# The 2021 ICPC Asia Kunming Regional Programming Contest Contest Editorial

Prepared by xxx

April 17, 2022

# A. Amino Acids

**Task**

You are given $M$ kinds of amino acids, print the structural formula of all possible peptide chains, with a molecular mass not greater than $N$, which can be formed by the given amino acids.

## A. Amino Acids

- Sort all given amino acids by lexicographical order and let $a_i$ be the molecule mass of the $i$-th amino acids.
- Any peptide chain can be described by a sequence $p_1, p_2, \cdots, p_c$ $(p_i \leq M)$ and its molecule mass is $\sum a_{p_i} - 18(c-1)$.
- Through dfs, we can enumerate all possible solutions by lexicographical order.
- It is obvious that all structural formulas have the same width. We only need to discard the first and last 2 columns and print the peptide bond between two amino acids.

## B. Blocks

**Task**

You are given a $W \times H$ grid map and $n$ rectangles. In each turn, we randomly choose a rectangle and paint it black. Calculate the expected number of turns to paint the whole map black.

## B. Blocks

- We can use dynamic programming to calculate the expectation.
- Note that two states are considered the same if the same set of rectangles has been painted.
- Define $dp_S$ the expected number of turns to paint the whole map black when rectangles in $S$ have already been painted, then we can calculate $dp_S$ as below:
- $dp_S =$
$$\begin{cases} 0 & \textit{The map has been painted black already} \\ \frac{\sum dp_{S \,\cup\, \{i\}}}{n} + 1 & \textit{otherwise} \end{cases}$$
- The second transformation results in an equaion, and can be solved as $dp_S = \frac{\sum_{i \notin S} dp_{S \,\cup\, \{i\}} + n}{|S|}$.
- We can use $O(n)$ binary numbers to describe which grids are painted black in state $S$ and check if any state is in the first case within a time complexity of $O(2^n \cdot n)$.
- The overall time and space complexity are $O(2^n \cdot n)$.

## C. Cup of Water

**Task**

There is a variable $s$, which is initially $0$. In each turn we will randomly select a real number $t$ between $0$ and $x$ and let $s$ be $s + t$. Calculate the expected number of turns when $s > 1$ is satisfied.

## C. Cup of Water

- Let $P(n)$ be the possibility that $s > 1$ is not satisfied after $n$ turns.Then the answer is $\sum_{n\geq 0} P(n)$.
- Then the subproblem is:

**subproblem**

There are $n$ variables $v_1, v_2, \cdots, v_n$, each of which is a randomly selected real number between $0$ and $x$. Calculate the possibility when $v_1 + v_2 + \cdots + v_n \leq 1$ is satisfied.

- Here is a combinatorial way to solve this problem.
- Cut $1$ in $N$ $(N = \infty)$ tiny pieces, then each variable is a randomly selected integer between $1$ and $Nx$, and we can add another variable $v_{n+1}$ .Then the problem is:
- Calc the number of solutions of equation $v_1 + v_2 + \cdots + v_n + v_{n+1} = N, \forall i \leq 1 \leq v_i \leq Nx, v_{n+1} \geq 0$.

## C. Cup of Water

- This is a classic combinatorial problem, which can be solved by the principle of inclusion and exclusion.
- Forall $i \geq 0, xi \leq 1$, assume there is at least $i$ variables bigger than $Nx$. The number of ways is $\binom{n}{i} \cdot \binom{N - i \cdot xN}{n}$
- Note that $N = \infty$, then $\alpha N - c = \alpha N, \binom{N - i \cdot xN}{n} = \frac{(N - i \cdot xN)^n}{n!}$
- So the number of solutions is:
- $$\sum_{ix \leq 1} (-1)^i \frac{\binom{n}{i} \cdot ((1 - ix)N)^n}{n!}$$
- And the possibility is:
- $$\sum_{ix \leq 1} (-1)^i \frac{\binom{n}{i} \cdot ((1 - ix)N)^n}{n!(Nx)^n} = \sum_{ix \leq 1} (-1)^i \frac{\binom{n}{i} (\frac{1 - ix}{x})^n}{n!}$$

## C. Cup of Water

- Then the answer is:
- $$\begin{aligned}\sum_{n\geq 0} P(n) &= \sum_{n\geq 0}\sum_{ix\leq 1}(-1)^i\frac{\binom{n}{i}\left(\frac{1-ix}{x}\right)^n}{n!}\\ &= \sum_{ix\leq 1}\frac{(-1)^i}{i!}\sum_{n\geq 0}\frac{\left(\frac{1-ix}{x}\right)^n}{(n-i)!}\\ &= \sum_{ix\leq 1}\frac{(-1)^i}{i!}\left(\frac{1-ix}{x}\right)^i\sum_{n\geq i}\frac{\left(\frac{1-ix}{x}\right)^{n-i}}{(n-i)!}\\ &= \sum_{ix\leq 1}\frac{(-1)^i}{i!}\left(\frac{1-ix}{x}\right)^i\sum_{n\geq 0}\frac{\left(\frac{1-ix}{x}\right)^{n}}{n!}\\ &= \sum_{ix\leq 1}\frac{(-1)^i}{i!}\left(\frac{1-ix}{x}\right)^i e^{\frac{1-ix}{x}}\end{aligned}$$

## D. Divisions

**Task**

For any sequence $S_1, S_2, \cdots, S_n$, define $f(S)$ the number of ways to divide it into a non-increasing and a non-decreasing subsequence. You are asked to construct a sequence $S$ with $f(S) = k$, where $k$ is a given integer.

## D. Divisions

- $k = 0$
- Construct the sequence - $1, 1, 4, 5, 1, 4, 1, 1, 4, 5, 1, 4$.
- $k = 1$
- Construct the sequence - $1, 1, 4, 5, 1, 4$.
- $k > 1$
- Construct a non-decreasing sequence - $x_1 \cdots x_1, x_2 \cdots x_2, \cdots, x_m \cdots x_m$, where $x_i < x_{i+1}$ and $x_i$ appears $a_i$ times in the sequence. For a non-decreasing sequence $S$, the element in $S_B$ must be equal. The greatness of $S$ is equal to $(\sum_{i=1}^{m} 2^{a_i} - 1) + 1$. After knowing that, it is easy to construct the answer.

## E. Easy String Problem

**Task**

You are given a string with length $n$, and the size of the alphabet also is $n$.
There are $q$ queries. For a query, you are given two integers $l, r$ and you need to answer the number of different strings (which can be empty) that can be obtained by removing a substring containing $[l, r]$.

## E. Easy String Problem

- Let $[l, r]$ be the string we obtain after deleting the substring $\{a_l a_{l+1} ... a_r\}$.

**Lemma**

Lemma: If $[l, r] = [l + k, r + k]$,then $[l, r] = [l + 1, r + 1] = ... = [l + k, r + k]$.

- So the answer of $L, R$ is how many $[l, r], l \leq L, R \leq r$ satisfied $[l, r] \neq [l - 1, r - 1]$, which is equal to $L \cdot (n - R + 1) - \{How\ many\ [l, r]\ satisfied\ [l, r] = [l - 1, r - 1]\}$.
- Because $[l, r] = [l - 1, r - 1]$ iff $a_{l-1} = a_r$, the number of $[l, r] = [l - 1, r - 1]$ is equal to the number of pairs $(p, q), 1 \leq p \leq L - 1, R + 1 \leq q \leq n$ satisfied $a_p = a_q$, which can be computed by the Mo's algorithm in $O(n\sqrt{n})$.

# F. Find the Maximum

## Task

Given a tree, each vertice $i$ has a value $b_i$, define $b(u, v)$ the average value of vertices on path $(u, v)$.
Calculate $\max_{u \neq v, x \in R}\{-x^2 + b(u, v)x\}$

## F. Find the Maximum

- We only need to calculate the maximum and minimum $b(u, v)$.

**Lemma**

There must exist a path whose length is no more than $3$ that has the maximum $b(u, v)$.

- Now we only consider the maximum. If we let $b'_i = -b_i$ then we can also get the minimum.
- For every vertice, we only need to calculate the maximum and second maximum weight next to it and solve this problem in $O(n)$.

## G. Glass Bead Game

### Task

There are $n$ beads on a line, each has a value $p_i$, $\sum_{p_i} = 1$.
In each turn, choose a bead and move it to the front.
In each turn, bead $i$ is moved with probability $p_i$.
Let the weight of this move be the number of bead in front of the chosen bead.
Let $E_m$ be the expected weight of the $i$-th turn.
Calculate $\lim_{m\to\infty} E_m$

## G. Glass Bead Game

- Let $X$ be the cost of an access. To calculate $\mathbb{E}[X]$, we rely on the fact that
- $$X = \sum_{i \neq j} X_{i,j}$$
- where $X_{i,j}$ is an indicator variable which is 1 when bead $B_i$ is the next accessed bead and bead $B_j$ is ahead of it in the current list. The important observation is that $\mathbb{P}\left(X_{i,j} = 1\right) = p_i \frac{p_j}{p_i + p_j}$ (the probability that $B_i$ is the next accessed bead, times the probability that $B_j$ was accessed more recently than $R_i$ ). So we have
- $$E[X] = \mathbb{E}\left[\sum_{i \neq j} X_{i,j}\right] = \sum_{i \neq j} \mathbb{E}\left[X_{i,j}\right] = \sum_{i \neq j} p_i \frac{p_j}{p_i + p_j} = \sum_{i \neq j} \frac{p_i p_j}{p_i + p_j}$$

## H. Helesta

### Task

You are given $n$ points $(x_i, y_i),\ 1 \le i \le n$ and $m$ sets $S_j = \{(x_i, y_i) \mid A_j x_i + B_j y_i + C_j > 0\}$ $(1 \le j \le m)$.
You need to find a permutation $p_1, \ldots, p_m$ of $1, 2, \ldots, m$, such that $|S_{p_1}| + \sum_{i=2}^{m} |S_{p_i} \oplus S_{p_{i-1}}| \le M$, where $M = 1.8 \times 10^8$ is a given constant and $A \oplus B$ means $(A \cup B) - (A \cap B)$.

## H. Helesta

- Optimal Mo's algorithm on half-plane range.
- The problem is transformed into maintaining point set, which is initially empty, and supports adding and deleting points no more than $M$ times, so that the point set corresponding to each half-plane appears at least once in this process.
- First, we randomly select $B$ points, on each point we perform a "rotating scanline algorithm" ( this algorithm can be used to calculate the number of points in a half-plane range in $O(n\sqrt{n}\log n)$ time ) centered on this point (cost $B \cdot 2n$), each half-plane needs expectedly $2\frac{n}{B}$ cost to move to the closest half-plane which we have calculated in the "rotating scanline algorithm" process. $B$ rotated scanlines all start and end at the same slope, then the cost of connecting them does not exceed $n$.

## H. Helesta

- The expected total cost is $n + 2nB + 2\frac{mn}{B} = 4n\sqrt{m}$, if you think the variance is large, you can randomize it several times.
- The time complexity of the naive implementation $O(mB) = O(m^{1.5})$, you can pass this question.
- It can be optimized to $O((B^2 + m)\log B) = O(m \log m)$.

## I. Interval Mex

**Task**

We are given a sequence $a_1, a_2, \cdots, a_n$, define the weight of interval $[l, r]$ $\text{mex}(\{a_l, a_{l+1}, \cdots, a_r\})$. Given $q$ queries, calculate the $k_j$-th smallest weight among all subintervals of $[L_j, R_j]$.

## I. Interval Mex

- First, we can consider the case then $L = 1, R = n$ and use binary search to solve the $k$-th problem.
- Then we only need the calculate how many subintervals have a weight $\geq m$.
- Let $last_{i,j}$ be the $\max\{k \mid k \leq i, a_k = j\}$.
- For any right endpoint $r$, the maximum legal left endpoint $t_r = \min_{0 \leq j < m}\{last_{r,j}\}$ and the answer is $\sum t_r$.
- For $j = 0, 1, \cdots, m - 1$, let $p_1, p_2, \cdots, p_c$ be the position of $j$ in the sequence and that is $\forall p_k \leq v < p_{k+1}, t_v = \min\{t_v, p_k\}(p_0 = 0, p_{c+1} = n + 1)$.
- It is obivous that $t_r$ is a non-descreasing sequence, so this can actually be turned into an interval-assignment operation.

## I. Interval Mex

- Now we can do the problem at an interval. We still use binary search and the same method to calculate $t_r$.
- For any interval $[L_j, R_j]$, the answer is $\sum_{i \le R_j, t_i \ge L_j} t_i - L_j + 1$.
- Ideally, we can store all history information and do this subproblem online on a functional segment tree. But this solution may result in high time and space cost.

## I. Interval Mex

- Now we try to do binary search offline.Let $lb_j, rb_j, m_j$ be the binary search bound and middle of $j$-th query.
- Sort all queries by $m_j$, and do the above operations on a segment tree,and update $lb_j, rb_j, m_j$.
- Do this for $\log n$ turns and we can get all answers.
- The overall time and space complexity is $O(n \log^2 n)$ and $O(n)$.

# J. Just Another String Problem

## Task

You are given a string $s$ of length $n$ consisting of lowercase English letters and a non-decreasing array $\{v_1, v_2, \cdots, v_n\}$.
A set $T$ of non-empty strings is called good if and only if there does not exist a pair of strings in $T$ such that one is a suffix of the other. For a good set $T$, denote the value of $T$ as $\sum_{t \in T} v_{|t|}$.
For a string $t$ and some positive integer $k$, define $f(t, k)$ as the maximum value among all good sets $T$ that satisfies the following conditions:

1. $|T| \leq k$;
2. $T$ only contains substrings of $t$.

You need to answer $q$ queries. For each of them, you are given two values $l, k$ $(1 \leq l \leq n, 1 \leq k \leq 10^9)$, and you need to find out $f(s[1, l], k)$, where $s[1, l]$ denotes the prefix with $l$ letters of $s$.

## J. Just Another String Problem

- It can be found that you just need to choose the prefix. For some prefix $i$, if exists some prefix $j$ such that $j > i$ and $i$ is a border of $j$. Use KMP to calculate this.
- Then, you can use binary search in segment tree to solve this,
- Time complexity is $O(n \log n)$.

## K. King of Gamers

**Task**

Let $x = \frac{a}{b}$ and we will play $n$ games. When playing the $i$-th game, if his current winning rate is lower than or equal to $x$, he will be eager to win and win the game easily. Otherwise, he will enjoy the game and lose it.
Calculate how many games we will win.

## K. King of Gamers

**Lemma and Proposition**

$\lfloor nx \rfloor = \lfloor (n-1)x \rfloor$ or $\lfloor nx \rfloor = \lceil (n-1)x \rceil$.
The answer of $n$ is either $\lfloor nx \rfloor$ or $\lceil nx \rceil + 1$.

- Using Mathematical induction to prove(without many details):
- The proposition holds when $n = 0$.
- If the proposition holds when $n = k$:
- 1.$\lfloor kx \rfloor = \lceil kx \rceil$: then Little G will win a game, and the answer of $k+1$ is $\lceil (k+1)x \rceil$.
- 2.the answer of $k$ is $\lfloor kx \rfloor$: then Little G will win a game, and the answer is $\lceil kx \rceil$, according to the Lemma, $\lceil kx \rceil = \lceil (k+1)x \rceil$ or $\lceil kx \rceil = \lfloor (k+1)x \rfloor$ .
- 3.the answer of $k$ is $\lceil kx \rceil$: then Little G will lose a game, and the answer is $\lceil kx \rceil$, according to the Lemma, $\lceil kx \rceil = \lceil (k+1)x \rceil$ or $\lceil kx \rceil = \lfloor (k+1)x \rfloor$ .
- So the proposition holds for all natural number $n$.
- If $x = 0$, then the answer is $1$.

## K. King of Gamers

- Lets consider what happened when playing the $n^{th}$ game$(x \neq 0)$.
- 1.If $\lfloor nx \rfloor = \lfloor (n-1)x \rfloor$, then the answer of $n$ is always $\lceil nx \rceil$;
- 2.If $\lfloor nx \rfloor = \lceil (n-1)x \rceil$, then the answer of $n$ is always $\lfloor nx \rfloor$.
- To avoid classified discussion, you can also assume the answer of $n-1$ is $\lfloor nx \rfloor$(even if this can be wrong or invalid) and judge whether Little G will win or lose in the $n^{th}$ game, which will lead to the correct answer of $n$.
- The simplest solution is just print $\lfloor \frac{(n-1)a}{b} \rfloor + 1$.

# L. Light of Stars

## Task

There are $n$ stars on a 2D plane. The $i$-th star can be seen as a point located at $(x_i, y_i)$. It's guaranteed no two stars occupy the same place.
Each star emits the **same** regions of light that don't intersect. Every region of light is given in the form of an angle. Now you want to know, for every star, the number of times it is lightened by other stars. Note that stars will not block the light.

## L. Light of Stars

- Firstly, we calculate the contribution of each light separately. For the light in an interval, we use the straight line of its two rays as the new coordinate axis, then this problem becomes a simple two-dimensional point counting problem.
- However, there will still be a small problem. If any two rays are selected as axes, trigonometric functions and division will appear in the calculation process. There may be minor problems with accuracy. A feasible solution is to determine a coordinate axis as the positive half axis of the X- axis, and then subtract the answer of L from the answer of R, so that in the operation process, there is only the tangent and multiplication of the angle that needs to be preprocessed initially. The accuracy of tangent operation is about $10^{-13}$, while the angle within 50000 coordinates closest to the integer degree (non-integer degree) is about $10^{-7}$, By setting EPS, there would be no accuracy problem.

## M. Mountain Is Quiet and Alone

**Task**

The mountain is a monotone slope described by $n$ points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. The slope contains the following segments: $((-\infty, 0), (0, 0))$, $((0, 0), (x_1, y_1))$, and $((x_i, y_i), (x_{i+1}, y_{i+1}))$ for all $1 \le i \le n-1$. A segment $((x_n, y_n), (x_n, y_n+1))$ of length $1$, and we call the $(x_n, y_n+1)$ end of the segment head.

The segment always rotates counter-clockwise (downhill) around its lowest contact point with the hill. Whenever it has a new (lower) contact point with the hill, it starts rolling from that point. It only stops rolling when its head has $y$-coordinate $0$.

Calculate the total length its head travels in the rolling process.

## M. Mountain Is Quiet and Alone

- Simulate the whole process and divide each rotation into two cases according to whether the end of the segment can get in touch with the hill. See std implementation for details.

Thank you!