

First thing to say is: W(T)Fs like this have happened before: 06, 08, ~10, ~12, 14, ~17, 18

Second meta question: if i only want to prepare for NAC, what should my takeaway from this be:

1. There are problems that you can train for, there are also problems that you can’t really train for by doing CF and etc

2. Some problems have to be solved by playing around with them

3. Binary search & brute force can be really hard, have a look at the problems on that guide.

Third takeaway: for probability problems involving exponential/complicated state space that are doable (which everyone on a contest is), they often have nice closed form potentials. Try to guess them.

~~A: Archery - 0/0 + 1/4~~

~~B: City Planning - 1/12 + 7/31~~

C: \shuffle{PerfectShuffle} - 1/3 + 1/2

~~D: Monkey Wars - 0/0 + 2/2~~

~~E: Singland - 4/16 + 4/9~~

~~F: Time is Money - 2/20 + 6/8~~

~~G: Juggling Troupe - 1/3 + 10/13~~

~~H: Mathemagicians - 11/45 + 20/72~~

~~I: Dalarna - 0/0 + 0/0~~

~~J: Slackingoffagain (0/12, 6/7 solves)~~

F: Time is Money - 2/20 + 6/8

Have undirected graph where edges have 2 costs, time & money

Find spanning tree that minimizes time \* money (aka. (sum of cost 1) \* (sum of cost 2))

Solution:

Let \lambda be a Lagrange multipliers

Pick some \lambda, minimize over time + money \* \lambda

Then ternary search / hill climb over \lambda until you get to the max of time \* money (might have to be an exhaustive traversal / recursive descent)

Theorem: if i have a point set (x\_i, y\_i), the minimizer of x\_i \* y\_i must lie on the convex hull

(false is maximize x\_i \* y\_i, counter example (1, 1), (2-eps, 1/2), (1/2, 2-eps)

Similar-looking problem that can be solved with convex hull: http://serjudging.vanb.org/wp-content/uploads/mobilization.pdf

Takeaway message: whenever you get asked 2 function optimization, try f(x) + \lambda g(x) if: 0) you have seen that combination of objectives before

1) you see other teams get it

2) you are in last hour and have nothing to do

H: Mathemagicians - 11/45 + 20/72

You have a binary string, you can at each step replace a letter by a neighbor’s value,

Check if you can get from S1 to S2

Sol: write a simulator that takes a arbitrary initial string, see what comes out.

Result: count # of locations that are different, this goes downward

special case everything same & everything different

G: Juggling Troupe - 1/3 + 10/13

Abelian sandpile on a path, with 0 & n + 1 as sinks

Add balls at the right one at a time, observe that the states are just segments of 1s, maintain those segments in a stack-ish thing,

(how to see this: write down n^2, see a pattern)

Alternatively, track where the 0s are: if you have a binary string and then you add a ball where someone already has one, it removes two zeroes and injects a zero at (L + r - index) (the removed zeros are the closest zeroes on both ends)

B: City Planning - 1/12 + 7/31

Grid, can build k things at each cell, the kth thing costs c\_k to build

Need to build a total of N things, minimize building cost + total distance of these things.

Greedy clearly works: each cell tracks what’s the next cost of building on it, then always take mincost to build.

Problem: N is 10^{12}.

Faster: bin search on the cost at which this greedy process stops

Decision function is: for cost limit L, are there N things to be built whose total cost is less than L

(loop through each k to find this)

Most fixed this by doing to int128, it’s actually possible to AC this using int64, highly recommend everyone to go back & solve this using 64-bit, just have to compare w. MAXVAL / y before you take x \* y.

D: Monkey Wars - 0/0 + 2/2

There are n cards, distributed among 2 people.

Each pair of cards have a win probability.

They pick random cards, winner keeps both

Find probability that person 1 ends up with all the cards.

Observations:

1. What if all cards are fair, 50/50: probability of a random walk starting at x on a unit path from 0..n ends up at n before 0. This prob is x / n.

2. What if there is a card that beats everything: then whoever starts with that card wins.

3. What if there are 3 cards that beat everything, but are fair among themselves: then whoever gets all 3 of those win.

Other thing you can do is run size 5 simus, and realize that

WinProb(00111) = WinProb(00101) + WinProb(00010)

Aka. the win chances vectors are additive

So you guess that every card has a value v[i], and prob(S) = v[S]

This v[i] value ends up being.... If i start with a card, play against a random card, goto that card if that card beats me, repeat... at a random moment of time (after arbitrarily long), what is the probability that I hold card i.

To find this stationary, just take P^{2^{16}} where P is the probability transition matrix

E: Singland - 4/16 + 4/9

Given a string, find the expected length until it occurs as a substring in a random string.

Strange: answer is always integer.

How to even simulate this:

For a string, build a KMP failure function / automata, that tracks state of `this is the longest prefix of my string that has already been generated’

Then do some probability propagation on this automata.

~~Don’t submit this solution, run it locally for alphabet size 3.~~

You can actually solve this system of linear equations mod 10000: it’s almost DAG like, so you can actually solve it...

Run this code on some small data, observe that a lot of solutions are:

A^{L}

And then with A = 10, see that many of the rest have few digits

A^{L} + A^{L\_1}...

Realize L\_1 is the failure function value.... t

So answer is \sum\_{i: length i suffix = length i prefix} A^{i}

https://math.stackexchange.com/questions/3149271/markov-chain-approach-for-expected-amount-of-time-when-the-monkey-types-a-sequen?noredirect=1&lq=1

J: Slackingoffagain (0/12, 6/7 solves)

Step 1: by brute force, answer is always 0

Step 2: simplify brute force by saying that grid[i][j] = s[i + j]

Can prove that if s is overlap-tree (no two overlapping substrings of s are the same), then s works for the above construction

Then set s to be s(0) = 1, s(1) = s(0) flip(s(0))....

s[i] = parity(popcount(i))

Before we talk about I, I want to mention that the psychology of doing these ‘the easy problem is impossible’ type contests is very different than your typical NA contest. You want to have some problem that keeps the keyboard occupied, so 2 people are bored, and can actually go solve stuff.

I: Dalarna - 0/0 + 0/0

Given a point in a simple polygon (with <= 50 vertices), find total perimeter visible after k (k <= 5) reflections.

Standard: pick a sequence of sides that you ray bounces on, reflect polygon across those. Then get rid of things that obscure vision, and do visibility.

Easier way: if for angle a\_1 & a\_2, my ray hits the same sequence of sides, then for any a between a\_1 & a\_2, my ray also hits that sequence. Use this to do recursive descent on angle of ray.

Richard’s code: https://pastebin.com/7kVAvY0c

A: Archery - 0/0 + 1/4

Make a lot of observations, write a lot of code, refer to IOI solutions.