**Solutions**

**Problem: CF 426B**

Origin: http://codeforces.com/contest/834/submission/29027867

Learned: sweeps can be easier done with ordered map and set which erases.

**Problem: CF 426C (Math)**

Origin: http://codeforces.com/blog/entry/53567

Learned: use math before brute forcing solutions, number theory problems are hard, they almost never involve calculating divisors… math with cubic roots and non cubic roots binary search on 10^9, for math problems look at tricks!

**Contest Summary 426C: Please think of doing it with a trick before using brute force methods**

**Problem: CF 427B**

Origin: <http://codeforces.com/blog/entry/53588>

Learned: Simplify and do it with less coding if you can 20 minutes too long

**Problem: CF 427C (Partial Sums)**

Origin: <http://codeforces.com/blog/entry/53588>

Learned: psums implementation good, coding fast, thinking too long go faster!

2) Alternate way of implementing with rectangles = psum[x2][y2] – psum[x1-1][y2] – psum[x2][y1-1] + psum[x1-1][y1-1]

**Problem: CF 427D (DP Palindromes)**

Origin: http://codeforces.com/contest/835/problem/D

Learned: 1) Learned how to do Hashing in O(n) 2) Fix hashing it has problems in it later 3) Can do DP on palindromes in O(n^2) time and memory 4) use recursive relationships for everything to find the mark and f[][] 5) sum it all together!

**Contest Summary CF 427**: Think of DP before using hashing, learned hashing with palindromic substrings, code faster = good -> great

Segment Tree Practice

**Problem: Haybales USACO Platinum December 2015 RMQ, Lazy, Sum Seg Tree**

Origin: http://www.usaco.org/current/data/sol\_haybales\_platinum\_dec15.html

Learned: Practice with rmq and sum lazy propagation. Argh long long be careful bc 1LL << 60 not 1 << 60 (long long) error make sure to fix. Argh forgot to update both in lazy propagation. Bug! Very tricky

**Problem: Promotion Counting USACO Platinum January 2017**

Origin: <http://www.usaco.org/current/data/sol_promote_platinum_jan17.html>

Learned: 1) Tree and segment / bit tree problem clearly n = 100000 2) tree + segment/bit always involves a certain preorder/inorder traversal which puts the objects in a proper ordering 3) preorder traversal = simplifies tree problem into an array problem 4) then sort the states in decreasing order then use bit to update the result.

**Problem: CF 424D (greedy)**

Origin: <http://codeforces.com/contest/831/problem/D>

Learned: 1) Make observations first, does not have to be DP problem even if it looks like DP 2) Make observation that all the segments have to be in order 2) DP Solution also correct 3) memorization may TLE so do not use 4) DP solution just make sure you initialize with big enough integer 0x3f to small needs INT\_MAX as INF

**Problem: CF 424A (brute force)**

Origin: <http://codeforces.com/contest/831/problem/A>

Learned: 1) took 20 minutes too long 2) could have done it with one for loop sweep.

**Problem: CF 424C (prefix sums)**

Origin: <http://codeforces.com/contest/831/status/C>

Learned: 1) you can also use prefix sums to solve this problem

**Contest Summary CF 423:** Good:Coding really fast in first two problems! Nice job! Last problem Problem C = got stuck because TLE bc reading in 2 million strings Bad: Chuck you need to remember to use fast i/o when you are switching to cin/cout. Switch to cin/cout

**Problem: CF 423C (string problem, union find problem)**

Origin: <http://codeforces.com/contest/828/status/C>

Learned: 1) switch to cin / cout with ios::sync\_with\_stdio(false) 2) can do it with other method of sorting it and then processing strings too 3) can also use the union-find get\_next method to solve it even faster (cool) 4) whenever there is a continuous line segment and you are updating intervals with same value you should think of union - find

**Problem: CF 423D (greedy graph problem)**

Origin: http://codeforces.com/contest/828/problem/D  
  
Learned: 1) make a greedy observation that the tree should just be a star – graph which is optimal. 2) coding is really simple.

**Problem: CF 275D (BIT + segment + prefix sums)**

Origin: <http://codeforces.com/blog/entry/14417>

Learned: 1) release and do it for and with God!! 2) rest!!! :D :D 10x more productive at least 3) work in spurts and work hard!!! 4) make cool observations about and that all 1’s have to be 1’s and all 0’s have to have at least 1 0 in interval 5) can split it up into individual bits to solve 6) then realize you can process all the 1’s first to evaluate and make sure the 1’s condition is satisfied 7) then with bit shift operators you can verify if 0’s situation can be satisfied 8) then you solve!

**Problem: CF 242E (Segment Tree + XOR + Lazy Updates)**

Origin: <http://codeforces.com/contest/242/status/E/page/4?order=BY_CONSUMED_TIME_ASC>

Learned: 1) when you xor you are doing it bit by bit so you can split it into the 20 bits of the xor, tree[x][i] stands for the xor of the value at node x (which represents a range of values) and at bit i 2) after you decide to update bit by bit, you realize that summing is really easy with lazy propagation and so is building 3) only update is a little tricky and requires you to realize that you can only update when there value you are pushing has a bit at this bit value. 4) write helper push function to update.

Problem: CF 422C (Sorting + Query Updates)

Origin: http://codeforces.com/contest/822/status/C

Learned: 1) Important observation is that you don’t need to check both sides!!! You can only check if ex < sy because the other situation is covered 2) Then you can process it in forms of queries with l and r, queries. 3) You can solve it with sorting, with pq, and with two pointer.

**Contest Summary CF 422**: Good: Got my highest rank 601 Solved Problem D with guess and check Bad: Was not very focused, not present, kind of gave up on solving problem C LOL, should have focused on finding a different sort instead of debugging, I need to think a little bit more before I code, find the trick (don’t just jump on a solution when you see one), 2 improvements, 1) be present + 10 min 2) don’t give up after you get WA, 3) work on finding the trick/optimal solution – 10 min 4) be greedy when you can 5) for sorting questions/query questions always think psums, two pointers, binary search. Also, think pq solution and sorting according to l and r index/intervals. Maybe getf() solution also. Finally, DP

**Problem: KQuery SPOJ (Offline Query w/Segment + BIT Tree) (#2 Vector Segment Tree)**

Origin: <http://www.spoj.com/problems/KQUERY/>

Learned: 1) switch to scanf and AC in one go 2) coding very fast (but could be faster) 3) got the general idea of segment tree problems 4) basic = straight implementation of seg tree 5) other = preprocess/ don’t preprocess and sort in some arbitrary order to get the valid ordering then, use segment tree to update values in increasing/decreasing order. Harder problems = figure out when to sort and when to add stuff.

**Problem: CF 182D (Offline Query + Sieve)**

Origin: <http://codeforces.com/contest/301/problem/D>

Learned: 1) Really hard but nice problem 2) have to make the observation that we use offline queries starting from l-1 with bit updates to get the answer… sweep from right to left3) find the update/transition graph with sieve of erasthonas 4) sweep and update to find answer. Bit indexing a little tricky.

**Problem: CF 216E (Offline Query + BIT)**

Origin: http://codeforces.com/contest/369/problem/E

Learned: 1) Got the transformation that I need to solve for pl+1, pr-1. 2) Solve by sorting in a specific order. 3) Use offline query again!!!! If hard to do online -> offline query 4) After realizing that I can use offline to query I switch to processing one by one in left 5) use bit tree to update! 6) 10^6 index bounds usually indicates that you have to sweep across the index somehow.

Problem: CF 225C (Segment Tree + Tree)

Origin: <http://codeforces.com/contest/383/problem/C>

Learned: 1) Bit update on intervals tricky!!! 2) +val, -val, tricky 3) solve by keeping two arrays of even and odd height level bits 4) After this, and dfs, it becomes simple lazy segment/bit tree problem 5) updating ranges of consecutive elements by adding intervals. Cool way http://codeforces.com/blog/entry/10476

Problem: CF 200D (Segment Tree + Tree)

Origin: <http://codeforces.com/contest/343/problem/D>  
  
Learned: 1) You cannot just update ranges with one segment tree in a naïve manner. 2) I got the idea but implementation was all wrong 3) Just use two segment trees one to keep track of the last subtracted point and one to keep track of the last updated range. 4) use two range maximum query, one normal, and one with range updates

Problem: CF 169E (Segment Tree + BIT + Tree)

Origin: <http://codeforces.com/contest/276/status/E>

Learned: 1) I got the right idea I just missed an implementation detail where you are double counting when you update both the root and the tree 2) tried different ways to alleviate this issue, cute trick of not updating the tree all the way helps 3) write classes for LCA, segment tree, sparse table etc. helps a lot in coding contest when you have to have multiple seg trees 4) got the problem idea where you update the tree and the root separately 5) another way of solving it involves breaking the tree into chains and updating each chain/bit correspondingly using bit range updates +l +x, + r+1 –x etc. really cool!

Problem: CF 428B Greedy

Origin: <http://codeforces.com/blog/entry/53815>

Learned: 1) Greedy solution very tricky 2) made observation that 7, 6, 5, 4 all should use 4 because only they can use all of 4 3) missed greedy observation that 3 should also use 4 because of the fact that 3 can only be split up into 2 2’s or a 4 4) this leads to only 2 cases left to consider (where I had 3 which made it impossible to do) greedy!!!! 2 and 1’s 5) 2’s should obviously use the two side 2’s and then 4 + 1 while 1’s have to be greater than 2 \* 4 + 1 + 2 left

**Contest Summary CF 428:** This contest was really bad for me… performed at 1400 rating again. 1) May need to consider taking more contests… we will see on Wednesday 2) Biggest problem was that I slept too late only got like 5 hours of sleep and yeah was so tired when I woke up… started at 7:35 barely could read problem statements 3) Vow: not to come back next summer and have to hang out ugh 4) Vow: sleep earlier and treat contests more seriously + 7 hours of sleep before 5) Good did not waste time after renap 6) When school starts = treat life + sin + happiness + contests + everything more seriously… this is last time this contest is not legit. 7) Only got one problem C right ☹ 8) at least 1600 next round treat seriously = sleep

Learned: 1) read the question carefully ugh. Number one was so easy, if got right rating = + 100 2) Problem B greedy was tricky! 3) clique + gcd/number theory/math relearn!

Problem: CF 232E Bit/Segment + Tree

Origin: http://codeforces.com/contest/396/status/C

Learned: 1) h[v] to update and h[u] to update if each level is different from the rest 2) two bit trees/segment trees to update ranges 3) the TLE will happen if you use lazy segment tree because of the fact that segment tree build takes a lot of time 4) can do it with normal segment tree and switching query and update 5) can also do it with bit tree. Range update and single element query 6) use bit tree and also use height to account for different level differences. 7) Gosh darn it wasted so much time debugging and it was just a simple mistake… need to learn how to debug better 8) think twice code once 9) should have gotten the height implementation and implemented with bit tree always 10) in coding think twice code once – 20 minutes per implementation… (5 plan 15 coding, timer) 11) debug: look through problem 3-> logic 5 -> code 2-> debug 10 12) debug pretty fast with template + binary search

Problem: CF 52C Segment Tree Lazy

Origin: http://codeforces.com/contest/52/problem/C

Learned: 1) RMQ implementation took way too long 2) tiny tiny bug in order of push which heavily skewed results… 3) coding speed = fast enough but very inaccurate = slow down. Think twice code once. 4) sscanf and gets() 5) circle -> array 6) rmq structure built and fully functional now. 7) better coding. And debugging = more focused

Problem: CF 145E Segment Tree Lazy

Origin: <http://codeforces.com/contest/145/problem/E>

Learned: 1) Looking at 5 solutions good pretty much does same thing, learned more about classes and using struct, and also building segment tree iteratively + N etc. 2) I got most of it except for key observation that you use an opposite… like opposite increasing value… keep that in mind that for these problems when you flip something it might all become opposite so you might have to store opposite. 3) try more examples to test 4) ask yourself what does each node store to solve this problem 5) got the cool math formulas by contradiction why each case must be so 6) timer good 7) implement 12 minutes :D 8) write helper functions to push, modify, and swap 9) build, query mx[1], if within range swap, if out of range then push, query, and then merge. 10) can use the nodes as classes to update to make life easier 11) push function messes up because you can’t swap the nodes and then swap the children too (should just swap children). 12) gave up lol 13) UGHHHHHHH no the previous comments were wrong. The real reason why it did not work is because of Array out of bounds fk. My range update does update the ranges, and I push at top so it should be fine, and more importantly I combine and merge correctly so it should be fine. What is wrong is index out of bounds. For large test case error look into index out of bounds etc. large problems. Basically code more carefully Chuck.