Algolympics 2018

Solution Sketches

Problem F: Man-in-the-Middle Attack

- Sol. 1: Lots of if-else cases.
    - Not recommended; prone to bugs.

Problem F: Man-in-the-Middle Attack

- Sol. 2: Sort then take middle.
  - Use built-in sort.
    - `qsort` (C)
    - `std::sort` (C++)
    - `Arrays.sort` or `Collections.sort` (Java)

Problem F: Man-in-the-Middle Attack

- Sol. 3: a + b + c - min(a,b,c) - max(a,b,c)
  - Maybe easier/faster to code?

Problem F: Man-in-the-Middle Attack

- Sol. 4: $a \oplus b \oplus c \oplus \min(a,b,c) \oplus \max(a,b,c)$
  - $\oplus$ = XOR. (^ in most languages)
  - Even easier/faster to code?

Problem C: Jejeland

- Sol. 1: Custom compare.
  - Be careful with implementation.

Problem C: Jejeland

- Sol. 2: Make lowercase, then normal compare.
  - Keep the untouched version for printing.
    - Make a copy for comparison.

Problem L: Zoolander

- Breadth-first Search (BFS) from starting point.
- Remember “direction” as well.
  - (x, y, d) or (i, j, d)
- Needs “right turn”.
  - (x, y) ➜ (y, -x) (normal coordinates)
  - (i, j) ➜ (j, -i) (row-column coordinates)
  - d ➜ (d + 1) mod 4
- O(4nm) = O(nm).

Problem L: Zoolander

- Some interesting cases.

```
F . . . . . . . . . . . . . . . . . . . . . . . . . # . .
> . . . . . . . . . . . . . . . . . . . . . . . . . . . .
# # # # # # # # # # # # # # # # # # # # # # # # # # . . #
. . # . . . . . . . . . . . . . . . . . . . . # . . # . .
. . . . . . . . . . . . . . . . . . . . . . . . . . # . .
# . . # # # # # # # # # # # # # # # # # # # # . . # # . .
. . # . . # . . . . . . . . . . . . . . # . . # . . # . .
. . # . . . . . . . . . . . . . . . . . . . . # . . # . .
. . # # . . # # # # # # # # # # # # # # . . # # . . # . .
. . # . . # . . # . . . . . . . . # . . # . . # . . # . .
. . # . . # . . . . . . . . . . . . . . # . . # . . # . .
. . # . . # # . . # # # # # # # # . . # # . . # . . # . .
. . # . . # . . # . . # . . # . . # . . # . . # . . # . .
. . # . . # . . # . . . . . . . . # . . # . . # . . # . .
. . # . . # . . # # . . # # . . # # . . # . . # . . # . .
. . # . . # . . # . . # . . # . # # . . # . . # . . # . .
. . # . . # . . # . . # . . . . . # . . # . . # . . # . .
. . # . . # . . # . . # . . # . . # . . # . . # . . # . .
. . # . . # . . # # . . # # # # # . . # # . . # . . # . .
. . # . . # . . # . . . . . . . . . . . # . . # . . # . .
. . # . . # . . # . . # . . . . . # . . # . . # . . # . .
. . # . . # # . . # # # # # # # # # # # . . # # . . # . .
. . # . . # . . . . . . . . . . . . . . . . . # . . # . .
. . # . . # . . # . . . . . . . . . . . # . . # . . # . .
. . # # . . # # # # # # # # # # # # # # # # # . . # # . .
. . # . . . . . . . . . . . . . . . . . . . . . . . # . .
. . # . . # . . . . . . . . . . . . . . . . . # . . # . .
# . . # # # # # # # # # # # # # # # # # # # # # # . . #
. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . # . . . . . . . . . . . . . . . . . . . . . . . # . .
```

```
F . . . . . . . . . . . . . . . . . . . . . . . . . .
. # . . # # . . # # . . # . # # . . # # . . # # . . .
. # . . . . . . . . . . . . . . . . . . . . . . . # .
. . . # . . # # . . # # . # . . # # . . # # . . . # .
. . . # . . . . . . . . . . . . . . . . . . . # . . .
. # . . . # . . # # . . # . # # . . # # . . . # . . .
. # . . . # . . . . . . . . . . . . . . . # . . . # .
. . . # . . . # . . # # . # . . # # . . . # . . . # .
. . . # . . . # . . . . . . . . . . . # . . . # . . .
. # . . . # . . . # . . # . # # . . . # . . . # . . .
. # . . . # . . . # . . . . . . . # . . . # . . . # .
. # . . . # . . . # . . . # . . . # . . . # . . . # .
. . . # . . . # . . . # . . . # . . . # . . . # . . .
. . . # . . . # . . . # . ^ . # . . . # . . . # . . .
. # . . . # . . . # . . . # # . . # . . . # . . . # .
. # . . . # . . . # . . . . . . . . . # . . . # . . .
. . . # . . . # . . . # # . . # # . . # . . . # . . .
. . . # . . . # . . . . . . . . . . . . . # . . . # .
. # . . . # . . . # # . . # # . . # # . . # . . . # .
. # . . . # . . . . . . . . . . . . . . . . . # . . .
. . . # . . . # # . . # # . . # # . . # # . . # . . .
. . . # . . . . . . . . . . . . . . . . . . . . . # .
. # . . . # # . . # # . . # # . . # # . . # # . . # .
. # . . . . . . . . . . . . . . . . . . . . . . . . .
. . . # # . . # # . . # # . . # # . . # # . . # # . .
. . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Problem A: Triangles

- Order doesn't matter. Sort.
- Then for $i < j < k$:
  - min(a[i], a[j], a[k]) = a[i].
  - max(a[i], a[j], a[k]) = a[k].
- Thus, the answer is:
  - sum(a[i]*a[k]) for $1 \le i < j < k \le n$

Problem A: Triangles

$$\sum_{1 \le i < j < k \le n} a_i a_k = \sum_{j=1}^{n} \sum_{i=1}^{j-1} \sum_{k=j+1}^{n} a_i a_k$$

$$= \sum_{j=1}^{n} \sum_{i=1}^{j-1} a_i \sum_{k=j+1}^{n} a_k$$

$$= \sum_{j=1}^{n} \left(\sum_{i=1}^{j-1} a_i \right) \left(\sum_{k=j+1}^{n} a_k \right)$$

Problem A: Triangles

$$\text{answer} = \sum_{j=1}^{n} \left(\sum_{i=1}^{j-1} a_i \right) \left(\sum_{k=j+1}^{n} a_k \right)$$

- Let s[j] = a[1] + a[2] + ... + a[j - 1].
- Let t[j] = a[j + 1] + a[j + 2] + ... + a[n].
- The answer is the sum of s[j]*t[j] for all j = 1..n.
- O(n) (after O(n log n) sort)

Problem A: Triangles

- Alternatively,
- Let s[j] = a[1] + a[2] + ... + a[j - 1].
- Let t[j] = a[j + 1] + a[j + 2] + ... + a[n].
- Then s[j]*t[j] = sum of wow factors of all triangles with median point j.

Problem K: Bananagrams

- Let freq(s) be the frequency counts of letters in string s.
- Only need to consider substrings with distinct freq.
- Windowing to go through all substrings.
    - Maintain freq array and collect distinct ones.
    - freq is always only 26 in size.

Problem K: Bananagrams

- Then add contributions of each freq array.
- Let $c = [c_0, c_1, \dots, c_{25}]$ be the freq array.
- Then add $(c_0+c_1+\dots+c_{25})! / (c_0!\, c_1! \dots c_{25}!)$
    - Multinomial coefficient.
- Precompute factorials and stuff.
- $O(\alpha n \log n)$ or $O(\alpha n)$
    - α = alphabet size (= 26).

Problem E: Cookie

- **Nim** game.
- Answer = # of subarrays that are winning positions for player 2.
- Player 2 wins iff bitwise XOR = 0. (Well-known)
- Thus, answer = # of subarrays that have XOR 0.

Problem E: Cookie

- Thus, answer = # of subarrays that have XOR 0.
- Let $x_i = a_1 \oplus a_2 \oplus \ldots \oplus a_i$, $x_0 = 0$
- Then $a_{i+1} \oplus a_{i+2} \oplus \ldots \oplus a_j = x_i \oplus x_j$
- $x_i \oplus x_j = 0$ iff $x_i = x_j$
- Thus, answer = # of (i, j), $0 \le i < j \le n$ where $x_i = x_j$.

Problem E: Cookie

- Thus, answer = # of (i, j), $0 \leq i < j \leq n$ where $x_i = x_j$.
- Insight: $x_{2m} = 0$, so there are $\leq 2m$ distinct x_i's.
- For each distinct prefix XOR value, count how many times it appears as x_i.
- Sum $c(c-1)/2$ for all counts c.
- $O(m \log m)$ or $O(m)$

Problem G: The Wedding Guests

- Binary search the answer.
- Given g, can you find partition (A,B) where max weight edge on each side is $\leq g$?
- Now, only weight comparison with g matters.
- There are just two kinds of edges.

Problem G: The Wedding Guests

- Insight: Consider only edges with weight > g.
- These edges *must* be from A to B.
- Just check if bipartite!
- $O(n^2 \log ans)$ or $O(n^2 \log n)$.
  - log from binary search

Problem G: The Wedding Guests

- Alternative: Add edges one by one in decreasing weight until the graph is no longer bicolorable.
- For every new edge, possibly update the bicoloring.
- $O(n^3)$; up to n-1 recolors and recolor takes $O(n^2)$. Too slow.

Problem G: The Wedding Guests

- Key: Forget the edges; on recoloring, either keep all or flip all!
- Each recoloring only takes $O(n)$, so $O(n^2)$ total.
- $O(n^2 \log n)$, dominated by sorting.
- Key: On union, recolor just the *smaller* component!
- Now, recoloring takes $O(n \log n)$ total.
    - Still $O(n^2 \log n)$ overall, though.

Problem B: Superconstructible

- We want $\phi(n) = 2^r 3^s$.
- $\phi(n) = n\,(1 - 1/p_1)\,(1 - 1/p_2)$ ... for all primes $p_i \mid n$.
- n can have any number of 2 and 3 factors.
- Let $p \mid n$ where $p > 3$. Then:
  - If $p^2 \mid n$, then $p \mid \phi(n)$, impossible.
  - Also, $(p - 1) \mid \phi(n)$, so $p - 1 = 2^a 3^b \rightarrow p = 2^a 3^b + 1$.

Problem B: Superconstructible

- Hence, $n \geq 3$ is superconstructible iff
- $n = 2^x 3^y p_1 p_2 \ldots p_t$ where $p_1, \ldots, p_t$ are distinct primes > 3 of the form $2^a 3^b + 1$.

Problem B: Superconstructible

- Enumerate all primes $2^a 3^b + 1$ up to 10^{18}.
  - Only 141 of them.
- Enumerate all products $p_1 p_2 \ldots p_t$ up to 10^{18}.
  - Only 3508893 of them.
- Enumerate all $2^x 3^y p_1 p_2 \ldots p_t$ up to 10^{18}.
  - Only 86414585 of them. (incl. 1 and 2)
- Lookup to answer queries.

Problem B: Superconstructible

- Alternatively, binary search n. Now, given n, we need to count superconstructible numbers ≤ n.
- For each such query:
  - Enumerate all $2^x 3^y \leq n$. For each (x,y):
    - Find # of products $p_1 \ldots p_t \leq n/(2^x 3^y)$. (Binary search)
- Slower query, but smaller memory.

Problem B: Superconstructible

- Primes of the form $2^a3^b + 1$ are called **Pierpont primes**.

Problem H: Religious War Prevention

- Given tree T with n nodes, color with up to k colors such that all nodes with same color form a connected subgraph.

Problem H: Religious War Prevention

- Call an edge hot if it connects nodes of different colors.
- Then choosing colors is the same as:
  - Choosing the hot edges, then
  - Choosing the colors of the resulting subtrees.
- This is independent of the tree T! Only n and k matters. Hence, m = M.

Problem H: Religious War Prevention

- To compute the answer for (n,k):
  - Choose how many groups g.
  - Then out of n-1 edges, g-1 will be hot. Choose.
  - Then out of k colors, g will be used. Choose.
  - g! ways to assign groups to colors.

$$\sum_{g=1}^{\min(n,k)} \binom{n-1}{g-1} \binom{k}{g} g!$$

Problem H: Religious War Prevention

$$\sum_{g=1}^{\min(n,k)} \binom{n-1}{g-1}\binom{k}{g} g!$$

- O(min(n,k)).
  - Precompute inverses, etc.
  - $nk \leq 10^9$ ➡ $\min(n,k) \leq \text{sqrt}(10^9) < 32000$.

Problem D: Crab Product

- Goal: Find $S * T$ where $(S * T)_i = \sum_{j \otimes k = i} S_j T_k$.
- Let $x = (x_0, \ldots, x_{n-1})$ and $y = (y_0, \ldots, y_{n-1})$. We say $x \preceq y$ iff $x_i \leq y_i$ for all i.
- Given U, define U' such that $U'_i = \sum_{i \preceq j} U_j$.
- Then: $(S * T)' = S' \cdot T'$, where $\cdot$ is pointwise product!

Problem D: Crab Product

- Compute U′ from U in $O(nb^n)$ time using DP.
- Compute U from U′ in $O(nb^n)$ time using DP.
  - Basically, reverse of each other.
- $O(nb^n)$.

Problem D: Crab Product

- Strategy is similar to Fourier transform and Hadamard transform methods.
  - Hadamard transforms for "XOR"-type operations. The current problem is "AND" (on b = 2).
  - Can be generalized to higher bases as well. XOR becomes "no-carry addition base b".
  - Mirror algorithm for "OR" (and max).

Problem D: Crab Product

- XOR: AND: OR:

$$\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}^{\otimes n} \quad \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}^{\otimes n} \quad \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}^{\otimes n}$$

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \omega & \omega^2 & \omega^3\\ 1 & \omega^2 & \omega^4 & \omega^6\\ 1 & \omega^3 & \omega^6 & \omega^9\end{bmatrix}^{\otimes n} \quad \begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}^{\otimes n} \quad \begin{bmatrix}1 & 0 & 0 & 0\\ 1 & 1 & 0 & 0\\ 1 & 1 & 1 & 0\\ 1 & 1 & 1 & 1\end{bmatrix}^{\otimes n}$$

- $\otimes$ = Kronecker product.

Problem I: The Pangets

- Given s, t, what’s the minimum maximum edge to connect them?
- Add edges in increasing weight until s and t join.
- Thus, only **MST** matters.
- Take MST, ignore other edges.

Problem I: The Pangets

- Compute **“union find” tree** by performing MST but making a new node for each successful union.
- Each connected component under a certain weight is now a subtree.
- The cost to connect s and t is now (roughly) the LCA of s and t in this tree.
  - Watch out for edges of equal weights!

Problem I: The Pangets

- To perform queries, we need subtree sum and subtree updates. Flatten this tree and build **segment tree** with lazy propagation on the preorder traversal.
  - Alternatively, Euler tour techniques.
- O(b log b + q log n).

Problem J: Fififibobobobonacci Sequence

- Use generating functions on $f(n) = a\,f(n-1) + b\,f(n-2)$.
- Let r and s be the roots of $x^2 - ax - b$.
- Two kinds of closed-forms:
  - $f(n) = cr^n + ds^n$ if $r \neq s$.
  - $f(n) = (c + dn)r^n$ if $r = s$.
- Compute c and d from $f(0)$ and $f(1)$.

Problem J: Fififibobobobonacci Sequence

- For example, for Fibonacci numbers:
  - f(n) = f(n-1) + f(n-2)

$$f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Problem J: Fifififibobobobonacci Sequence

- The required sum can then be decomposed into:
  - Arithmetic series.
  - Geometric series.
  - Arithmetic-geometric series.
  - Quadratic-geometric series.
- All can be computed in O(log u) time with something similar to “binary exponentiation”.

Problem J: Fififibobobobonacci Sequence

- Now, r and s could be irrational, even complex.
- Work on field extension $\frac{\mathbb{Z}}{p\mathbb{Z}}[r]$ by adjoining r (and thus s).
  - Here, $p = 10^9+7$.
- Edge case:
  - r = 0 or s = 0 (or both). Slight care needed.
- O(log u).

Problem J: Fifififibobobobonacci Sequence

- There's also a matrix-based solution, based on:

$$\begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} a & b \\ 1 & 0 \end{bmatrix}^m \begin{bmatrix} f_1 \\ f_0 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} a & b \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} f_1 \\ f_0 \end{bmatrix} = [f_m f_n]$$

- Same complexity, though possibly higher constant.
- Left to the reader as exercise.

Thank you!

- Kevin Charles Atienza
  - Problem setter + Judge
- Jared Guissmo Asuncion, M.Sc.
  - Theme supervisor
- Joseph Daniel Dantes
  - Tester
- Payton Robin Yao, M.Comp.
  - Judge