Содержание

В некоторых задачах большой ввод и вывод. Пользуйтесь быстрым вводом-выводом.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу.

Задачи

Задача 9A. Мостостроение [0.2 sec, 256 mb]

Давным давно, в 2009-м году...

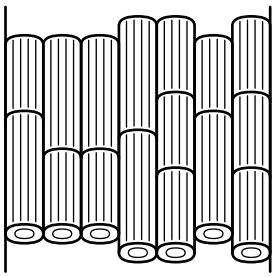
В деревне Зайкино регулярно идут проливные дожди, в результате чего речка Дубровка, которую обычно можно просто перешагнуть, выходит из берегов. Чтобы можно было перейти разлившуюся реку, планируется построить плавучий мост из брёвен, оставшихся от строительства бани бизнесмена, поселившегося неподалёку.

Все оставшиеся брёвна имеют одинаковую толщину. При этом есть x брёвен длины a и y брёвен длины b.

Построенный мост должен состоять из l рядов, каждый из которых составлен из одного или нескольких брёвен. Пилить брёвна нельзя, так как последняя пила утонула при разливе Дубровки.

Главный инженер хочет построить мост максимальной возможной ширины. Ширина моста определяется по минимальной ширине ряда брёвен в нём.

Например, если нужно построить мост из семи рядов, и при этом есть шесть брёвен длины 3 и десять брёвен длины 2, то можно построить мост ширины 5.



Формат входных данных

Ввод состоит из одного или нескольких тестовых случаев. Каждый тестовый случай состоит из пяти целых положительных чисел x, a, y, b и l. Каждое число не превосходит 500. Общее количество брёвен в каждом тестовом случае не меньше l.

Обозначим $d = \max(x, a, y, b, l)$. Гарантируется, что сумма d по всем тестам не превосходит 5000.

Формат выходных данных

Для каждого тестового случая на отдельной строке выведите одно число — максимальную возможную ширину моста.

Пример

stdin	stdout
6 3 10 2 7 10 7 20 9 25 106 126 135 28 137	5 9 112

Задача 9B. Две лесопилки [0.05 sec, 256 mb]

От вершины до подножья холма растет N старых деревьев. Районная администрация решила в санитарных целях срубить эти деревья, а чтобы снизить стоимость мероприятия перевезти все древесину на лесопилки. Деревья могут быть перевезены только в одном направлении – вниз. У подножья холма находится лесопилка, а также две дополнительные лесопилки могут быть построены на холме вдоль дороги. Вам предстоит определить, где наиболее выгодно построить эти лесопилки, чтобы минимизировать стоимость транспортировки древесины. Перевозка 1 килограмма древесины на 1 метр стоит 1 копейку.

Формат входных данных

Первая строка входного файла содержит натуральное число N – количество деревьев ($1 \leqslant N \leqslant 20\,000$). Деревья занумерованы от 1 до N начиная с вершины холма. Следующие N линий содержат по два целых числа w_i и d_i ($1 \leqslant w_i, d_i \leqslant 10\,000$) – вес дерева номер i и расстояние между деревьями i и i+1. Последнее из этих чисел (d_n) задает расстояние от нижнего дерева до лесопилки.

Формат выходных данных

Выведите единственное число – минимальную стоимость сплава деревьев вниз по реке.

Пример

stdin	stdout
9 1 2 2 1 3 3 1 1 3 2 1 6 2 1 1 2 1 1	26

Пояснение к примеру

В примере выгодно поставить лесопилки у деревьев с номерами 3 и 6.

Задача 9C. Order-Preserving Codes [1.5 sec, 256 mb]

Binary code is a mapping of characters of some alphabet to the set of finite length bit sequences. For example, standard ASCII code is a fixed length code, where each character is encoded using 8 bits.

Variable length codes are often used to compress texts taking into account the frequencies of occurence of different characters. Characters that occur more often get shorter codes, while characters occuring less often — longer ones.

To ensure unique decoding of variable length codes so called *prefix codes* are usually used. In a prefix code no code sequence is a proper prefix of another sequence. Prefix code can be easily decoded scanning the encoded sequence from left to right, since no code is the prefix of another, one always knows where the code for the current character ends and the new character starts.

Among prefix codes, the optimal code is known, so called Huffman code. It provides the shortest possible length of the text among all prefix codes that separatly encode each character with an integer number of bits.

However, as many other codes, Huffman code does not preserve character order. That is, Huffman codes for lexicographically ordered characters are not necessarily lexicographicaly ordered.

In this problem you are asked to develop a prefix code that would be optimal for the given text among all order-preserving prefix codes. Code is called order-preserving if for any two characters the code sequence for the character that goes earlier in the alphabet is lexicographically smaller.

Since text itself is not essential for finding the code, only the number of occurences of each character is important, only this data is given.

Формат входных данных

The first line of the input file contains n — the number of characters in the alphabet ($2 \leqslant n \leqslant 2000$). The next line contains n integer numbers — the number of occurences of the characters in the text for which the code must be developed (numbers are positive and do not exceed 10^9). Characters are described in the alphabetical order.

Формат выходных данных

Output n bit sequences, one on a line — the optimal order-preserving prefix code for the described text.

Пример

stdin	stdout
5 1 8 2 3 1	00 01 10 110 111

Задача 9D. Калила и Димна на лесозаготовках [0.2 sec, 256 mb]

Спонсор сегодняшней задачи — codeforces round 189. Codeforces — мечты сбываются!

Калила и Димна — два шакала. Они живут в огромных джунглях. Однажды шакалы решили устроиться на завод лесозаготовки и подработать.

Управляющий завода хочет, чтобы они отправились в джунгли и срубили n деревьев высотой $a_1, a_2, \ldots, a_n$. Для этого Калила и Димна купили цепную пилу в магазине. Каждый раз, когда они используют пилу на дереве номер i, они уменьшают высоту этого дерева на единицу. Каждый раз Калила и Димна должны заправить пилу для использования. Цена заправки зависит от того, какие деревья полностью спилены (дерево считается полностью спиленным, если его высота равна 0). Если максимальный идентификатор полностью срубленного дерева равняется i (первоначально это дерево имело высоту a_i), то цена заправки пилы равняется b_i. Если ни одно дерево не срублено полностью, то заправлять пилу запрещается. Изначально пила заправлена. Известно, что для каждого $i < j$, $a_i < a_j$ и $b_i > b_j$, а также $b_n = 0$ и $a_1 = 1$.

Калила и Димна хотят полностью срубить все деревья с минимальными затратами. Они ждут Вашей помощи! Поможете?

Формат входных данных

В первой строке записано целое число n ($1 \leqslant n \leqslant 10^5$). Во второй строке записано n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 10^9$). В третьей строке записано n целых чисел $b_1, b_2, \ldots, b_n$ ($0 \leqslant b_i \leqslant 10^9$).

Гарантируется, что $a_1 = 1$, $b_n = 0$, $a_1 < a_2 < \cdots < a_n$ и $b_1 > b_2 > \cdots > b_n$.

Формат выходных данных

В единственной строке должна быть записана минимальная стоимость вырубания всех деревьев.

Примеры

stdin	stdout
5 1 2 3 4 5 5 4 3 2 0	25
6 1 2 3 10 20 30 6 5 4 3 2 0	138

Задача 9E. Копилка [3.5 sec, 256 mb]

Дракон хранит свои сбережения в копилке. Каждый день Дракон или кладёт ровно один доллар в копилку, или берёт из копилки сколько-то денег. Дракону интересно, могут ли его сбережения стать значительно больше, если бы он вкладывал в копилку больше.

Говоря более точно, Дракон задаёт вам серию вопросов: "Если вкладывать в копилку дополнительных z долларов каждый день, каково максимальное количество денег в копилке в период с дня a по день b?"

Формат входных данных

Первая строка содержит числа n и m ($1 \leqslant n, m \leqslant 500\,000$) – число дней и количество вопросов. Следующая строка содержит n целых чисел $s_1, s_2, \ldots, s_n$ – количество денег в копилке после i-го дня ($0 \leqslant s_i \leqslant s_{i-1} + 1, s_0 = 0$). Следующие m строк содержат тройки чисел a_i, b_i, z_i: $1 \leqslant a_i \leqslant b_i \leqslant n$ и $1 \leqslant z_i \leqslant 10^9$. Учтите, что Дракон не властен над временем, поэтому каждый запрос нужно обрабатывать независимо от остальных, используя исходную последовательность s.

Формат выходных данных

Выведите m строк, содержащие ответы на вопросы.

Примеры

stdin	stdout
9 3 1 2 1 2 3 4 1 2 0 1 9 1 5 7 4 4 4 2	10 13 4

Гробы

Задача 9F. Свёртка [1.5 sec, 256 mb]

Рассмотрим все подмножества множества $U = \{0, 1, 2, \ldots, n-1\}$. Каждому подмножеству $A = \{a_1, a_2, \ldots, a_k\}$ соответствует уникальное целое число, равное $p(A) = \sum_{i=1}^{k} 2^{a_i}$. Функцию F от n-элементного множества будем задавать массивом целых чисел f длины 2^n так, что значение функции $F(A)$ равно $f[p(A)]$.

Вам даны две функции F и G, нужно найти функцию H такую, что

$$H(A) = \sum_{B \cup C = A} F(B)G(C).$$

Формат входных данных

В первой строке заданы два целых числа n и t ($1 \leqslant n \leqslant 16$, $1 \leqslant t \leqslant 100$). Здесь n — размер множества U, а t — количество тестовых случаев. Во второй строке заданы целые числа a и b, каждое от 1 до 10^9. Эти числа используются в следующем генераторе псевдослучайных чисел:

```
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand16() {
3.     cur = cur * a + b; // вычисляется по модулю 2^32
4.     return cur / 2^16; // целое число от 0 до 2^16 - 1
5. }
```

Тестовые случаи генерируются последовательно. В каждом из них сперва генерируются по порядку элементы массива f (значения функции F), а затем генерируются по порядку элементы массива g (значения функции G). Каждое следующее целое число генерируется вызовом функции `nextRand16()`.

Формат выходных данных

В ответ на каждый тестовый случай выведите в отдельной строке одно целое число: $\left(\sum_A H(A) \cdot (p(A) + 1)\right) \bmod 2^{32}$.

Примеры

stdin	stdout
3 2 30 239017	2723387430 3167905008
16 2 239 17	551267264 1632349120

Пояснения к примерам

Массивы в первом тесте из примера:
f_1: 3, 113, 3395, 36331, 41370, 61471, 9130, 11774
g_1: 25547, 45526, 55066, 13590, 14501, 41817, 9356, 18543
h_1: 76641, 8167827, 273846333, 5284992017, 1656829263, 11450721456, 3699971823, 14260048942
f_2: 32024, 43238, 51978, 52034, 53714, 38578, 43250, 52338
g_2: 62834, 50034, 59250, 8050, 44914, 36722, 53106, 20338
h_2: 2012196016, 6482475400, 8243104152, 15561662464, 7225902008, 16869349792, 22350138288, 44342816072