Руководитель олимпиады: Шульгина Галина Михайловна

Председатель жюри: Базилевич Григорий
Состав жюри: Кориненко Матвей
Коробко Семен
Пупков Кирилл
Девятериков Иван

Спонсоры олимпиады: XSOLLA
PARMA Technologies Group
Knoema

Отдельная благодарность тем, кто протестировал задачи: Поспелов Сергей
Субботин Игорь
Перескокова Марина

Задача A. Витя и велосипед

Автор и разработчик: Матвей Кориненко

Для начала необходимо перевести все величины в одни и те же единицы измерения путем простейших математических операция. R_1, R_2, R_3, s — в метры, а t — в секунды.

Заметим, что подходящее нам время движения на велосипеде и есть t, тогда средняя скорость движения велосипеда будет равна $v = \frac{s}{t}$ $^{\text{М}}/_{\text{С}}$. Это же и будет линейной скоростью вращения колес велосипеда, т.к. они не проскальзывают по земле. Тогда угловая скорость вращения заднего колеса и шестерни — $\omega = \frac{v}{R_3}$. Линейная скорость движения цепи велосипеда — $v = \omega \times R_2$, к тому же, поскольку цепь не скользит по шестерням и не растягивается, то эта же скорость цепи будет и около первой шестерни. Тогда угловая скорость ее вращения (вращения педалей, ответ на задачу) будет равна $\frac{v}{R_1}$.

Задача B. Фокус с зеркалом

Автор и разработчик: Кирилл Пупков

Назовем множество позиций кубиков правильным, если существует точка, такая, что относительно нее это множество симметрично (то есть для него ответ это 0). Ясно, что если размер наибольшего правильного множества равен M, то ответ на задачу $N - M$. Действительно, для правильного множества кубиков добавлять не придется, а для всех остальных придется добавить ровно по одному симметричному.

Наибольшее правильное множество можно найти за $O(2^n \cdot n)$ перебором или за $O(n^3)$ методом двух указателей. Но квадратичный алгоритм проще: достаточно просто найти точку, такую, что она является серединой наибольшего количества отрезков. Тогда она будет центром искомого множества. Это можно легко сделать, используя std::map, std::unordered_map или массив. В зависимости от выбранной структуры решение набирает от 50 до 100 баллов. При этом нужно не забывать учитывать, что центр не может находиться внутри кубика.

Задача C. Генеральная репетиция

Автор и разработчик: Григорий Базилевич

Для решения первой и второй подзадач можно перебрать все возможные подмножества человек средней группы и проверить их средние погрешности по танцу. Асимптотика: $O(2^k \times n \times k)$.

Для дальнейшего решения нам необходимо доказать пару лемм. Сначала заметим, что для любого $\forall i \; \forall j \quad a_{ij} \geqslant b_j$.

Лемма 1 Погрешность удара k человек можно заменить средним арифметическим погрешностей ударов каждого человека.

$$\left|\frac{a_1+a_2+...+a_k}{k}-b\right| = \left|\frac{(b-b+a_1)+...+(b-b+a_k)}{k}-b\right| =$$
$$= \left|\frac{b\times k+(a_1-b)+...+(a_k-b))}{k}-b\right| = \left|b+\frac{(a_1-b)+...+(a_k-b)}{k}-b\right| =$$
$$= \left|\frac{(a_1-b)+...+(a_k-b)}{k}\right| = \frac{|(a_1-b)+...+(a_k-b)|}{k} = \frac{(a_1-b)+...+(a_k-b)}{k}$$

Лемма 1 доказана.

Лемма 2 Средняя погрешность танца k человек равна среднему арифметическому средних погрешностей танца каждого человека.

$$\frac{\left|\frac{a_{11}+...+a_{k1}}{k}-b_1\right|+...+\left|\frac{a_{1n}+...+a_{kn}}{k}-b_n\right|}{n} =$$
$$= \frac{\left(\frac{(a_{11}-b_1)+...+(a_{k1}-b_1)}{k}\right)+...+\left(\frac{(a_{1n}-b_n)+...+(a_{kn}-b_n)}{k}\right)}{n} =$$
$$= \frac{((a_{11}-b_1)+...+(a_{k1}-b_1))+...+((a_{1n}-b_n)+...+(a_{kn}-b_n))}{kn} =$$
$$= \frac{((a_{11}-b_1)+...+(a_{1n}-b_n))+...+((a_{k1}-b_1)+...+(a_{kn}-b_n))}{kn} =$$
$$= \frac{\left(\frac{(a_{11}-b_1)+...+(a_{1n}-b_n)}{n}\right)+...+\left(\frac{(a_{k1}-b_1)+...+(a_{kn}-b_n)}{n}\right)}{k}$$

Лемма 2 доказана.

Для решения третьей подзадачи необходимо было заметить, что если поставить в танец человека, у которого погрешность по танцу будет $\geqslant 0$, то мы уже не сможем получить среднюю погрешность танца среди всех человек $\leqslant d = 0$. Значит необходимо вывести только тех, у кого танец исполнен идеально.

Для решения четвертой и пятой подзадач воспользуемся жадными идеями — будем добавлять людей, пока средняя погрешность танца $\leqslant d$. Для этого в четвертой подзадаче будем на i-м шаге выбирать человека для добавления в танец за линейное время, а в пятой подзадаче отсортируем массив и пройдем по нему. Итоговая асимптотика: $O(k^2)$ или $O(k \log k)$ в зависимости от реализации.

Задача D. Ваня и быстрая игра

Автор: Иван Девятериков
Разработчик: Семён Коробко

Во-первых заметим, что $a_i >$ n мы точно никогда не используем. Действительно, так как при решении для какого-то i мы используем числа, которые меньше данного i, то мы будем использовать только числа $a_i \leqslant i \leqslant n$.

Выкинем все $a_i > n$. Пусть m теперь — это количество оставшихся чисел.

Давайте создадим массив $left$ длины n+1, где предпосчитаем ближайшее слева a_i для каждого i.

Решение №1 Давайте сразу искать ответ для текущего i.

Давайте, пока $teck > 0$ и $left[i] > 0$ делать следующую вещь.

Заметим, что из текущего i мы будем вычитать $left[i] - min(teck, \frac{i}{left[i]})$ ходов. Если $teck < \frac{i}{left[i]}$, то присвоим $i = i - teck \times left[i]$ — это i и будет ответом. А иначе, присвоим из $teck = teck - \frac{i}{left[i]}$ и i = i mod left[i].

Заметим, что мы сделаем не больше log k операций, потому что каждое «иначе» уменьшает i хотя бы в два раза, таким образом не более чем через log k ходов left[i] будет равен 0.

Данный алгоритм работает за $\mathcal{O}(n \log k)$ по времени и $\mathcal{O}(n)$ по памяти.

Решение №2 Представим переходы $X \to X_{\texttt{новый}}$ в виде ориентированного графа. Заметим, что граф у нас ациклический (причем из каждой вершины выходит ровно одно ребро).

Значит наша задача сводится к тому, чтобы для каждой вершины найти потомка на расстоянии k. Давайте инвертируем все ребра и теперь нам нужно найти предка на расстоянии k.

Для этого можно воспользоваться техникой «двоичные спуски/подъёмы». Тогда решение будет работать за $\mathcal{O}(n \log n)$. Подробнее про эту технику можно почитать здесь: `https://neerc.ifmo.ru/wiki/index.php?title=Метод_двоичного_подъёма`

Но так как k фиксировано, мы можем для каждой вершины посчитать предка на расстоянии k, просто запустив DFS и поддерживая текущий путь до вершины. Попадая в вершину, мы сохраним ее предка на нужном расстоянии. Такое решение работает за $\mathcal{O}(n)$.

Замечание Хоть асимптотически решение с помощью DFS и быстрее, но на практике первое решение работает либо также, либо даже лучше.

Задача E. История и XAX

Автор и разработчик: Григорий Базилевич

Для решения первой подзадачи можно было для каждого запроса проверить каждое из n событий. Асимптотика: $O(q \times n \times k)$.

Для решения второй подзадачи можно было для каждого запроса сохранить принадлежность к какому-то событию, а потом для каждого события и ключевого слова выбирать ближайшее. Асимптотика: $O(n^2 \times k)$.

Перейдем к полному решению задачи. Очевидно, что ответом является событие, находящееся ближе всего слева или справа к нашему текущему событию.

Подход 1

Можно было сохранить пары (ключевое слово; события, в которых встретилось). Это можно было сделать с помощью структуры map в C++ или dict в python. Для хранения событий можно было использовать set либо массив.

Научимся быстро находить ближайшее событие — с помощью find в set или бинарного поиска найдем позицию рассматриваемого события в массиве. Посмотрим на события, находящиеся слева и справа от события.

Асимптотика: $O(\sum c_i \log n)$ на сохранение данных, $O(\log n)$ на запрос. Итог — $O((\sum c_i + q) \log n) \simeq O((n + q) \log n)$.

Подход 2

Отсортируем все события по возрастанию, ключевые слова для каждого из событий по возрастанию. Отсортируем запросы по возрастанию сначала по t_x, потом по y.

Теперь можно решить задачу за один проход по всем событиям. Будем поддерживать для каждого ключевого слова пару (время последнего вхождения; событие последнего вхождения) (на префиксе). Тогда, когда нам встретится еще раз это ключевое слово, мы попытаемся улучшить ответ для предыдущего события (если надо) — как правое значение и запишем предыдущее событие как ответ для текущего значения. Заметим, что для этого нам необходимо пройти ключевое слово каждого события всего один разю

Асимптотика: $O(n \log n + \sum c_i \log \sum c_i + q \log q)$ на сохранение данных, $O(\sum c_i)$ на ответ на все запросы. Итог — $O((n+q) \log n + \sum c_i)$ (при учете что $n \simeq q \simeq \sum c_i$).

Задача F. Забывчивый Матвей

Авторы: Матвей Кориненко, Семен Коробко, Кирилл Пупков
Разработчик: Семён Коробко

Заметим, что всего возможных вариантов пароля может быть M^N (так как длина пароля — N символов, на месте каждого из которых может быть один из M символов множества A).

Но поскольку число комбинаций может не быть кратным числу K, а ответ все равно необходимо взять по модулю MOD, то это число мы можем домножить на K^{MOD-2} и итоговое также взять по модулю MOD. Это работает как следствие из малой теоремы Ферма ($a^{p-1} \equiv 1 \mod p$):

$$\frac{M^N}{K} = M^N \times \frac{1}{K} = M^N \times K^{-1} \equiv M^N \times K^{MOD-2} \equiv ans \mod MOD$$

Для полного решения этой задачи необходимо было воспользоваться быстрым возведением в степень, в процессе которого необходимо брать число по модулю MOD.

Задача G. Абсолютный призёр

Автор и разработчик: Семён Коробко

Докажем известное утверждение, что $\sum_{i=0}^{N} C_N^i = 2^N$.

Действительно, ведь C_N^i — это количество способов выбрать i предметов из N. Мы сначала из N выбираем 0 предметов, затем 1, затем 2, ..., затем $N-1$, затем N.

Таким образом мы посчитаем количество подмножеств множества размера N. А это именно 2^N.

Также можно провести аналогию с двоичной системой счисления. Например, при $N = 3$ представим, что у нас есть 3 предмета. Скажем, что если мы взяли предмет номер i, то соответсвующая цифра равна 1, иначе 0.

Количество способов взять 0 предметов при $N = 3$ равно 1. Вот сам этот способ: 000. Количество способов взять 1 предмет: 3. Способы: 100, 010, 001. Количество способов взять 2 предмета: 3. Способы: 110, 011, 101. Количество способов взять 3 предмета: 1. Способы: 111.

Заметим, что мы использовали все числа по основанию 2 от 000, до 111. Не трудно убедится, что их ровно $2^3 = 8$.

Всё ровно так же будет и в общем случае.

Таким образом, $\sum_{i=1}^{N} \sum_{j=l_i}^{r_i} \sum_{k=0}^{j} C_j^k = \sum_{i=1}^{N} \sum_{j=l_i}^{r_i} 2^j$.

Переформулируем задачу: у нас есть переменная $s = 0$. Нам поступает N запросов найти сумму всех степеней двойки от l до r.

Заметим, что эта задача практически идентична задаче, где у нас есть N запросов на добавление 1 на отрезки от l до r, только в самом конце нам нужно провести «каррирование», т.е. привести ответ в двоичный вид. Например, так:

```
const int base = 2;

void carry(int* a, int len_of_a) {
    int d = 0;
    for (int i = 0; i < len_of_a; i++) {
        a[i] += d;
        d = a[i] % base;
        a[i] /= base;
    }
}
```

Задачу с запросами добавления на отрезке очень просто решает дерево отрезков, но, вероятно, оно у вас не пройдёт на полный балл из-за больших ограничений.

Можно решить задачу с добавлениями при помощи сканирующей прямой за асимптотику $O(N \log N)$. Это проходит 4ую, возможно 5ую, подгруппы.

Решение на полный балл за $O(N)$: заведём массив ans, в нём добавим 1 для каждого l_i, вычтем 1 для каждого $r_i + 1$. Заведём переменную $tec = 0$, и для каждого i от 1 до максимального значения r присвоим:

```
tec = tec + ans_i;
ans_i = tec;
```

Затем проведём «каррирование».