.com/contest/48/problem/B (2) //2D Prefix Sum [BF prolly works too]

http://codeforces.com/problemset/problem/761/F

VA 10360 //2D Prefix Sum

UVA 983 //2D Prefix Sum

http://codeforces.com/contest/777/problem/C (4) //NICE

http://codeforces.com/contest/818/problem/C (4) //Prefix Sum

http://codeforces.com/contest/834/problem/B (2) //26 queries — NICE rozehrivacka pro prvaky

<http://www.spoj.com/problems/RANGESUM/> (4) //NICE: Offline (delta) + Prefix Sum

<http://www.spoj.com/problems/RANDG/> (3) //NICE [but too low bounds] [PrefixSum] [Try all indexes]

<http://www.spoj.com/problems/HARSHAD/> (3) //Sieve + simple function

<http://www.spoj.com/problems/PUCMM210/> (3) //Number theory (thinking not necessary)

<http://www.spoj.com/problems/BCAKE/> (3) //Prefix sum of rectangle + N^4

<http://www.spoj.com/problems/PAIRSUM/> (4) //VERY NICE — Prefix sum + Prefix PW sum

<http://www.spoj.com/problems/MAIN111/> (3) //Sieve + Brute Force (answer in O(1))

<http://www.spoj.com/problems/PLUSEVI/> (4) //NICE — There are not many of them

http://codeforces.com/contest/18/problem/C (2) //Prefix sum End==2\*i

http://codeforces.com/contest/873/problem/B (2) //Search for same prefix (+/-1)

http://codeforces.com/contest/33/problem/C (4) //Prefix sum + Sweep from back + sweep from front

http://codeforces.com/contest/872/problem/B (3) //Sweep from both sides (RMQ works too)

prime-count

<http://www.spoj.com/problems/NTHPRIME/>

<http://www.spoj.com/problems/DIVFACT4/>

http://codeforces.com/problemset/problem/665/F

<https://www.codechef.com/problems/CNTPRIME>

<http://www.spoj.com/problems/SUMPRIM1/>

<http://www.spoj.com/problems/SUMPRIM2/> [NO — hard]

<https://www.hackerrank.com/contests/projecteuler/challenges/euler010/problem> //easy

prime-testing

<http://www.spoj.com/problems/ADAGIFT/>

<https://www.spoj.com/problems/PRMFN/> (5) //[VERY NICE][GREEDY]

<http://www.spoj.com/problems/PON/en/>

<http://www.spoj.com/problems/DAYOUT2C/en/> [EASY]

Project Euler #60: Prime pair sets

Project Euler #58: Spiral primes

Project Euler #131: Prime cube partnership

<https://devskill.com/CodingProblems/ViewProblem/229>

<https://devskill.com/CodingProblems/ViewProblem/327>

Gym 100753K [2015 German Collegiate Programming Contest (GCPC 15) + POI 10-T3]

Project Euler #130: Composites with prime repunit property

<http://www.spoj.com/problems/ABA12A/> (3)

10871 UVA (3) //Easy — fermat not necessary

<http://www.spoj.com/problems/POP1/> (4) //Fast primality testing (or somehow)

<http://www.spoj.com/problems/POP2/> (5) //NICE — same as above (yet with ll)

<http://www.spoj.com/problems/POP3/> (6) //same as above (yet with big)

<http://www.spoj.com/problems/DCEPC203/> (6) //NICE [optimalisation]

<http://www.spoj.com/problems/PRIMPERM/> (4) //NICE (next\_perm + sieve)

probability

<https://codeforces.com/contest/1173/problem/E1> (5) //[NICE][DP]

<https://codeforces.com/contest/1194/problem/F> (6) //[NICE][COMBINATORICS][PREPROCESS]

http://codeforces.com/gym/101982 (5) [K](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][DP]

http://codeforces.com/gym/101845/problem/G (4) //[DP] Classical

http://codeforces.com/gym/101845/problem/M (3) //[NICE][OBSERVATION]

http://codeforces.com/gym/101726/problem/B (4) //[NICE][DP][BRUTE-FORCE]

<https://www.devskill.com/CodingProblems/ViewProblem/470> (3) //Brute Force

8262 — Knockout Tournament (4) [LA] //[NICE][SIMULATION] //Might not be working on LA (but on CF y)

http://codeforces.com/contest/912/problem/D (4) //[VERY NICE][EXPECT][DIJKSTRA]

http://codeforces.com/contest/908/problem/D (5) //[DP][MATH][INVERSION]

7821 — Elections (4) //[EASY][DP]

http://codeforces.com/gym/101620 {G}(5) //[VERY NICE][DIJKSTRA][EXPECTED VALUE]

http://codeforces.com/contest/110/problem/D (4) //[NICE][COMBINATORICS]

http://codeforces.com/contest/108/problem/D (4) //[NICE][SIMPLE][COMBINATORICS]

7619 — Guessing the Dice Roll (5) //[NICE][AHO][MATRIX EXPO]

7998 — Election (3) //Math works too [probability][DP]

LightOJ 1104 //Birthday Paradox

Gym 101064K [2016 USP Try-outs] //Birthday Paradox

11762 UVA (5)

11427 UVA (5)

11348 UVA (2)

http://codeforces.com/contest/768/problem/D (4) //With DP

<http://www.spoj.com/problems/IITWPC4J/> (4) //with DP

10828 UVA (5) //Nice problem but bad statemend: Expected value of visits MC

10777 UVA (4) //NICE — yet solvable with DP

http://codeforces.com/contest/839/problem/C (3) //NICE & Easy => Tree

<http://www.spoj.com/problems/ZCR/> (3) //Easy (+DP)

<http://www.spoj.com/problems/IITKWPCN/> (2) //Easy — Odd/Eve (black balls)

http://codeforces.com/contest/846/problem/F (5) // Expected number of unique elements

<http://www.spoj.com/problems/BTCODE_H/> (4) //DP (but main GROW is idea)

http://codeforces.com/contest/867/problem/D (5) //VERY NICE [DP]

http://codeforces.com/contest/24/problem/D (5) //VERY NICE [DP]+[TIME]

http://codeforces.com/contest/28/problem/C (4) //VERY NICE [DP]

recursion

<https://codeforces.com/contest/1175/problem/B> (3) //[STACK]

<https://www.codechef.com/problems/HPIRATES> (4) //[VERY NICE][OBSERVATION][GRAPHS]

8255 — Dunglish [LA](http://cfrp.azurewebsites.net/blog/entry/3) //Finding all possibilities

http://codeforces.com/contest/915/problem/C (4) //[NICE][DP-works-too][GREEDY]

http://codeforces.com/contest/134/problem/B (4) //Number Theory

http://codeforces.com/contest/897/problem/C (4) //+Slightly [DP]

6585 — Draughts (4) //[NICE][BACKTRACK][DFS]

http://codeforces.com/contest/68/problem/D (5) //[NICE] Keep max and kill branches

UVA 536 //Tree from dfs

UVA 12347 //BST from preorder

<http://www.spoj.com/problems/GOC11A/> 4

13170 UVA (7) //heavy implementation — but NICE!

10854 UVA (3) //if/else

http://codeforces.com/contest/31/problem/D (4) //[NICE] Brute-force by recursion

http://codeforces.com/contest/36/problem/B (2) //[NICE][SIMPLE]

RMQ

<https://www.spoj.com/problems/MOZPWS/> (4) //[NICE][IMPLEMENTATION][MQ]

<https://codeforces.com/gym/101972/problem/G> (4) //[NICE][SWEEP WORKS TOO]

http://codeforces.com/problemset/problem/514/D //+BS

http://codeforces.com/problemset/problem/872/B

<http://www.spoj.com/problems/TNVFC1M/>

<https://devskill.com/CodingProblems/ViewProblem/19>

http://codeforces.com/contest/713/problem/D 6

http://codeforces.com/contest/675/problem/E 5

<http://www.spoj.com/problems/POSTERIN/> 5 //VERY NICE — Delete all minimas

<http://www.spoj.com/problems/RPLN/> (3) //RMQ only

<http://www.spoj.com/problems/CITY2/> (4) //RMQ + MAP [NICE][VAGUE STATEMENT]

<http://www.spoj.com/problems/DIFERENC/> (4) //Solve separately (linear D&C)

http://codeforces.com/contest/863/problem/E (4) //OR some Queue / sorting

http://codeforces.com/contest/5/problem/C (4) //NICE — many other options

http://codeforces.com/contest/15/problem/D (5) //VERY NICE 2D RM [sliding-windw][monotone-queue]

http://codeforces.com/contest/873/problem/E (5) //[NICE][Brute-Force + RMQ]

rope

<http://www.spoj.com/problems/AROPE/> 4

<http://www.spoj.com/problems/AROPE2/> 5 //same as above (+time)

scc

<http://www.spoj.com/problems/ADAPANEL/> [combinatorics]

13301 — Impossible Communication (4) //[CUTE][SAMPLE][OBSERVATION][EASY]

http://codeforces.com/problemset/problem/427/C proposed by [**unofficial10**](http://cfrp.azurewebsites.net/profile/unofficial10)

http://codeforces.com/contest/894/problem/E (5) //[NICE][DP]

<https://devskill.com/CodingProblems/ViewProblem/79>

UVA 11838

UVA 247

UVA 13057

UVA 12645

UVA 11770

UVA 12926

UVA 11324

UVA 11709

UVA 12745

<http://www.spoj.com/problems/TFRIENDS/> (4) //just scc size

<http://www.spoj.com/problems/CAPCITY/> (4) //scc destination [WEAK TC]

http://codeforces.com/contest/22/problem/E (5) //[NICE][make it strongly connected][SRC>DST]

segment\_tree

<http://www.spoj.com/problems/ADABERRY/> [+TREAP]

<https://codeforces.com/contest/1180/problem/E> (6) //[VERY NICE]

<https://codeforces.com/contest/1187/problem/D> (5) //[VERY NICE][GREEDY][STACK]

<https://toph.co/p/maintain-the-queue> (5) //[NICE][QUEUE]

http://codeforces.com/gym/101992/problem/L (5) //[VERY NICE]

http://codeforces.com/gym/101982 [F](http://cfrp.azurewebsites.net/blog/entry/6) //[SWEEP][NORMALIZE][NICE][XOR]

http://codeforces.com/gym/101962/problem/I (4) //[BS]

http://codeforces.com/gym/101801 [G](http://cfrp.azurewebsites.net/blog/entry/5) //Two ST [SEQUENCE]

http://codeforces.com/gym/101879/problem/G (5) //Merge / Special

http://codeforces.com/gym/101741/problem/J (6) //[NICE][COMBINATORICS]

http://codeforces.com/contest/914/problem/D (5) //[NICE][GCD][LOG]

http://codeforces.com/contest/915/problem/E (5) //[VERY NICE][UNLIMITED]

http://codeforces.com/contest/145/problem/E (5) //[NICE]//Bit swap + subsequence

<http://www.spoj.com/problems/ADAGF/>

<http://www.spoj.com/problems/ADATREE/>

http://codeforces.com/contest/911/problem/G (7) //Segment tree over queries

http://codeforces.com/contest/895/problem/E (5) //[PRECISION]

http://codeforces.com/contest/52/problem/C (4) //Easy [MIN]+[INCREASE]

http://codeforces.com/contest/56/problem/E (5) //[NICE][NORMALIZE][MAX]

http://codeforces.com/contest/877/problem/E (5) //[VERY NICE][EULER TOUR TREE]

<https://devskill.com/CodingProblems/ViewProblem/283>

<https://devskill.com/CodingProblems/ViewProblem/315>

http://codeforces.com/problemset/problem/756/C

http://codeforces.com/contest/739/problem/C (8)

http://codeforces.com/contest/718/problem/C (8)

http://codeforces.com/contest/750/problem/E (7)

http://codeforces.com/contest/759/problem/C (7)

11165 UVA (5)

http://codeforces.com/contest/763/problem/E (8) //VERY NICE — [non-trivial]

<http://www.spoj.com/problems/BGSHOOT/> (5) //normalize — then easy

<http://www.spoj.com/problems/KGSS/> (4)

http://codeforces.com/contest/765/problem/F (7) //VERY NICE — CASCADE

<http://www.spoj.com/problems/GSS1/> (5) //Idea — then easy

<http://www.spoj.com/problems/KQUERYO/> (5) //Seg-tree of vectors

http://codeforces.com/contest/633/problem/G (8) //EulerTree+Seg+Bitset

<http://www.spoj.com/problems/NAJ0001/> (7) //10^8 int — memory (and worked)

<http://www.spoj.com/problems/PRMQUER/> (5) //2 segment trees + sieve

<http://www.spoj.com/problems/EC_DIVS/> (5) //dunno if intended

<http://www.spoj.com/problems/DCEPC11I/> (5) //NICE — 1,2,3,4,5,.. inc

<http://www.spoj.com/problems/QUE2/> (4) //kth number

http://codeforces.com/contest/785/problem/E (6) //Seg+Treap [and faster]

http://codeforces.com/contest/786/problem/B (6) //+Dijkstra

13183 UVA (6) //Merge-Sort-Tree [dunno]

http://codeforces.com/contest/121/problem/E (7) //Dunno — done by vectorisation

http://codeforces.com/contest/803/problem/G (5) //VERY NICE!! — ST 10^9 + ST/RMQ 10^5

http://codeforces.com/contest/794/problem/F (7) //Digit by digit! (N\*log(N)\*100 )

http://codeforces.com/contest/811/problem/E (6) //VERY NICE — DSU (easier Timofey + animals)

http://codeforces.com/contest/817/problem/F (7) //10^18 + MEX ~~ NICE yet problematic

http://codeforces.com/contest/816/problem/B (3) //Or offline trick makes it easier

http://codeforces.com/contest/834/problem/D (5) //+Dynamic Programming | NICE

<http://www.spoj.com/problems/SBO/> (5) //preLast→ last (-1), last→ now (+1) — VERY NICE

<http://www.spoj.com/problems/GOODE/> (5) //NICE: Inversion + L-Mex

<http://www.spoj.com/problems/CNTPRIME/> (3) //ST+Sieve (short range)

<http://www.spoj.com/problems/SEGSQRSS/> (4) //NICE {weak data} ~~ SQRT works too

<http://www.spoj.com/problems/MON2012/> (5) //NICE [Online][10^9 Range]

<http://www.spoj.com/problems/PARSUMS/> (4) //But other approaches work too

<http://www.spoj.com/problems/THRBL/> (4) //Simple SA — maximum on range <= A[a]

<http://www.spoj.com/problems/HORRIBLE/> (3) //Totally classical

<http://www.spoj.com/problems/MULTQ3/> (4) //NICE (interesting operation)

<http://www.spoj.com/problems/PERMPATT/> (4) //NICE [minimum][+IDEA]

http://codeforces.com/contest/869/problem/E (5) //NICE — 2D [random][XOR]

http://codeforces.com/contest/19/problem/D (5) //NICE [+BS][+SET] {bs not necessary}

sequences

11885 UVA 7 //Previous problem requested for statement

11522 UVA 3 //Trick — low numbers only :P

sieve

<http://www.spoj.com/problems/ADAPRIME/>

<https://devskill.com/CodingProblems/ViewProblem/577> (4) //[NICE][TWO POINTERS]

<https://www.spoj.com/problems/EVENSEMIP/> (5) //[VERY NICE][SEGMENTED]

http://codeforces.com/gym/101982 [H](http://cfrp.azurewebsites.net/blog/entry/3) //[GOLDBACH]

http://codeforces.com/contest/114/problem/E (5) //[NICE][FAST THINGS NEEDED][SEGMENTED]

http://codeforces.com/contest/58/problem/B (3) //[NICE][GREEDY][LEAST PRIME FACTOR]

Project Euler #134: Prime pair connection //Segmented

11610 UVA (5)

11353 UVA (3)

<http://www.spoj.com/problems/TDPRIMES/> (4)

<http://www.spoj.com/problems/VECTAR8/> (3)

<http://www.spoj.com/problems/NFACTOR/> (4)

<http://www.spoj.com/problems/HS08PAUL/> (4) //simply generate

http://codeforces.com/contest/776/problem/B (3) //Easy — trict: PM-1/ELSE-2

<http://www.spoj.com/problems/GGD/> (4) // N/lowestDiv\*(lowestDiv-1)

http://codeforces.com/contest/822/problem/D (4) //DP + Lowest factor

<http://www.spoj.com/problems/NGIRL/> (4) //Squares — Primes + BS == Easiest

<http://www.spoj.com/problems/PTRI/> (5) //Very fast sieve necessary:/

<http://www.spoj.com/problems/AFS/> (3) //Sum of divisort + DP

<http://www.spoj.com/problems/BSPRIME/> (4) //Very fast sieve needed

<http://www.spoj.com/problems/DCEPC505/> (4) //NICE — at most 10527450

<http://www.spoj.com/problems/CUBEFR/> (3) //NICE — Sieve out k^3 numbers

<http://www.spoj.com/problems/PRIMES2/> (8) //VERY NICE — Some hell-shit optimizing

http://codeforces.com/contest/26/problem/A (2) //Easy — many ways to solve it

simulation

<https://codeforces.com/contest/1180/problem/C> (4) //[MODULO][QUEUE]

<https://codeforces.com/gym/102001/problem/F> (4) //[NICE][GREEDY]

<https://codeforces.com/gym/101991/problem/D> (4) //[NICE][NORMALIZE][PREFIX]

<https://codeforces.com/gym/102021> [L](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE][JUST DO IT]

<https://codeforces.com/gym/102021> [D](http://cfrp.azurewebsites.net/blog/entry/4) //Try one and see

<https://codeforces.com/gym/101972/problem/K> (3) //[EASY][STRINGS]

http://codeforces.com/gym/101962/problem/K (3) //[NICE] //Complexity

http://codeforces.com/gym/101650 [A](http://cfrp.azurewebsites.net/blog/entry/7) //[VERY NICE][TREAP][PROBABILITY]

http://codeforces.com/gym/101650 [K](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE] //Perhaps weak TC

<https://abc084.contest.atcoder.jp/tasks/abc084_c> (3) //Brute-Force

http://codeforces.com/contest/908/problem/B (2) //[EASY][NICE]

http://codeforces.com/contest/141/problem/B (3) //[NICE][IF-PARTY]

7921 — Anticlockwise Motion (4) //Simulate in sqrt

7691 — Falling Apples (3)

7977 — The Key to Cryptography (2)

http://codeforces.com/contest/129/problem/B (2) //Graph

http://codeforces.com/contest/903/problem/B (2) //[NICE]

http://codeforces.com/contest/120/problem/C (1)

http://codeforces.com/contest/118/problem/A (1)

8012 — Voting Fraud (1)

http://codeforces.com/contest/897/problem/A (1) //BF — do as they say

7985 — Bumper-to-Bumper Traffic (4) //FINE — We have whole time-lapse

7988 Flow Shop (3) //Do as they say. Trivial but not bad.

7703 — Reading Digits (2) //Simple simulate what they ask for

http://codeforces.com/contest/893/problem/A (1) //Easy but nice

http://codeforces.com/contest/102/problem/B (2) //Do as they write — log-convergence

http://codeforces.com/contest/92/problem/A (1) //Way too easy

http://codeforces.com/contest/88/problem/C (3) //[NUMBER THEORY]

http://codeforces.com/contest/84/problem/D (4) //Priority queue by min\*size

http://codeforces.com/contest/79/problem/A (1) //Simulate rules

http://codeforces.com/contest/879/problem/A (1) //Iterate day-by-day

http://codeforces.com/contest/879/problem/B (3) //Either at most N^2 or the biggest element [NICE]

http://codeforces.com/contest/879/problem/D (4) //[NICE][Array elimination]

http://codeforces.com/contest/46/problem/A (2) //[EASY][MODULO]

http://codeforces.com/contest/46/problem/B (3) //[EASY][SEARCH-EACH-QUERY]

http://codeforces.com/contest/55/problem/A (2) //Simple (long) simulation

http://codeforces.com/contest/60/problem/A (1) //Moving LR bounds

12187 UVA (2)

http://codeforces.com/contest/724/problem/C 5

http://codeforces.com/contest/746/problem/C 3

11093 UVA (2)

http://codeforces.com/contest/768/problem/C (4)

<http://www.spoj.com/problems/WRONG/> (5) //VERY NICE — precalculate from back, then go from front

http://codeforces.com/contest/864/problem/C (4) //Not nice — just iffs

<http://www.spoj.com/problems/WAGE/> (3) //Simple Game Of Life Modification

http://codeforces.com/contest/6/problem/C (2) //Simple simulate from both sides

http://codeforces.com/contest/9/problem/B (2) //Simulate what is given (+ doubles)

http://codeforces.com/contest/11/problem/B (3) //sqrt(X) [diff must be even]

http://codeforces.com/contest/30/problem/A (2) //Simply simulate process [-1000→ 1000]

sorting

<http://www.spoj.com/problems/ADATOMAT/>

<http://www.spoj.com/problems/ADAUSORT/>

<http://www.spoj.com/problems/ADACUT/>

<http://www.spoj.com/problems/ADAHLIA/>

<https://codeforces.com/contest/1174/problem/B> (3) //[OBSERVATION] Nice yet simple

<https://codeforces.com/contest/1174/problem/A> (2) //[EASY]

<https://codeforces.com/contest/1203/problem/E> (4) //[GREEDY][SIMPLE]

<https://codeforces.com/contest/1203/problem/B> (3) //[GREEDY][NICE]

<https://codeforces.com/contest/1185/problem/D> (4) //[BF][IMPLE][MATH]

<https://codeforces.com/contest/1185/problem/C1> (3) //[GREEDY]

<https://codeforces.com/contest/1197/problem/C> (4) //[NICE][EASY][OBSERVATION]

<https://codeforces.com/contest/1197/problem/A> (3) //[SIMPLE][GREEDY][MATH]

<https://codeforces.com/contest/1189/problem/B> (3) //[EASY][OBSERVATION]

<https://www.spoj.com/problems/SWAPDIFF1/> (4) //[CLASSICAL][INVERSIONS]

<https://www.spoj.com/problems/SHINCARD/> (4) //[NICE]

<https://www.spoj.com/problems/POTATOPL/> (4) //[NICE][GREEDY]

<https://codeforces.com/gym/101991/problem/I> (3)

http://codeforces.com/gym/101962/problem/M (4)

http://codeforces.com/gym/101726/problem/E (3) //[STRUCTURES][IMPLEMENTATION]

13282 Cakey McCakeFace (3) //Brute Force

8260 Installing Apps (4) //[NICE] //Sorting + some DP

http://codeforces.com/contest/922/problem/D (3) //[NICE][GREEDY][EASY]

http://codeforces.com/contest/920/problem/C (3) //[EASY] Sortable by parts

http://codeforces.com/contest/913/problem/D (4) //[NICE][BS][2P][FW]

http://codeforces.com/contest/149/problem/A (2) //Sorting|Greedy

<http://www.spoj.com/problems/SEUG/> (2) //Bad statement

http://codeforces.com/contest/141/problem/A (2) //Or frequency

http://codeforces.com/contest/137/problem/E (4) //[NICE][PREFIX SUM MATCHING]

http://codeforces.com/contest/137/problem/C (4) //[NICE][FENWICK WORKS TOO]

http://codeforces.com/contest/136/problem/C (3) //Last to 1(2) |OR| first to INF

7601 — Football (3) //Greedy

7673 — What a Simple Research (2) //[EASY][IMPLE]

http://codeforces.com/contest/108/problem/B (2) //Easy & Adjacent

8025 — Stacking Cups (2) //Sorting+input

http://codeforces.com/contest/892/problem/A (2) //[EASY][SUM]

<https://www.urionlinejudge.com.br/judge/en/problems/view/2290> (3) //Sort+check [fast]

http://codeforces.com/contest/81/problem/C (3) //MATH[Lesser=greater][comparator]

http://codeforces.com/contest/53/problem/D (3) //Bubble sort

http://codeforces.com/contest/58/problem/D (4) //[BUCKET][GREEDY][STRING]

UVA 10810 //INV

UVA 11858 //INV

http://codeforces.com/problemset/problem/645/B //INV

UVA 10327 //INV

<http://www.spoj.com/problems/BUBBLESORT/> //INV

UVA 11495 //INV [GAME]

<http://www.spoj.com/problems/CODESPTB/> //INV

UVA 13212 //INV

http://codeforces.com/problemset/problem/749/E //INV

12189 UVA (3)

12196 UVA (4)

http://codeforces.com/contest/731/problem/D 7

11925 UVA (4)

11979 UVA (3)

http://codeforces.com/contest/747/problem/D (4)

11890 UVA (4)

<http://www.spoj.com/problems/KAOS/> (4) //INV — GOOD problem!!!!

<http://www.spoj.com/problems/KSMALL/> (5) //fast sort /or/ quick-select

<http://www.spoj.com/problems/RKS/> (3) //use map

<http://www.spoj.com/problems/SPCJ/> (4) //reverse + go from back

http://codeforces.com/contest/785/problem/B (2) //last-first + vice versa

http://codeforces.com/contest/798/problem/D (4) //Take 1st then take best B of every pair (sort by A)

http://codeforces.com/contest/810/problem/B (2)

http://codeforces.com/contest/810/problem/C (3) //+Math

http://codeforces.com/contest/814/problem/A (1) //Pro prváky — but nice observation

http://codeforces.com/contest/817/problem/B (3) //Frequency of TOP 3

10769 UVA (3) //Sadly N^4 passes

13208 UVA (4) //Sort + Prefix Sum

13212 UVA (3) //Number of inversions

http://codeforces.com/contest/831/problem/C (3) //NICE ~ Check all "add" against first

http://codeforces.com/contest/831/problem/D (4) //Can be solved with BS+Max-Match

http://codeforces.com/contest/841/problem/C (3) //NICE — match greatest to lowest

http://codeforces.com/contest/845/problem/C (2) //EASY — pro prvaky

<http://www.spoj.com/problems/HSHW/> (4) //Test every big/low pair + big/big low/low on +/-

<http://www.spoj.com/problems/CODESPTB/> (3) //Count inversions [BASIC]

http://codeforces.com/contest/863/problem/B (2) //Sort and omit 2

<http://www.spoj.com/problems/AMR10G/> (2) //Easy yet NICE

http://codeforces.com/contest/12/problem/C (2) //Very simple

http://codeforces.com/contest/16/problem/B (1) //[EASY]

http://codeforces.com/contest/22/problem/D (3) //Sort by begin + sweep

http://codeforces.com/contest/23/problem/C (3) //Take them by pairs + add last

http://codeforces.com/contest/24/problem/B (3) //Simple follow the rules

http://codeforces.com/contest/27/problem/B (3) //Compare number of victories

http://codeforces.com/contest/27/problem/C (4) //[NICE] Find next bigger/lesser (sort)

spanning\_tree

<https://www.urionlinejudge.com.br/judge/en/problems/view/2683> (3) //[BASIC][MAXIMUM]

http://codeforces.com/contest/908/problem/F (5) //[VERY NICE] //Not exactly MST but similar[GREEDY]

http://codeforces.com/contest/125/problem/E (5) //[BS]

http://codeforces.com/contest/76/problem/A (4) //[VERY NICE] Sort by A and KEEP spanning + one edge

LA 6622 — Absurdistan Roads (4) //Plus one edge

UVA 12176

UVA 10600

UVA 10724

<https://www.hackerrank.com/contests/june-world-codesprint/challenges/johnland>

UVA 11710

Gym 101252C [2014-2015 CT S02E05: Codeforces Trainings Season 2 Episode 5 — 2009-2010 ACM-ICPC]

Gym 101047I [2015 USP Try-outs][HARD]

UVA 11183 [Directed]

LightOJ 1101

<https://www.codechef.com/problems/CHEFELEC>

UVA 10307

http://codeforces.com/problemset/problem/598/D

http://codeforces.com/problemset/problem/32/C

http://codeforces.com/problemset/problem/744/A

<https://devskill.com/CodingProblems/ViewProblem/344>

908 — Re-connecting Computer Sites [UVA]

1208 — Oreon [UVA]

1235 — Anti Brute Force Lock [UVA]

10034 — Freckles [UVA]

11228 — Transportation system [UVA]

11631 — Dark roads [UVA]

11733 — Airports [UVA]

11747 — Heavy Cycle Edges [UVA]

BLINNET SPOJ (3)

11183 UVA (4) //Directed [need to know algo!]

<http://www.spoj.com/problems/ULM09/> (3) //Sum-Kruskal

<http://www.spoj.com/problems/IITKWPCG/> (4) //VERY NICE [log instead of price]

http://codeforces.com/contest/17/problem/B (3) //Spanning tree [no dsu]

spfa

LightOJ 1074

UVA 1171

UVA 11478

UVA 12768

11478 UVA (5)

sqrt

<https://codeforces.com/contest/1199/problem/D> (4) //[NICE][SIMPLE IMPLE]

<https://www.spoj.com/problems/ZING01/> (5) //[VERY NICE][STRING][SEGMENT]

http://codeforces.com/gym/101889 [D](http://cfrp.azurewebsites.net/blog/entry/6) //[NICE][STL]

http://codeforces.com/contest/916/problem/D (6) //[NICE][BS][BINARY-LIFTING]

12003 UVA 7

11990 UVA (5)

<http://www.spoj.com/problems/GIVEAWAY/> (7) //SQRT + BS > [or Seg+Trie]

http://codeforces.com/contest/786/problem/C (5) //Nsqrn (bg) + sqrSegs (end)

http://codeforces.com/contest/840/problem/D (5) //NICE — Either frequent OR brute-force

http://codeforces.com/contest/13/problem/E //VERY NICE [SQRT-BLOCK UPDATE/JUMP]

http://codeforces.com/contest/85/problem/D (4) //NICE [ST shall work too]

stl

<http://www.spoj.com/problems/ADABASET/> [IO-testing][ARRAY]

<http://www.spoj.com/problems/ADALIST/> [ROPE]

<http://www.spoj.com/problems/ADAFIELD/> [SET]

<http://www.spoj.com/problems/ADAQUEUE/>

<http://www.spoj.com/problems/ADAFRIEN/>

<https://www.spoj.com/problems/REDARR2/> (4) //[NICE][PQ]

<https://www.spoj.com/problems/MINSTOCK/> (3) //[EASY][MAP]

http://codeforces.com/gym/101864 [F](http://cfrp.azurewebsites.net/blog/entry/4) //[SET]

http://codeforces.com/contest/920/problem/B (3) //[NICE][QUEUE][SIMPLE]

http://codeforces.com/contest/911/problem/E (4) //[NICE][STACK][SET]

http://codeforces.com/contest/909/problem/D (4) //[VERY NICE][SIMULATION][GROUP]

http://codeforces.com/contest/899/problem/E (4) //[VERY NICE][SET]

<https://arc087.contest.atcoder.jp/tasks/arc087_a> (2) //Map

http://codeforces.com/contest/903/problem/C (2) //Map

http://codeforces.com/contest/900/problem/C (3) //[NICE][SET]

http://codeforces.com/contest/112/problem/A (1) //tolower [trivial]

3396 — Alphabetics (2) //Getline, letters identification

http://codeforces.com/contest/81/problem/A (2) //Stack OR string

http://codeforces.com/contest/78/problem/A (1) //fgets + imple

http://codeforces.com/contest/69/problem/E (3) //NICE [2POINTERS][SET][MAP]

<http://www.spoj.com/problems/RMID/> //Dynamic Median

<http://www.spoj.com/problems/RMID2/> //Dynamic Median

http://codeforces.com/problemset/problem/713/C //Dynamic Median

<http://www.spoj.com/problems/EC_ESTA/> //Dynamic median

http://codeforces.com/contest/799/problem/B (2) //EASY — MAP

http://codeforces.com/contest/808/problem/D (3) //MAP

10887 (2) //string + map

10730 (3) //Easy with hash-map

http://codeforces.com/contest/821/problem/C (3) //STACK (vector) Nice+Easy

<http://www.spoj.com/problems/SOLVEIT/> (3) //Set + lower\_bound

<http://www.spoj.com/problems/IITKWPCA/> (2) //Set + getline

http://codeforces.com/contest/849/problem/D (5) //Queue

<http://www.spoj.com/problems/CRAN02/> (4) //Map (+Math)

<http://www.spoj.com/problems/MAX_NUM/> (4) //Queue (possibly multiple ways)

<http://www.spoj.com/problems/SID/> (5) //Sort + Vector (or similar) [strict TLE]

<http://www.spoj.com/problems/RPLD/> (2) //Map of sets

http://codeforces.com/contest/861/problem/D (4) //unordered map of sets

<http://www.spoj.com/problems/FACEFRND/> (2) //Set or Bitset

<http://www.spoj.com/problems/HACKRNDM/> (3) //Easy — map

http://codeforces.com/contest/847/problem/K (4) //NICE Map+Queue

http://codeforces.com/contest/855/problem/A (1) //set

http://codeforces.com/contest/4/problem/C (2) //map+string

http://codeforces.com/contest/5/problem/E (6) //iffs + RMQ+BS+SET [or other sol]

http://codeforces.com/contest/44/problem/A (1) //Set + pair

http://codeforces.com/contest/45/problem/C (4) //NICE — Handling with sets

http://codeforces.com/contest/918/problem/B (2) //[MAP] //TLDR

strings

<https://codeforces.com/contest/1189/problem/A> (2) //[OBSERVATION]

<https://codeforces.com/contest/1194/problem/C> (3) //[EASY][FREQUENCY][2POINTERS]

<https://www.spoj.com/problems/MAGSUB1/> (3) //[EASY][SYMPATHIC][COMBINATORICS]

<https://www.urionlinejudge.com.br/judge/en/problems/view/2866> (1) //Reverse

<https://codeforces.com/gym/101972/problem/H> (3) //[NICE][PREPROCESS]

http://codeforces.com/contest/909/problem/A (1) //Lexicographical comparision [EASY]

7948 Periodic Strings (2) //Brute-force

7892 — Game of Matchings (6) //No idea — heuristic works — but seems to be nice

8017 — Mancunian and Sonu Generate Strings (5) //[NICE][DP]//+Something: Trie/SA/SET...?

http://codeforces.com/contest/890/problem/D (4) //[NICE][SET][SORTING][26][BRUTE-FORCE]

http://codeforces.com/contest/43/problem/B (2) //Frequency

http://codeforces.com/contest/50/problem/B (2) //Frequency + Power

<http://www.spoj.com/problems/MINMOVE/> (3) //Minimal lexicographical rotation

<http://www.spoj.com/problems/BWHEELER/> //Burrows Wheeler

<http://www.spoj.com/problems/EDIST/> //Edit Distance

3189 LA //Edit Distance

<https://www.hackerrank.com/contests/morgan-stanley-2015/challenges/minimum-transformation-cost/problem> //Edit Distance

http://codeforces.com/gym/101492/problem/L //Laevenstein Distance

UVA 13068 //Lexicographically lowest rotation

2755 [LA]//Lexicographically lowest rotation

LightOJ 1073 //Lexicographically Shortest Superstring

http://codeforces.com/contest/762/problem/C 5

<http://www.spoj.com/problems/LCS0/> 10 //LCS

<http://www.spoj.com/problems/IITWPC4H/> 2 //Frequence array

13186 UVA (6) //Bitset + Trie ~ NICE [6-7 mby?]

http://codeforces.com/contest/798/problem/B (2) //Brute-force .. pro prváky

10745 UVA (4) //Frequency (N^2 possible if efficient!!)

http://codeforces.com/contest/822/problem/B (2) //Easy pro prvaky (slightly imple.)

http://codeforces.com/contest/828/problem/C (4) //+Sorting (process only necessary!)

http://codeforces.com/contest/832/problem/B (3) //Naive compare back+front [+freq]

<http://www.spoj.com/problems/STC04/> (5) //Next + pairs O(N\*26) [frist look O(26^2\*N)]

<http://www.spoj.com/problems/IITKWPCJ/> (4) //GCD or HASHING

<http://www.spoj.com/problems/SUBSN/> (4) //Next (NICE — bad input):

<http://www.spoj.com/problems/BOGGLE/> (2) //EASY [MAP][STREAM]

<http://www.spoj.com/problems/MAIN8_E/> (4) //VERY NICE — Next function

<http://www.spoj.com/problems/STRMATCH/> (3) //Nice matching, yet low constraints on "N"

<http://www.spoj.com/problems/FINDSR/> (3) //Clever bruteforce works here (NlogN)

http://codeforces.com/contest/39/problem/J (2) //Simple iteration over string

http://codeforces.com/contest/876/problem/E (4) //Compare from back + make fail-pairs

suffix\_array

<http://www.spoj.com/problems/ADAPHOTO/> [or hashing after turboanalisis]

<http://www.spoj.com/problems/ADASTRNG/>

<https://codeforces.com/contest/1202/problem/E> (5) //[NICE][OBSERVATION][RMQ]

<https://codeforces.com/gym/101991/problem/E> (6) //[VERY NICE][RMQ]Prolly an easier alternative

<https://toph.co/p/distinctness> (6) //[VERY NICE][SET][LCP][RMQ]

http://codeforces.com/gym/101889 [M](http://cfrp.azurewebsites.net/blog/entry/6) //[VERY NICE][OBSERVATION]

http://codeforces.com/gym/101840 [B](http://cfrp.azurewebsites.net/blog/entry/7) //[NICE][ST][BIT][LCP]

http://codeforces.com/contest/129/problem/D (4) //[NICE]//Suffix Automaton / Brute Force works too

http://codeforces.com/contest/123/problem/D (5) //[NICE][RMQ][D&C]

8086 Substring Sorting (7) //[VERY NICE][RMQ][BS][PST]

7702 — Castle //6 [VERY NICE][RMQ][BS][FW]

7701 — Favorite music //8 [VERY NICE][RMQ][SEARCH][OPTI] //WRONG TIME-LIMIT==OPTI

<https://www.urionlinejudge.com.br/judge/en/problems/view/1530> //7 [VERY NICE][RMQ][SET][OFFLINE]

<https://www.urionlinejudge.com.br/judge/en/problems/view/2000> (5) //Basic disubstr counting

UVA 760 //LCS

UVA 1227 //LCS

<http://www.spoj.com/problems/LCS/en/> //LCS

UVA 11512

7502 [LA]

Gym 100923D [2015 ACM National Contest Romania — Round 1]

UVA 1254

UVA 12191

UVA 12206

<https://www.codechef.com/problems/INSQ16F>

3943 LA

UVA 11107

UVA 12974

UVA 10526

Davos and Reading [INSOMNIA] //Awesome problem but can't find link [hard] — non of regular judges

UVA 12338

<https://devskill.com/CodingProblems/ViewProblem/328>

12191 UVA 5

SARRAY SPOJ 3

4513 LA 6

<http://www.spoj.com/problems/LCS2/> 7 // must be linear (SA+LCP+MQ)

http://codeforces.com/contest/802/problem/I 7 //NICE! SA+LCP+(Segmentree/queue)

<http://www.spoj.com/problems/LONGCS/> //5 LCS of multiple strings (NICE)

<http://www.spoj.com/problems/SUBLEX/> //5 VERY NICE: Kth substring

http://codeforces.com/contest/873/problem/F //6 VERY NICE: SA+Histogram

http://codeforces.com/contest/30/problem/E //7 VERY NICE: [many other sols: Manach/KMP/HASH...]

6856 — Circle of digits //7 [VERY NICE]

ternary\_search

LightOJ 1146

Gym 101482G [2014-2015 Northwestern European Regional Contest (NWERC 2014)]

Gym 101309D [2010-2011 ACM-ICPC Northeastern European Regional Contest (NEERC 10)]

UVA 13010 //+Dijkstra

2015-2016 CTU Open Contest: Chasing the Cheetah

12197 UVA (4)

<http://www.spoj.com/problems/KOPC12A/> (4) //TS (sadly brute-force works too N^2)

topo

<https://codeforces.com/gym/102058/problem/A> (5) //[NICE][2TIMES][PRIORITY]

UVA 11686

LightOJ 1034

UVA 10305

124 — Following Orders [UVA]

200 — Rare Order [UVA]

872 — Ordering [UVA]

11060 — Beverages [UVA]

http://codeforces.com/contest/765/problem/E 5

http://codeforces.com/contest/770/problem/C 4 //reduce + toposort

http://codeforces.com/contest/825/problem/E 4 //Toposort from biggest/backward

<http://www.spoj.com/problems/CODESPTI/> (6) //VERY NICE — Hard/Weak children

http://codeforces.com/contest/47/problem/B (2) //[EASY][Toposort on 3 elements]

http://codeforces.com/contest/909/problem/E (4) //[NICE][BFS][RECURSION]

treap

<https://codeforces.com/contest/1181/problem/D> (5) //[VERY NICE][SORTING][OFFLINE]

<https://www.hackerrank.com/challenges/find-the-running-median/problem> (4) //[NICE][HEAPS]

http://codeforces.com/gym/101992/problem/M (4) //[NICE][OFFLINE][SORTING]

http://codeforces.com/gym/100112 [C](http://cfrp.azurewebsites.net/blog/entry/5) //[VERY NICE]

http://codeforces.com/gym/101864 [B](http://cfrp.azurewebsites.net/blog/entry/5) //[NICE] Lesser than

http://codeforces.com/contest/899/problem/F (5) //[VERY NICE][DATA STRUCTURE] (FW or Treap)

http://codeforces.com/contest/879/problem/E (6) //[VERY NICE][Making somponents]

<http://www.spoj.com/problems/ADAAPHID/> [or clever fenwick][or SQRT-tree]

<http://www.spoj.com/problems/ADACROP/>

http://codeforces.com/contest/762/problem/E 6

<http://www.spoj.com/problems/COUNT1IT/> 5

<http://www.spoj.com/problems/IITWPC4D/> 4 //From end — pick i-th + del i-th

<http://www.spoj.com/problems/ALLIN1/> 4 //Typical treap operations

http://codeforces.com/contest/847/problem/D 5 //Prolly overkill — VERY NICE

http://codeforces.com/contest/863/problem/D 4 //Not desired solution

<http://www.spoj.com/problems/KOILINE/> (4) //VERY NICE — Iterate from back — get+remove

<http://www.spoj.com/problems/TWIST/> (5) //NICE: Reverse

<http://www.spoj.com/problems/MEANARR/> (5) //VERY NICE! [POLICY][SHORT]

tree

<http://www.spoj.com/problems/ADATOMEL/>

<https://codeforces.com/contest/1189/problem/D1> (4) //[OBSERVATION]

http://codeforces.com/contest/913/problem/B (2) //Simple check

http://codeforces.com/contest/894/problem/D (5) //[NICE][MERGING][BINARY]

http://codeforces.com/contest/746/problem/G 5

http://codeforces.com/contest/750/problem/F 9

<http://www.spoj.com/problems/RTREE/> 3 //longest path tree — query

13175 UVA (2) //something like preorder build

http://codeforces.com/contest/796/problem/C (3) //Just counting — inc by at most 2

http://codeforces.com/contest/797/problem/D (4) //VERY NICE — sort + D&C all

http://codeforces.com/contest/805/problem/E (4) //NICE — Treewidth coloring (greedy)

http://codeforces.com/contest/828/problem/D (3) //Star construction

<http://www.spoj.com/problems/TREEDEGREE/> (3) //Degree from euler tree

<http://www.spoj.com/problems/UCV2013J/> (3) //Find what is leaf in Binary Tree

<http://www.spoj.com/problems/GCPC11J/> (3) //Finding ceter

http://codeforces.com/contest/34/problem/D (3) //Simple reconstruction + DFS

tree-dp

<http://www.spoj.com/problems/ADASALES/>

<https://codeforces.com/contest/1187/problem/E> (5) //[VERY NICE][DFS]

<http://www.spoj.com/problems/PT07X/> (4) //Classical- VC on tree [NICE]

http://codeforces.com/contest/81/problem/E (6) //[NICE][PSEUDOFOREST REDUCTION]

http://codeforces.com/contest/61/problem/D (4) //dfs-only might works too

LA 6631 — Jingle Balls (4) //[NICE][SIMPLE]

<https://www.codechef.com/problems/TRANDED> //By [**sahil070197**](http://cfrp.azurewebsites.net/profile/sahil070197)

13089 — Golden Coins (UVA)

http://codeforces.com/problemset/problem/855/C

http://codeforces.com/problemset/problem/718/D

<https://www.codechef.com/problems/TWOCOINS>

<https://www.hackerrank.com/contests/101hack35/challenges/road-maintenance/problem>

7649 — Performance Review (LA)

http://codeforces.com/problemset/problem/741/D

http://codeforces.com/problemset/problem/592/D

<https://www.codechef.com/problems/TOMJERGA>

http://codeforces.com/problemset/problem/814/D

1220 — Party at Hali-Bula (UVA)

<https://www.hackerrank.com/contests/june-world-codesprint/challenges/r-tree-decoration/problem>

12452 — Plants vs. Zombies HD SP (UVA)

http://codeforces.com/problemset/problem/735/E

<https://www.codechef.com/problems/COLTREE>

12466 — Ancestors (UVA)

6829 — Intrepid climber (LA)

<https://www.hackerrank.com/contests/101hack35/challenges/jeanies-route>

12257 — The Queue (UVA)

http://codeforces.com/problemset/problem/805/F

http://codeforces.com/problemset/problem/763/D

1218 — Perfect Service

3346 — Perfect Domination on Trees (same as above -\_-)

12093 — Protecting Zonk

10859 — Placing Lampposts

http://codeforces.com/problemset/problem/23/E //NICE [but requires big int]

http://codeforces.com/problemset/problem/14/D (5) //NICE [sorting-one][2DFS]

<http://www.spoj.com/problems/TWOPATHS/> (6) //VERY NICE Same as above ~ bigger constraints

http://codeforces.com/contest/868/problem/E (8) //VERY NICE — HARD — on tree

trie\_bit

<https://www.spoj.com/problems/XORX/> (4) //[NICE][BASIC][PREFIXXOR]

http://codeforces.com/contest/888/problem/G (5) //NICE[BIT-BY-BIT D&C]

http://codeforces.com/contest/842/problem/D //Proposed by [**usernameson**](http://cfrp.azurewebsites.net/profile/usernameson)

http://codeforces.com/problemset/problem/706/D

<https://www.hackerrank.com/contests/w8/challenges/black-box-1>

<https://csacademy.com/contest/round-42/task/xor-submatrix/>

http://codeforces.com/contest/714/problem/C 5

<http://www.spoj.com/problems/SUBXOR/> (4)

http://codeforces.com/contest/817/problem/E (5) //Classical — remember sum! NICE!

http://codeforces.com/contest/37/problem/C (4) //NICE — Prefix dictionary [or math]

trie\_string

<http://www.spoj.com/problems/ADAINDEX/>

http://codeforces.com/gym/101628/problem/K (5) //[NICE][IMPLEMENTATION]

8015 — Alice and Bob play Contact (5) //[VERY NICE] 1stBob

2642 [LA]

UVA 10860

UVA 10295

UVA 13186 //+Bitset

UVA 1123

UVA 12506

UVA 11539

UVA 1142

UVA 12359

UVA 10282

11732 UVA (5)

11539 UVA (5)

11488 UVA (4)

<http://www.spoj.com/problems/TRYCOMP/> (4)

10860 UVA (4) //DP + Trie [nice — slightly generic]

<http://www.spoj.com/problems/DICT/> (4) //Sample trie — but slightly weak/wrong data-set

TSP

<https://www.urionlinejudge.com.br/judge/en/problems/view/2810> (5) //[NICE][DOUBLE]

Gym 100818E [2015-2016 ACM-ICPC Southeastern European Regional Programming Contest (SEERC 2015)]

3725 [LA]

UVA 10496

Gym 101020H [2015 Syrian Private Universities Collegiate Programming Contest] N!

LightOJ 1057

UVA 11643 //[NICE][BFS]

3305 [LA] //On plane

10937 UVA (4) //find '!' / BFS / TSP — NICE!

10944 UVA (4)

10818 UVA (5) //Easy — but not-easy implementation: ++Dijkstra [LEX!]

<http://www.spoj.com/problems/A_W_S_N/> (4) //BFS + TSP (path) — NICE

two-pointers

<http://www.spoj.com/problems/ADAFENCE/> [Multiple Pointers]

<http://www.spoj.com/problems/ADAPLANT/>

<https://codeforces.com/contest/1175/problem/C> (4) //[NICE][GREEDY]

<https://codeforces.com/contest/1203/problem/D1> (4) //[VERY NICE][STRINGS][PREPROCESS]

<https://codeforces.com/contest/1203/problem/D2> (5) //[VERY NICE][STRINGS][PREPROCESS]

<https://codeforces.com/contest/1186/problem/C> (4) //[VERY NICE][BITS][OBSERVATION]

<https://www.spoj.com/problems/MOZHCAN/> (5) //[VERY NICE]

http://codeforces.com/gym/101864 [L](http://cfrp.azurewebsites.net/blog/entry/4) //Static Size [SORTING]

Dev Skills-499: Closest Pair Point (4) //[NICE][GEOMETRY][STRUCTURES]

http://codeforces.com/contest/919/problem/C (3) //Similar — simple sweep [EASY]

http://codeforces.com/contest/144/problem/C (3) //[NICE][FREQUENCY][STRINGS]

http://codeforces.com/contest/895/problem/B (3) //[OR BS]

http://codeforces.com/contest/84/problem/B (2) //EASY //ll

http://codeforces.com/contest/79/problem/C (4) //NICE — [STRINGS][SET][COMPARE]

http://codeforces.com/contest/746/problem/F 6

11436 UVA (5)

http://codeforces.com/contest/760/problem/D 4

11386 UVA (4)

<http://www.spoj.com/problems/WOWSUBSTR2/> 3

<http://www.spoj.com/problems/ARRAYSUB/> 4

<http://www.spoj.com/problems/CODFURY/> 3 //easy — ukazkove

http://codeforces.com/contest/769/problem/B 3 //sort + TP

http://codeforces.com/contest/814/problem/C 4 //NICE — maybe some DP +/-

<http://www.spoj.com/problems/CRAN04/> 4 //NICE — (more or less) 3 pointers

<http://www.spoj.com/problems/OPCPIZZA/> 3 //NICE [EASY] [AGAINS EACH OTHER]

<http://www.spoj.com/problems/ALIEN/> (3) //Classical

<http://www.spoj.com/problems/HOTELS/> (3) //Classical & Easy

<http://www.spoj.com/problems/KOIREP/> (4) //VERY NICE — N buckedt find mid diff

http://codeforces.com/contest/6/problem/E (4) //NICE — Multiset

http://codeforces.com/contest/873/problem/C (3) //NICE — M times 2P tenchique

wavelet\_tree

UVA 1480

<http://www.spoj.com/problems/ILKQUERY/>

http://codeforces.com/contest/840/problem/D //Proposed by [**GreenGrape**](http://cfrp.azurewebsites.net/profile/GreenGrape)

Zfunction

<https://www.codechef.com/problems/CHSTR> //Proposed by [**Apptica**](http://cfrp.azurewebsites.net/profile/Apptica)

<http://www.spoj.com/problems/SUFEQPRE/>

http://codeforces.com/problemset/problem/126/B

http://codeforces.com/contest/119/problem/D (7) //[VERY NICE]

2SAT

<https://www.urionlinejudge.com.br/judge/en/problems/view/1348> (4) //[CLASSICAL][NICE][SCC]

11930 UVA (4)

http://codeforces.com/contest/776/problem/D (5)

Finally if you would like to add some problem to the list — even though I would be glad, please do so only in case of:

1. It is very interesting
2. There is nothing, or low number of problems in the topic
3. You add it in "bigger amount" at once

Thank you.

Offcourse if you have any remarks, questionns or requests, don't hesitate to ask.

PS: I'm sorry but there might be some duplicities. In that case, either report it or ignore it (unless they are in different topics, then it have reason :) )

FFT Questions :

<https://codeforces.com/problemset/problem/1096/G>

<https://codeforces.com/problemset/problem/632/E>

<https://codeforces.com/problemset/problem/1251/F>

|  |
| --- |
| 250C greedy easy |
|  | 679C dp (sliding window) + dfs medium |
|  | 631E dp(convex\_hull\_trick) hard |
|  | 196D greedy + fenwick hard |
|  | 653F suffixArray + binary search hard |
|  | 525E meet\_in\_the\_middle medium |
|  | 198B bfs easy |
|  | 577E constructive medium |
|  | 662B bipartite\_test easy |
|  | 652E biconnected\_components medium |
|  | 631D kmp or Hash easy |
|  | 245H dp medium |
|  | 580E segment + Hash medium |
|  | 580D dp easy |
|  | 445C greedy easy |
|  | 555E 'Bridge Tree' + LCA + dp medium |
|  | 621E matrix\_exponentiation easy |
|  | 665E trie easy |
|  | 575H combinatorics medium |
|  | 620E segment + lazy + dfs easy |
|  | 145B greedy easy |
|  | 235B dp + probability medium |
|  | 223C dp + induction medium |
|  | 510C topological\_sort easy |
|  | 675D binary\_search\_tree easy |
|  | 103D dp + sqrt medium |
|  | 59E dp + graph + bfs easy |
|  | 177\_D2 ad\_hoc easy |
|  | 490F dfs\_Up/Down medium |
|  | 496E greedy + sorting medium |
|  | 605D segment + bfs medium |
|  | 670F constructive medium |
|  | 413B implementation easy |
|  | 670E segment or pointer easy |
|  | 509F dp medium |
|  | 486E fenwick or segment easy |
|  | 365C combinatorics easy |
|  | 365D dp + greedy easy |
|  | 489F dp easy |
|  | 678D matrix\_exponentiation easy |
|  | 681D dfs + sorting easy |
|  | 677E binary\_search + partial sum + brute force easy |
|  | 298C constructive easy |
|  | 212E dp + tree medium |
|  | 682C dfs easy |
|  | 682D dp easy |
|  | 222C number\_theory easy |
|  | 222E matrix\_exponentiation easy |
|  | 222D greedy + two\_pointer easy |
|  | 78E max\_flow easy |
|  | 576D matrix\_exponentiation + dp medium |
|  | 242E segment easy |
|  | 56E segment + dp easy |
|  | 396C segment + dfs easy |
|  | 12D fenwick easy(?) |
|  | 444C segment + lazy medium \* |
|  | 375D guni easy |
|  | 438D segment medium |
|  | 271D trie easy |
|  | 471D kmp medium |
|  | 463D dp + topological sort medium |
|  | 463E dfs + number\_theory medium |
|  | 442C greedy + sorting medium |
|  | 13E sqrt easy |
|  | 126B Hash or kmp easy(?) |
|  | 343D segment or guni medium(?) |
|  | 600E guni easy |
|  | 570D guni easy |
|  | 246E guni easy |
|  | 208E guni easy |
|  | 510D number\_theory + dp\_bitmask easy |
|  | 685B centroid easy |
|  | 685E dp easy |
|  | 506B scc easy |
|  | 427C scc easy |
|  | 369C dfs easy |
|  | 526F segment medium |
|  | 526E dp medium |
|  | 526D Hash + binary search easy |
|  | 526C greedy easy |
|  | 117C dfs medium |
|  | 507E dijkstra easy(?) |
|  | 292E segment easy |
|  | 514C trie easy |
|  | 587C LCA easy(?) |
|  | 687C dp medium |
|  | 687A graph + dfs easy |
|  | 687B chinese\_remainder\_theorem medium |
|  | 587B dp + sorting + two\_pointers medium(?) |
|  | 484D dp medium(?) |
|  | 261B dp + probability medium |
|  | 9D dp easy |
|  | 623B dp + TOF medium |
|  | 208C dp + bfs + dfs medium |
|  | 484B sorting + math medium |
|  | 687D union\_find medium |
|  | 165E dp\_bitmask easy |
|  | 19B dp\_knapsack easy |
|  | 534F dp + bitmask + memorization + Hash medium |
|  | 239E dfs + dp medium(?) |
|  | 145C dp easy |
|  | 101B no\_tag\_yet medium(?) |
|  | 57A bfs\_dfs or math easy |
|  | 507D dp easy |
|  | 27E dp + number\_theory easy |
|  | 467D dp + graph easy |
|  | 335B dp easy |
|  | 18E dp easy |
|  | 611C dp easy |
|  | 611D dp easy |
|  | 652D fenwick + compression or segment easy |
|  | 554E graph + dsu medium(?) |
|  | 560E combinatorics + dp easy (?) |
|  | 177C2 graphs + dfs\_and\_similar easy |
|  | 177B2 number\_theory easy |
|  | 269C graph + flows + topological\_sort medium |
|  | 243B graph easy |
|  | 689D sparse\_table easy |
|  | 689C binary\_search easy |
|  | 689B bfs easy |
|  | 689A implementation easy |
|  | 346D bfs medium |
|  | 276E graphs + fenwick medium |
|  | 436C dsu easy |
|  | 591C constructive easy(?) |
|  | 18D bigNum easy |
|  | 598C geometry easy |
|  | 533B dp + tree medium(?) |
|  | 379D brute\_force + dp medium |
|  | 558E segment + lazy easy |
|  | 35C bfs easy |
|  | 255D binary\_search + math medium |
|  | 234F dp easy |
|  | 229C combinatorics + graph medium |
|  | 303B math easy |
|  | 319C convex\_hull\_trick medium |
|  | 592D dfs\_and\_similar + trees medium |
|  | 492E math medium |
|  | 492D math or binary\_search medium(?) |
|  | 630E math easy |
|  | 95C dijkstra medium |
|  | 10D dp easy |
|  | 617E mo easy |
|  | 89A greedy + math medium |
|  | 340B geometry(ccw) easy(?) |
|  | 292D graphs + dsu medium |
|  | 367C graphs + sorting easy |
|  | 536B Hash easy |
|  | 379F graphs + LCA easy |
|  | 690C3 graphs + LCA easy |
|  | 550D constructive + graphs easy |
|  | 414C merge\_sort medium |
|  | 550E constructive medium(?) |
|  | 474E fenwick medium(?) |
|  | 690F1 graphs + combinatorics easy |
|  | 690C2 graphs + trees easy |
|  | 690C1 graphs easy |
|  | 404D dp easy |
|  | 285D bruteforce easy |
|  | 650C graphs + DAG + dp + dsu medium(?) |
|  | 39H math easy |
|  | 552E dp medium |
|  | 552D geometry easy |
|  | 449C number\_theory medium(?) |
|  | 603C grundy easy |
|  | 699C dp easy |
|  | 699D graph + dfs easy |
|  | 698C bitmask + probability easy (?) |
|  | 196B dfs medium(?) |
|  | 196C sorting + geometry + divide\_and\_conquer medium |
|  | 27D dfs easy |
|  | 295C bfs easy |
|  | 291D dfs easy |
|  | 402D greedy easy(?) |
|  | 402E graph + dfs easy |
|  | 691D graph + dfs + sorting easy |
|  | 223B dp easy |
|  | 691E Matrix easy |
|  | 53E dp medium(?) |
|  | 487E hld + biconnected\_components + segment hard(?) |
|  | 487C number\_theory + constructive medium |
|  | 487B dp medium |
|  | 148E dp easy(?) |
|  | 264C dp medium |
|  | 464C dp medium |
|  | 701F dfs or max\_flow + tarjan\_algorithm easy |
|  | 696C math medium |
|  | 117E hld + segment hard(?) |
|  | 489E binary search + dp medium(?) |
|  | 148D dp + probability easy |
|  | 378C dfs easy |
|  | 165D segment\_on\_tree easy |
|  | 581F dp\_on\_tree easy |
|  | 360C dp medium |
|  | 48E dp + topological\_sort + bfs easy |
|  | 85D segment medium |
|  | 553D greedy + graph medium |
|  | 164A dfs easy |
|  | 74C union\_find medium |
|  | 269B dp easy(?) |
|  | 382D dfs easy |
|  | 444B probabilites medium |
|  | 487D sqrt\_decomposition medium |
|  | 623C dp + binary search medium |
|  | 696E hld + segment + fenwick hard |
|  | 467E segment easy |
|  | 587D 2\_sat + graphs hard |
|  | 547D graphs + eulerian\_tour medium |
|  | 350E floyd\_warshall medium |
|  | 190E dfs\_and\_similar easy(?) |
|  | 22E dfs\_and\_similar medium |
|  | 22D dp easy(?) |
|  | 659G dp medium |
|  | 245G graphs easy |
|  | 266D graphs + floyd\_warshall medium |
|  | 103B dfs\_and\_similar easy |
|  | 87D union\_find + dfs\_and\_similar medium |
|  | 263D dfs\_and\_similar easy |
|  | 543B bfs + graphs easy(?) |
|  | 95E graphs + dp medium |
|  | 455C union\_find easy |
|  | 274B dp easy |
|  | 383C segment easy |
|  | 6E two\_pointers easy |
|  | 321C centroid\_decomposition\_of\_tree medium(?) |
|  | 555C segment or STL medium |
|  | 549B graphs + greedy + constructive easy |
|  | 587F aho\_corasick hard(?) |
|  | 542C dfs\_and\_similar + math easy |
|  | 466E union\_find + starting\_finishing\_time + dfs\_and\_similar + graphs easy |
|  | 163E aho\_corasick + segment or fenwick hard |
|  | 232B dp + combinatorics medium |
|  | 369E segment easy |
|  | 474C geometry(rotating point) easy |
|  | 703D segment or fenwick easy |
|  | 522D segment easy |
|  | 52C segment easy |
|  | 282E trie medium |
|  | 696D aho\_corasick + matrix\_exponentiation hard |
|  | 372C sliding\_window + dp medium |
|  | 536E HLD medium |
|  | 704A implementation easy |
|  | 106E ternary\_search medium |
|  | 30C dp easy |
|  | 149D dp easy |
|  | 149E kmp easy |
|  | 291E kmp medium |
|  | 508C greedy easy |
|  | 311B CHT medium |
|  | 312C constructive easy |
|  | 575F greedy medium |
|  | 600E guni easy |
|  | 427D string + dp easy |
|  | 427E ternary\_search easy |
|  | 453B dp + bitmask medium(?) |
|  | 645E dp medium |
|  | 571B dp medium(?) |
|  | 25E hashing medium |
|  | 557E trie + dp medium |
|  | 607C strings + hash + kmp medium |
|  | 235C suffix\_automaton + kmp medium |
|  | 296D graphs + floyd easy |
|  | 296B combinatorics + math easy |
|  | 296C partial\_sum easy |
|  | 452E suffix\_automaton hard |
|  | 356D dp medium |
|  | 706E implementation medium |
|  | 706D trie easy |
|  | 260E segment + fractional\_cascading + binary\_search hard(?!) |
|  | 343C binary\_search medium(?) |
|  | 707D persistent\_segment\_tree or dfs medium |
|  | 707E 2d\_fenwick easy(?) |
|  | 404E binary\_search easy |
|  | 709C greedy easy |
|  | 709D greedy easy |
|  | 709E dfs\_up\_down easy |
|  | 710F aho\_corasick hard |
|  | 299C games + greedy easy(for me it’s a bit harder that easy :D) |
|  | 299D back\_track + combinatorics medium |
|  | 258E persistent\_segment\_tree or even segment tree medium |
|  | 254D bfs easy |
|  | 484E persistent\_segment\_tree + binary\_search medium |
|  | 132C dp easy |
|  | 671C segment\_tree medium |
|  | 494B dp easy(man baram yekam dar hadde medium bud !) |
|  | 186C binpow or matrix\_exponentiation easy |
|  | 204E suffix\_automaton medium(shayadam hard !) |
|  | 711D pdfs + combinatorics easy |
|  | dp easy |
|  | 527E euler\_tour medium |
|  | 123D suffix\_automaton medium |
|  | 348E dfs\_up\_down hard |
|  | 633G pdfs + segment + lazy easy |
|  | 32C math easy |
|  | 32D implementation easy |
|  | 195E dsu easy |
|  | 175C greedy + sorting easy |
|  | 615C trie easy |
|  | 566A trie + greedy medium |
|  | 712D dp easy |
|  | 717E dfs easy |
|  | 108D combinatorics easy |
|  | 88E grundy easy |
|  | 82D dp easy |
|  | 500E fenwick or segment medium |
|  | 21C dp or even segment! medium |
|  | 713B binary\_search easy |
|  | 714E flows or dp medium |
|  | 449D dp + combinatorics(exclusion\_inclusion) medium |
|  | 717G flows medium |
|  | 506D DSU medium |
|  | 510E flows medium |
|  | 546E flows medium |
|  | 277E flows medium |
|  | 164C flows medium |
|  | 73C dp easy |
|  | 201C dp easy |
|  | 347E dp medium |
|  | 316C2 flows medium |
|  | 316G3 suffix\_automaton medium |
|  | 240F segment\_tree easy |
|  | 111C dp easy |
|  | 715B dijkstra easy |
|  | 297B greedy easy |
|  | 715A constructive chert!(chert ro sath bandi nemitoonam konam !) |
|  | 3B greedy easy |
|  | 291D greedy easy |
|  | 534D greedy easy |
|  | 388C greedy medium |
|  | 389C greedy easy(?) |
|  | 720B flows medium |
|  | 504E hash + binary\_search + graphs medium |
|  | 504C binary\_search easy |
|  | 576E dynamic\_connectivity medium |
|  | 551E sqrt\_decomposition easy(?) |
|  | 444D hashing + binary\_search + two\_pointers |
|  | 392B dp easy |
|  | 392A add\_hoc easy |
|  | 460D brute\_force + constructive easy |
|  | 360B dp easy |
|  | 131E sorting easy |
|  | 139D implementation medium(?) |
|  | 190C implementation easy |
|  | 190D two\_pointers easy |
|  | 558D sorting + implementation medium(?) |
|  | 524D two\_pointers + greedy easy |
|  | 721C topological\_sort + bfs + bellman\_ford easy |
|  | 721D greedy easy(?) |
|  | 722C union\_find easy |
|  | 722D trie + binary\_search easy |
|  | 56C stack easy |
|  | 455D sqrt\_decomposition medium |
|  | 723E dfs\_and\_similar easy |
|  | 723D dfs\_and\_similar easy |
|  | 663E dp + bitmask medium(or hard !?) |
|  | 687E graphs + scc medium |
|  | 141E graphs + mst medium |
|  | 718C matrix + segment medium |
|  | 329C constructive or probability medium |
|  | 329B bfs easy |
|  | 93E dp easy |
|  | 332C data\_structure easy |
|  | 174C segment easy |
|  | 348C sqrt easy |
|  | 639D data\_structure easy |
|  | 400E segment easy |
|  | 601D hash medium |
|  | 498D segment medium |
|  | 331B2 segment medium |
|  | 387E segment or fenwick medium |
|  | 46E dp easy |
|  | 231E tartan easy |
|  | 220E two\_pointers easy |
|  | 724D greedy easy |
|  | 724C implementation medium |
|  | 372D two\_pointers or HLD + dfs\_and\_similar medium |
|  | 226C fibonacci\_numbers + matrix\_exponentiation medium |
|  | 45C segment easy |
|  | 650D fenwick medium |
|  | 341D fenwick medium |
|  | 294E dfs\_and\_similar easy |
|  | 219E data\_structures easy |
|  | 191E segment or fenwick + binary\_search easy(!?) |
|  | 121E segment medium |
|  | 121D two\_pointers + tof medium |
|  | 121B add\_hoc easy |
|  | 121A implementation + math easy |
|  | 300E number\_theory + binary\_search easy |
|  | 414D graphs + trees + two\_pointers easy |
|  | 89C pointers easy |
|  | 398D sqrt\_decomposition medium |
|  | 452F hash hard(?) |
|  | 594D number\_theory + fenwick medium |
|  | 607D number\_theory + segment + dfs\_ans\_similar medium |
|  | 266E segment + combinatorics medium |
|  | 256E segment easy |
|  | 547E segment + aho\_corasick medium |
|  | 91E segment + cht medium |
|  | 455E segment + cht medium |
|  | 717D matrix\_exponentiation easy |
|  | 76A dsu medium |
|  | 549G add\_hoc easy(?) |
|  | 279E add\_hoc + greedy medium |
|  | 132E min\_cost\_max\_flow medium |
|  | 549F ds medium |
|  | 295E segment medium |
|  | 713D 2d\_rmq + binary\_search medium |
|  | 338E segment + lazy medium |
|  | 94C add\_hoc easy |
|  | 458C ternary\_search medium |
|  | 724E max\_flow\_min\_cut + dp medium |
|  | 309C greedy medium |
|  | 309B two\_pointers exponentiation medium ! (what a tag !) |
|  | 425C dp medium |
|  | 626F dp medium |
|  | 626E ternary\_search medium |
|  | 727E hash easy(faghat moshkel ine ke double hash mikhad ) |
|  | 727D greedy easy |
|  | 731C graphs + dfs\_and\_similar easy |
|  | 731F math easy |
|  | 731D greedy medium |
|  | 731E dp easy |
|  | 173B bfs easy |
|  | 716E gooni + hash or centroid\_decomposition medium |
|  | 160E segment medium |
|  | 160C math + sorting easy |
|  | 377D greedy easy |
|  | 724G gaussian\_elimination + graphs medium |
|  | 732F dfs\_and\_similar medium(?) |
|  | 76E math easy |
|  | 316E3 segment + fibonacci\_numbers medium (ro be bala :D) |
|  | 33C dp easy |
|  | 507C dfs\_and\_similar easy |
|  | 377E cht medium |
|  | 436E data\_structures (segment or bit or even set(I think so !)) + greedy medium |
|  | 730H implementation easy |
|  | 704D l-r flow medium (roo be bala) |
|  | 476D math easy |
|  | 476C math easy |
|  | 555B greedy + sorting medium |
|  | 535C binary\_search + math easy |
|  | 514D binary\_search easy |
|  | 363D binary\_search easy |
|  | 514E matrix\_exponentiation medium |
|  | 283E segment medium |
|  | 36E euler\_tour medium |
|  | 710E dp medium |
|  | 725D greedy medium |
|  | 47D brute\_force easy |
|  | 54D dp + kmp easy |
|  | 60D math medium |
|  | 60E matrix\_exponentiation + fibonacci\_numbers medium |
|  | 625E math + geometry(moadele khat) medium |
|  | 367E dp medium |
|  | 66D math easy |
|  | 733D brute\_force easy |
|  | 733C greedy easy |
|  | 406A math easy |
|  | 406B math + constructive easy |
|  | 406C graph medium |
|  | 406D graph + ccw medium |
|  | 733F graph + trees + dsu medium |
|  | 568B dp easy |
|  | 568A math + number\_theory easy |
|  | 568C 2sat + greedy medium |
|  | 568D brute\_force(back\_track) + math(geometry) medium |
|  | 317A math easy |
|  | 317B dfs easy(roo be bala) |
|  | 317C dfs\_and\_similar + constructive medium |
|  | 95B dp + greedy easy |
|  | 671D graph(tree) + gooni + greedy medium |
|  | 516D dfs\_and\_similar + segment(haddeaghal man segment zadam :D) medium |
|  | 741D gooni + bitmask medium |
|  | 741A graphs + math easy |
|  | 741B graphs + dp easy |
|  | 741C graphs + dfs\_and\_similar + constructive medium |
|  | 590E aho\_corasick + graph\_matching hard(?) |
|  | 590D dp medium |
|  | 734F bitmask medium |
|  | 34C implementation easy |
|  | 660F cht medium |
|  | 758E guni medium |
|  | 540E fenwick medium |
|  | 660E combinatorics medium |
|  | 767D segment + binary\_search medium |
|  | 767C dfs easy |
|  | 765E dfs medium |
|  | 711E probability + math medium |
|  | 331C1 dp easy |
|  | 340C math easy |
|  | 340E dp medium |
|  | 359D dp medium |
|  | 437D dsu easy |
|  | 366C dp easy |
|  | 305D dp medium |
|  | 305E grundy medium |
|  | 305C math easy |
|  | 158E dp medium |
|  | 115B greedy easy |
|  | 766D dsu medium |
|  | 766E trie + dfs medium |
|  | 468B greedy easy |
|  | 446B dp + math medium |
|  | 267B fleury algorithm easy |
|  | 245E implementation + math easy |
|  | 444E binary\_search + graphs medium |
|  | 451D math medium |
|  | 761E constructive medium |
|  | 323C segment medium |
|  | 761D greedy + sorting easy |
|  | 63E dp medium |
|  | 75C math easy |
|  | 676E math(polynomial) medium |
|  | 92E dsu medium |
|  | 92D binary\_search easy |
|  | 439D binary\_search easy |
|  | 528D fft + dp medium \* |
|  | 763A dfs easy |
|  | 763B constructive easy |
|  | 599D math + combinatorics medium |
|  | 272D math + combinatorics + sorting easy |
|  | 729E greedy medium |
|  | 601B data\_structure medium |
|  | 430A constructive easy |
|  | 496D binary\_search medium |
|  | 578C ternary\_search easy |
|  | 439C constructive medium |
|  | 451C brute\_force easy |
|  | 271C constructive easy |
|  | 581D brute\_force medium |
|  | 71B math easy |
|  | 518E greedy hard(man kheili codam kasif bud haddeaghal !) |
|  | 518D dp medium |
|  | 518C implementation easy |
|  | 120I greedy medium |
|  | 120H maximum\_matching easy |
|  | 120J tof! medium |
|  | 120G implementation easy |
|  | 224C data\_structure easy |
|  | 313D dp medium |
|  | 313E greedy + data\_structure medium |
|  | 538F graphs(trees) + segment easy |
|  | 113B kmp easy |
|  | 125E graphs + binary\_search hard(?) \* |
|  | 125D constructive hard(codam kheili kasif bud mojaddadan !) |
|  | 125C combinatorics easy |
|  | 124E dp hard \* |
|  | 124D number\_theory + geometry medium |
|  | 124C dsu easy |
|  | 762C kmp easy |
|  | 59D constructive + greedy easy |
|  | 59C strings + greedy easy |
|  | 29D dfs medium |
|  | 14E dp easy |
|  | 762E sorting + ordered\_set medium |
|  | 237D graphs + greedy + constructive easy |
|  | 353C math easy |
|  | 204B data\_structure easy |
|  | 442B probability medium |
|  | 234C dp easy |
|  | 231C binary\_search + two\_pointers easy |
|  | 567D binary\_search + sorting easy |
|  | 229A brute\_force easy |
|  | 111A math easy |
|  | 182D hashing + brute\_force easy |
|  | 182D constructive + dfs\_and\_similar medium |
|  | 313C greedy + sorting + math easy |
|  | 759C segment easy |
|  | 759B dp easy |
|  | 759A dfs\_and\_similar + math easy |
|  | 9E greedy + graphs medium |
|  | 367D dp(bitmask) + binary\_search medium |
|  | 367B segment easy |
|  | 758D dp easy |
|  | 364D probabilities + math medium |
|  | 643F dp hard \* |
|  | 74D segment hard \* |
|  | 279D dp medium |
|  | 357D number\_theory medium |
|  | 356C greedy + implementation medium |
|  | 755E constructive + graphs medium |
|  | 755D fenwick medium |
|  | 238A math easy |
|  | 283C dp + graphs medium |
|  | 258C number\_theory medium |
|  | 448D binary\_search easy \* |
|  | 768G guni medium \* |
|  | 768E grundy easy |
|  | 768B math easy |
|  | 768C implementation + math easy |
|  | 15D dp + stack + sorting easy |
|  | 35E sweep\_line(?) medium |
|  | 498E matrix\_exponentiation + bitmask hard |
|  | 498C number\_theory + maximum\_matching medium |
|  | 245D math easy |
|  | 288C math medium \* |
|  | 549C games + math easy \* |
|  | 359C math easy |
|  | 549H binary\_search easy |
|  | 466D greedy(jalebe hame dp zadan !) medium |
|  | 98D constructive medium |
|  | 97B divide\_and\_conquer + constructive medium |
|  | 776C math easy |
|  | 776D graphs (2sat or check kardane 2 bakhshi boodan) easy |
|  | 776E number\_theory (phi) easy |
|  | 650E graphs + trees + dsu medium \* |
|  | 777E segment or fenwick + sorting easy |
|  | 777D greedy easy |
|  | 777C dp easy |
|  | 778A binary\_search + strings easy |
|  | 778B math + string easy |
|  | 385E matrix easy |
|  | 778C guni medium |
|  | 150E centroid\_decomposition\_of\_tree + segment medium!(valla bara man asoon bood faghat hey time mishod :/) |
|  | 779C sorting easy |
|  |  |
|  |  |
|  | gym : |
|  | http://codeforces.com/group/qcIqFPYhVr/contest/203881/problem/J centroid easy |
|  | http://codeforces.com/group/qcIqFPYhVr/contest/203881/problem/P centroid hard |
|  | http://codeforces.com/group/qcIqFPYhVr/contest/203881/problem/X centroid hard |
|  | http://codeforces.com/group/qcIqFPYhVr/contest/203881/problem/U bruteforce + bitmask medium |
|  | http://codeforces.com/gym/100712/attachments/download/3454/acm-amman-collegiate-programming-contest-en.pdf problem H -> 'Bridge Tree' |
|  | http://codeforces.com/gym/100676/attachments/download/3333/acm-arabella-collegiate-programming-contest-en.pdf problem H -> 'Bridge Tree' |
|  | 101016D HLD medium |
|  | http://codeforces.com/problemset/gymProblem/100796/E segment medium |
|  | http://codeforces.com/gym/101026 problem\_E segment medium |
|  | http://codeforces.com/gym/100589/problem/A sqrt\_decomposition + dfs medium |
|  | http://codeforces.com/gym/100623/problem/B segment easy |
|  | 100199B l-r flow easy |
|  |  |
|  | topcoder : |
|  | div2 contests : |
|  | 457\_1 implementation easy |
|  | https://community.topcoder.com/stat?c=problem\_statement&pm=14304 dp easy |
|  | https://community.topcoder.com/stat?c=problem\_statement&pm=14305 bitmask easy |
|  |  |
|  |  |
|  | div1 contests : |
|  | 692\_1 binary search + floyd warshall or scc easy |
|  | https://community.topcoder.com/stat?c=problem\_statement&pm=2987&rd=5862 game\_theory + bitmask + grundy\_numbers hard |
|  | 600\_easy math(binary\_numbers - in didgahesh ke masale ro bara bit ha , joda hal koni) easy |
|  |  |
|  | quera : |
|  | https://quera.ir/course/assignments/825/problems/2505 two\_pointers easy |
|  | https://quera.ir/course/assignments/825/problems/2511 dp easy |
|  | https://quera.ir/course/assignments/840/problems/2562 dp + probability easy |
|  | soalaye dore : |
|  | https://goo.gl/xOeqh4 man ba segment zadam valla ! medium(?) |
|  | https://goo.gl/UkmDYO sqrt medium |
|  | https://goo.gl/h5EN9y parallel\_binary\_search + dsu medium |
|  |  |
|  |  |
|  | spoj : |
|  | http://www.spoj.com/problems/MKTHNUM/ segment medium(?) |
|  | http://www.spoj.com/problems/DQUERY/ mo's or persistent\_segment\_tree or segment\_tree easy |
|  | http://www.spoj.com/problems/FINDSR/ kmp medium(?) |
|  | http://www.spoj.com/problems/SUBXOR/ trie easy |
|  | http://www.spoj.com/problems/PHONELST/ trie easy |
|  | http://www.spoj.com/problems/DYNACON1/ dynamic\_connectivity easy |
|  | http://www.spoj.com/problems/DYNACON2/ dynamic\_connectivity easy |
|  | http://www.spoj.com/problems/GSS1/ segment easy |
|  | http://www.spoj.com/problems/SUBLEX/ suffix array + binary search easy |
|  | http://www.spoj.com/problems/PT07X/ matching easy |
|  | http://www.spoj.com/problems/QTREE5/ centroid easy |
|  | http://www.spoj.com/problems/MATCHING/ matching easy |
|  | http://www.spoj.com/problems/QTREE/ HLD easy |
|  | http://www.spoj.com/problems/XXXXXXXX/ segment hard(?) |
|  | http://www.spoj.com/problems/FASTFLOW/ max\_flow easy |
|  | http://www.spoj.com/problems/SUBST1/ suffix array easy |
|  | http://www.spoj.com/problems/QTREE2/ LCA easy |
|  | http://www.spoj.com/problems/GSS3/ segment easy |
|  | http://www.spoj.com/problems/MATGAME/ grundy\_numbers medium(?) |
|  | http://www.spoj.com/problems/IM/ max\_flow medium |
|  | http://www.spoj.com/problems/MKTHNUM/ fractional cascading + segment\_tree + binary search |
|  | medium |
|  | http://www.spoj.com/problems/FREQ2/ Mo's algorithm easy |
|  | http://www.spoj.com/problems/DICT/ trie easy |
|  | http://www.spoj.com/problems/SHPATH/ dijkstra easy |
|  | http://www.spoj.com/problems/GSS2/ segment + lazy\_propagation hard |
|  | http://www.spoj.com/problems/QTREE4/ centroid\_decomposition\_of\_tree hard |
|  | http://www.spoj.com/problems/QTREE3/ HLD easy |
|  | http://www.spoj.com/problems/TRAFFICN/ dijkstra easy |
|  | http://www.spoj.com/problems/MST/ kruskal super easy |
|  | http://www.spoj.com/problems/PA06ANT/ matrix\_exponentiation + graph easy |
|  | http://www.spoj.com/problems/KQUERYO/ segment\_tree easy |
|  | http://www.spoj.com/problems/MORSE/ trie medium |
|  | http://www.spoj.com/problems/XMAX/ gaussian\_elimination easy |
|  |  |
|  |  |
|  | codechef : |
|  | https://www.codechef.com/problems/ASTRGAME grundy\_numbers medium(?) |
|  | https://www.codechef.com/problems/GCD2 math + gcd easy |
|  | https://www.codechef.com/problems/MXZERO easy |
|  | https://www.codechef.com/problems/UTMOPR easy |
|  | https://www.codechef.com/problems/MAXDIFF sorting easy |
|  | https://www.codechef.com/problems/ENTEXAM sorting easy |
|  | https://www.codechef.com/problems/NOTINCOM easy |
|  | https://www.codechef.com/problems/MNMX easy |
|  | https://www.codechef.com/problems/FRUITS easy |
|  | https://www.codechef.com/problems/MISSP easy |
|  | https://www.codechef.com/problems/SIMPSTAT easy |
|  | https://www.codechef.com/problems/FDIVGAME grundy\_numbers medium |
|  | https://www.codechef.com/problems/PPTREE trie easy |
|  | https://www.codechef.com/problems/KGP13G trie easy |
|  | https://www.codechef.com/problems/MRXCHOC/ max\_flow medium |
|  | https://www.codechef.com/CDJA2016/problems/KM01 graphs + connected\_components + sorting easy |
|  | https://www.codechef.com/problems/IN01 math easy |
|  | https://www.codechef.com/problems/SUBLCM two\_pointers + sieve easy |
|  | https://www.codechef.com/OCT16/problems/BGQRS segment + lazy medium |
|  |  |
|  |  |
|  | BWC : |
|  | http://black.white-crow.ir/problemset/problem/DO\_RE\_1 sieve + number\_theory easy |
|  |  |
|  | uva : |
|  | https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&page=show\_problem&problem=470 backtrack medium(?) |
|  | https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&page=show\_problem&problem=45 geometry + convex\_hull medium |
|  | https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&page=show\_problem&problem=2483 trie easy |
|  | https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&page=show\_problem&problem=1620 kmp easy |
|  | https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&page=show\_problem&problem=3904 probability easy(?) |
|  | https://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&page=show\_problem&problem=1246 topological\_sort easy |
|  |  |
|  | sgu : |
|  | http://acm.sgu.ru/problem.php?contest=0&problem=507 gooni easy |
|  |  |
|  | IPSC : |
|  | https://ipsc.ksp.sk/2009/real/problems/l.html HLD easy ! |
|  |  |
|  |  |
|  | main : |
|  | http://main.edu.pl/en/archive/oi/2/drz graphs easy |
|  | http://main.edu.pl/en/archive/oi/13/okr z\_function + set easy(?) |
|  |  |
|  | lightoj : |
|  | http://lightoj.com/volume\_showproblem.php?problem=1279 gaussian\_elimination + number\_theory medium |
|  |  |
|  | hackerrank: |
|  | https://www.hackerrank.com/challenges/greedy-florist greedy easy |

### Ad Hoc

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Ad Hoc | [Sleepy Cow Herding](https://starcoder.org/usaco/USACO-2019-Jan-Bronze/#problem-2-Sleepy-Cow-Sorting) | Easy |

### Simulation

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Simulation | [Hoof, Paper, Scissors](https://starcoder.org/usaco/USACO-2017-Jan-Bronze/#problem-2-hoof-paper-scissors) | Easy |
| Simulation | [The Bovine Shuffle](https://starcoder.org/usaco/USACO-2017-Dec-Bronze/#problem-2-the-bovine-shuffle) | Easy |
| Simulation | [Milk Measurement](https://starcoder.org/usaco/USACO-2017-Dec-Bronze/#problem-3-milk-measurement) | Easy |
| Simulation | [Mixing Milk](https://starcoder.org/usaco/USACO-2018-Dec-Bronze/#problem-1-mixing-milk) | Easy |
| Simulation | [Shell Game](https://starcoder.org/usaco/USACO-2019-Jan-Bronze/#problem-1-shell-game) | Easy |

### Brute Force

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Brute Force | [Triangles](https://starcoder.org/usaco/USACO-2020-Feb-Bronze/#problem-1-triangles) | Easy |
| Brute Force | [The Blocked Billboard](https://starcoder.org/usaco/USACO-2017-Dec-Bronze/#problem-1-the-blocked-billboard) | Easy |
| Brute Force | [Team Tic Tac Toe](https://starcoder.org/usaco/USACO-2018-Open-Bronze/#problem-1-team-tic-tac-toe) | Easy |
| Brute Force | [Photoshoot](https://starcoder.org/usaco/USACO-2020-Jan-Bronze/#problem-2-photoshoot) | Easy |

### Greedy

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Greedy | [Boyer-Moore’s Majority Vote Algorithm](https://starcoder.org/programming/boyer-moore-majority-vote/) | Easy |
| Greedy | [Mad Scientist](https://starcoder.org/usaco/USACO-2020-Feb-Bronze/#problem-2-mad-scientist) | Hard |

### Coordinate Compression

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Coordinate Compression | [Range Sum](https://starcoder.org/algorithm/algorithm_coordinate_compression_range_sum/) | Easy |
| Sliding Window | [Min/Max of Sliding Window](https://starcoder.org/algorithm/algorithm_sliding_window_min_max/) | Medium |

# Intermediate Algorithm (USACO Silver Level)

## Video Solutions for Silver Questions

I selected several interesting questions in USACO Silver and made explained videos for them.  
Checkout my solution videos for selected [Silver Questions](https://starcoder.org/usaco_silver_video/)

### Prefix Sum

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Prefix Sum | [Prefix Sum](https://starcoder.org/programming/algorithm-prefixsum/) | Easy |
| Prefix Sum | [Subarray Sums Divisible by K](https://starcoder.org/programming/algorithm-prefixsum/) | Easy |
| Prefix Sum | [Maximum Subarray Sum](https://starcoder.org/programming/algorithm-subarraysum/) | Easy |
| Prefix Sum | [Div7](https://starcoder.org/usaco/USACO-2016-Jan-Silver/#problem-2-subsequences-summing-to-sevens) | Easy |
| Prefix Sum | [breedcounting](https://starcoder.org/usaco/USACO-2015-Dec-Silver/#problem-3-breedcounting) | Easy |
| Prefix Sum | [hoof paper scissors](https://starcoder.org/usaco/USACO-2017-Jan-Silver/#problem-2-hoof-paper-scissors) | Easy |
| Prefix Sum | [maxcross](https://starcoder.org/usaco/USACO-2017-Feb-Silver/#problem-2-why-did-the-cow-cross-the-road-ii) | Easy |
| Prefix Sum | [homework](https://starcoder.org/usaco/USACO-2017-Dec-Silver/#problem-1-my-cow-ate-my-homework) | Easy |
| 2D Prefix | [paintbarn](https://starcoder.org/usaco/USACO-2019-Feb-Silver/#problem-2-painting-the-barn) | Medium |

### Binary Search

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Binary Search | [Binary Search C++](https://starcoder.org/programming/algorithm-binarysearch-cpp/) | Easy |
| Binary Search | [Maximum Median](https://starcoder.org/algorithm-maximum-medium/) | Easy |
| Binary Search | [B.Books](https://starcoder.org/programming/algorithm-binary-search/#b-books) | Easy |
| Binary Search | [C.Cellular Network](https://starcoder.org/programming/algorithm-binary-search/#c-cellular-network) | Easy |
| Binary Search | [tractor](https://starcoder.org/usaco/USACO-2013-Feb-Silver/#problem-2-tractor) | Easy |
| Binary Search | [Cross Country Skiing](https://starcoder.org/usaco/USACO-2014-Jan-Silver/#problem-2-cross-country-skiing) | Easy |
| Binary Search | [Auto Complete](https://starcoder.org/usaco/USACO-2014-Feb-Silver/#problem-1-auto-complete) | Easy |
| Binary Search | [Angry Cows](https://starcoder.org/usaco/USACO-2016-Jan-Silver/#problem-1-angry-cows) | Easy |
| Binary Search | [Haybales](https://starcoder.org/usaco/USACO-2016-Dec-Silver/#problem-1--counting-haybales) | Easy |
| Binary Search | [Cow Dance Show](https://starcoder.org/usaco/USACO-2017-Jan-Silver/#problem-1-cow-dance-show) | Easy |
| Binary Search | [Secret Cow Code](https://starcoder.org/usaco/USACO-2017-Jan-Silver/#problem-3-secret-cow-code) | Easy |
| Binary Search | [Convention](https://starcoder.org/usaco/USACO-2018-Dec-Silver/#problem-1-convention) | Easy |
| Binary Search | [Berry Picking](https://starcoder.org/usaco/USACO-2020-Jan-Silver/#problem-1-berry-picking) | Easy |
| Binary Search | [Loan Repayment](https://starcoder.org/USACO-2020-Jan-Silver/#problem-2-loan-repayment) | Easy |
| Binary Search | [Social Distancing](https://starcoder.org/usaco/USACO-2020-Open-Silver/#problem-1-social-distancing) | Easy |

### Two Pointers

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Two Pointers | [Circulation of Affection](https://starcoder.org/programming/an-impassioned-circulation-of-affection/) | Easy |
| Two Pointers | [Cow Lineup](https://starcoder.org/usaco/USACO-2011-Nov-Silver/#problem-2-cow-lineup) | Easy |
| Two Pointers | [Diamond Collector](https://starcoder.org/usaco/USACO-2016-Open-Silver/#problem-2-diamond-collector) | Easy |
| Two Pointers | [Why did the Cow Cross the Road?](https://starcoder.org/usaco/USACO-2017-Feb-Silver/#problem-1-why-did-the-cow-cross-the-road) | Easy |
| Two Pointers | [Paired Up](https://starcoder.org/usaco/USACO-2017-Open-Silver/#problem-1-paired-up) | Easy |
| Two Pointers | [Diamond Collector](https://starcoder.org/usaco/USACO-2016-Open-Silver/#problem-2-diamond-collector) | Hard |
| Two Pointers | [Layout](https://starcoder.org/usaco/USACO-2005-Dec-Gold-P3-Layout/) | Hard |

### DFS

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| DFS | [Grid Paths](https://starcoder.org/programming/grid-paths/) | Easy |
| FloodFill | [Flood Fill](https://starcoder.org/programming/flood-fill/) | Easy |
| FloodFill | [Grid Path](https://starcoder.org/programming/grid-paths/) | Easy |
| DFS | [Luxury River Cruise](https://starcoder.org/usaco/USACO-2013-Open-Silver/#problem-3-luxury-river-cruise) | Easy |
| FloodFill | [Switching on the Lights](https://starcoder.org/usaco/USACO-2015-Dec-Silver/#problem-1-switching-on-the-lights) | Easy |
| FloodFill | [Build Gates](https://starcoder.org/usaco/USACO-2016-Jan-Silver/#problem-3-build-gates) | Easy |
| FloodFill | [Moocast](https://starcoder.org/usaco/USACO-2016-Dec-Silver/#problem-3-moocast) | Easy |
| FloodFill | [Why did the Cow Cross the Road III](https://starcoder.org/usaco/USACO-2017-Feb-Silver/#problem-3-why-did-the-cow-cross-the-road-iii) | Easy |
| FloodFill | [Where’s Bessie](https://starcoder.org/usaco/USACO-2017-Open-Silver/#problem-3-wheres-bessie) | Easy |
| FloodFill | [Mootube](https://starcoder.org/usaco/USACO-2018-Jan-Silver/#problem-3-mootube) | Easy |
| FloodFill | [Mooyo Mooyo](https://starcoder.org/usaco/USACO-2018-Dec-Silver/#problem-3-mooyo-mooyo) | Easy |
| FloodFill | [Icy Perimeter](https://starcoder.org/usaco/USACO-2019-Jan-Silver/#problem-2-icy-perimeter) | Easy |
| DFS/Flood Fill | [Fence Planning](https://starcoder.org/usaco/USACO-2019-Open-Silver/#problem-3-fence-planning) | Easy |
| DFS | [Milk Visits](https://starcoder.org/usaco/USACO-2019-Dec-Silver/#problem-3-milk-visits) | Easy |
| DFS/Flood Fill | [Wormhole Sort](https://starcoder.org/usaco/USACO-2020-Jan-Silver/#problem-3-wormhole-sort) | Easy |

### BFS

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| BFS | [BFS Distance](https://starcoder.org/programming/bfs-graph/) | Easy |
| BFS | [Piggyback](https://starcoder.org/usaco/USACO-2014-Dec-Silver/#problem-1-piggyback) | Easy |
| BFS | [Closing the Farm](https://starcoder.org/usaco/USACO-2016-Open-Silver/#closing-the-farm) | Easy |
| BFS | [Muliplayer Moo](https://starcoder.org/usaco/USACO-2018-Open-Silver/#problem-3-multiplayer-moo) | Easy |
| BFS | [The Great Revegetation](https://starcoder.org/usaco/USACO-2019-Feb-Silver/#problem-3-the-great-revegetation) | Easy |
| BFS | [Clock Tree](https://starcoder.org/usaco/USACO-2020-Feb-Silver/#problem-3-clock-tree) | Easy |

### Tree

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Tree | [Tree Traversal](https://starcoder.org/programming/tree-traversals/) | Easy |
| Tree | [Tree Algorithms](https://starcoder.org/programming/tree-algorithms/) | Easy |

### Sorting

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Sorting | [Field Reduction](https://starcoder.org/usaco/USACO-2016-Open-Silver/#problem-1-field-reduction) | Easy |
| Sorting | [The Bovine Shuffle](https://starcoder.org/usaco/USACO-2017-Dec-Silver/#problem-3-the-bovine-shuffle) | Easy |
| Sorting | [Rental Service](https://starcoder.org/usaco/USACO-2018-Jan-Silver/#problem-2--rental-service) | Easy |
| Sorting | [Out of Sorts](https://starcoder.org/USACO-2018-Open-Silver/#problem-1-out-of-sorts) | Easy |
| Sorting | [Mountain View](https://starcoder.org/usaco/USACO-2019-Jan-Silver/#problem-3-mountain-view) | Easy |

### Geometry

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Geometry | [Load Balancing](https://starcoder.org/algorithm2/usaco/USACO-2016-Feb-Silver/#problem-2-load-balancing) | Easy |
| Geometry | [Triangles](https://starcoder.org/usaco/USACO-2020-Feb-Silver/#problem-2-triangles) | Easy |
| Geometry | [Field Reduction](https://starcoder.org/usaco/USACO-2016-Open-Silver/#problem-1-field-reduction) | Easy |

# Hard Algorithms (USACO Gold Level)

## Video Solutions for Gold Algorithms

[Dynamic Programming](https://starcoder.org/usaco_dp)

### Easy DP

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| DP | [Climbing Stairs](https://starcoder.org/programming/dp-climbing-stairs/) | Easy |
| DP | [Unique Paths II](https://starcoder.org/programming/dp-unique-path/) | Easy |
| DP | [House Robber](https://starcoder.org/programming/dp-house-robber/) | Medium |
| DP | [Marathon](https://starcoder.org/usaco/USACO-2014-Dec-Silver/#problem-2-marathon) | Medium |

### Backtrack

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Backtrack | [Subsets](https://starcoder.org/programming/backtrack-subsets/) | Medium |
| Backtrack | [Permutation](https://starcoder.org/programming/backtrack-permutation/) | Easy |
| Backtrack | [Combination](https://starcoder.org/programming/backtrack-combination-sum/) | Easy |
| Backtrack | [Knights Tour](https://starcoder.org/programming/knights-tour-problem/) | Easy |
| Backtrack | [Hamiltonean Cycles](https://starcoder.org/programming/hamiltonean-cycles/) | Easy |
| Backtrack | [Palindrome Partitioning](https://starcoder.org/programming/backtrack-palindrome-partitioning/) | Easy |

### Subsequences (LIS, LCS)

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| DP | [Passing Notes](https://starcoder.org/algorithm/algorithm_dp_passing_notes/) | Medium |
| DP | [LIS (Longest Increasing Sequence, N^2 Algorithm)](https://starcoder.org/algorithm/algorithm_dp_LIS/) | Medium |
| Binary Search | [LIS (Longest Increasing Sequence, NLogN Algorithm)](https://starcoder.org/algorithm/algorithm_dp_LIS_nlogn/) | Hard |
| LIS,DFS | [Missile Defense System](https://starcoder.org/algorithm/algorithm_dp_LIS_nlogn/) | Hard |
| DP | [LCS (Longest Common Sequence)](https://starcoder.org/algorithm/algorithm_dp_LCS/) | Medium |
| LIS | [Sister Cities](https://starcoder.org/algorithm/algorithm_dp_LIS_sister_cities/) | Medium |
| LIS | [Chorus](https://starcoder.org/algorithm/algorithm_dp_LIS_chrous/) | Medium |
| LIS | [Best Time to Buy and Sell Stock](https://starcoder.org/programming/dp-buy-sell-stock/) | Easy |
| Backtrack, LIS | [Missile Defense System](https://starcoder.org/algorithm/algorithm_dfs_missile_defense_system/) | Medium |
| LIS/LCS | [LCIS(Longest Common Increasing Sequence)](https://starcoder.org/algorithm/algorithm_dp_LCIS/) | Hard |

### Monotonous Stack

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Monotonous Stack | [Stock Spanner](https://starcoder.org/programming/algorithm-monotonous-stack/) | Easy |

### Knapsack DP

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| 0/1 | [0/1 Knapsack Problem](https://starcoder.org/algorithm/algorithm_dp_0_1_knapsack/) | Easy |
| Unbounded | [Currency System](https://starcoder.org/algorithm/algorithm_dp_currency_system/) | Easy |
| Unbounded | [Number Combination](https://starcoder.org/algorithm/algorithm_dp_number_system/) | Easy |
| Multiple | [Multiple Knapsack Problem](https://starcoder.org/algorithm/algorithm_dp_multiple_knapsack/) | Easy |
| Mixed | [Mixed Knapsack Problem](https://starcoder.org/algorithm/algorithm_dp_mixed_knapsack/) | Medium |
| Multiple Constraits | [Knapsack with Multiple Constraits](https://starcoder.org/algorithm/algorithm_dp_multiple_constraits/) | Hard |
| Group | [Group Knapsack](https://starcoder.org/algorithm/algorithm_dp_group_knapsack/) | Easy |

### Interval DP

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Interval DP | [Stone Merger](https://starcoder.org/algorithm/algorithm_interval_dp_illustrated_stone_merge/) | Easy |
| Interval DP | [Bonus Binary Tree](https://starcoder.org/algorithm/algorithm_interval_dp_scored_binary_tree/) | Easy |

### DP on Finite Automata

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| DP on FA | [Corn Yard](https://starcoder.org/algorithm/algorithm_dp_fa_corn_yard/) | Medium |
| DP on FA | [Little King](https://starcoder.org/algorithm/algorithm_dp_fa_little_king/) | Medium |
| DP on FA | [Password Counting](https://starcoder.org/algorithm/algorithm_dp_fa_password_counting/) | Medium |
| DP on AC FA | [DNA Repairing](https://starcoder.org/algorithm/algorithm_dp_fa_dna_repairing/) | Medium |

### State Compression DP

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| State Compression DP | [Treasure Hunt](https://starcoder.org/algorithm/algorithm_dp_fa_treasure_hunt/) | Medium |
| State Compression DP w/ Rolling Array | [Artillary Position](https://starcoder.org/algorithm/algorithm_dp_fa_artillery_position/) | Medium |
| State Compression DP | [Traveling by Stagecoach](https://starcoder.org/usaco/POJ-2686-Traveling-by-Stagecoach/) | Medium |

### DP on Digits

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| DP on Digits | [Sum of Digits](https://starcoder.org/usaco/DP-digit-dp-sum-of-digits/) | Medium |

### DP & Combinatorics

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Combinatorics | [KK’s Chemicals](https://starcoder.org/algorithm/algorithm_dp_kk_chemicals/) | Medium |

### DP on a Tree

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| DP on a Tree | [A Party Without a Boss](https://starcoder.org/algorithm/algorithm_dp_tree_a-party-without-boss) | Medium |
| DP on a Tree | [Course Selection](https://starcoder.org/algorithm/algorithm_dp_tree_course-selection/) | Medium |

### Graph

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Dijkstra | [Dijkstra’s Shortest Path](https://starcoder.org/programming/dijkstra's-shortest-path/) | Medium |
| Shortest Path Faster Algorithm (SPFA) | [Layout](https://starcoder.org/usaco/USACO-2005-Dec-Gold-P3-Layout/) | Medium |

### TopologySort

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| TopologySort | [Milking Order](https://starcoder.org/video/usacovideo-usaco-2018-gold-open-p2/) | Medium |

### Point Update Range Query

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| BIT | [Cow Haircut](https://starcoder.org/video/usacovideo-usaco-2020-gold-open-p1) | Medium |

### Euler Tour, LCA

| **Algorithm** | **Example** | **Level** |
| --- | --- | --- |
| Euler Tour, LCA,BIT | [Cow Land](https://starcoder.org/video/usacovideo-usaco-2019-gold-feburary-p1) | Medium |
| Euler Tour, LCA, Custom Hash | [Milk Visits](https://starcoder.org/video/usacovideo-usaco-2019-gold-december-p2/) | Medium |

# Hard Algorithms (USACO Platinum Level)

| **Algorithm** | **Problem** | **Solution** | **Level** |
| --- | --- | --- | --- |
| Segment Tree | [CSES 1649](https://cses.fi/problemset/task/1649/) | [Dynamic Range Minimum Queries](https://starcoder.org/cses/segtree-cses-1649/) | Easy |
| Monotonous Queue, Segment Tree | [USACO 2013 Open, Gold](http://www.usaco.org/index.php?page=viewproblem2&cpid=285) | [Max Query with Monotonous Queue](https://starcoder.org/usaco/USACO-2017-open-285/) | Medium |
| Bitmask DP | [USACO 2014 December, Gold](http://www.usaco.org/index.php?page=viewproblem2&cpid=494) | [Guard Mark](https://starcoder.org/USACO-2014-Gold-December-P1/) | Easy |
| Advanced DP | [USACO 2014 November P1, Gold](http://www.usaco.org/index.php?page=viewproblem2&cpid=193) | [Balanced Cow Breeds](https://starcoder.org/video/USACO-2012-Gold-November-P1) | Easy |
| Game strategy using DP | USACO 2010 Gold January P1 | [Taking Turns](https://starcoder.org/video/USACO-2010-Gold-Jan-P1) | Medium |
| Bitmask DP | [USACO 2015 Gold January P2](http://www.usaco.org/index.php?page=viewproblem2&cpid=515) | [Moovie Mooving](https://starcoder.org/video/USACO-2015-Gold-December-P1) | Medium |
| BIT, DFS | [USACO 2017 Platinum January P1](http://usaco.org/index.php?page=viewproblem2&cpid=696) | [Promotion Counting](https://starcoder.org/USACO-2017-Platinum-January-P1) | Medium |
| DP, MaxQueue | USACO 2010 Gold Open | [Cow Hopscotch](https://starcoder.org/USACO-2010-gold-open-CowHopscotch) | Medium |
| DP, Sliding Window | USACO 2019 Platinum January P3 | [Train Tracking 2](https://starcoder.org/video/usacovideo-usaco-2019-platinum-january-p3/) | Medium |
| DP | USACO 2016 Platinum Open P1 | [262144](https://starcoder.org/video/usacovideo-usaco-2016-platinum-open-p1) | Medium |
| DP, BIT/SEG | USACO 2015 Gold Feburary P1 | [Cow Hopscotch](https://starcoder.org/video/usacovideo-usaco-2015-gold-feburary-p1/) | Hard |
| HLD | CSES 2134: Path Queries II | [Path Queries II](https://starcoder.org/CSES-2134-path-queries-ii/) | Easy |
| Merge From Small to Large | USACO 2018 Open P3 | [Disruption](https://starcoder.org/video/usacovideo-usaco-2018-platinum-open-p3/) | Easy |
| LCA DSU | USACO 2018 Open P3 | [Disruption](https://starcoder.org/video/usacovideo-usaco-2018-platinum-open-p3-dsu/) | Easy |

<https://train.usaco.org/usacoprob3?S=0820&a=0MYQjFcMd6o>

<https://train.usaco.org/usacoprob3?a=0MYQjFcMd6o&amp;S=0819>