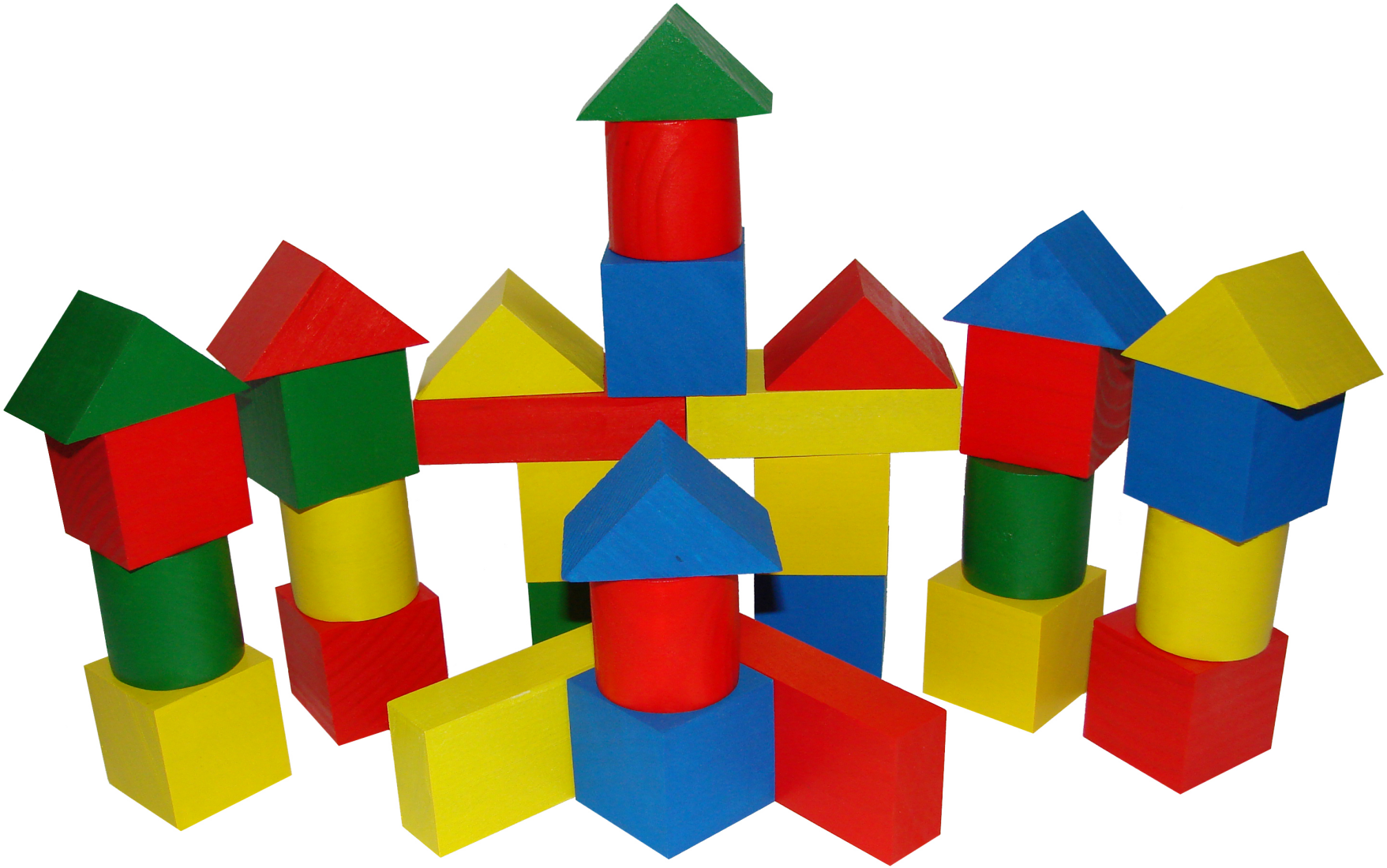
ИОИП 2015: Разбор задач

29 марта 2015 года

Задача «Кубики»



Над задачей работали

- Идея задачи: Евгений Замятин
- Текст условия: Евгений Замятин
- Тесты, проверяющая программа и др.: Евгений Замятин
- Текст разбора: Евгений Замятин

Постановка задачи

- Дано n башенок из кубиков, размер каждой a_i
- За одно действие можно переложить кубик с одной башенки на другую, убрать кубик с какой-то башенки или добавить из набора
- Необходимо посчитать минимальное число действий, после выполнения которых все башенки будут одинаковой высоты

100 баллов

- sum — сумма всех a_i
- Минимальный ответ достигается при высоте всех башенок равной $\lfloor \frac{sum}{n} \rfloor$ или $\lceil \frac{sum}{n} \rceil$
- Можно посчитать ответ для обоих случаев и выбрать минимальный

100 баллов

- Как считать ответ?
- Пусть мы хотим узнать сколько операций понадобится, чтобы высота каждой башенки была M.
- Тогда посчитаем $Great = \sum_{a_i > M}(a_i - M)$ — суммарное количество кубиков, которые нужно убрать и $Less = \sum_{a_i < M}(a_i - M)$ — суммарное количество кубиков, которые нужно добавить.
- Заметим, что, по возможности, выгодно перекладывать, так как эта операция равнаценна тому, что мы один кубик уберем, а другой добавим.
- Тогда ответ это $\max(Great, Less)$, так как все, что сможем мы переложим (Перекладываний будет $\min(Great, Less)$), а остальное либо уберем, либо добавим в зависимости от ситуации.

Асимптотика

- Итоговая асимптотика — O(n)

Вопросы

- Вопросы?

Задача «Очередь в банк»

Над задачей работали

- Идея задачи: Дмитрий Филиппов
- Текст условия: Дмитрий Филиппов
- Тесты, проверяющая программа и др.: Дмитрий Филиппов
- Текст разбора: Дмитрий Филиппов

Постановка задачи

- Дана очередь из людей, у каждого человека какое-то настроение
- Настроение считается хорошим, если оно не меньше x
- В очередь приходят и уходят люди
- Надо научиться отвечать на запрос «Сколько человек с хорошим настроением стоит в очереди перед данным»

40 баллов

- Будем хранить очередь в массиве
- Добавление человека в конец очереди и удаление из начала — $O(1)$.
- Ответ на запрос — посмотреть на всех людей в очереди перед данным и посчитать, сколько из них в хорошим настроением
- Ответ на запрос в худшем случае за $O(n)$.

100 баллов

- Поддерживаем префиксные суммы массива, где в i-ом элементе записано 1, если у i-го человека хорошее настроение, и 0, если плохое
- $prefixSum[maxIndex]$ — сколько человек с хорошим настроением стоит перед человеком с номером $maxIndex$
- Поддерживаем человека, который стоит первым в очереди

100 баллов

- Добавление в конец очереди — подсчет новой префиксной суммы
- Удаление из начала очереди — увеличение номера первого человека в очереди на 1
- Запрос — разность двух префиксных сумм:

$$answer = prefixSum[queryId] - prefixSum[firstManIndex]$$

Асимптотика

- Все операции выполняются за $O(1)$
- Итоговая асимптотика — $O(m)$

Вопросы

- Вопросы?

Задача «Выборы президента»

Над задачей работали

- Идея задачи: Илья Збань, Илья Пересадин
- Текст условия: Илья Збань
- Тесты, проверяющая программа и др.: Илья Збань
- Текст разбора: Илья Збань

Постановка задачи

- В задаче дано, сколько голосов получил каждый из n политиков, и известно, что политики из одной партии не могут голосовать друг за друга
- Нужно найти любое разбиение на партии, удовлетворяющее условию

20 баллов

- Пусть зафиксировано множество человек в одной партии
- Заметим, что разбиение корректно, если $\sum(a_i + 1) = n$, где a_i — число голосов членов выбранной партии
- 20 баллов: перебрать все подмножества, относящиеся к одной партии

60 баллов

- 60 баллов: ДП за куб или за квадрат
- Например, $dp_{i,j}$ — можно ли рассмотрев первые i (в любом порядке) человек, набрать в одной партии $\sum(a_i + 1) = j$ голосов
- Состояний — n^2, переходов — $O(1)$

100 баллов

- Жадный алгоритм
- Отсортируем все элементы по убыванию
- Идем слева направо, пытаемся добавить новый элемент в множество. Если сумма не превышает n — добавляем, иначе — нет
- Если в конце не набрали сумму n, ответ — NO
- Время работы — $O(n \ log \ n)$

Доказательство

- Ответ NO бывает только тогда, когда исходный массив имеет нечетную длину и состоит из единиц — очевидно, разбиения не существует
- Иначе всегда можно построить ответ конструктивно. Пусть $b_i = a_i + 1$. Заметим, что каждому числу b_i, большему единицы, можно сопоставить $b_i - 2$ единиц, и так как $\sum a_i = n$, все единицы будут использованы
- Задача — выбрать некоторые b_j так, чтобы $\sum b_j = n$

Доказательство

- Заметим, что на первом шаге возьмем максимум, и запомним, что ему соответствовала хотя бы одна единица ($b_1 > 2$ по предположению). Предположим, что в какой-то момент жадный алгоритм не смог добавить число x, т.е. $sum + x > n$, где sum — текущая сумма. Тогда заметим, что в массиве есть хотя бы $x - 2$ единиц, соответствующих этому числу, и еще одна единица от максимума. То есть, алгоритм в любом случае сможет набрать сумму n единицами, так как $sum + x - 1 >= n$

100 баллов

- Другие решения:
- Рюкзак за $O(n\sqrt{n})$
- Битовые оптимизации предыдущей динамики за $n^2/64$

Вопросы

- Вопросы?

Задача «Пирожные»

Над задачей работали

- Идея задачи: Илья Пересадин
- Текст условия: Захар Войт
- Тесты, проверяющая программа и др.: Захар Войт
- Текст разбора: Захар Войт

Постановка задачи

- Дано n пирожных, для каждого известна координата x_i и время t_i, за которое его можно съесть.
- Начиная из координаты 0, нужно съесть как можно больше пирожных за время T.

Решение на 15 баллов ($n \leq 20$)

- Фиксируем подмножество пирожных S.
- Пирожные оптимально есть в порядке неубывания x_i.
- Минимальное время за которое можно съесть подмножество S:

$$t = \max_{j \in S}(x_j) + \sum_{j \in S} t_j$$

- Ответом будет минимальное t среди всех S.
- Сложность решения $O(2^n \cdot n)$
- Это решение проходит только первую подгруппу.

Решение на 20 баллов ($n \leq 1\,000$, $T \leq 1\,000$)

- Отсортируем пирожные по x_i.
- Будем считать ДП $d[i, t]$ — какое максимальное количество пирожных среди пирожных от первого до i-го можно съесть за время t.
- База: $d[0, 0] = 0$
- Переход:

$$d[i, t] = \max(d[i-1, t-\Delta x], d[i-1, t-\Delta x - t_{i-1}] + 1)$$

где $\Delta x = x_i - x_{i-1}$

- Сложность решения $O(n \cdot T)$
- Это решение проходит только вторую подгруппу (но не первую)

Решение на 60 баллов ($n \leq 1\,000$, $T \leq 10^9$)

- Будем считать ДП $d[i, k]$ — за какое минимальное время можно съесть k пирожных из пирожных до i-го.
- База: $d[0, 0] = 0$
- Переход:

$$d[i, k] = \max(d[i-1, k] + \Delta x, d[i-1, k-1] + \Delta x + t_{i-1})$$

где $\Delta x = x_i - x_{i-1}$

- Ответ на задачу — максимальное k, такое, что $\exists i : d[i, k] \leq T$.
- Сложность решения $O(n^2)$
- Это решение проходит первую, вторую и третью подгруппы.

Решение на 100 баллов ($n \leq 100\,000$, $T \leq 10^9$)

- Предположим, что мы съедим i-е пирожное и оно будет последним (с максимальной координатой).
- Пусть S — множество съеденных нами пирожных, тогда время, за которое мы их съедим равно:

$$t = \sum_{j \in S} t_j + x_i$$

- Так как x_i мы фиксировали, нужно выбрать в качестве S максимальное по включению множество с суммой меньшей либо равной $T - x_i$.
- Очевидно, оптимально взять как можно больше пирожных с минимальными t_i.

Решение на 100 баллов ($n \leq 100\,000$, $T \leq 10^9$)

- Будем бежать по пирожным по неубыванию x_i и поддерживать множество S.
- Пусть S — максимальное множество с предыдущего шага $(\forall j \in S : x_j \leq x_{i-1})$.
- Тогда будем убирать элемент с максимальным t_i из S, пока $\sum\limits_{j \in S} t_j + x_i > T$.
- Добавим i в множество S, если $\sum\limits_{j \in S} t_j + t_i + x_i$.
- Полученное множество будет являться максимальным на текущем шаге.

Решение на 100 баллов ($n \leq 100\,000$, $T \leq 10^9$)

- Множество S можно поддерживать при помощи двоичной кучи или сбалансированного дерева поиска (например std::set/TreeSet)
- Сложность решения $O(n \log n)$

Вопросы

- Вопросы?

Задача «Счета дядюшки Скруджа»

Над задачей работали

- Идея задачи: Илья Пересадин
- Текст условия: Илья Пересадин
- Тесты, проверяющая программа и др.: Илья Пересадин
- Текст разбора: Илья Пересадин

Постановка задачи

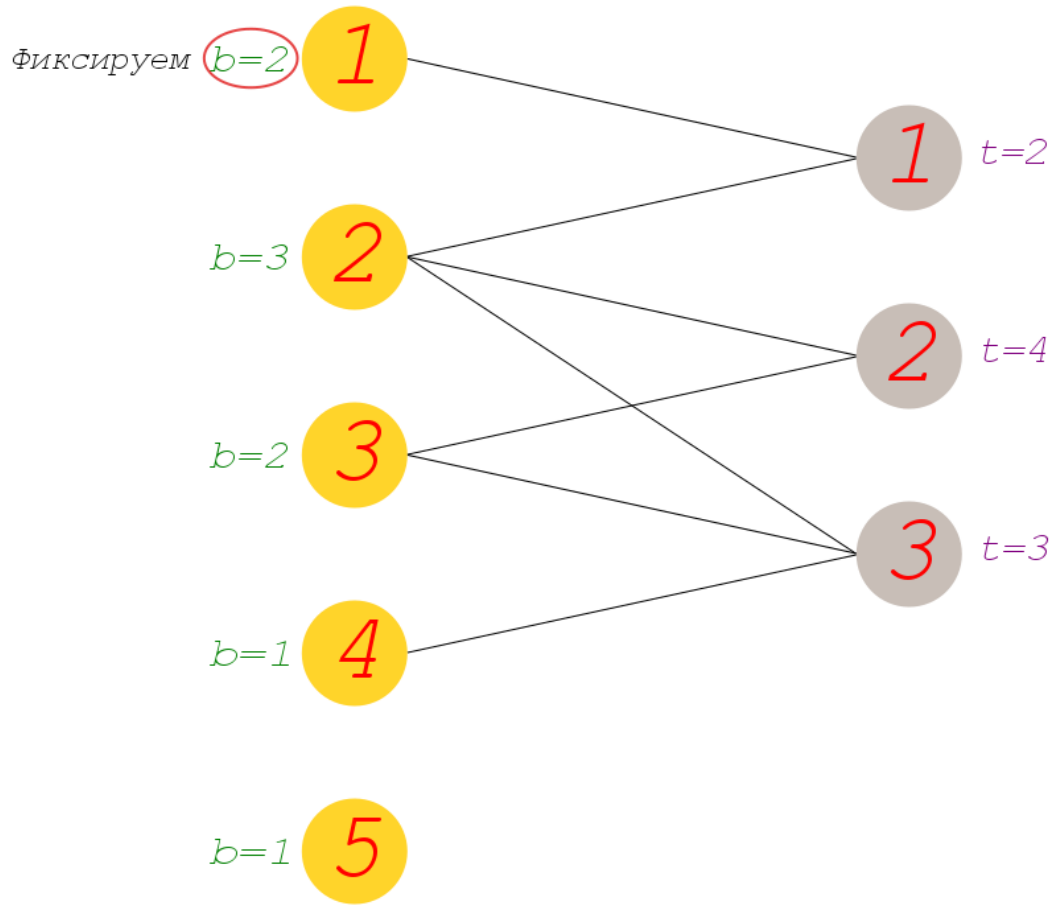
- Дано n банковских счетов, на каждый ежедневно приходит b_i долларов
- Значения b_i неизвестны
- Дано m подсказок: $(c_1, x_1), \ldots, (c_k, x_k)$, где x_j это сумма на c_j счету в некоторый день t. Значение t неизвестно. Для каждой подсказки свое значение t
- Требуется восстановить b_i

Решение на 50 баллов ($n \leq 10^5, MaxVal \leq 10^5$)

- Сохраним для каждого счета значение суммы на этом счету в какой-то из дней. Обозначим эту величину за d_i
- Возможное значение b_i является делителем d_i
- Зафиксируем какой-то делитель d_i
- Проверим, что этот делитель подходит в качестве b_i
- Построим двудольный граф, левая доля — счета, правая — подсказки. На ребре между вершинами — значение суммы на счету в подсказке

Решение на 50 баллов ($n \leq 10^5$, $MaxVal \leq 10^5$)

Пример графа




```
Тест
5 3
2
1 4 2 6
2
2 12 3 8
3
2 9 3 6 4 3
```

Решение на 50 баллов ($n \leq 10^5, MaxVal \leq 10^5$)

- Каждая вершина левой доли характеризуется величиной b_i, каждая вершина правой доли величиной t_i. Эти значения будем восстанавливать обходом графа в глубину
- Поддерживаем следующий инвариант: при посещении вершины левой доли известно b_i для нее, правой доли — t_i
- Для каждой вершины обхода в глубину находим значения b_i (либо t_i) для смежных с ней вершин

Решение на 50 баллов ($n \leq 10^5, MaxVal \leq 10^5$)

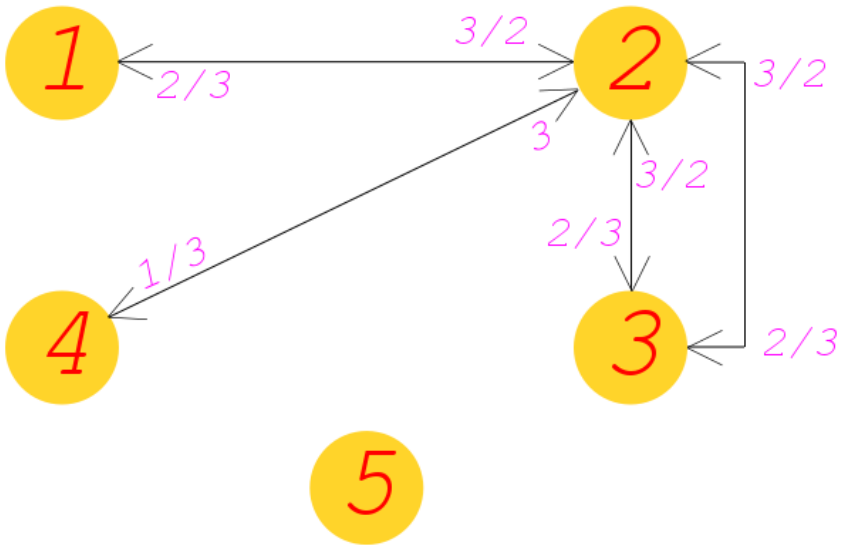
- Если в какой-то из смежных вершин уже посчитанное ранее значение не совпало с вычисленным на данный момент — делитель d_i не подходит в качестве b_i
- В графе могут быть несколько компонент связанности. Для каждой из компонент независимо восстановим значения b_i.
- Асимптотика этого решения $O(n \cdot D)$, где D — количество делителей. Количество делителей у чисел до 10^5 не превосходит 200.

Решение на 100 баллов ($n \leq 10^5, MaxVal \leq 10^{18}$)

- Обратим внимание на следующий факт — если для какого-то счета в подсказке мы знаем значение b_i, то значения остальных b_j в этой подсказке равны $\frac{x_j}{x_i} b_i$
- Построим граф. Вершины графа - это банковские счета. Ребра проведем между вершинами каждой подсказки: от первой вершины во всех остальных и от всех вершин в первую. Этих ребер будет достаточно, потому что внутри каждой подсказки нас волнует лишь связность. Весом ребра (u, v) будет сокращенная дробь $\frac{x_v}{x_u}$
- Обозначим некоторую вершину за стартовую и от нее запустим обход в глубину

Решение на 100 баллов ($n \leq 10^5, MaxVal \leq 10^{18}$)

Пример графа




```
Тест
5 3
2
1 4 2 6
2
2 12 3 8
3
2 9 3 6 4 3
```

Решение на 100 баллов ($n \leq 10^5, MaxVal \leq 10^{18}$)

- По ходу выполнения обхода в глубину считаем произведение дробей от стартовой вершины до текущей
- После выполения обхода в глубину для каждой вершины известны произведения $\frac{p_v}{q_v}$
- Пусть b_i стартовой вершины равно C. C должно быть таким, что $\frac{p_v}{q_v} C$ целое число для каждой вершины

Решение на 100 баллов ($n \leq 10^5, MaxVal \leq 10^{18}$)

- Этому требованию удовлетворяет $C = lcm(q_1, q_2, \ldots, q_k)$, где q_v — это знаменатели вершин
- Может быть несколько компонент связанности. Вычисляем значение всех b_v, независимо в каждой компоненте связанности
- После вычисления всех b_v, проверяем, что они походят под все подсказки. Если да — ответ *YES*, иначе *NO*

Технические детали реализации

- Произведение дробей нужно находить аккуратно: сначала делить числители и знаменатели на gcd, после чего перемножать дроби
- Если для какой-то вершины v числитель или знаменатель дроби стал больше 10^{18}, то решения нет, потому что в этом случае значение b_v будет больше 10^{18}
- Чтобы избежать переполнения 64 битного типа, произведение можно вычислить в вещественном типе и сравнить его с 10^{18}

Асимптотика

- Обход графа происходит за $O(n + \sum k_i)$, где $\sum k_i$ суммарная длина подсказок
- Вычисление произведения двух дробей выполняется за сложность вычисления gcd. Его можно произвести за $\log(min(a, b))$, где a и b не превосходят $MaxVal$
- Итоговая сложность $O(nlog(MaxVal) + \sum k_i)$

Вопросы

- Вопросы?

Спасибо за внимание!

- Спасибо за внимание!
- Вопросы?