Algolympics 2017

Solution Sketches

Problem H: Spoiled Children

- Everyone gets either *floor(W/N)* or *ceil(W/N)*. *ceil(W/N)* for the younger ones, just enough so that the total is *W*.
- Solution 1: Give everyone *floor(W/N)*. Then increase the first few until the total is *W*.
- Solution 2: Give *ceil(W/N)* to the first *W mod N*, then *floor(W/N)* to the rest.

Problem H: Spoiled Children

- Floor:
  - `W / N`
- Ceil:
  - `(W + N - 1) / N`
  - `(W / N) + (W % N ? 1 : 0)`

Problem G: Superstitious

- Each single letter substring is already a palindrome, giving us N already.
- Any more will make the string bad.

Problem G: Superstitious

- `xx` is a palindrome.
- `xyx` is a palindrome.
- Any palindrome of length ≥ 2 has one of these substrings.
- So we just need to ensure `xx` or `xyx` doesn't appear.

Problem G: Superstitious

- `ABCABCABC...`
- `ABCDEFGABCDEFG...`
- `CAFECAFECAFE...`
- `DECAFDECAFDECAF...`
- etc.

Problem D: Never Forget The C

- Looks like scary calculus at first, but it’s not!
- Just straightforward simulation.

Problem D: Never Forget The C

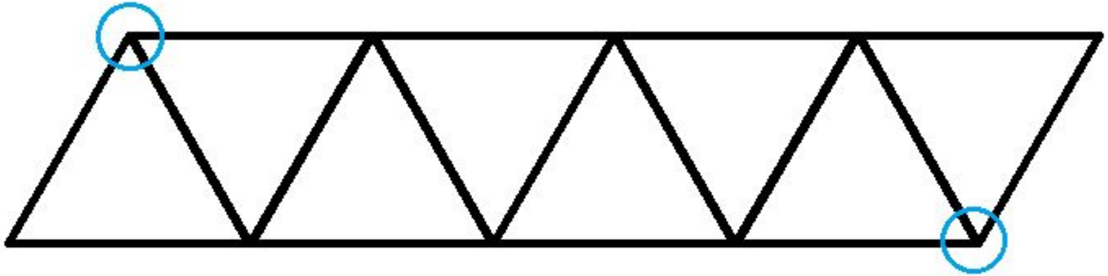
- Solution 1: Combine each component then perform breadth-first search (BFS).

Problem D: Never Forget The C

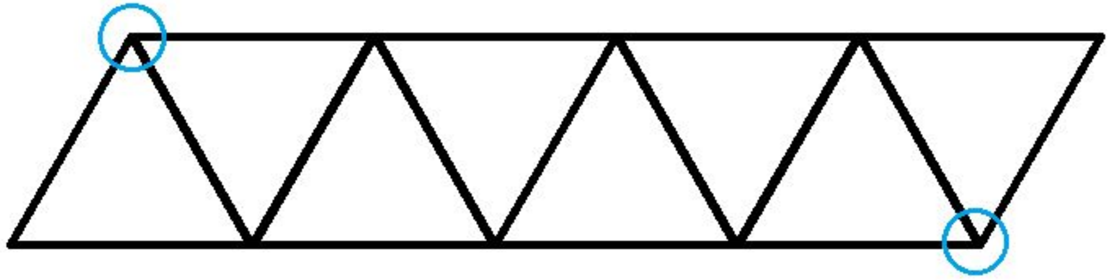
- Solution 2: BFS with transition cost 0 or 1, depending on which, push in front or at the back of the queue.
  - Or just Dijkstra if you’re lazy, but that risks TLE.

Problem F: Illuminati

- Eulerian path
- Eulerian path exists iff at most two nodes have odd degree.
- In our graph, exactly two nodes with odd degree:



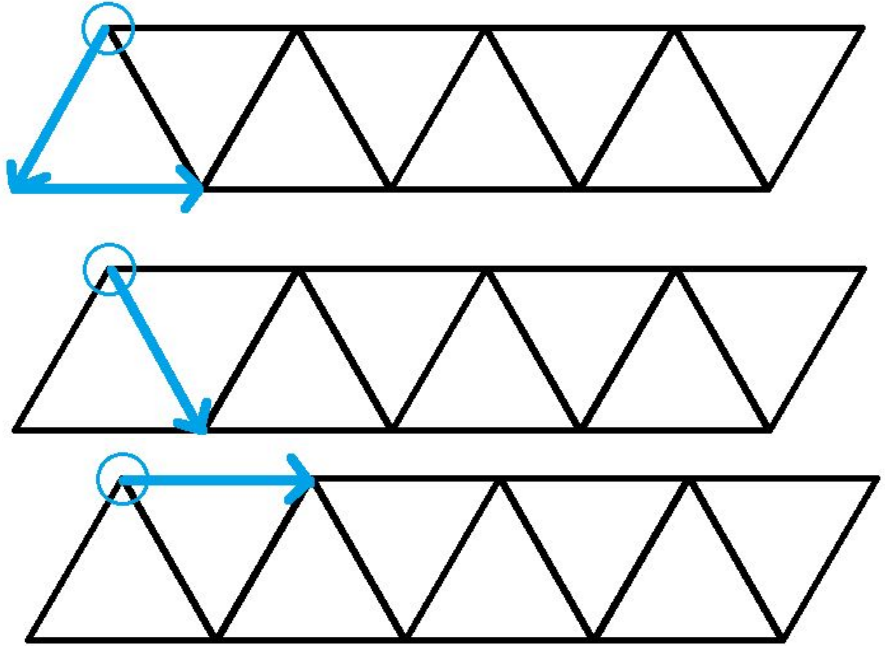
Problem F: Illuminati



- The path starts and ends there.
- Let $f(n)$ be the answer for n.

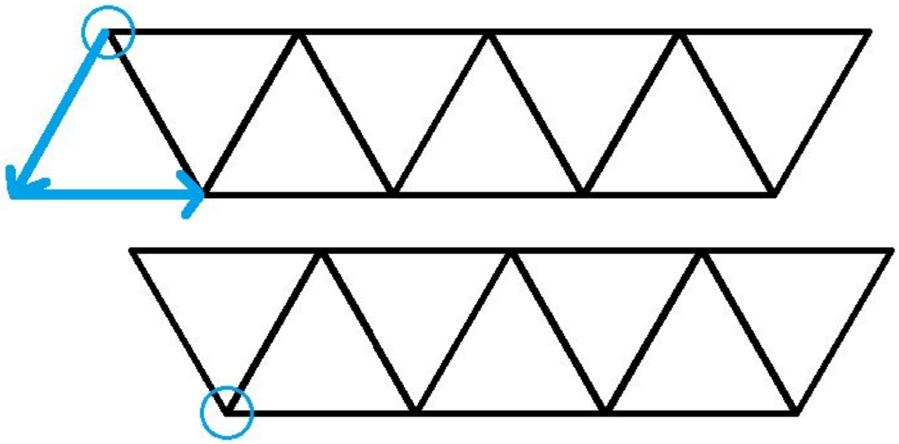
Problem F: Illuminati

- The path can begin in three ways:



Problem F: Illuminati

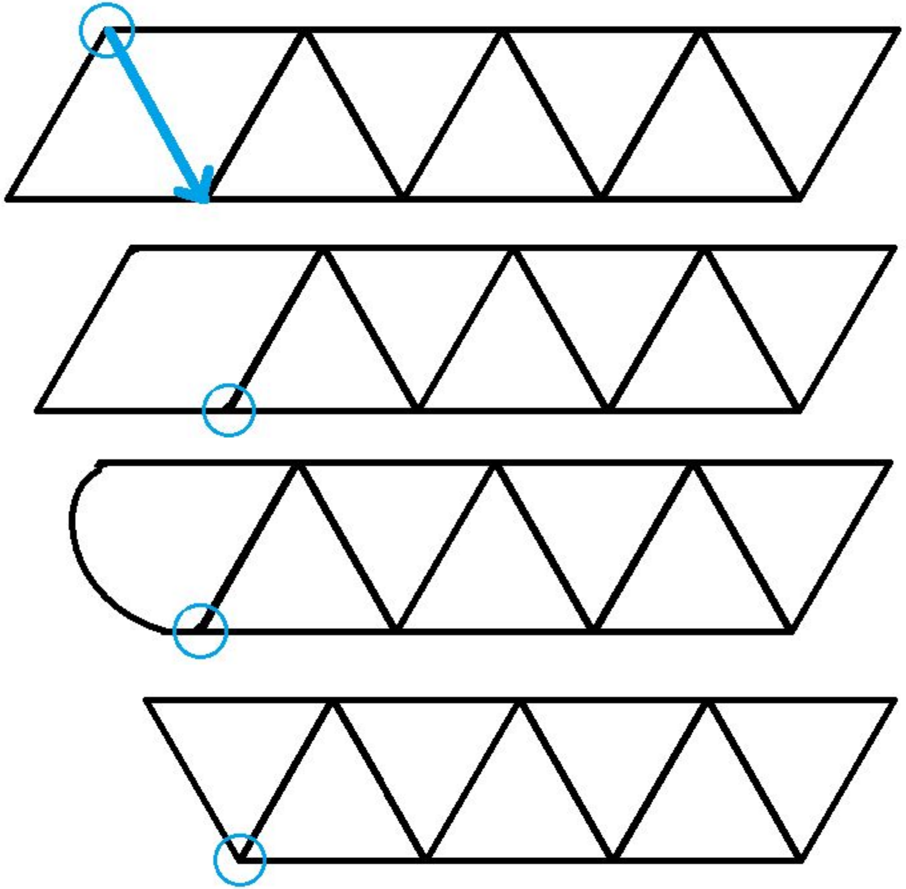
- Case 1:



- f(n - 1)

Problem F: Illumina

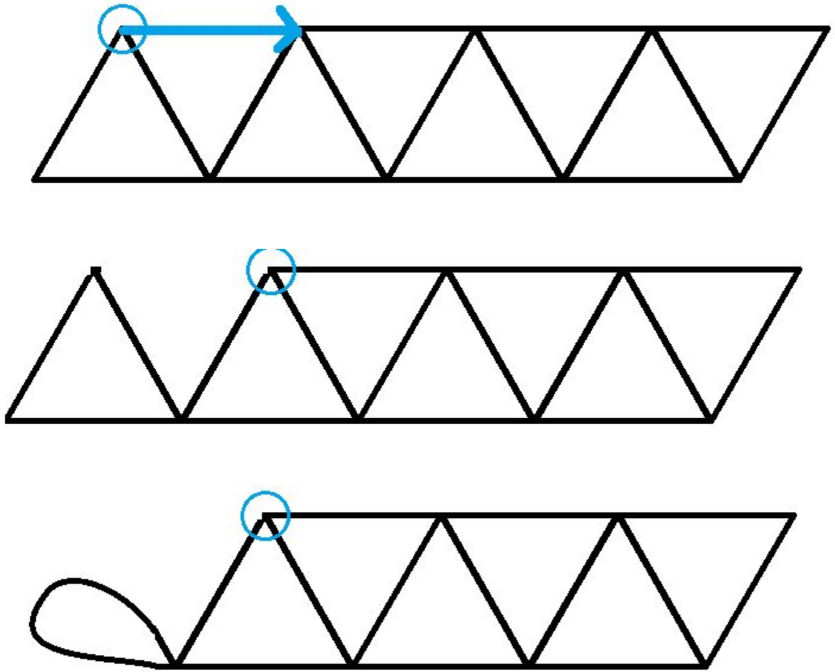
- Case 2:



- f(n - 1)

Problem F: Illuminati

- Case 3:



- 2 choices on how to traverse the loop
- $2 \cdot f(n - 2)$

Problem F: Illuminati

- $f(n) = 2 \cdot f(n - 1) + 2 \cdot f(n - 2)$
- Base case: $f(2) = 12$, $f(3) = 32$.
- Special case: $f(1) = 6$, doesn't follow recurrence.
    - Reason: no odd degrees, etc.

Problem F: Illuminati

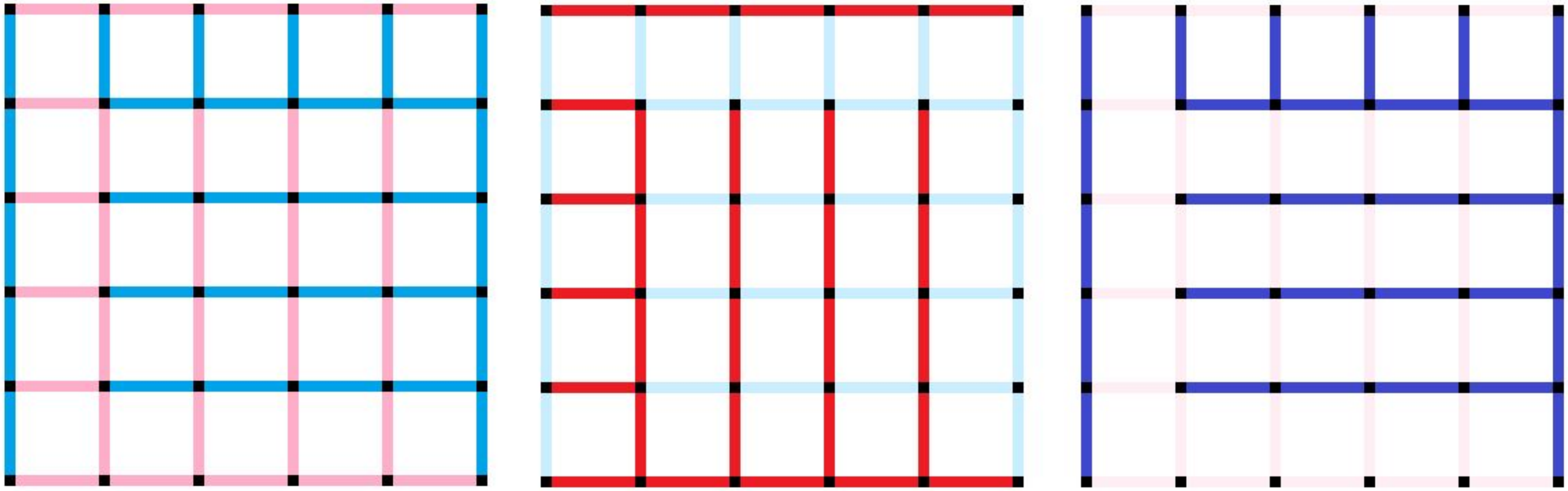
- f(n) = 2·f(n - 1) + 2·f(n - 2)
- DP can give up to n ≤ 10^6.
- Use matrix exponentiation:

$$\begin{bmatrix} 2 & 2 \\ 1 & 0 \end{bmatrix}^{n-2} \begin{bmatrix} f(3) \\ f(2) \end{bmatrix} = \begin{bmatrix} f(n+1) \\ f(n) \end{bmatrix}$$

- O(log n).

Problem A: Donut Cross

- Everything possible.



Problem E: MechaMocha

- Left-right sweep.
    - Use x-coordinate as time.
- Events:
    - On rectangle left edge, range increment.
    - On rectangle right edge, range decrement.
- Between events:
    - Range query “how many 1s”.

Problem E: MechaMocha

- Insight: “1” only appears as either the minimum or the second minimum.
- **Segment tree with lazy propagation:** Keep track of minimum and second minimum, and their counts.
- (m_1, c_1, m_2, c_2)
  - (minimum, count, 2nd minimum, 2nd count)

Problem E: MechaMocha

- On increment $(m_1, c_1, m_2, c_2) \rightarrow (m_1 + 1, c_1, m_2 + 1, c_2)$
- On decrement $(m_1, c_1, m_2, c_2) \rightarrow (m_1 - 1, c_1, m_2 - 1, c_2)$
- On combine $(m_1, c_1, m_2, c_2) + (m_1', c_1', m_2', c_2')$, handle all cases

Problem L: Slip'n'Slide

- $n \leq 8$ suggests brute force.
- Cayley's formula gives $n^{n-2} \leq 8^6 = 262{,}144$ trees. Just need to know how to enumerate them.
- No need to ensure each tree is enumerated only once, since we're taking max.

Problem L: Slip'n'Slide

- Insight: Enumerate *rooted* trees instead. Easier.
- Idea 1: Guess a topological sort then guess the backward edges.
  - $n! \cdot (n - 1)! \leq 203212800$, too much.
- Improvement: The “root” can be node 1.
  - $(n - 1)! \cdot (n - 1)! \leq 25401600$, might still be too tight.

Problem L: Slip'n'Slide

- Idea 2: Enumerate rooted trees by enumerating each level recursively.
    - $n^{n-1} \leq 2097152$, good enough!

Problem L: Slip'n'Slide

- More sophisticated method involves **Prüfer sequences**
  - Enumerates each tree exactly once
  - But might be tough to code, hence not recommended for this problem.
    - Still, learning it is useful!

Problem B: Adding Time

- 0: 0
- 1: 10
- 2: 19
- 3: 199
- 4: 19999999999999999999999
- 5: 199999999999...99999999999 [lots of 9s]
- 6: 199999999999...99999999999 [*lots* of 9s]

Problem B: Adding Time

- Each number goes to the previous.
- Recurrence: $f(n) = 2 \cdot 10^{(f(n-1) - 1)/9} - 1$.
- Euler-Fermat theorem
  - If $\gcd(a,m) = 1$, then $a^x = a^y \pmod{m}$ iff $x = y \pmod{\varphi(m)}$
    - φ denotes Euler's totient theorem
- Need to extend to case $\gcd(a,m) \neq 1$.

Problem B: Adding Time

- Extension:
  - If $x, y \geq \lg m$, then $a^x = a^y \pmod m$ iff $x = y \pmod{\varphi(m)}$
- Mostly true in our case since $\lg m \leq 2 \cdot 10^7 < 25$.
- Recursion!
- $f(n) \bmod m = (2 \cdot 10^{((f(n-1) \bmod 9\varphi(m)) - 1) / 9) \mathit{mod}\ m} - 1) \bmod m$.
  - For *mod*, just ensure the result is ≥ 25, adjust if necessary

Problem K: Planet Link

- $f(T, k)$ = # of ways to split tree T into up to k paths.
- Insight: each way of splitting into t paths corresponds to a choice of t - 1 edges to remove.

Problem K: Planet Link

- Upper bound: $f(T, k) \leq \sum_{g=1}^{\min(n,k)} \binom{n-1}{g-1} \binom{k}{g} g!$
    - All selections of up to k - 1 edges might be valid.
- Achieved by **path graph**.
    - Thus, this is the maximum.

Problem K: Planet Link

- Lower bound: $$f(T,k) \geq \sum_{g=n-2}^{n} \binom{n-1}{g-1}\binom{k}{g} g!$$
  - Keeping up to two edges always results in lines. [Why?]
- Achieved by **star graph**.
  - Thus, this is the minimum.

Problem K: Planet Link

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

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

Problem C: Wildcard List

- Problem: Find $$\sum_{k=0}^{n} \binom{ak+b}{ck+d}$$
- Lucas's Theorem: For prime p,

$$\binom{n}{r} \equiv \binom{\lfloor n/p \rfloor}{\lfloor r/p \rfloor} \binom{n \bmod p}{r \bmod p} \pmod{p}$$

Problem C: Wildcard List

$$\sum \binom{ak+b}{ck+d} \equiv \sum \binom{\lfloor \frac{ak+b}{p} \rfloor}{\lfloor \frac{ck+d}{p} \rfloor} \binom{(ak+b) \bmod p}{(ck+d) \bmod p}$$

- Solution: Let k = qp + r, and enumerate q and r independently.
  - Assume L ≡ -1 (mod p) so they're truly independent. We can add the "leftover terms" separately.

Problem C: Wildcard List

$$\sum \binom{ak+b}{ck+d} \equiv \sum \binom{\lfloor \frac{ak+b}{p} \rfloor}{\lfloor \frac{ck+d}{p} \rfloor} \binom{(ak+b) \bmod p}{(ck+d) \bmod p}$$

- $k = qp + r$
- ($ak + b \equiv ar + b$) and ($ck + d \equiv cr + d$), so the second term is constant. Factor it out.
- The sum reduces to a recursive call!
    - Just memoize. There will only be a few distinct calls.

Problem J: Expando

- Range update: range increase by x.
- Range query: sum $a[i] \cdot a[i+1]$.
- **Segment tree with lazy propagation**.
- Range update is tricky.

Problem J: Expando

- On range update, $\Sigma a[i] \cdot a[i+1]$ becomes
  - $\Sigma(a[i] + x)(a[i+1] + x)$
  - $\Sigma a[i] \cdot a[i+1] + x\Sigma a[i] + x\Sigma a[i+1] + x^2\Sigma 1$
- So we need to keep track of subarray sums and counts as well.

Problem I: Tagolympics

- Problem: $\displaystyle\sum_{x=n}^{2n} \sum_{\substack{y=n \\ \gcd(x,y)=1}}^{2n} 1$

Problem I: Tagolympics

- Variant: $$\sum_{\substack{n/2<x,y\le n\\ \gcd(x,y)=1}} 1$$
- This is almost the same (easy to convert), but turns out to give a cleaner solution.
- Define: $$f(n) := \sum_{\substack{n/2<x,y\le n\\ \gcd(x,y)=1}} 1$$

Problem I: Tagolympics

$$
\begin{aligned}
(n - \lfloor n/2 \rfloor)^2 &= \sum_{n/2 < x, y \le n} 1 \\
&= \sum_g \sum_{\substack{n/2 < x, y \le n \\ \gcd(x,y) = g}} 1 \\
&= \sum_g \sum_{\substack{n/g/2 < x, y \le n/g \\ \gcd(x,y) = 1}} 1 \\
&= \sum_g f(\lfloor n/g \rfloor)
\end{aligned}
$$

Problem I: Tagolympics

- We get: $$f(n) = (n - \lfloor n/2 \rfloor)^2 - \sum_{g \geq 2} f(\lfloor n/g \rfloor)$$
- At this point, simple memoization already gives O(n log n), though this is not enough.

Problem I: Tagolympics

- We get: $$f(n) = (n - \lfloor n/2 \rfloor)^2 - \sum_{g \geq 2} f(\lfloor n/g \rfloor)$$
- Insight: $\lfloor n/g \rfloor$ has at most $2\sqrt{n}$ distinct values.
- By cleverly compressing sums with the same arguments, we get $O(\sqrt{n})$ per n, and overall:

$$O\left(\sum_{i=1}^{\sqrt{n}} \sqrt{i} + \sum_{i=1}^{\sqrt{n}} \sqrt{\frac{n}{i}}\right) = O\left(\int_1^{\sqrt{n}} \left(\sqrt{x} + \sqrt{\frac{n}{x}}\right) dx\right) = O(n^{3/4})$$

Problem I: Tagolympics

- $O(n^{3/4})$ is fast, but it can still be improved! By sieving small values, we can get $O(n^{2/3}(\log n)^{1/3})$!
    - or, less precisely, simply $\tilde{O}(n^{2/3})$
- I won't give all the details, but one can learn these sorts of techniques from Project Euler!

Thank you!

- Kevin Charles Atienza
    - Problem setter + Judge
- Jared Guissmo Asuncion, M.Sc.
    - Theme supervisor