XXII Командная олимпиада школьников Санкт-Петербурга по программированию

9 ноября 2014 года

Задача A. «ABCD-код»

Задача A. «ABCD-код»

- Идея задачи — Андрей Станкевич, Николай Ведерников
- Подготовка тестов — Николай Ведерников
- Разбор задачи — Николай Ведерников

Постановка задачи

Проверить для каждого четырехзначного числа «ABCD», что «AB»^{2} + «CD»^{2} имеет остаток 1 от деления на 7.

Решение задачи

- Чтобы получить «AB», надо «ABCD» поделить на 100 и взять целую часть.
- Чтобы получить «CD», надо «ABCD» поделить на 100 и взять остаток от деления.
- Затем возвести в квадрат каждую часть и проверить, равен ли отстаток от деления на 7 одному.

Задача B. «Шахматы»

Задача B. «Шахматы»

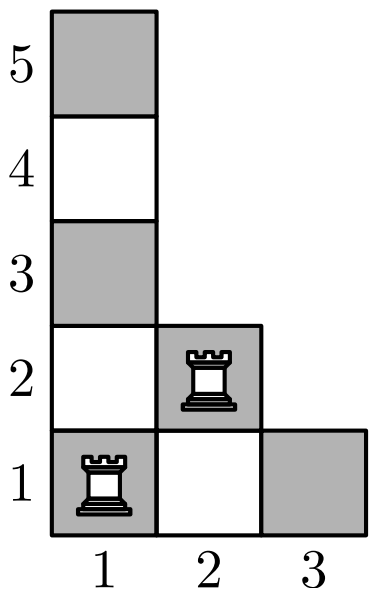
- Идея задачи — Геннадий Короткевич
- Подготовка тестов — Анна Малова
- Разбор задачи — Анна Малова

Постановка задачи

Дана ступенчатая клетчатая фигура. Нужно поставить минимальное число ладей, которые будут бить все клетки доски.

Решение задачи

- Каждая ладья бьёт строку и столбец, но не более.
- После этого понятно, что нам выгодно выставлять ладьи на диагонали, которая выходит из левого нижнего угла.



Задача C. «Завоевание»

Задача C. «Завоевание»

- Идея задачи — Антон Гардер
- Подготовка тестов — Антон Гардер
- Разбор задачи — Антон Гардер

Постановка задачи

Города:

- Есть n городов
- В i-м городе a_i воинов, можно покупать по одному за c_i монет
- Если в городе осталось меньше солдат, чем уже есть у нас, получаем их бесплатно

Цель:

- Получить всех воинов

Идея задачи

- Занумеруем города по возрастанию количества воинов в них
- Динамическое программирование. $d_{i,k}$ - минимальное количество монет, которое нужно, чтобы забрать всех воинов из первых i городов, причем из городов с номерами строго больше i мы забрали уже k воинов
- Ответ — $d_{n,0}$

Как пересчитывать

Вычисляем $d_{i,k}$:

- В i-м городе можем купить t воинов $(t \leqslant a_i)$
- Можем полностью захватить город, если $\sum_{j=1}^{i-1} a_j + t + k > a_i - t$
- Покупку этих t воинов мы уже учитывали раньше
- $d_{i,k} = min_{\{t | \sum_{j=1}^{i-1} a_j + t + k > a_i - t\}} (d_{i-1,k+t} + t \cdot c_i)$
- Заметим, что нам никогда не нужно покупать больше чем $max(a_i)$ воинов
- Время работы: $O(n \cdot max^2(a_i))$

Альтернативное решение

- Занумеруем города по возрастанию количества воинов в них
- Жадность. Обрабатываем города, начиная с конца. Для последнего города знаем, что захватили все меньшие, значит можем посчитать, сколько воинов мы обязаны купить в этом городе.
- Для i-го города знаем, сколько воинов захватили из меньших городов и сколько должны купить в больших. Значит, можем посчитать, сколько должны купить в i-м городе.
- Время работы: $O(n \cdot log(n))$

Задача D. «Дюны»

Задача D. «Дюны»

- Идея задачи — Михаил Дворкин
- Подготовка тестов — Михаил Дворкин
- Разбор задачи — Михаил Дворкин

Постановка задачи

Порывы ветра:

- Было n порывов ветра
- i-ый порыв менял высоту участков дюны с l_i-го по r_i-й
- Изменение высоты: $+x_i, -x_i, +x_i, -x_i, \ldots$

Запросы:

- m запросов: «какая высота в итоге у q_i-го участка?»

Количество участков от 1 до **миллиарда**!

Решение задачи

- Не надо моделировать миллиард участков дюны!
- Для каждого запроса — участка номер q_i — вычислим его высоту
- Просмотрим заново все n порывов ветра, про каждый посчитаем, менял ли он высоту участка q_i и как именно
- Время работы: $O(nm)$

Задача E. «Игра»

Задача E. «Игра»

- Идея задачи — Анна Малова
- Подготовка тестов — Анна Малова
- Разбор задачи — Анна Малова

Постановка задачи

- Есть n столбиков разной высоты.
- Петя может закидывать кольца только на столбики высоты не менее l_1 и не более r_1.
- Вася может закидывать кольца только на столбики высоты не менее l_2 и не более r_2.
- Если на столбик уже закинуто кольцо, то ещё раз туда закидывать нельзя.
- Дети кидают по очереди. Первым ходит Петя.
- Выигрывает тот, кто закинет больше колец.

Решение задачи

- Мальчикам выгодно начать кидать на столбики, которые есть у обоих.
- Считаем *both* — количество столбиков, на которые оба могут закинуть. N_{Petya} и N_{Vasya} — количества столбиков, на которые может закинуть только Петя и на которые может закинуть только Вася, соответственно.

Решение задачи

- Далее моделируем ходы по очереди.
  - $both > 0$, то обязательно закидываем на общие.
  - $both = 0$ и $N_{Player} > 0$, то закидываем на соответствующий столбик.
  - $both = 0$ и $N_{Player} = 0$, то больше не закинуть.

Сложность решения — $O(n + m)$.

Задача F. «НОД и НОК»

Задача F. «НОД и НОК»

- Идея задачи — Сергей Копелиович
- Подготовка тестов — Дмитрий Филиппов
- Разбор задачи — Дмитрий Филиппов

Постановка задачи

Входные данные:

- Даны два натуральных числа a, b до 10^9

Нужно найти два натуральных числа x, y, таких, что

- $x \leqslant y$
- $\text{НОД}(a, b) = \text{НОД}(x, y)$
- $\text{НОК}(a, b) = \text{НОК}(x, y)$
- $y - x$ — минимально среди всех таких пар (x, y)

Решение задачи

- Факторизуем числа a, b — получим два массива f_a, f_b, состоящие из пар (простое число, его степень вхождения)
- Заметим, что если степень вхождения простого числа p в f_a равна α, а в f_b — β, то минимальная степень вхождения числа p в каждое из чисел x и y равна $\min(\alpha, \beta)$, а максимальная — $\max(\alpha, \beta)$
- Отсортируем и сольем массивы f_a, f_b

Что дальше?

- Степень вхождения простого числа p в одно из чисел должна быть минимально возможной, чтобы сошёлся НОД
- Степень вхождения простого числа p в другое должна быть максимально возможной, чтобы сошёлся НОК

Что дальше?

- Простых чисел в факторизации каждого из чисел a, b мало — не больше 10
- Значит, в массиве, полученном слиянием двух факторизаций, их не более 20
- Пользуясь свойством, что каждое простое число входит в числа x, y либо в минимальной степени, либо в максимальной, сделаем полный перебор
- Найдем такие числа x, y, что $x \leqslant y, y - x$ — минимально

Время работы

- Время работы: $O(\sqrt{max(a, b)} + 2^{primes})$

Возможные ошибки

- В ответе $x > y$
- Переполнение 32-битного типа данных во время подсчета чисел x, y в рекурсии

Задача G. «Мерлин»

Задача G. «Мерлин»

- Идея задачи — Виталий Аксенов
- Подготовка тестов — Григорий Шовкопляс
- Разбор задачи — Григорий Шовкопляс

Постановка задачи

Входные данные:

- Есть n сосудов
- Сосуд с номером i характеризуется числом a_i — количестом литров эликсира в нем.

Нужно выбрать k сосудов таким образом, что:

- Можно перелить жидкость из них в оставшиеся, чтобы количество эликсира везде было одинаково.
- Эти k сосудов нужно опустошить и разбить.
- k должно быть наименьшим возможным.

Для начала

- Заметим, что выгоднее всего выливать эликсир из тех сосудов, в котором его больше всего.
- Отсортируем массив сосудов по возрастанию. (Нужно использовать одну из быстрых сортировок, например `quick sort`, `merge sort`, `heap sort`, этого требуют ограничения)

Что дальше?

- Если мы выльем k последних (в отсортированном порядке) сосудов, то останется $n - k$ первых.
- Чтобы в $n - k$ первых сосудов было поровну эликсира, нужно долить до каждого сосуда a_i $(1 \leqslant i \leqslant n - k)$ хотя бы $a_{n-k} - a_i$ литров эликсира.
- Тогда на k накладывается следующее условие: $(a_{n-k} - a_1) + (a_{n-k} - a_2) + \ldots + (a_{n-k} - a_{n-k}) \leqslant a_{n-k+1} + \ldots + a_n$
- Это выражение можно преобразовать:

$$a_{n-k} \cdot (n - k) - \sum_{i=1}^{n-k} a_i \leqslant \sum_{i=n-k+1}^{n} a_i$$

Что осталось?

- Теперь мы можем перебирать k, начиная с нуля до тех пор, пока это условие не выполнится.
- Ограничения на n не позволят каждый раз считать эти суммы заново, поэтому просто предподсчитаем суммы элементов на всех префиксах и суффиксах массива.
- Также нужно обратить внимание, что эти суммы могут не поместится в 32-битный тип данных, и использовать 64-битный.

Итоговая сложность решения — $O(n \log n)$

Второе решение

- Нам хватит посчитать sum — сумма всех элементов.
- Нам просто нужно найти такое максимальное i, что $a_i \cdot i \leq sum$.

Итоговая сложность решения — $O(n \log n)$

Задача H. «Регистрация на олимпиаду»

Задача H. «Регистрация на олимпиаду»

- Идея задачи — Михаил Дворкин
- Подготовка тестов — Андрей Комаров
- Разбор задачи — Андрей Комаров

Постановка задачи

Входные данные:

- Дан список троек ⟨ фамилия, имя, отчество ⟩
- Иногда нарушен порядок и вместо ⟨ фамилия, имя, отчество ⟩ дано ⟨ имя, отчество, фамилия ⟩
- Все фамилии уникальны
- Все имена встречаются хотя бы по два раза
- Множества имён, фамилий и отчеств не пересекаются

Нужно вернуть везде правильный порядок и отсортировать список лексикографически.

Решение

Две задачи:

- Преобразовать неправильные записи в правильные
- Отсортиовать результат

Вторая часть стандартна и не должна вызвать затруднений.

Решение: починка записей

Посмотрим на первую колонку. В ней записаны

- Имена: встречаются хотя бы два раза
- Фамилии: встречаются ровно один раз

Как определить, это имя или фамилия?

- Встречается в файле ровно один раз — фамилия
- Иначе — имя. В этом случае, приведём тройку в правильный порядок

Реализация

- Как определить, сколько раз встречалось слово?
  - С помощью стандартной библиотеки (`std::map`, `HashMap`, `dict`)
    - $O(n \log n)$
  - Пройтись по всем элементам и сравнить
    - $O(n^2)$
- Сортировка
  - Любая быстрая — $O(n \log n)$
  - Любая медленная — $O(n^2)$

Итоговая сложность решения — $O(n \log n)$ или $O(n^2)$.
Оба решения укладывались в отведённое время.

Задача I. «Безопасный путь»

Задача I. «Безопасный путь»

- Идея задачи — Антон Гардер
- Подготовка тестов — Борис Минаев
- Разбор задачи — Борис Минаев

Постановка задачи

Входные данные:

- n прямых
- Две различные точки
- Каждая точка лежит ровно на одной прямой

Нужно найти путь между двумя точками такой, что:

- Путь проходит только по линиям из условия
- Сумма всех углов, на которые надо повернуть при движении вдоль пути, минимальна

Решение

Две подзадачи:

- Построить граф
- Найти кратчайший путь

Решение: построение графа

- Пересечь все пары прямых
- Объединить одинаковые точки
- Вершины графа — направленные отрезки между точками пересечений
- Если конец одного отрезка совпадает с началом другого — добавляем ребро
- Стоимость ребра — 180° - угол между отрезками

Решение: нахождение кратчайшего пути

- $O(n^2)$ вершин
- $O(n^2)$ ребер
- Алгоритм Дейкстры за $O(n^2 \log n)$
- Алгоритм Форда-Беллмана за $O(n^4)$

Задача J. «Преобразование последовательности»

Задача J. «Преобразование последовательности»

- Идея задачи — Георгий Корнеев
- Подготовка тестов — Виталий Демьянюк
- Разбор задачи — Виталий Демьянюк

Постановка задачи

- Дана последовательность из n целых неотрицательных чисел.
- За одну операцию можно увеличить любое число последовательности на единицу.
- Требуется за минимальное количество операций добится того, чтобы последовательность содержала подпоследовательность $1, 2, \ldots, h$

Решение

- Если подпоследовательность a_i, a_{i+1}, ..., a_{i+h-1} содержит a_{i+j}, такой что $a_{i+j} > j + 1$, то ее нельзя привести к последовательности 1, 2, ..., h.
- Иначе, для ее преобразования потребуется $T(i) = \frac{h \cdot (h+1)}{2} - \sum_{j=0}^{h-1} a_{i+j}$ операций.
- Будем рассматривать подпоследовательности в порядке убывания позиций их первых элементов.

Решение

- Заметим, что если $a_i > 1$, то последовательности с началом в позициях $i - min(a_i - 1, h) + 1, \ldots, i$ нельзя преобразовать.
- В переменной L будем хранить минимум величины $i - min(a_i - 1, h) + 1$
- Переход от a_{i+1} к a_i :
  - $T(i) \leftarrow T(i+1) - a_i + a_{i+h}$.
  - Если $i < L$, и $a_i \leqslant 1$, то $res \leftarrow min(res, T(a_i))$
  - $L \leftarrow min(L, i - min(a_i - 1, h) + 1)$
- Время работы : $O(n)$

Задача K. «Крестики-нолики»

Задача K. «Крестики-нолики»

- Идея задачи — Демид Кучеренко
- Подготовка тестов — Демид Кучеренко
- Разбор задачи — Демид Кучеренко

Постановка задачи

- Дана ситуация в игре крестики-нолики на бесконечном поле
- Необходимо определить количество ходов крестиков таких, что крестики либо выигрывают сразу, либо следующим ходом

Решение

- Задача требовала аккуратной реализации
- Сначала определим можем ли мы выиграть за один ход
- Просто переберем все пустые клетки, поставим туда крестик и проверим не получилось ли выигрышной ситуации
- Для этого можно просто пройти на четыре клетки в каждую из восьми сторон
- Не забываем, что ходить можно и вне данного прямоугольника, однако нет смысла уходить дальше чем на одну клетку от него

Решение: победа ноликов

- Если крестики не могут выиграть одним ходом, то посчитаем количество выигрышных ходов для ноликов
- Сделать это можно точно также, как и для крестиков
- Если этих ходов больше одного, то очевидно, что мы не сможем закрыть их одним ходом, то есть нолики выиграли
- Если у ноликов ровно один выигрышный ход, то первым ходом крестики обязаны занять эту клетку
- Если у ноликов выигрышных ходов нет, то переберем первый ход крестиков

Решение: у ноликов есть выигрышный ход I

- Рассмотрим ситуацию, когда у ноликов есть выигрышный ход
- Тогда первый ход крестиков жестко зафиксирован
- Поставим туда крестик, и посчитаем количество победных ходов для крестиков в новой ситуации
- Заметим, что так как изначально победных ходов для крестиков не было, то и сейчас они могли появиться только благодаря вновь поставленному крестику

Решение: у ноликов есть выигрышный ход II

- Поэтому победные ходы следует искать только в небольшом квадрате (пять клеток в каждую из четырех сторон) вокруг вновь поставленного крестика
- Если победных ходов больше одного, то предполагаемый ход является выигрышным, добавляем его в ответ.

Решение: у ноликов нет выигрышного хода I

- Рассмотрим ситуацию, когда у ноликов нет выигрышного хода
- Тогда перебираем первый ход крестиков
- Далее обрабатываем их так же, как и в предыдущем случае
- Благодаря тому, что мы ищем новые выигрышные позиции в небольшом квадрате, сложность решения не увеличится.

Решение: у ноликов нет выигрышного хода II

- Не забываем, что теперь есть смысл уходить на две клетки от границ данного прямоугольника, поскольку мы можем сделать два хода за крестиков.

Итог

- В итоге получается сложное с точки зрения реализации решение, которое работает за $O(nm)$.

Вопросы?