Задача A. Додайте їх усі!

Спочатку обчисліть проміжну послідовність D:

$$D_k = \sum_{i+j=k} (A_i + B_j)$$

Потім отримайте відповіді, взявши префіксні суми D. Значення D_k можна виразити як:

$$D_k = \sum_{i=\max(1,k-N)}^{\min(k-1,N)} A_i + \sum_{j=\max(1,k-N)}^{\min(k-1,N)} B_j$$

Завдяки попередньому обчисленню префіксних сум, це можна обчислити за $O(N)$ часу.

Задача B. Дужки та Реверсований рядок

Ключові спостереження

- Перетворіть дужки на послідовність ± 1: $a_i = \begin{cases} +1 & \text{якщо } S_i = \text{‘(’} \\ -1 & \text{якщо } S_i = \text{‘)’} \end{cases}$
- Нехай $S_0, S_1, \ldots, S_N$ будуть префіксними сумами
- X є дійсним $\iff$ $S_k \geqslant 0 \ \forall k$ та $S_N = 0$

Формула підрахунку операцій

$f(X) = x + \frac{|S_N + 2x|}{2}$ де: $x = \left\lceil -\frac{\min\{S_0, \ldots, S_N\}}{2} \right\rceil$

- x: Операції для того, щоб зробити всі префіксні суми ненегативними
- $\frac{|S_N+2x|}{2}$: Додаткові операції для балансування загальної суми

Ефективні обчислення

Для кожної ротації T_i:

1. Обчисліть для суфікса $S_{i+1} \cdots S_N$:
   - Мінімальна префіксна сума
   - Загальна сума
2. Обчисліть для префікса $S_i \cdots S_1$:
   - Мінімальна префіксна сума
   - Загальна сума
3. Об’єднайте результати для обчислення $f(T_i)$

Може бути обчислено за $O(N)$ часу, обробляючи в зворотному та прямому порядку.

Задача C. Кастомні банкноти

Підхід динамічного програмування

Визначте $f(a, b, c)$ як мінімальну загальну кількість банкнот для сум (a, b, c).

Рекурентне співвідношення

Для наступного номіналу d:

- Потрібно $\lfloor a/d \rfloor + \lfloor b/d \rfloor + \lfloor c/d \rfloor$ банкнот номіналом d
- Залишки $a\%d$, $b\%d$, $c\%d$ обробляються рекурсивно

$$f(a,b,c) = \min_d \left\{ f\left(\left\lfloor \frac{a}{d} \right\rfloor, \left\lfloor \frac{b}{d} \right\rfloor, \left\lfloor \frac{c}{d} \right\rfloor \right) + a\%d + b\%d + c\%d \right\}$$

Оптимізації

- Розглядайте лише $(a,b,c) = \left(\frac{A}{g}, \frac{B}{g}, \frac{C}{g}\right)$ для різних g
- Кількість унікальних станів: $O(\max(A,B,C))$
- Для переходів потрібно розглядати лише максимальні значення d
- Реалізуйте з використанням мемоізованої рекурсії

Аналіз складності

Обчислення відбувається за формулою:

$$g(x) = \min\{g(x/2), g(x/3), \ldots, g(\lfloor \sqrt{x} \rfloor), \ldots, g(1)\}$$

- Часова складність: $O(\max(A,B,C)^{3/4})$
- Просторова складність: $O(\max(A,B,C)^{1/2})$

Задача D. Робота з послідовністю

Рішення

Метод динамічного програмування

Визначимо:

$$dp[i][x] := \text{Мінімальні операції, щоб зробити вузол } i \text{ та його піддерево дійсними з } A_i = x$$

Базовий випадок (Листя)

Для листя:

$$dp[i][x] = |A_i - x|$$

Рекурентне співвідношення

Для внутрішніх вузлів:

$$dp[i][x] = \min_y \left(dp[2i][y] + dp[2i+1][x-y] \right) + |A_i - x|$$

Ключові спостереження

- Для всіх вузлів i, $dp[i]$ є опуклою функцією від x
- Для вузла k рівнів вище листя:
  - Нахил $dp[i]$ варіюється від $-k$ до k
  - Має k точок, де нахил збільшується на 2
- Дозволяє $O(k)$ обчислення для кожного вузла

Ефективне обчислення

- **Точки зміни**: Критичні значення x, де змінюється нахил
- Для вузла i:
  - Точки зміни = Мінковський добуток точок зміни дітей
  - Вставити A_i як додаткову точку зміни

Аналіз складності

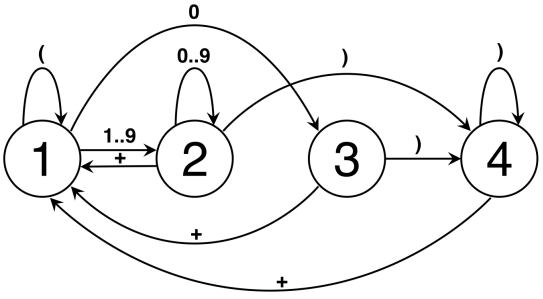
- **Попередня обробка**:
$$\sum_{k=1}^{N} k2^{N-k} = O(2^N)$$
- **Запити на оновлення**:
$$\sum_{k=1}^{N} k = O(N^2) \text{ за запит}$$
- **Всього**:
$$O(2^N + QN^2)$$

Задача E. Генерація виразів

Рішення

Оскільки вираз містить лише операцію +, нам не важливо, в якому порядку виконуються операції. Це означає, що нам потрібно обчислити суму всіх чисел у всіх дійсних виразах, які можуть бути сформовані.

Зазвичай для розбору виразів ми можемо використовувати автомат + стек (це називається автомат з розширенням). Тут ми можемо спростити це і використовувати автомат + лічильник глибини. Найпростіший автомат виглядає так (зверніть увагу, що нам потрібен додатковий стан для обробки одиничних нулів):



Щоб розібрати вираз, ми проходимо через символи і робимо відповідні переходи. Додатково ми запам'ятовуємо поточну глибину (баланс дужок). Таким чином, поточний стан процесу розбору можна описати як пару (x, d), де x — це стан автомата, а d — поточна глибина. Глибина не перевищує n, тому у нас є $O(n)$ можливих станів.

Тепер ми можемо зробити динамічне програмування. Для кожної пари (x, d) ми запам'ятовуємо три значення (w, s, c): w — це кількість способів досягти цього стану, s — це сума всіх повністю розібраних чисел на шляху до цього стану, а c — це сума чисел, які наразі розбираються.

Після цього нам потрібно ретельно реалізувати всі переходи автомата. Коли ми читаємо якийсь символ a, ми робимо $c' = c \cdot 10 + a \cdot w$, оскільки кожне поточне число буде помножене на 10, а потім всі w з них будуть збільшені на a, а коли ми читаємо "+" або ")", ми робимо $s' = s + c$ і $c' = 0$.

Коли ми бачимо символ ?, ми просто намагаємося замінити його на всі можливі символи і робимо всі переходи в ДП. Щоб прискорити алгоритм, ми можемо згрупувати всі цифри тут і зробити один великий перехід замість 10.

Загальна складність часу становить $O(n^2)$.

Задача F. Інвертування

Рішення

Рішення початкового стану

Визначте індикатори інвертування:

$$f_i = \begin{cases} 1 & \text{якщо шлях до } i \text{ інвертовано} \\ 0 & \text{в іншому випадку} \end{cases}$$

Ключові властивості:

- Кожна конфігурація $(f_1, ..., f_N)$ унікально визначає значення вершин
- Можна обчислити знизу вгору від листя
- Мінімальні операції $= \sum_{i=1}^{N} f_i$

Обробка запитів на інвертування

Коли значення вершини i інвертується:

- Інвертуйте як f_i, так і f_{p_i} (якщо існує)
- Спеціальний випадок: Тільки f_1 інвертується, коли корінь інвертується
- Оновлення відповіді за $O(1)$ на запит

Аналіз складності

- Початкове рішення: $O(N)$
- Обробка запитів: $O(1)$ на запит
- Загальна: $O(N + Q)$

Задача G. Генерація трикутників

Дійсні типи трикутників

Сім можливих конфігурацій трикутників:

- Рівносторонні трикутники:
  - (1,1,1)
  - (2,2,2)
  - (3,3,3)
- Рівнобедрені трикутники:
  - (1,2,2)
  - (1,3,3)
  - (2,2,3)
  - (2,3,3)

Ключове усвідомлення

- **Принцип оптимізації**: Існує оптимальне рішення, де кожен рівнобедрений тип з'являється не більше двох разів
- **Перетворення**: 3 рівнобедрені трикутники можна перетворити на 2 рівносторонні трикутники
  - Приклад: $3 \times (1, 1, 2) \to 2 \times (1, 1, 1) + 1(2, 2, 2)$

Алгоритм

- Брутфорс-пошук по можливим кількостям рівнобедрених трикутників (до 81 комбінації)
- Альтернативний підхід:
  - Спочатку максимізуйте рівносторонні трикутники
  - Потім використовуйте мемоізовану рекурсію для залишкових паличок
- Належним чином обробляйте негативні кількості

Задача H. Розфарбування півплощин

Ключові спостереження

- Потрібно розглядати операції лише вздовж країв багатокутника
- Критичне обмеження:
  - Не можна вибрати півплощину, яка скасує попередні операції з краями
  - Необхідно зберігати всі раніше розфарбовані області

Підхід до розв'язання

Стратегія зворотного порядку

- **Проблема прямого порядку**: Можна застрягти в процесі
- **Інсайт зворотного порядку**:
  - Край можна вибрати лише тоді, коли всі краї в одній півплощині вже оброблені
  - Більше оброблених країв $\Rightarrow$ легше вибрати залишилися краї
  - Це свідчить про те, що жадібний вибір є оптимальним

Оптимізація складності

- Наївний підхід: $O(N^3)$ часу
- Оптимізований підхід:
  - Відстежуйте кількість необроблених країв у кожній півплощині
  - Використовуйте диференційні оновлення, щоб досягти $O(N^2)$ часу

Задача I. Цілі числа на краях

Визначення

- Пара (s, t) є *гарною*, якщо всі шляхи s-t мають квадратні добутки ваг
- $w(P) :=$ добуток ваг ребер у шляху P
- $f(x) :=$ частина x, вільна від квадратів (поділити на всі квадратні фактори)
  - $f(48) = 3$, $f(120) = 30$, $f(4) = 1$
  - $f(x) = 1 \Leftrightarrow x$ є досконалим квадратом

Ключове усвідомлення

- Будь-які два шляхи s-t можуть бути перетворені один в одного шляхом перевертання ребер трикутника
- Для гарних пар:
  1. Існує шлях P з $f(w(P)) = 1$
  2. Усі трикутники мають добутки ваг ребер, які є досконалими квадратами
- Ці умови є як необхідними, так і достатніми

Підхід до розв'язання

Коли умови виконуються

- Візьміть остовне дерево, що виходить з (1,1)
- Визначте $w(v) :=$ добуток ваг ребер від кореня до v в дереві
- (s, t) є гарною $\iff f(w(s)) = f(w(t))$

Імплементація

- Обробляйте великі числа за допомогою хешування:
  - Призначте хеш-значення простим числам
  - Використовуйте хешування Зобріста для управління добутками шляхів
- Часова складність: $O(\max(A) + N^2 \log N)$

Задача J. Джунглі та Піраміда

Комбінаторне підрахування

Для даної послідовності порівнянь з:

- a "менших" порівнянь
- b "більших" порівнянь

Кількість дійсних перестановок:

$$\binom{K-1}{a} \times \binom{N-K}{b} \times (N-1-a-b)!$$

Умови дійсної послідовності

Послідовність порівнянь є дійсною, якщо:

- Підтримує інтервал пошуку (l, r), де $r - l \geqslant 2$ протягом всього часу
- Початковий інтервал: $N + 1$ (віртуальні межі)
- Оновлення:
  - "Менше": $r \leftarrow \left\lfloor \frac{r-l+1}{2} \right\rfloor$
  - "Більше": $r \leftarrow \left\lfloor \frac{r-l}{2} \right\rfloor$

Підхід динамічного програмування

- Параметри простору станів:
  - Довжина послідовності
  - Кількість "менших" порівнянь
  - Поточний розмір інтервалу
- Потрібно розглянути лише $O(\log^2 N)$ різних станів
- Можна вирішити за допомогою ДП або мемоізованої рекурсії
- Часова складність: $O(\log^2 N)$ на тестовий випадок (оптимізується до $O(\log N)$)

Задача K. Король, Джокер та Подільність

Ключове усвідомлення

Число x не ділить жодного з $[L, R]$, якщо і тільки якщо:

$$\left\lfloor \frac{L-1}{x} \right\rfloor = \left\lfloor \frac{R}{x} \right\rfloor$$

Алгоритм

- Перерахувати діапазони часток для обох $L - 1$ та R
- Знайти інтервали, де частки рівні
- Часова складність: $O(\sqrt{R})$ на запит

Альтернативний підхід

- Обробляти запити офлайн у порядку зростання L
- Використовувати 2D сегментне дерево з бінарним пошуком
- Складніша реалізація з більш жорсткими часовими обмеженнями

Задача L. Давайте виміряємо перестановки

Ключові інсайти

- Для будь-якої перестановки: Ліза$(P) \geqslant$ Лео(P)
- Рівність виконується, коли перестановка не має спадної підпослідовності довжини 3 ($|\text{LDS}| \leqslant 2$)

Як виглядає хороша перестановка

- Ітеруйте елементи зліва направо
- Кожен елемент повинен бути або
  - більшим за всі попередні елементи, або
  - мінімальним невикористаним елементом
- Тож ми дивимося на наступний елемент
  - якщо це новий максимум, ми "зберігаємо" всі елементи менші за нього,
  - якщо ні, перевірте, що це мінімальний збережений елемент, і видаліть його зі сховища
- Щоб підтримувати сховище, нам потрібна проста черга

Підхід динамічного програмування

- Визначте стан DP:
  - $DP[i, j]$:= Кількість хороших префіксів розміру i з j збереженими елементами
- Перехід:
  - $(i, j) \to (i+1, j')$ для всіх $j' \geqslant j$: зберігайте деякі елементи, потім додайте новий максимум
  - $(i, j) \to (i+1, j-1)$: візьміть мінімальний збережений елемент
- Це дає складність: $O(n^3)$ (або $O(n^2)$), що занадто повільно

Оптимізоване рішення

- Зверніть увагу, що процес виглядає подібно до послідовності дужок
- Насправді існує бієкція з послідовностями дужок:
  - послідовність "`((((`" з $k+1$ відкритими дужками означає "зберегти k елементів, потім додати новий максимум",
  - одна дужка "`)`" означає "взяти елемент зі сховища",
- Відповідь - це кількість решіткових шляхів від $(k, \max(a))$ до (n, n)
- Може бути обчислена за допомогою того ж трюку, що й для чисел Каталана.

- Часова складність: $O(n)$

Задача M. Магентові та Цианові Цілі Числа

Ключові Ідеї

Всі однакові елементи повинні мати один і той же колір (виняток, коли всі елементи однакові, цей випадок тривіальний).

Тепер ми можемо припустити, що всі елементи різні, давайте відсортуємо їх: $x_1 < x_2 < ... < x_m$. Давайте доведемо, що x_{m-1} та x_m повинні мати різні кольори.

- Припустимо, що x_{m-1} та x_m обидва магента.
- $d_m \leqslant \gcd(x_{m-1}, x_m) \leqslant x_m - x_{m-1}$.
- $d_c \leqslant x_{m-2} < x_{m-1}$.
- $d_m + d_c \leqslant x_m$.
- Це гірше, ніж якщо ми пофарбуємо x_m у магенту, а все інше - у цианový.

Давайте доведемо, що всі $x_1, x_2, ..., x_{m-2}$ повинні мати один і той же колір.

- Припустимо, що x_{m-1} - цианový, x_m - магента, і у нас є принаймні один елемент для кожного кольору.
- $d_m \leqslant x_m/2$, оскільки це дільник x_m, менший за x_m.
- $d_c \leqslant x_{m-1}/2$, з тієї ж причини.
- $d_m + d_c \leqslant x_m$.
- Це гірше, ніж якщо ми пофарбуємо x_m у магенту, а все інше - у цианový.

Це означає, що у нас є лише два варіанти для розгляду: (x_m, все інше) і (x_{m-1}, все інше).

Розв'язок

Реалізуйте за допомогою сегментного дерева, що зберігає:

- Максимальні та другі максимальні значення
- НСД всіх решти елементів

Для кожного запиту обчисліть результат на сегменті та візьміть максимум з двох варіантів. Часова складність $O(n + q \log n)$ (обчислення НСД).

Задача N. Гарні підпослідовності

Розв'язок

Обчислення максимальної довжини

Визначте стан DP:

$dp(i, p) :=$ Максимальна довжина гарної підпослідовності в $(A_1, \ldots, A_i)$, що закінчується кратним p

Рекурентне співвідношення:

$$dp(i+1, p) = \begin{cases} \max\{dp(i, q) \mid q \text{ взаємно просте з } A_{i+1}\} + 1 & \text{якщо } p \mid A_{i+1} \\ dp(i, p) & \text{в іншому випадку} \end{cases}$$

- Відповідь: $\max dp(N, *)$
- Складність: $O(\sum \log A_i)$ з використанням інлайн DP

Підрахунок підпослідовностей максимальної довжини

Визначте стан DP:

$dp2(i, p) :=$ Кількість підпослідовностей максимальної довжини в $(A_1, \ldots, A_i)$, що закінчується кратним p

- Використовуйте інверсію Мебіуса для підрахунку дільників
- Складність: $O(\sum 2^{\omega(A_i)})$, де $\omega(n)$ підраховує різні прості множники