# 2017 Virginia Tech High School Contest Solution Outlines

The Judges

Dec 10, 2017

## A – Shiritori – First solved at 0:03

**Problem**

Given a sequence of up to 100 000 words $W_i$ played in a game of Shiritori, find out if the game was played according to the rules:

- the last letter of the $W_{i-1}$ is equal to the first letter of $W_i$.
- no words repeated

## A – Shiritori – First solved at 0:03

### Solution

- Condition that last letter is equal to first letter of next word is checked easily by storing a reference to the last word and performing a character comparison.
- Condition of "no repeated words" requires a way to determine, in constant time, whether a word has already been played.
- Must use a set/map/hash set/hash map (e.g. Java `HashSet` or `HashMap`) that provides $\mathcal{O}(1)$ or $\mathcal{O}(\log n)$ lookup time.
- Cannot use a data structure such as an ArrayList. Why? To determine if $W_i$ has already been played, requires $i-1$ comparisons. This is a total of $\sum_{i=1\ldots n} i-1 = \frac{n(n-1)}{2} = \mathcal{O}(n^2)$ which takes too long for 100 000 words.

## C – Saving Princess Peach – First solved at 0:06

**Problem**

Given an unsorted list of up to 200 numbers that contains a subset of "obstacles" numbered $0 \ldots N-1$, find the obstacles not listed and output them in sorted order. Also output the number of distinct elements in the list given.

## C – Saving Princess Peach – First solved at 0:06

### Solution

- Place all numbers listed in a set (to eliminate duplicates), then loop over $0 \ldots N - 1$ and output those numbers not in the set.
- Unlike in problem A, using a less efficient data structure (e.g., a loop over the elements in a list) to check for set membership would be acceptable here because of the small problem size.

## K – Provinces and Gold – First solved at 0:16

**Problem**

Given the number of Coppers $C$, Silvers $S$, and Golds $G$ in a hand played in the board game Dominion (URL), find out which treasure and victory cards a player can buy that turn.

## K – Provinces and Gold – First solved at 0:16

### Solution

- Compute the monetary value: $v = 3G + 2S + C$.
- If $v \geq 8$, output `Province or`.
- Else if $v \geq 5$, output `Duchy or`.
- Else if $v \geq 2$, output `Estate or`.
- If $v \geq 6$, output `Gold`.
- Else if $v \geq 3$, output `Silver`.
- Else output `Copper`.

## F – The Weight Of Words – First solved at 0:21

**Problem**

Given the sum $w$ of the weights of a word's letters, and given its length $l$, find a word with that weight and that length. Otherwise, output `impossible`.

## F – The Weight Of Words – First solved at 0:21

### Solution

- Lightest words are aaaaa. . . , Heaviest words are zzzz. . . . , hence any combination where either $w < l$ or $w > 26l$ is impossible.
- Otherwise, there are multiple ways of constructing such a words.
- For instance, could greedily add z until the remaining letters can be written as a + one other letter.
- Or, could start with aaa. . . .aaa and distribute the weight by incrementing each letter (wrapping around) until all weight is distributed.

## E – Through the Grapevine – First solved at 0:21

### Problem

Given an undirected social graph with $n$ nodes and $m$ edges ($n, m \leq 100\,000$) of people who know each other, find out to how many people a rumor will have spread after $d$ days ($d \leq 10\,000$). A person $k$ that has heard the rumor on day $i$ will begin spreading it on day $i + 1$, but only if they have heard it from $C_k$ distinct people.

## E – Through the Grapevine – First solved at 0:21

### Solution

- Use a BFS (breadth-first search). Record, for each person, the day at which they've heard the rumor and whether they are spreading the rumor already.
- When following an edge, if the recipient is not already spreading the rumor, decrement their remaining skepticism, letting them spread the rumor the next day, and add them to the BFS queue.
- BFS queue contains the people that *start* spreading the rumor at any given day. Hence, every node is put into the queue at most once: complexity $\mathcal{O}(m)$. No need to record who someone is hearing the rumor from.
- Side note: complexity does not depend on number of days $d$.
- Letting everyone spread the rumor over and over should time out.

## B – Stop Card – First solved at 0:30

**Problem**

Given a randomly shuffled deck of $n$ cards with values $a_i$. Start drawing cards from the top. Consider an optimal stopping algorithm that skips the first $c$ drawn cards, then selects the first card following that is better than the maximum of the skipped choices, what is the expected value that is eventually chosen? NB: if no better card appears after the skipped ones, you are stuck with last choice offered/last card of the deck.

## B – Stop Card – First solved at 0:30

### Solution

- Related to so-called "Secretary" problem of maximizing the likelihood of making the best choice when selecting from among $n$ candidates that are presented sequentially and where each candidate must either be rejected or accepted (in which case the selection is over).
- Insight: expected value does not depend on the numbers written on the cards, just on their relative rank, i.e. $E = \sum_{r=1\ldots n} a_r \; P(r)$ where $P(r)$ is the likelihood that the card with "rank" $r$ is being picked. ($r$ goes from 1 to $n$, with $r = 1$ is the smallest card and $r = n$ the highest card - note rank is usually defined in the other direction.)

# B – Stop Card – First solved at 0:30

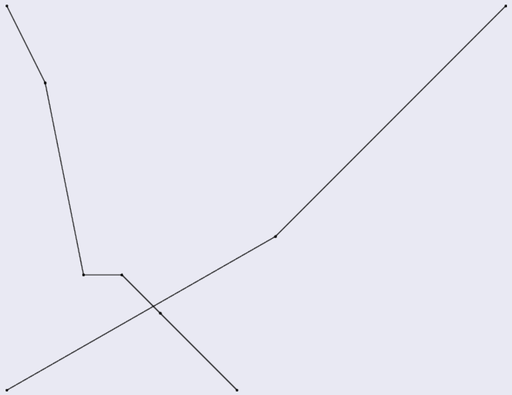
## Solution

- Break down $P(r) = P_l(r) + \sum_{i=c\ldots\min(n-1,r)} P_i(r)$ where $P_l(r)$ is the probability of the card with rank $r$ being picked by default and $P_i(r)$ is the probability of card with rank $r$ being picked in the $i^{\text{th}}$ position.
- $P_l(r) = \frac{c}{n-1}\frac{1}{n} = \frac{c}{n(n-1)}$: for a card to be chosen by default, it must be last in the deck (probability $\frac{1}{n}$) and the maximum of the remaining $n-1$ cards must be among the $c$ skipped cards (probability $c\frac{1}{n-1}$).
- $P_i(r) = \frac{\binom{r-1}{i}}{\binom{n-1}{i}}\frac{c}{i}\frac{c}{n}$. Similar argument: cards appears in position $i$ with probability $\frac{1}{n}$, the maximum of the first $i$ cards must be within the first $c$ cards (probability: $\frac{c}{i}$). In addition, all cards before position $i$ must be smaller than the card with rank $r$, or else $r$ would not be chosen at $i$. There are $\binom{n-1}{i}$ ways to select $i$ cards from the remaining $n-1$, and $\binom{r-1}{i}$ ways to select $i$ from the $r-1$ cards that are smaller than $r$.
- Handle $c = 0$ specially (answer is average $(\sum a_i)/n$ in this case).

## J – Traveling Monk – First solved at 0:40

### Problem

Given 2 piecewise linear monotonic functions, one increasing ($A(t)$) and one decreasing ($D(t)$) with $a$ and $d$ pieces, resp., find the smallest $t$ at which they intersect.



## J – Traveling Monk – First solved at 0:40

### Solution

- Two approaches: binary search or sweep.
- Binary search is probably easiest: The function $f(t) = a(t) - d(t)$ is monotonically decreasing, can use binary search to look for where $f(t) = 0$. $a(t)$ and $d(t)$ can be computed in $\mathcal{O}(\max(a, d))$.
- Also possible by keeping track of overlapping ascending and descending segments, processing them in order of time until crossing segment is found.

## H – Coconut Splat – First solved at 0:45

**Problem**

A counting-out game with $n$ players ($2 \leq n \leq 100$) that uses a rhyme and in which players that lose each round move through stages: split folded hands in fists, open fists, remove hand, out. Find out who wins the game.

## H – Coconut Splat – First solved at 0:45

### Solution

Multiple ways of solving it, small problem size allows simulation.

- Can represent players as lists of pairs $(p, s)$ where $1 \leq p \leq n$ is the number of the player and $s$ the state of the hand ($s \in \{\text{FOLDED}, \text{FIST}, \text{PALM}\}$)
- Rhyme selects player $(i + s - 1) \bmod n$ after player $i$.
- if $(p, \text{FOLDED})$ is hit, remove it and insert two copies of $(p, \text{FIST})$ at current position.
- if $(p, \text{FIST})$ is hit, replace with $(p, \text{PALM})$.
- if $(p, \text{PALM})$ is hit, remove.
- Repeat until only one hand is left.

Complexity of insert/remove operations does not matter due to small problem size.

## L – Criss-Cross – First solved at 1:19

**Problem**

Given up to 1 000 line segments whose endpoints have integer coordinates, count the number of intersections or touch points.
Additional stipulations include:

- ignore intersection points created by pairs of segments that overlap or touch at more than 1 point.
- may assume that a pair of segments that touches at end points does not have 3 collinear points.

## L – Criss-Cross – First solved at 1:19

### Solution

The intended solution was this:

- Insight: if the end points have integer coordinates, then any touch or intersection points will have rational coordinates $(\frac{x}{w}, \frac{y}{w})$.
- Can reduce each fraction to obtain a unique representation for each point (divide by $\gcd(x, w)$ and $\gcd(y, w)$, respectively), and store points in hash map.
- Can use Python's `fractions.Fraction` class, for instance.

### Side Note

Unfortunately, our test data was too weak to break all solutions that attempted to use inexact computations using `double` or `long double`. We'll post the test data we used, but we will fix the problem before we upload it for Kattis to make sure all double approaches fail.

## G – RA Duty Scheduler – First solved at 1:51

### Problem

There are 2 resident advisors (RAs) needed for each of $d$ days. Each RA has a list of days on which they can work. Find a schedule for a month that ensures all needed slots are filled and that minimizes the maximum number of days any RA works that month.

### Aside

The problem author was asked to implement this in real life.

## G – RA Duty Scheduler – First solved at 1:51

### Solution

Classic application of maximum flow.

- Consider $k$ the maximum number of days any RA will serve. Build a standard flow network.
    - Connect source to each day with capacity 2.
    - Connect each day to the RAs available on that day with capacity 1.
    - Connect each RA to sink with capacity $k$.
- If maximum flow from src to snk is $2 \times d$, all slots are assigned. If not, must increase $k$.
- Minimize $k$ - can use binary search, but for problems this small can simply loop $k = 1 \ldots$.... and stop at first feasible $k$.

## D – Swish – First solved at 3:34

**Problem**

Given 20 cards from the game Swish, find the minimum possible number of swishes that can be formed with all cards, or output -1 is not possible.
A swish is cycle of cards so that a ring in one card overlaps with a circle in the next one.
Each card has one ring and one circle in one of 12 positions on a $3 \times 4$ grid. Can rotate and flip the cards.

## D – Swish – First solved at 3:34

### Solution

First, build a routine to test whether a subset of cards can form a swish.

- Assign each dot and ring a "color" so that dots and rings which can be rotated to have the same position are the same color. There are 4 possible colors, representing positions $(0, 2, 9, 11)$, $(3, 5, 6, 8)$, $(1, 10)$, and $(4, 7)$, resp.
- It does not suffice to simply check whether the number of dots and rings of each color is equal. (Consider cards $\{(0, 2), (0, 11)\}$).
- However, this is a necessary condition which can rule out many subsets of cards.
- A second condition is that there cannot be more dots of a given color than there are possible positions for that color.
- If both conditions are met, use exhaustive exploration via backtracking to find out if there is a possible swish for that subset.

## D – Swish – First solved at 3:34

### Solution (cont'd)

- To determine what is the fewest number of swishes to cover all the cards, use subset dynamic programming with the subsets which make exactly 1 swish.
- During the subset DP, we must also check that the swish we are adding is disjoint from the other subsets that make swishes, because each card cannot be a part of more than one swish.

## I – Alien Codebreaking – Not solved

### Problem

"Break" an alien encryption that uses a shift-based cipher combined with a one-time pad. One-time pad is computed using repeated application of linear congruence $f(x) = (33x + 1) \mod 2^{20}$ in an $X \times X$ grid ($X \leq 250\,000$), then summing the columns and concatenating the result expressed in base 10. In the next step, the base 10 number is converted to base 27 to given the one-time pad for the shift cipher.
This was our reach/challenge problem.

## I – Alien Codebreaking – Not solved

### Solution

The solution involves multiple parts.

- Insight: there are only $2^{20}$ possible values for $f^k(0)$ and they repeat (are periodic). Compute them all and store them in an array $a_i$. Cost: $\mathcal{O}(\text{MOD})$.
- Can thus compute the column sum $C_1$ of column 1 in $\mathcal{O}(X)$.
- The remaining column sums are computed using $C_{i+1} = (33C_i + X) \bmod 2^{20}$ also in $\mathcal{O}(X)$.
- The result is a decimal number with approximately 1.5M digits. `java.math.BigInteger` takes about 2 minutes for this conversion.
- Devise a fast algorithm to convert it to base 27 using a divide-and-conquer algorithm that uses a number-theoretic transform (NTT) as a subroutine.

## I – Alien Codebreaking – Not solved

### Solution (cont'd)

- Divide-and-conquer: split number $n = \overline{ab}$ in two parts $\bar{a}$ and $\bar{b}$ of roughly equal length. If $\bar{b}$ has $k$ digits, then $n = 10^k \bar{a} + \bar{b}$.
- Can compute, in base 27, $C(n) = C(10^k) \times C(\bar{a}) + C(\bar{b})$.
- $C(\bar{a})$ and $C(\bar{b})$ are computed recursively (divide part).
- Stop dividing when number of digits is sufficiently small (or even 1).
- $C(10^k)$ can be computed recursively as well: $C(10^k) = C(10^{k/2}) \times C(10^{k-k/2})$ and $C(1_{10}) = 1_{27}$
- Must compute the multiplication of a $m$ bit base 27 number with a $n$ bit base 27 in $\mathcal{O}(\max(m,n)\log(\max(m,n)))$.
- Treat base 27 digits of both number as coefficients of a polynomial, and use an NTT-based fast polynomial multiplication routine. Then normalize the coefficients to be within the range of $0 \ldots 26$.
- Finally, apply the shift-based cipher to construct the output.