Algolympics 2016

Solution Sketches

Problem E: Noel

- Easy!
  - Make all lowercase (or all uppercase).
  - Remove duplicates.
  - Compare.

Problem A: Gag Olympics

- Also easy! Just careful implementation.
- Be careful! Exactly copy the (awesome) ASCII art.

Problem B: The Tale of the Extremely Bored Fairies

- Translation of problem: Given *a* and *n*, find the unique *x* such that $(a+1) \oplus (a+2) \oplus \ldots \oplus (a+n) \oplus x = 0$
- $\oplus$ denotes bitwise XOR.
- Properties of $\oplus$:
  - Associative, commutative.
  - Has identity 0.
  - The inverse of every x is itself, i.e., $x \oplus x = 0$.

Problem B: The Tale of the Extremely Bored Fairies

- Thus,
  - $x \oplus y = 0 \leftrightarrow$
  - $(x \oplus y) \oplus y = 0 \oplus y \leftrightarrow$
  - $x \oplus (y \oplus y) = y \leftrightarrow$
  - $x \oplus 0 = y \leftrightarrow$
  - $x = y$
- Hence: the answer is $x = (a+1) \oplus (a+2) \oplus \ldots \oplus (a+n)$

Problem B: The Tale of the Extremely Bored Fairies

- However, we can’t compute it with a simple loop since n is large!
- Now, at this point, it’s already possible to compute $(a+1) \oplus (a+2) \oplus \ldots \oplus (a+n)$ quickly by analyzing each bit separately, but that’s quite complicated. I will show a better way.

Problem B: The Tale of the Extremely Bored Fairies

- Thus,
  - $(a+1) \oplus \ldots \oplus (a+n) =$
  - $(a+1) \oplus \ldots \oplus (a+n) \oplus (0 \oplus 1 \oplus \ldots \oplus a) \oplus (0 \oplus 1 \oplus \ldots \oplus a) =$
  - $(0 \oplus 1 \oplus \ldots \oplus (a+n)) \oplus (0 \oplus 1 \oplus \ldots \oplus a) =$
- Thus, we want to compute $0 \oplus 1 \oplus \ldots \oplus n$ given n.

Problem B: The Tale of the Extremely Bored Fairies

- Insight: $(2m) \oplus (2m + 1) = 1$.
  - Proof: All bits cancel out except the last one.
- Insight: $(4m) \oplus (4m + 1) \oplus (4m + 2) \oplus (4m + 3) = 0$
  - Proof: $((4m)\oplus(4m+1)) \oplus ((4m+2)\oplus(4m+3)) = 1 \oplus 1 = 0$
- Thus, every block of four numbers disappear! We only need to consider the last three elements (or less)!
- $O(1)$.

Problem F: Print F

- Problem: Find the sum of squares of proper divisors of $n := 2^{p-1}(2^p - 1)$ for $p = 74207281$, mod m.
- It's easier to sum for *all* divisors and just subtract n^2 in the end.

Problem F: Print F

- Important: $2^p - 1$ is prime (given in statement). Hence, the divisors are all either 2^i or $2^i(2^p - 1)$ for some $0 \le i < p$.
- We now want:

$$\sum_{i=0}^{p-1} (2^i)^2 + \sum_{i=0}^{p-1} (2^i(2^p - 1))^2$$

Problem F: Print F

$$\sum_{i=0}^{p-1}(2^i)^2 + \sum_{i=0}^{p-1}(2^i(2^p - 1))^2$$

$$= \sum_{i=0}^{p-1}(2^i)^2(1 + (2^p - 1)^2)$$

$$= (1 + (2^p - 1)^2)\sum_{i=0}^{p-1} 4^i$$

$$= (1 + (2^p - 1)^2)(4^p - 1)/3$$

Problem F: Print F

- Hence, answer: $(1 + (2^p - 1)^2)(4^p - 1)/3$
- Reduce mod m. Tricky! Especially if m is divisible by 3, since 3 is not invertible mod m.
- Use: $$\frac{a}{3} \bmod m = \frac{a \bmod 3m}{3}$$
- Be careful with overflow!

Problem D: Colored Tile Puzzle

- This is really just an elaborate BFS problem!
- Just keep track of your smell as part of your state.
- No need to worry about the transition being slow (especially for a long sequence of V’s) since each V is only visited a constant number of times.
- Tricky case: PVVVVY. This allows you to get lemon smell by using Y. So don’t just consider Y as a wall!

Problem I: Inside Down and Upside Out

- Another implementation problem.
- One can represent the setup with a 3D matrix, then simulate each letter *carefully*.
- O(LWHS) time.

Problem I: Inside Down and Upside Out

- This can also be solved in $O(LW \log (L + W) + S)$ time!
  - Left as exercise

Problem H: Timbre

- Problem: Insert into BST and find the final tree.
- You can’t simulate since that takes $O(n^2)$! Need to find something faster.

Problem H: Timbre

- Insight: We know the inorder traversal: 1, 2, ..., n
- Insight: The parent of x is either:
  - the nearest $y < x$ that is inserted earlier, or
  - the nearest $y > x$ that is inserted earlier.
  - Which one among these two? The one that’s inserted later!
- Hence, we can compute parents. After computing parents, a BFS/DFS at the end gives us the tree.

Problem H: Timbre

- Now, we need to solve the subproblem:
  - For each x, find the nearest $y < x$ such that $t[y] < t[x]$.
    - Here, $t[x]$ denotes insertion time.
- We can use segment tree + binary search.
  - $O(n \log^2 n)$
  - $O(n \log n)$ by incorporating the binary search with the segment tree

Problem H: Timbre

- We can also solve it in $O(n)$ by using a left-right sweep and a stack! Just keep track of the “peaks” with the stack.
    - Google “stock span problem” for more details.
- $O(n)$ is optimal.

Problem C: Godlike Multiplication

- Multiplication without carrying… just like polynomial multiplication!
- Solution: fast polynomial multiplication using **Fast Fourier Transform (FFT)**.
- Gotcha: Be careful with some “carries” because of the weird specifics of the described multiplication method.

Problem C: Godlike Multiplication

- $O(d \log d)$ where d is the total # of digits

Problem G: GGVV

- Range query: Given subinterval, follow instructions.
- Range update: Flip two kinds of characters
- Too slow to simulate all!
- **Segment tree with lazy propagation.**
- We need to discuss the details of the segment tree a bit.

Problem G: GGVV

- For each sequence, we keep track of the displacement assuming initial direction is north.
  - Other directions can be determined from this.
- We also keep track of the results assuming certain pairs of letters are flipped. (Four total.) On range updates, we simply rearrange the results.
- $O(n + d \log n)$

Thank you!

Judges:

- Kevin Charles Atienza
- Jared Guissmo Asuncion
- Karl Ezra Pilario