Problem A. Simplified Genome Translation

You could implement translation by a hash table where mRNA codons are keys, and the corresponding translated amino acids are values.

Problem B. Multi-Ladders

Let G be an undirected graph, and let λ be the number of colors that we have available for properly coloring the vertices of G. The objective is to find a polynomial function $P(G, \lambda)$, in the variable λ, called the chromatic polynomial of G, that will tell us in how many different ways we can properly color the vertices of G, using at most λ colors.

Using the deletion-contraction as described in the Wikipedia entry, we define the deletion edge (u, v) be the edge at the top of the ladder and let G_n be the ladder graph. Then we have $P(G_n, \lambda) = P(G_n - uv, \lambda) - P(G_n|uv, \lambda)$. But $G_n - uv$ is a ladder graph of height $n - 1$ with two pendant vertices at the top. Each of these can be colored using any one of $\lambda - 1$ colors, since they only need to differ in color from the vertex where they are attached to the ladder and there is no edge between them. Therefore, the following holds: $P(G_n - uv, \lambda) = (\lambda - 1)^2 P(G_{n-1}, \lambda)$. The graph $G_n uv$ has a vertex merged to the top two vertices to form a triangle. This vertex must have a different color from the two vertices where it is attached to the ladder. Hence, $P(G_n|uv, \lambda) = (\lambda - 2)P(G_{n-1}, \lambda)$. By combining the above two observations, we have $P(G_n, \lambda) = ((\lambda - 1)^2 - (\lambda - 2))P(G_{n-1}, \lambda) = (\lambda^2 - 3\lambda + 3)P(G_{n-1}, \lambda)$. Note that the base case of the ladder graph of height one is a path on two vertices and has $\lambda(\lambda - 1)$ colorings. Therefore,

$$P(G_n, \lambda) = \lambda(\lambda - 1)(\lambda^2 - 3\lambda + 3)^{n-1} \tag{1}$$

On the other hand, the chromatic polynomial $P(C_k, \lambda)$ for the cycle graph C_k is well-known as

$$P(C_k, \lambda) = (\lambda - 1)^k + (-1)^k(\lambda - 1) \tag{2}$$

for all positive integers $k \geq 1$ 1. We also need the following property 2. Let G be an undirected graph with subgraphs G_1 and G_2. If $G = G_1 \cup G_2$ and $G_1 \cap G_2 = K_2$, for some positive integer n, then

$$P(G, \lambda) = \frac{P(G_1, \lambda) \cdot P(G_2, \lambda)}{\lambda(\lambda - 1)} \tag{3}$$

Algorithm for Solving the Problem

Given k for a k-regular polygon C_k, n for a ladder graph L_n, and the number of available colors m, the proposed algorithm is described as follows:

Step 1: Let $G_0 = C_k \cup L_n$, where $C_k \cap L_n$ is an edge. Compute $P(G_0, \lambda) = \frac{P(C_k,\lambda)\cdot P(L_n,\lambda)}{\lambda(\lambda-1)}$ using Equations 1–3.

Step 2: For $i = 1$ to k, execute the following three operations: (1) compute $P(G_i, \lambda) = \frac{P(G_{i-1},\lambda)\cdot P(Ln,\lambda)}{\lambda(\lambda-1)}$ using Equation 3. (2) $i := i + 1$.

Step 3: Let $\lambda = m$ and return the value $P(G_k, m)$.

---

1. Korean J. Math. 27 (2019), No. 2, pp. 525–534 https://doi.org/10.11568/kjm.2019.27.2.525

2. Ralph P. Grimaldi, Discrete and Combinatorial Mathematics: An Applied Introduction, Chapter 6, PEARSON.

Problem C. Distance Calculator

- Bubble sort algorithms
- Let each city be a vertex in graph $G = (V, E)$ and each road built between two cities is an edge in E between two vertices in V. The edge set of graph G is constructed according to the sorting strategy swapping two consecutive numbers.
- The minimum number of swapping can be calculated with a bubble sort algorithm.

Problem D. Tangle: A DAG for storing transactions

We can solve this problem using DFS or BFS traversal algorithm after reading a transaction.

Problem E. Printing Stickers

- This problem can be formulated as a minimum isolating cut problem: there is a graph G with a set T of terminal vertices. The goal is to compute mincuts from each of the terminal $v \in T$ that separates v from all other vertices in $T - \{v\}$. A trivial algorithm requires $|T|$ maxflows.
- You can pass using $|T|$ maxflows if the implementation is efficient (notice that the graph we constructed here is a planar graph.)
- Alternatively, there is an algorithm that requires only $2 + \log |T|$ maxflow computations, described as the following. We first label each vertex from 0 to $|T| - 1$ using binary format. Then, for each $i = 0, 1, 2, \ldots, \log |T|$ we compute any mincut that separates all terminals of ith digit equal to 0 from all terminals of ith digit equal to 1.

  Each cut $(X_i, V \setminus X_i)$ has two sides: all terminals of ith digit equal to 0 belong to X_i and the remaining terminals belong to $V \setminus X_i$.

  For each terminal $v \in T$, we compute the intersection of all sides where v belongs to. Let X_v be the intersection of all such sides. It can be shown that if we contract everything outside of X_v into another vertex $\bar{v}$, the mincut that separates v with $\bar{v}$ is a minimum isolating cut that separates v with all other terminals. All these mincut computations add up to at most 2 maxflow computations from the original graph.
- There is a slightly efficient implementation to the above idea using divide and conquer. We split the stickers into two sets, and run a maxflow that separates the first set with the second set. The result of the maxflow also separates the graph into two parts. Then, the algorithm recursively splits the stickers again and again.

Problem F. AA Country and King Dreamoon

- The journey of Dreamoon is equivalent to some DFS(Depth-First Search) process.
- When doing DFS, there are two kinds of moves in each step. The first is visiting a new node. The second is going back to the prarent node.
- We can try which kind of move is better such that the lexicographical order is the smallest and greedily choose it.
- Be careful after choosing each step, the journey is still a valid DFS process.
- The time complexity can be done in $\Theta(n)$

Problem G. Repetitive Elements

- Dynamic programming
- Let $dp(i, j)$ represent the length of the common string ending at the i^{th} and j^{th} positions of the genome, S. Denote S_i as the i^{th} nucleic acid of S.

$$dp(i,j) = \begin{cases} 0 & S_i \neq S_j \\ dp(i-1, j-1) + 1 & S_i = S_j \end{cases}$$

- You can filter out overlaps by the condition $(j - i) > dp(i - 1, j - 1)$.

Problem H. Meeting Places

Consider $dp[K][N]$ to represent the optimal answer for dividing the first N points into K segments. It is obvious that we have $dp[K][N] = min(dp[K][N], dp[K-1][i-1] + cost(i, N)), \forall 1 \leq i \leq N$.

There are only expected O(1) of points that can make the minimum enclosing circle larger under random conditions. Hence, the selectivity of i is expected $O(1)$.

In conclusion, the time complexity is $O(N^2 + KN)$.

Problem I. Cell Nuclei Detection

- Build a bipartite graph with one partite of nodes representing the ground-truth bounding boxes and the other partite representing the detected bounding boxes.
- Use a map data structure to keep track of the coordinates of the centers of the detected bounding boxes.
- For each ground-truth bounding box, check the intersection area with its nearby detected bounding boxes to determine the "detected relation". Since the height and width of a ground-truth bounding box are bounded by 4, the nearby detected bounding boxes have centers near the center of the ground-truth bounding box within a distance of 2.
- Apply Dinic's algorithm to determine the maximum bipartite matching number in $O(\sqrt{V}E)$.

Problem J. Traveling in Jade City

- Partition the railroad system into three paths
- Use prefix sum on each path to maintain traveling time
- Use BST on each path to maintain connectivity
- Shortest path is computed by constant number of case checking

Problem K. Group Guests

- This problem can be formulated as partitioning the edge set of an undirected graph into paths of length 2 (P3), cycles of length 3 (C3), and stars of 4 nodes (S3).
- Partitioning the edge set of an undirected graph into P3s is equivalent to asking whether each component has an even number of edges (called an even component).

- Partitioning the edge set of an undirected graph into P3s and C3s is equivalent to asking whether there exists a C3 whose removal makes each component even.
- Partitioning the edge set of an undirected graph into P3s and S3s is equivalent to asking whether there exists an S3 whose removal makes each component even.
- Combing the above three facts, we can solve this problem in polynomial time. The bottleneck is to identify the existence of a feasible C3. This problem can be solved in $O(n\sqrt{n/\log n})$ time.

Problem L. Programmable Virus

Almost isomorphic to brainfuck (except the End mark)

Problem M. Connectivity Problem

We can solve this problem using Union-Find algorithm.