# December Contest Prep

Daniel Wisdom

December 15, 2017

## 1 General Contest Strategies

USACO gives partial credit for every test case you solve, so this is just as important as actually solving full problems. The first test case is always the sample case. Test cases 2-4 are also very small inputs that a naive solution should work on. The test cases then get larger and you will need a fast algorithm. Because all test cases are weighted, submitting naive algorithms to every problem and getting 4/10 is a 400/1000, while fully solving a single problem only gets 333/1000. This means you should always submit a brute force algorithm to every problem and get those easy points.

## 2 Algorithm Types

Every problem is unique, but USACO has common types of algorithms that can work to solve problems. If you are stumped on a problem, think about whether each strategy might work. Platinum problems sometimes require a combination of two of these strategies. This is not a full list of all possible strategies, you will need to entirely make up a unique algorithm for some problems. Especially in higher divisions, the solution is often a well-known algorithm with some optimization to finish in time.

1. **Greedy Algorithm**: Can you make the seemingly best decision at every step and still get the right answer at the end?
2. **Brute Force**: If the data is small, can you try all possible combinations in time?
3. **Line Sweep**: Can you sort and process the data in some way that makes the problem easier? Can you traverse a tree or graph to make the problem easier?
4. **Graph Theory**: Can you view the data as a graph or tree? This allows you to use shortest paths, max flow, min spanning trees, and many other well-known algorithms.
5. **Binary Search**: Can you easily check is a guess is above or below the answer? If so you can quickly find the answer. Note that we sometimes binary search on some other parameter that we later use to calculate the answer.
6. **Dynamic Programming**: Can you break the problem down into sub problems that are easier to calculate? This is very common in USACO.
7. **Segment Trees**: These are so common in USACO that they deserve their own category. Segment trees allow you to find the min, sum, and other associative operations in $O(\log N)$ time and also allow updates in $O(\log N)$ time. Also consider using BITs, which can match a segment tree for any operation with an inverse, such as product, sum, and xor.

# 3 Examples

When you start a problem, see if you can apply an algorithm you know or can easily come up with. If you can't come up with anything, find a naive solution that would work and then look for ways to speed up this solution. If none of this works, try restating the problem in a different way and repeat.

## 3.1 Radio Contact

USACO Gold, January 2016

Farmer John has lost his favorite cow bell, and Bessie the cow has agreed to help him find it! They both fan out and search the farm along different paths, but stay in contact via radio so they can keep in touch with each-other. Unfortunately, the batteries in their radios are running low, so they want to plan their movements so as to conserve power, by trying to stay always within a short distance apart. Farmer John starts at location (fx,fy) and plans to follow a path consisting of N steps, each of which is either 'N' (north), 'E' (east), 'S' (south), or 'W' west. Bessie starts at location (bx,by) and follows a similar path consisting of M steps. Both paths may share points in common. At each time step, Farmer John can either stay put at his current location, or take one step forward along his path, in whichever direction happens to be next (assuming he has not yet reached the final location in his path). Bessie can make a similar choice. At each time step (excluding the first step where they start at their initial locations), their radios consume energy equal to the square of the distance between them.

Please help FJ and Bessie plan a joint movement strategy that will minimize the total amount of energy consumed up to and including the final step where both of them first reach the final locations on their respective paths.

## 3.2 Moocast

USACO Gold, December 2016

Farmer John's N cows ($1 \leq N \leq 1000$) want to organize an emergency "moo-cast" system for broadcasting important messages among themselves. Instead of mooing at each-other over long distances, the cows decide to equip themselves with walkie-talkies, one for each cow. These walkie-talkies each have a limited transmission radius, but cows can relay messages to one-another along a path consisting of several hops, so it is not necessary for every cow to be able to transmit directly to every other cow.

The cows need to decide how much money to spend on their walkie-talkies. If they spend \$X, they will each get a walkie-talkie capable of transmitting up to a distance of $\sqrt{X}$. That is, the squared distance between two cows must be at most X for them to be able to communicate.

Please help the cows determine the minimum integer value of X such that a broadcast from any cow will ultimately be able to reach every other cow.

## 3.3 High Card Low Card

USACO Platinum, December 2015

Bessie the cow is a huge fan of card games, which is quite surprising, given her lack of opposable thumbs. Unfortunately, none of the other cows in the herd are good opponents. They are so bad, in fact, that they always play in a completely predictable fashion! Nonetheless, it can still be a challenge for Bessie to figure out how to win. Bessie and her friend Elsie are currently playing a simple card game where they take a deck of 2N cards, conveniently numbered 1...2N, and divide them into N cards for Bessie and N cards for Elsie. The two then play N rounds, where in each round Bessie and Elsie both play a single card. Initially, the player who plays the highest card earns a point. However, at

one point during the game, Bessie can decide to switch the rules so that for the rest of the game, the player who plays the lowest card wins a point. Bessie can choose not to use this option, leaving the entire game in "high card wins" mode, or she can even invoke the option right away, making the entire game follow the "low card wins" rule.

Given that Bessie can predict the order in which Elsie will play her cards, please determine the maximum number of points Bessie can win.

### 3.4 Cow Rectangles

USACO Gold, January 2015

The locations of Farmer John's N cows ($1 \leq N \leq 500$) are described by distinct points in the 2D plane. The cows belong to two different breeds: Holsteins and Guernseys. Farmer John wants to build a rectangular fence with sides parallel to the coordinate axes enclosing only Holsteins, with no Guernseys (a cow counts as enclosed even if it is on the boundary of the fence). Among all such fences, Farmer John wants to build a fence enclosing the maximum number of Holsteins. And among all these fences, Farmer John wants to build a fence of minimum possible area. Please determine this area. A fence of zero width or height is allowable.

### 3.5 Load Balancing

USACO Platinum, February 2016

Farmer John's N cows are each standing at distinct locations (x1,y1)...(xn,yn) (x1,y1)...(xn,yn) on his two-dimensional farm ($1 \leq N \leq 100,000$), and the xi's and yi's are positive odd integers of size at most 1,000,000. FJ wants to partition his field by building a long (effectively infinite-length) north-south fence with equation x=a (a will be an even integer, thus ensuring that he does not build the fence through the position of any cow). He also wants to build a long (effectively infinite-length) east-west fence with equation y=b, where b is an even integer. These two fences cross at the point (a,b), and together they partition his field into four regions. FJ wants to choose a and b so that the cows appearing in the four resulting regions are reasonably "balanced", with no region containing too many cows. Letting M be the maximum number of cows appearing in one of the four regions, FJ wants to make M as small as possible. Please help him determine this smallest possible value for M.

### 3.6 Promotion Counting

USACO Platinum, January 2017

The cows have once again tried to form a startup company, failing to remember from past experience that cows make terrible managers! The cows, conveniently numbered 1...N ($1 \leq N \leq 100,000$), organize the company as a tree, with cow 1 as the president (the root of the tree). Each cow except the president has a single manager (its "parent" in the tree). Each cow $i$ has a distinct proficiency rating, $p(i)$, which describes how good she is at her job. If cow $i$ is an ancestor (e.g., a manager of a manager of a manager) of cow $j$, then we say $j$ is a subordinate of $i$.

Unfortunately, the cows find that it is often the case that a manager has less proficiency than several of her subordinates, in which case the manager should consider promoting some of her subordinates. Your task is to help the cows figure out when this is happening. For each cow $i$ in the company, please count the number of subordinates $j$ where $p(j) > p(i)$.