ЦЕНТР
ОЛИМПИАДНОГО
ПРОГРАММИРОВАНИЯ
ЮУРГУ

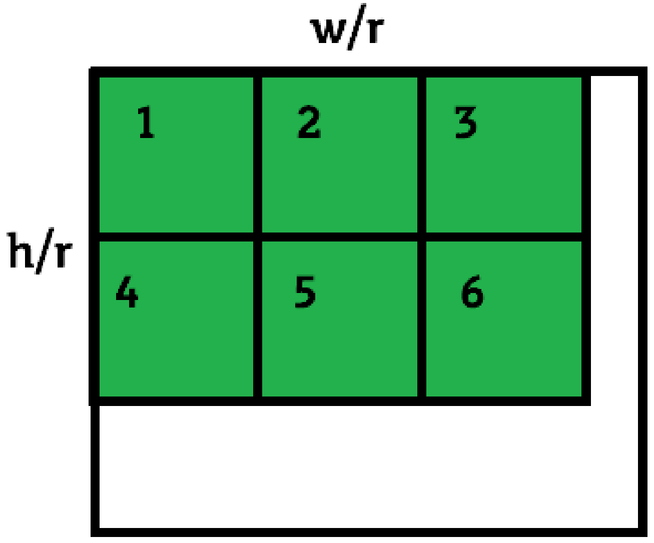
Prime Time

2025

Простые задачи
=
Простые номера

Gifts (2)

(w/h) * (h/r) / 6
(деление с округлением вниз)



Часы (7)

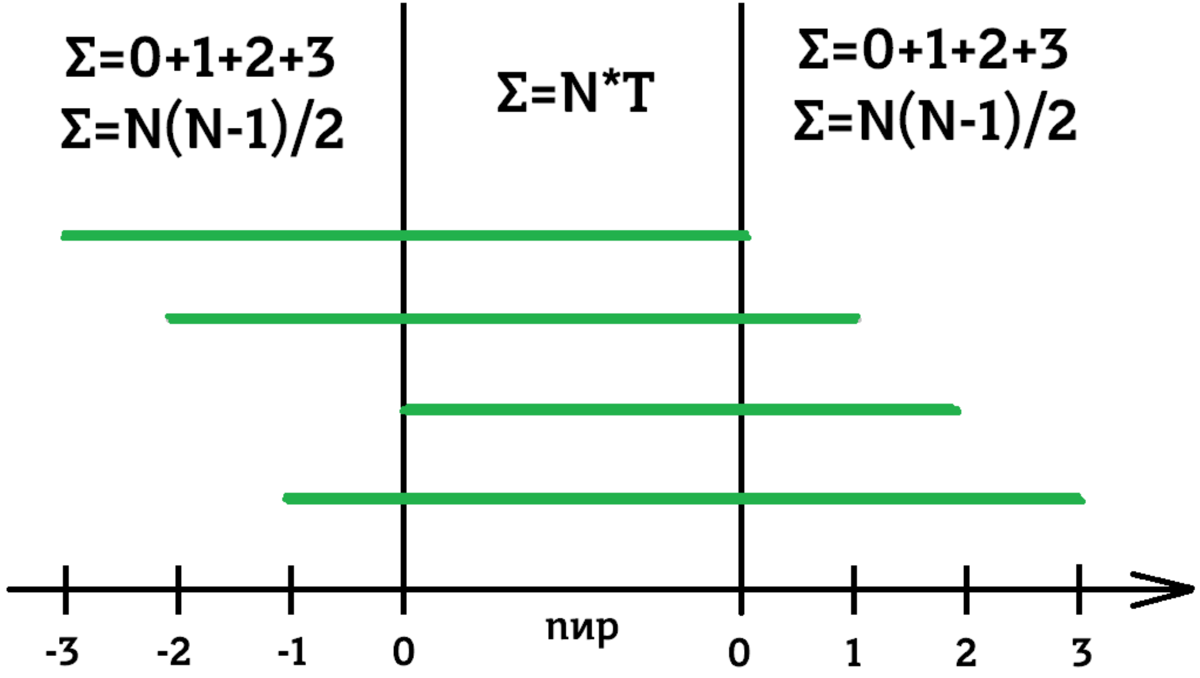
- Непосредственное моделирование
- Сначала секунды, затем минуты, затем часы
- Увеличения на 1 → не больше 24+60+60

T-shirts (3)

Жадный алгоритм:

- пойдем по возрастанию размера
- наденем сначала подходящие, затем на 1 размер больше
- нет смысла пропускать людей/футболки
- анализируется только текущий и следующий размер
- возможно переполнение при суммировании в int

Подземные короли (5)



$$(N(N-1) + NT) / N = N - 1 + T$$

Азартные прыгуны (11)

- Решение за $N^2 \log N$ неэффективно → $N \log N$
- Отсортируем массив, теперь разности p[R] - p[L]
- Группы разностей с одинаковым вычитаемым:
  p[1] - p[0], p[2] - p[0], …, p[n-1] - p[0]
  p[2] - p[1], p[3] - p[1], …, p[n-1] - p[1]
  …
  p[n-1] — p[n-2]
- Всего N-1 группа, разности в группе не убывают

Азартные прыгуны (11)

- Слияние упорядоченных массивов, N-1 указатель (по 1 в каждой группе)
- Структура с быстрым добавлением и извлечением минимума — например, куча или дерево
- В обратную сторону — аналогично

ЦЕНТР
ОЛИМПИАДНОГО
ПРОГРАММИРОВАНИЯ
ЮУРГУ

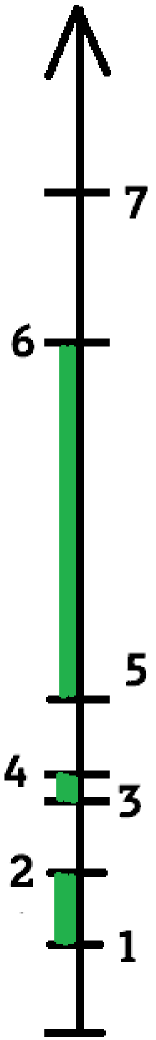
Дележ изумрудов (13)

- Нужно уравнять 1 распиленным изумрудом разность, накопленную на всех остальных изумрудах
- Чем больше изумруд, тем большую разность весов могут уравнять его части
- Остальные можно раздавать парами близкого друг к другу веса (чтобы суммы были «примерно равны»)
- Стратегия: отсортируем все по возрастанию, раздадим четные и нечетные через один, а распилим максимальный

ЦЕНТР
ОЛИМПИАДНОГО
ПРОГРАММИРОВАНИЯ
ЮУРГУ

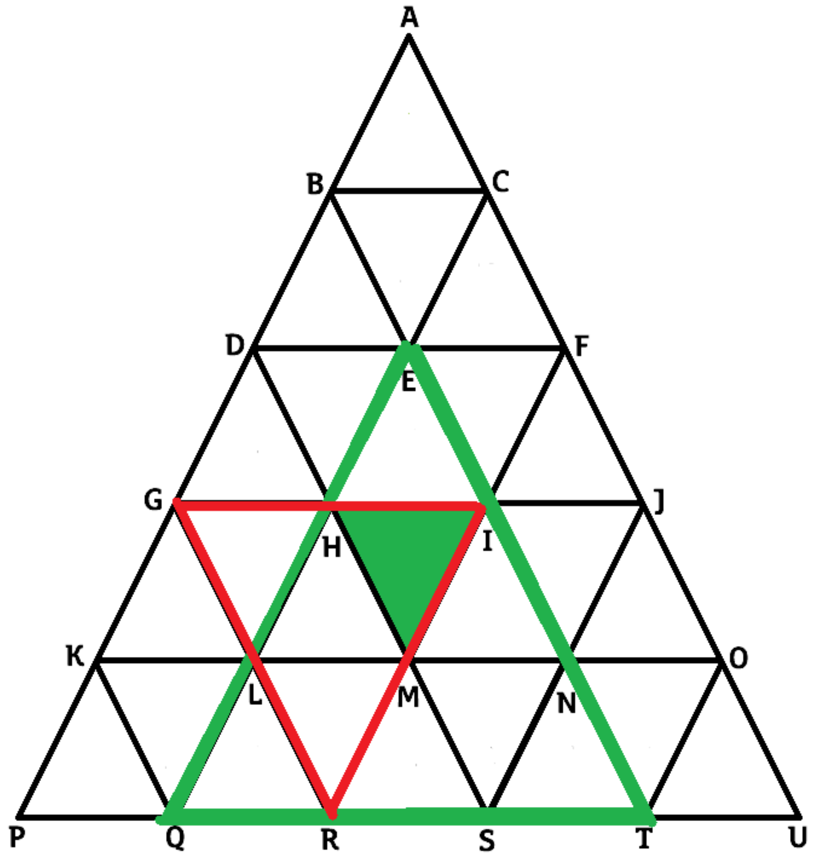
Дележ изумрудов (13)

- Стратегия всегда работает →
- Зеленые отрезки = разности в парах
- Отрезок [0; max] покрывает их все
- Его длина строго больше суммы, так как первый изумруд имеет ненулевой вес
- Можно было и не доказывать



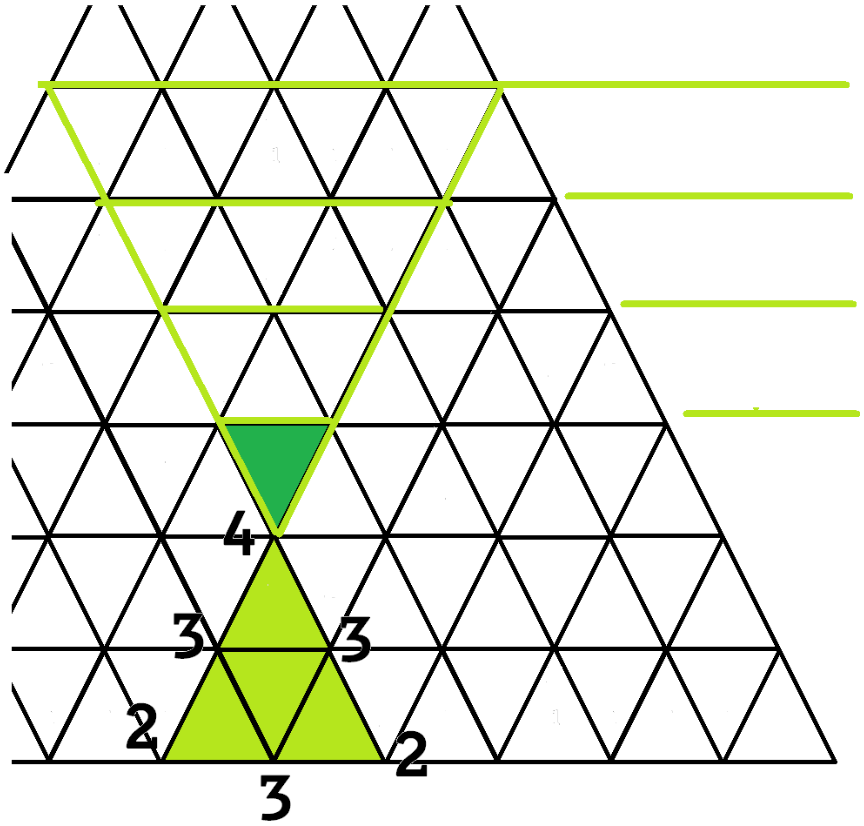
Изумрудная корона (1)

- Много тестов, размер до 10^6
  желательно отвечать за O(1)
- Треугольники 2 видов:
  вершиной вверх
  вершиной вниз
- Первый вид — легко:
  каждая сторона независима
  боковые $k = (N-h)/2 + 1$
  всего $h * k^2$



Изумрудная корона (1)

Второй вид:
переберем положение нижней вершины, для каждого её положения подсчитаем количество вариантов проведения верхней стороны →



Изумрудная корона (1)

Нужно быстро считать сумму в ромбе такого вида →

Но: в котором оставили только определенное число верхних строк (возможно, все)

```
    5
   4 4
  3 4 3
 2 3 3 2
1 2 3 2 1
 1 2 2 1
  1 2 1
   1 1
    1
```

ЦЕНТР ОЛИМПИАДНОГО ПРОГРАММИРОВАНИЯ ЮУРГУ

Изумрудная корона (1)

дополним до треугольника
найдем разность 2х
вычтем суммы в «уголках»

формулы выводятся
с использованием тождеств:

$\Sigma i = n(n+1)/2$

$\Sigma i^2 = n(n+1)(2n+1)/6$

1 2 3 4 5 4 3 2 1
1 2 3 4 4 3 2 1
1 2 3 4 3 2 1

---

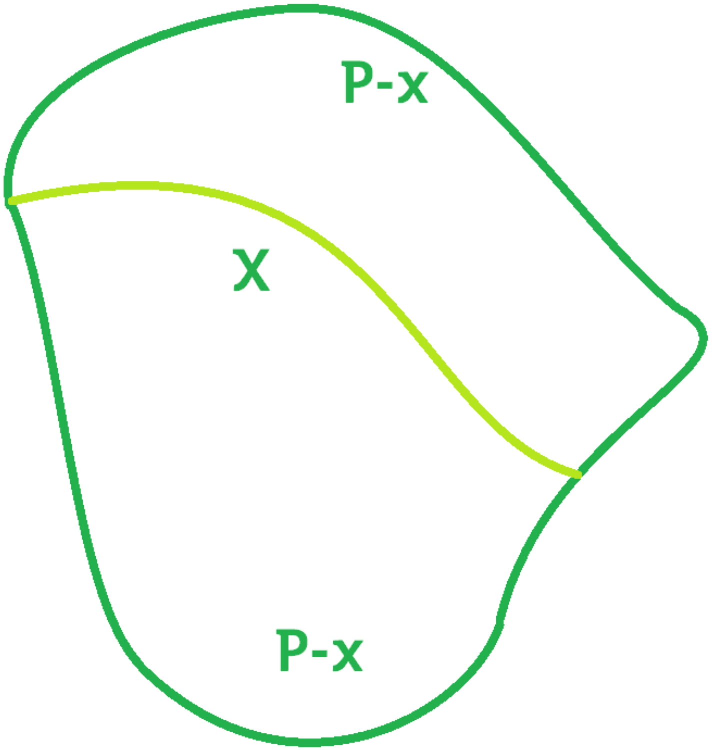
1 2 3 3 2 1
1 2 3 2 1
1 2 2 1
1 2 1
1 1
1

Сферы влияния (4)

- Всего 15 клеток → перебор за 2^{15}
- Самый строгий критерий - совпадение форм:
проверим 8 взаимных положений
(4 поворота * 2 варианта отражения)
каждый раз перенесем фигуру к началу
координат: x -= min(x), y -= min(y)
сравним наборы клеток как множества

Сферы влияния (4)

- Связность: dfs по клеткам одной из фигур
- Граница: можно вообще не проверять! →
- Общее время (грубая оценка) 2^k * 8 * k * log(k) при k=15, значение <10^7



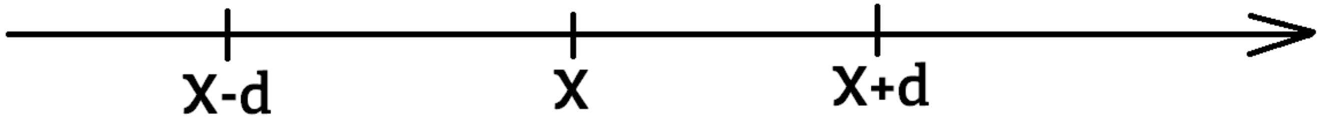
Поле Страшилы (9)

- Увеличим все C на 1 (получим длину периода)
- Длина периода = НОК периодов участков
- Запрос = НОК на прямоугольнике, без обновлений
- Авторское решение: 2d sparse table, N*M*logN*logM + Q
- Альтернатива: 2d дерево отрезков, Q*logN*logM
  явно не отсекалось, какие-то реализации могут проходить
  (но сложнее в реализации)
- Q=10^6, необходим эффективный I/O

ЦЕНТР
ОЛИМПИАДНОГО
ПРОГРАММИРОВАНИЯ
ЮУРГУ

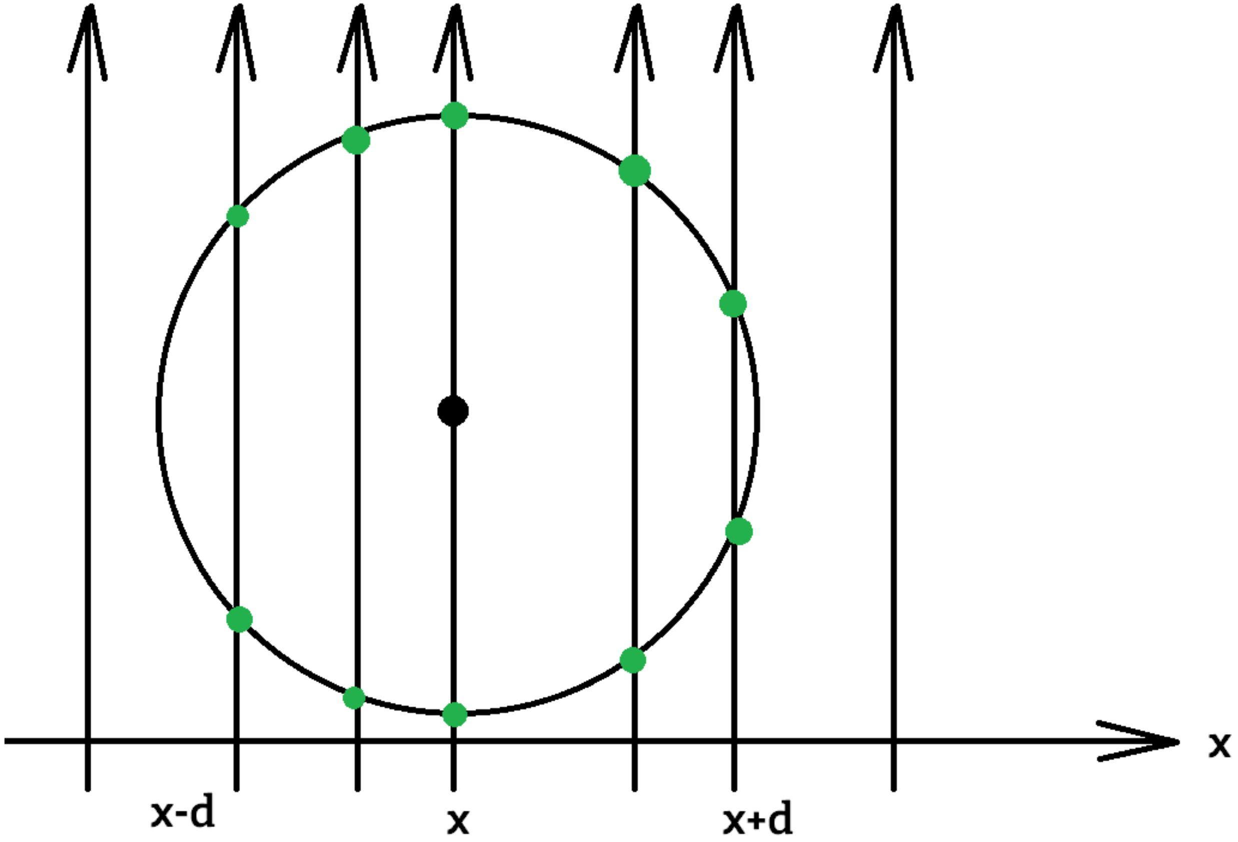
Летучие обезьяны (10)

- Аналогичная задача на прямой решалась бы методом 2 указателей → распространим на 2d



- Отсортируем все точки по возрастанию x, при равенстве x сгруппируем и отсортируем по y
- Массив массивов, пройдем по ним 2*(2d+1) указателями

Летучие обезьяны (10)



Летучие обезьяны (10)

- При переходе к следующей точке с тем же x: сместим все указатели вверх, скорректируем сумму в окрестности
- При переходе к точке с новым x: запустим указатели заново
- Общее время O(N*d), по каждому «вертикальному» вектору мы бежим указателями не более 2d+1 раз

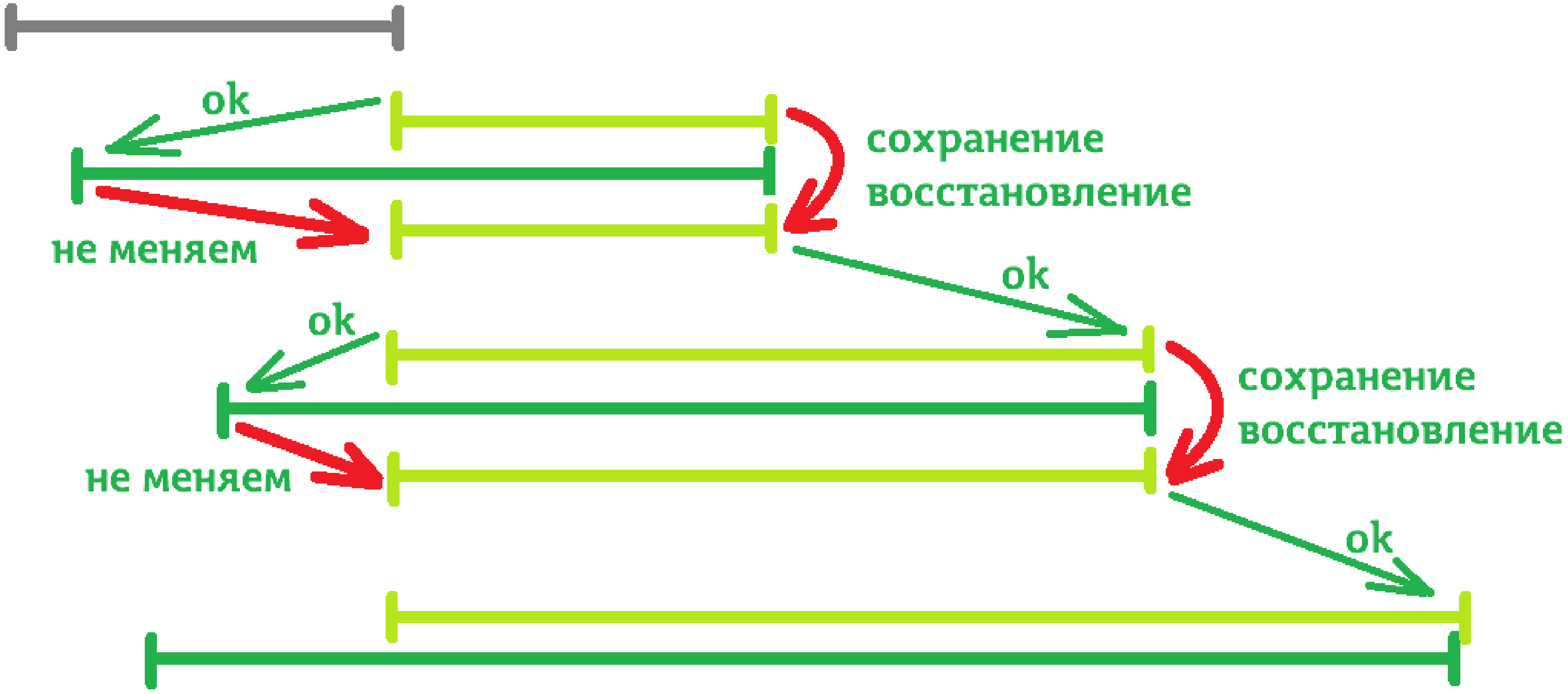
Разноцветные дуболомы (6)

- Очень похоже на стандартную задачу вычисления моды (самого частого значения) на отрезке
- Алгоритм Мо (корневая декомпозиция)
- Если поддерживать на отрезке набор чисел в нем и их частоты как словарь/множество с извлечением максимума и модификацией за log(N), то получим решение за N * sqrt(N) * log (N) (не должно проходить по TL)

Разноцветные дуболомы (6)

- Решение за N * sqrt(N) — Мо с откатами
- Легко обновлять максимум при «расширении» отрезка, но не при «сжатии»
- Для группы отрезков, начинающихся в блоке [L; R] будем поддерживать max на отрезках [R; Ri], каждый раз расширяя их влево до Li и откатывая максимум
- Короткие отрезки обработаем отдельно

Разноцветные дуболомы (6)

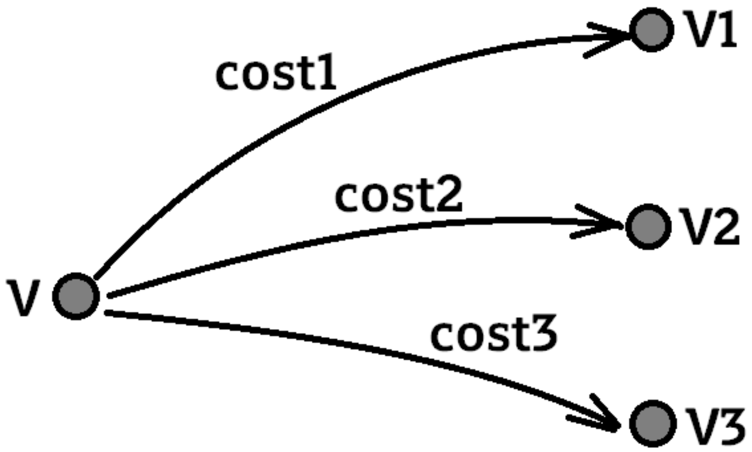


Зелье Гингемы (12)

- Условия задачи:
  ориентированный реберно-взвешенный граф
- Вершины = ингредиенты, эффекты = ребра
  нет петель и кратных ребер
- Найти путь (не обязательно простой) такой, что сумма ребер, по которым проходили хотя бы 1 раз, будет максимальна

Зелье Гингемы (12)

Если граф ациклический, то очевидная динамика



$$dp[v] = \max_i(dp[v_i] + cost_i)$$

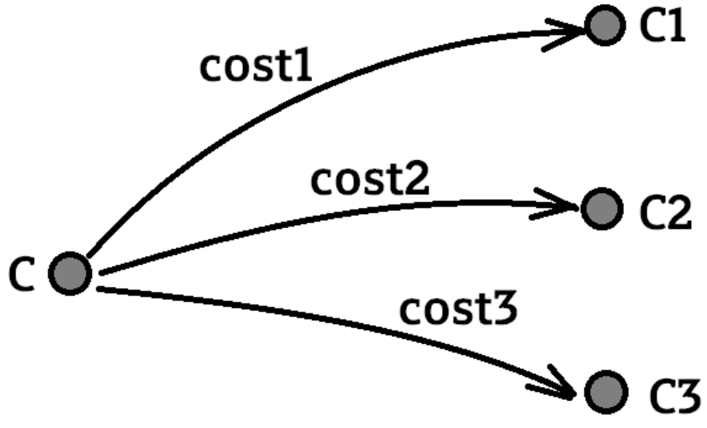
Зелье Гингемы (12)

- Найдем компоненты сильной связности (SCC)
- Граф компонент (aka конденсация) будет ациклическим, топологически отсортированным
- В пределах компоненты: можем зайти и выйти через любую вершину, в промежутке обойти все ребра
- Для компонент посчитаем сумму ребер внутри них

ЦЕНТР ОЛИМПИАДНОГО ПРОГРАММИРОВАНИЯ ЮУРГУ

Зелье Гингемы (12)

- SCC — O(M)
- Обход компонент — O(M)
- ДП по конденсации — O(M)
- Итого: O(M)



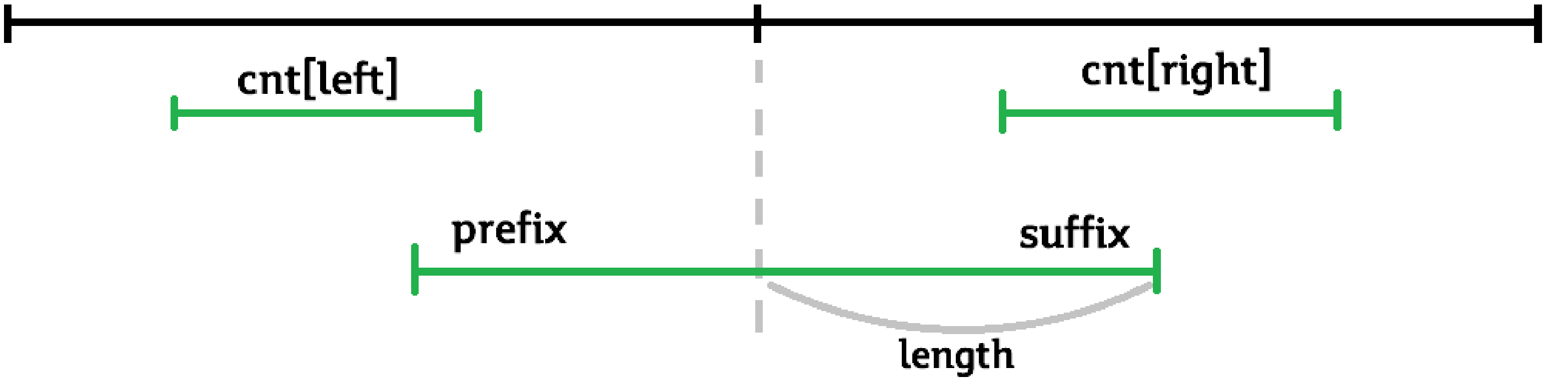
$$dp[C] = \max_i(dp[C_i] + cost_i) + \Sigma_C$$

Книга Виллины (8)

- Обновления и запросы → дерево отрезков
- Статистика на отрезке =
  количество его подотрезков, делящихся на k
- Массовое присваивание
- Как получать ответ на отрезке, имея данные по его частям?

ЦЕНТР
ОЛИМПИАДНОГО
ПРОГРАММИРОВАНИЯ
ЮУРГУ

Книга Виллины (8)



$$\underbrace{(\text{prefix} \% k)}_{R1} * \underbrace{10^{\text{length}}}_{L} + \underbrace{(\text{suffix} \% k)}_{R2}$$

$$R1*L + R2 = 0 \bmod k$$

$$R2 = -R1*L \bmod k$$

Книга Виллины (8)

pref[x] = количество префиксов отрезка, имеющих остаток x при делении на k

suff[x][t] = количество суффиксов отрезка, имеющих остаток x при делении на k
и такую длину len, что 10^len = t

rem = остаток от деления целого отрезка на k

cnt = ответ к задаче

Книга Виллины (8)

- Всего порядка k^2 значений для каждого отрезка
- Слияние отрезков также за k^2
- Удобно реализовать в виде класса/структуры «отрезок» и процедуры их слияния
- Храним по 1 экземпляру на каждый узел дерева
- Возвращаемое значение в рекурсивной процедуре спуска по дереву

Книга Виллины (8)

- Массовое присваивание — стандартным образом, отложенные операции
- Заранее построить структуры, соответствующие отрезкам вида CCCCCC… для всех цифр от 0 до 9 и всех длин вида 2^len, s = merge(s, s)
- Предварительный просчет за logN * 10, после этого присвоение узлу = использование готового значения
- Общее время работы NlogN * k^2

Ограда (14)

Две подзадачи:

1) как достичь максимальной площади, используя определенный набор отрезков?

2) какие из отрезков выбрать для использования, а какие выкинуть?

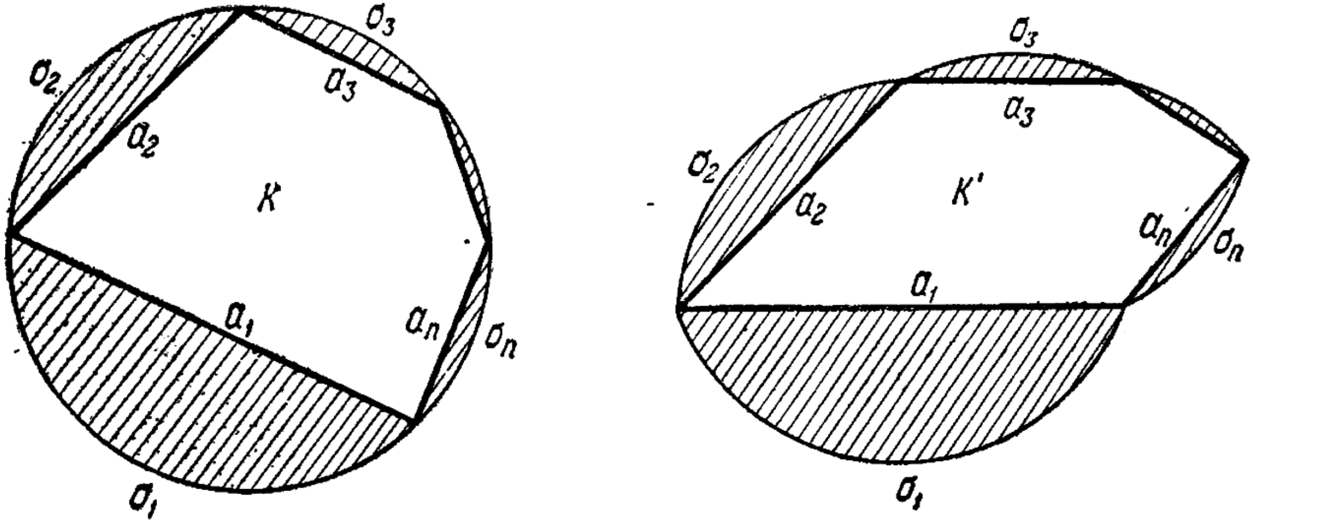
Ограда (14)

- Среди произвольных фигур при фиксированном периметре максимальной площадью обладает круг, «изопериметрическая задача» (Штейнер 1838, Гурвиц 1902, Шмидт 1938 и др.)
- Подзадача 1:
решение — вписанный многоугольник
(в некоторых источниках — «теорема Крамера»)

ЦЕНТР
ОЛИМПИАДНОГО
ПРОГРАММИРОВАНИЯ
ЮУРГУ

Ограда (14)

Теорема Крамера. *Из всех многоугольников с данными сторонами* $a_1, a_2, a_3, \ldots$ *наибольшую площадь имеет тот, около которого можно описать круг.*



Д.А.Крыжановский. Изопериметры. М. : Физматгиз, 1959. С. 52-53

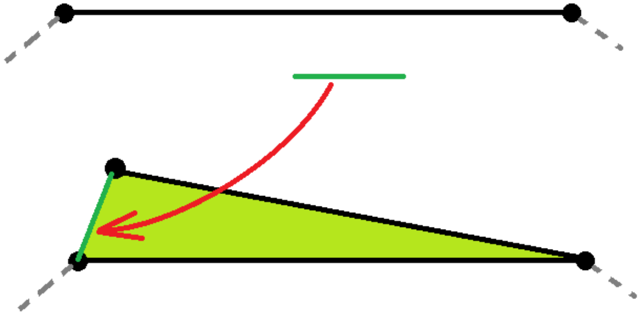
Ограда (14)

ЦЕНТР
ОЛИМПИАДНОГО
ПРОГРАММИРОВАНИЯ
ЮУРГУ

- Порядок сторон в многоугольнике не важен, важен только набор
- Подобрать радиус описанной окружности для фиксированного набора сторон можно двоичным поиском
- То есть, площадь находится за O(N*k)
  k — например, 60 для обычной точности

Ограда (14)

- Подзадача 2: набор используемых сторон — всегда какой-то префикс отсортированного по возрастанию набора
- Пусть это не так, и какая-то сторона пропущена; тогда возьмем любую сторону больше её, и заменим на пару сторон → площадь увеличится



Ограда (14)

- Перебор: выберем 3, 4, … N минимальных по длине отрезков, и попытаемся составить из них вписанный многоугольник
- Выберем максимальную среди всех вариантов площадь
- Время работы: $O(N^2 * k)$ — близко к TL, большая скрытая константа вычислений, тригонометрии и т.п.

Ограда (14)

- Отсечения: можно заметить, что почти всегда оптимальным является максимальный набор сторон, и выбрасывать лишние требуется ОЧЕНЬ редко
- Например, пойдем от максимальных наборов и прекратим перебор по какой-нибудь верхней оценке площади
- Также можно показать, что кандидатами на удаление являются только стороны, более чем в 2 раза превосходящие ближайших «соседей» по длине снизу, и перебирать нужно не более log(max(L)) вариантов

Спасибо за внимание!