Задача 1. Слияние последовательностей

Источник:	базовая*
Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В первых четырёх байтах входного файла задано целое число N — количество чисел в первой последовательности, а в следующих четырёх байтах задано целое число M — количество чисел во второй последовательности. Далее идут N четырёхбатовых целых чисел первой последовательности, и затем M чисел второй последовательности. Все числа знаковые, каждая последовательность упорядочена по неубыванию. Длины последовательностей лежат в диапазоне: $1 \leqslant N, M \leqslant 10^6$.

Требуется реализовать функцию слияния двух отсортированных последовательностей с сигнатурой:

```
//merges sorted arrays a[0..ak-1] and b[0..bk-1] into
//one sorted array res[0..rk-1], returning rk from function
int merge(const int *a, int ak, const int *b, int bk, int *res);
```

и применить её к заданным в файле последовательностям.

Требуется вывести в выходной файл ровно $N + M$ четырёхбайтовых целых чисел: полученная в результате слияния упорядоченная последовательность.

Пример

input.bin
05 00 00 00 04 00 00 00 FC FF FF FF FD FF FF FF 01 00 00 00 01 00 00 00 0A 00 00 00 F9 FF FF FF 00 00 00 00 07 00 00 00 08 00 00 00
output.bin
F9 FF FF FF FC FF FF FF FD FF FF FF 00 00 00 00 01 00 00 00 01 00 00 00 07 00 00 00 08 00 00 00 0A 00 00 00

Задача 2. Разбиение массива

Источник:	базовая*
Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В первых четырёх байтах входного файла задано целое число N — количество чисел в массиве. В следующих четырёх байтах записано целое число p — пивот-элемент. Далее идут N четырёхбатовых целых чисел — содержимое массива. Все числа знаковые, длина последовательности лежит в диапазоне: $1 \leqslant N \leqslant 10^6$.

Требуется реализовать функцию разбиения массива относительно заданного пивот-элемента с сигнатурой:

```
//partitions array a[0..n-1] into two subarrays, returning value k
// the subarray a[0..k-1] must have all elements <= pivot
// the subarray a[k..n-1] must have all elements >= pivot
int partition(int *a, int n, int pivot);
```

и применить её к заданной в файле последовательности. Внутри функции разрешается использовать дополнительную память.

Важно: Заметим, что элементы, которые в точности равны `pivot`, можно помещать как в левую, так и в правую часть массива. В данной задаче требуется распределить эти элементы примерно поровну. Если в левую часть попадает u элементов, равных пивоту, а в правую часть — v элементов, то должно выполняться: $|u - v| \leqslant 1$.

В первые 4 байта выходного файла нужно вывести целое число `k` — сколько элементов попадает в левую часть массива. Далее нужно вывести N четырёхбайтовых целых чисел: содержимое массива `a` после выполнения функции `partition`.

Пример

input.bin
09 00 00 00 04 00 00 00 06 00 00 00 F8 FF FF FF 09 00 00 00 F8 FF FF FF FA FF FF FF 05 00 00 00 02 00 00 00 09 00 00 00 FF FF FF FF
output.bin
05 00 00 00 F8 FF FF FF F8 FF FF FF FA FF FF FF 02 00 00 00 FF FF FF FF 05 00 00 00 06 00 00 00 09 00 00 00 09 00 00 00

Задача 3. Сумма минимумов

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда∗
Ограничение по памяти:	разумное

Дан массив A, в котором содержится n целых чисел. Нужно перебрать все пары чисел A_i и A_j в этом массиве, для каждой пары найти минимум $\min(A_i, A_j)$ и сложить вместе все эти минимумы.

Более формально, требуется найти сумму:

$$S = \sum_{i<j} \min(A_i, A_j) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \min(A_i, A_j)$$

Формат входных данных

В первой строке записано целое число n — сколько чисел в массиве ($1 \leqslant N \leqslant 300\,000$). В остальных n строках записаны самы эти числа в том порядке, в котором они идут в массиве A. Все числа по модулю не превышают 10^9.

Формат выходных данных

Нужно вывести одно целое число — искомую сумму минимумов S.

Осторожно: сумма S может быть довольно большой. Оцените максимально возможное значение S и выберите подходящий целочисленный тип.

Пример

input.txt	output.txt
5 1 4 3 5 6	26

Комментарий

Подсказка: подумайте, как решить задачу за $O(N)$, если массив A упорядочен по возрастанию.

Задача 4. Сортировка слиянием

Источник:	основная*
Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В первых четырёх байтах входного файла задано целое число N — количество чисел в массиве A. Далее идут N четырёхбайтовых целых чисел — содержимое массива A. Размер массива лежит в диапазоне: $0 \leqslant N \leqslant 500\,000$.

Требуется отсортировать массив A по неубыванию, используя **алгоритм сортировки слиянием**.

В выходной файл нужно вывести ровно N четырёхбайтовых целых чисел: содержимое массива A после сортировки.

Пример

input.bin
0A 00 00 00 1F 00 00 00 F2 FF FF FF 06 00 00 00 04 00 00 00 26 00 00 00 FD FF FF FF 1E 00 00 00 F6 FF FF FF 0A 00 00 00 F4 FF FF FF
output.bin
F2 FF FF FF F4 FF FF FF F6 FF FF FF FD FF FF FF 04 00 00 00 06 00 00 00 0A 00 00 00 1E 00 00 00 1F 00 00 00 26 00 00 00

Задача 5. Быстрая сортировка

Источник:	основная*
Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В первых четырёх байтах входного файла задано целое число N — количество чисел в массиве A. Далее идут N четырёхбайтовых целых чисел — содержимое массива A. Размер массива лежит в диапазоне: $0 \leqslant N \leqslant 500\,000$.

Требуется отсортировать массив A по неубыванию, используя **алгоритм быстрой сортировки**.

В выходной файл нужно вывести ровно N четырёхбайтовых целых чисел: содержимое массива A после сортировки.

Пример

input.bin
0A 00 00 00 1F 00 00 00 F2 FF FF FF 06 00 00 00 04 00 00 00 26 00 00 00 FD FF FF FF 1E 00 00 00 F6 FF FF FF 0A 00 00 00 F4 FF FF FF
output.bin
F2 FF FF FF F4 FF FF FF F6 FF FF FF FD FF FF FF 04 00 00 00 06 00 00 00 0A 00 00 00 1E 00 00 00 1F 00 00 00 26 00 00 00

Задача 6. Поразрядная сортировка

Источник:	основная∗
Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда∗
Ограничение по памяти:	разумное

В первых четырёх байтах входного файла задано целое число N — количество элементов в массиве A. Далее записано N элементов, каждый элемент состоит из ключа и значения. Ключ и значение являются **беззнаковыми** 32-битными целыми числами. Размер массива лежит в диапазоне: $0 \leqslant N \leqslant 10\,000\,000$ (внимание: 10^7).

Требуется отсортировать массив A по неубыванию ключа, используя **алгоритм поразрядной сортировки** (radix sort). Чтобы решение работало быстро, рекомендуется разбивать 32-битный ключ на четыре цифры, по одному байту каждая цифра.

В выходной файл нужно вывести N элементов: содержимое массива A после сортировки. Так же как и во входных данных, каждый элемент должен состоять из ключа и значения.

Пример

`input.bin`

```
0A 00 00 00
1F 00 00 00 1E 00 00 00 0E 00 00 00 24 00 00 00
05 00 00 00 1B 00 00 00 10 00 00 00 06 00 00 00
22 00 00 00 11 00 00 00 0A 00 00 00 11 00 00 00
25 00 00 00 05 00 00 00 0E 00 00 00 1D 00 00 00
25 00 00 00 1D 00 00 00 0E 00 00 00 1B 00 00 00
```

`output.bin`

```
05 00 00 00 1B 00 00 00 0A 00 00 00 11 00 00 00
0E 00 00 00 24 00 00 00 0E 00 00 00 1D 00 00 00
0E 00 00 00 1B 00 00 00 10 00 00 00 06 00 00 00
1F 00 00 00 1E 00 00 00 22 00 00 00 11 00 00 00
25 00 00 00 05 00 00 00 25 00 00 00 1D 00 00 00
```

Задача 7. Удаление дубликатов

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда*
Ограничение по памяти:	разумное

Дан массив A, в котором содержится n целых чисел. Нужно удалить из него дубликаты (т.е. повторы чисел), так чтобы в массиве каждое имеющееся в нём значение встречалось ровно один раз.

Формат входных данных

В первой строке записано целое число n — сколько чисел в массиве ($1 \leqslant N \leqslant 300\,000$). В остальных n строках записаны самы эти числа. Все числа по модулю не превышают 10^9.

Формат выходных данных

В первой строке нужно вывести целое число k — сколько различных чисел в массиве A. В оставшихся k строках нужно вывести сами эти различные числа в любом порядке.

Пример

input.txt	output.txt
10 1 1 -2 4 3 0 0 0 0 -2	5 1 3 -2 0 4

Комментарий

Подсказка: подумайте, как решить задачу за $O(N)$, если массив A упорядочен по возрастанию.

Задача 8. Выпуклый минимум

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда*
Ограничение по памяти:	разумное

Массив чисел $A_0, A_1, A_2, \dots A_{n-1}$ называется выпуклым вверх, если:

$$\forall\, i < k < j: \qquad A_k < \frac{(j-k)A_i + (k-i)A_j}{(j-i)}$$

Дан выпуклый вверх массив A и коэффициент C. Требуется найти индекс элемента массива, на котором достигается минимум линейной функции:

$$\operatorname*{argmin}_{i=0}^{n-1} (A_i + C \cdot i) = ?$$

Если минимальное значение достигается на нескольких элементах массива, нужно найти номер первого такого элемента.

Формат входных данных

В первой строке записано одно целое число n — размер выпуклого массива ($1 \leqslant n \leqslant 10^5$). Далее записаны элементы массива A_i (n целых чисел, $|A_i| \leqslant 10^{15}$). Затем записано целое число q — количество запросов, которые нужно обработать ($1 \leqslant q \leqslant 10^5$). В остальных q строках записаны целые числа C_j, определяющие значения коэффициента линейной функции ($|C_j| \leqslant 10^9$).

Формат выходных данных

Требуется вывести q целых чисел: для каждого коэффициента C_j, записанного во входных данных, нужно вывести номер i первого элемента A_i, на котором достигается минимум $(A_i + C \cdot i)$ при $C = C_j$.

Пример

input.txt	output.txt
10 9 4 0 -2 -2 -1 1 4 8 20 8 -5 1 -2 0 6 3 2 4	8 3 5 3 0 2 2 1

Пояснение к примеру

Рассмотрим коэффициент $C_2 = -2$. Выпишем значение соответствующей функции для всех элементов:

i=0: $9 - 2 * 0 = 9$
i=1: $4 - 2 * 1 = 2$
i=2: $0 - 2 * 2 = -4$
i=3: $-2 - 2 * 3 = -8$
i=4: $-2 - 2 * 4 = -10$
i=5: $-1 - 2 * 5 = -11$
i=6: $1 - 2 * 6 = -11$
i=7: $4 - 2 * 7 = -10$
i=8: $8 - 2 * 8 = -8$
i=9: $20 - 2 * 9 = 2$

Минимум достигается на двух элементах $i = 5$ и $i = 6$, и ответом является меньший номер $i = 5$.

Комментарий

Представьте себе, как бы вы решали задачу, если бы вместо массива A была дана гладкая функция $A(x)$, и нужно было бы найти минимум функции $(A(x) + Cx)$. Задача с массивом решается точно так же, нужно лишь найти дискретный аналог для понятия производной.

Задача 9. Булевы формулы

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Задано две булевы формулы, первая является полиномом Жегалкина, а вторая представлена в конъюнктивной нормальной форме. Требуется определить, эквивалентны они или нет.

Формат входных данных

В первой строке записано целое число T — количество тестов в файле ($1 \leqslant T \leqslant 100$). Далее описывается T тестов, по две строки на тест. В первой строке теста записан полином Жегалкина, а во второй — формула в КНФ.

Каждая формула записывается без пробелов и состоит только из маленьких латинских букв от `a` до `j` (первые 10 букв) и следующих символов:

1. `'&'` (ASCII 38) — конъюнкция,
2. `'|'` (ASCII 124) — дизъюнкция (только в КНФ),
3. `'!'` (ASCII 33) — отрицание (только в КНФ),
4. `'+'` (ASCII 43) — сложение по модулю два (только в полиноме Жегалкина),
5. `'1'` (ASCII 49) — истина (только в полиноме Жегалкина).

Буквы обозначают различные булевы переменные.

Обе формулы синтаксически корректны с точки зрения алгебры логики. В формулах отсутствуют скобки, так что в КНФ надо считать, что дизъюнкция имеет больший приоритет, чем конъюнкция. В КНФ перед каждой переменной либо стоит один символ отрицания, либо отрицания нет (гарантируется, что кратных символов отрицания нет). Гарантируется, что константа истины `1` не умножается ни на какую переменную, а встречается в сумме только сама по себе.

Длина каждой формулы не превышает 5 000 символов.

Формат выходных данных

Требуется вывести T строк, в каждой строке ответ на соответствующий тест. Если формулы эквивалентны, нужно написать `Equivalent`, а иначе — `Not equivalent`.

Пример

input.txt	output.txt
2 a&b&c+1 !a|b&a|!b&c a+b+c a|b|c&!a|!b|c&!a|b|!c&a|!b|!c	Not equivalent Equivalent

Пояснение к примеру

В первом тесте заданы неэквивалентные формулы: $(a \wedge b \wedge c) \oplus 1$ и $(\overline{a} \vee b) \wedge (a \vee \overline{b}) \wedge c$. Во втором тесте заданы эквивалентные формулы: $a \oplus b \oplus c$ и $(a \vee b \vee c) \wedge (\overline{a} \vee \overline{b} \vee c) \wedge (\overline{a} \vee b \vee \overline{c}) \wedge (a \vee \overline{b} \vee \overline{c})$.

Задача 10. Маленькая сортирующая машина

Источник:	космической сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда∗
Ограничение по памяти:	разумное

Вам предлагается испытать себя в оптимизации сортировки массива маленького размера.

В каждом тесте имеется массив из N элементов, у каждого элемента есть ключ и значение. И ключ, и значение являются беззнаковыми четырёхбайтовыми целыми числами.

Изначально во входном файле заданы только значения всех элементов массива. Далее нужно выполнить R раундов. На каждом раунде нужно:

1. Сгенерировать и записать N случайных чисел в ключи элементов массива.
2. Отсортировать элементы массива в порядке возрастания ключа.

В результате каждого раунда значения в массиве переставляются в некотором порядке, который зависит от генератора псевдослучайных чисел.

В данной задаче нужно использовать генератор псевдослучайных чисел xorwow. Исходный код этого генератора:

```
uint32_t xorwow(uint32_t state[5]) {
    uint32_t s, t = state[3];
    t ^= t >> 2;
    t ^= t << 1;
    state[3] = state[2]; state[2] = state[1]; state[1] = s = state[0];
    t ^= s;
    t ^= s << 4;
    state[0] = t;
    return t + (state[4] += 362437);
}
```

Начальное состояние генератора (пять чисел в `state`) задаётся в каждом тесте. В начале каждого раунда ключи генерируются для элементов в их текущем порядке, причём старшие два бита отбрасываются:

```
uint32_t state[5] = {..., ..., ..., ..., ...};
for (int i = 0; i < n; i++)
    elements[i].key = xorwow(state) & 0x3fffffff;
```

В конце теста нужно вывести значения всех элементов массива в их финальном порядке. Обратите внимание, что процесс полностью детерминирован, и только один финальный порядок является правильным.

Формат входных данных

В первой строке записано целое число Q — сколько тестов записано в файле ($1 \leqslant Q \leqslant 1\,000$). Далее описано Q тестов.

Описание теста начинается со строки с двумя целыми числами: N — размер массива и R — сколько раундов сортировки нужно выполнить ($1 \leqslant N \leqslant 64$, $0 \leqslant R$). Во второй строке теста записано пять шестнадцатеричных чисел, по восемь цифр в каждом — начальное содержимое массива `state` генератора `xorwow`. В третьей строке записано N целых чисел — значения элементов массива в том порядке, в котором они расположены изначально.

Гарантируется, что во всех раундах всех тестов все ключи будут различными. Суммарное количестов раундов R по всем тестам в файле не превышает 750 000.

Формат выходных данных

Для каждого из Q тестов нужно вывести ровно одну строку. В этой строке должно быть N целых чисел: значения элементов массива в их финальном порядке после всех раундов.

Пример

input.txt

```
5
15 0
b1c6114b f18c80b8 059cace1 24e9297b 5cab5281
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
15 1
b1c6114b f18c80b8 059cace1 24e9297b 5cab5281
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
15 2
b1c6114b f18c80b8 059cace1 24e9297b 5cab5281
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
15 3
b1c6114b f18c80b8 059cace1 24e9297b 5cab5281
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
10 7
4c373cdb 0102026b a8b5ef27 370796de 5840f014
135 12 13 11 10 17 10 7 1 5
```

output.txt

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 6 5 10 3 12 8 14 13 11 4 9 7 15 1
1 11 12 4 5 3 6 14 2 10 8 15 9 13 7
5 12 1 11 8 13 14 15 3 10 2 9 7 6 4
12 1 11 13 135 10 7 10 17 5
```

Комментарий

В данной задаче бессовестно жёсткое ограничение по времени. Скорее всего никто не сможет уложиться в TL — тогда задача перейдёт в разряд соревнования: кто сможет написать самое быстрое решение.

Внимание: не пытайтесь применять многопоточность! В nsuts замеряется суммарное процессорное время по всем потокам, поэтому многопоточность не поможет.

Задача 11. Большая сортирующая машина

Источник:	космической сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды*
Ограничение по памяти:	разумное

Вам предлагается испытать себя в оптимизации сортировки массива большого размера.

В каждом тесте имеется массив из N элементов, у каждого элемента есть ключ и значение. Ключ является 64-битным беззнаковым целым числом, а значение — 32-битным.

Изначально каждому элементу массива присваивается значение, равное его номеру (считая с нуля). Далее нужно выполнить R раундов. На каждом раунде нужно:

1. Сгенерировать и записать N случайных чисел в ключи элементов массива.
2. Отсортировать элементы массива в порядке возрастания ключа.

В результате каждого раунда значения в массиве переставляются в некотором порядке, который зависит от генератора псевдослучайных чисел.

В данной задаче нужно использовать генератор псевдослучайных чисел xorshift+. Исходный код этого генератора:

```
uint64_t xorshift128plus(uint64_t state[2]) {
    uint64_t x = state[0];
    uint64_t const y = state[1];
    state[0] = y;
    x ^= x << 23; // a
    state[1] = x ^ y ^ (x >> 17) ^ (y >> 26); // b, c
    return state[1] + y;
}
```

Начальное состояние генератора (два числа в `state`) задаётся в каждом тесте. В начале каждого раунда ключи генерируются для элементов в их текущем порядке, причём старшие два бита отбрасываются:

```
uint64_t state[2] = {..., ...};
for (int i = 0; i < n; i++)
    elements[i].key = xorshift128plus(state) & 0x3fffffffffffffffLL;
```

В конце теста нужно вывести значения всех элементов массива в их финальном порядке. Обратите внимание, что процесс полностью детерминирован, и только один финальный порядок является правильным.

Формат входных данных

В первой строке записано целое число Q — сколько тестов записано в файле ($1 \leqslant Q \leqslant 10$). Далее описано Q тестов.

Описание теста начинается со строки с двумя целыми числами: N — размер массива и R — сколько раундов сортировки нужно выполнить ($1 \leqslant N \leqslant 10^6$, $0 \leqslant R$). Во второй строке теста записано два шестнадцатеричных числа, по шестнадцать цифр в каждом — начальное содержимое массива `state` генератора `xorshift+`.

Гарантируется, что во всех раундах всех тестов все ключи будут различными. Суммарное количестов раундов R по всем тестам в файле не превышает 18. Кроме того, сумма всех N в файле не превышает 10^6.

Формат выходных данных

Для каждого из Q тестов нужно вывести ровно одну строку. В этой строке должно быть N целых чисел: значения элементов массива в их финальном порядке после всех раундов.

Пример

input.txt
5 15 0 b1c6114bf18c80b8 059cace124e9297b 15 1 b1c6114bf18c80b8 059cace124e9297b 15 2 b1c6114bf18c80b8 059cace124e9297b 15 3 b1c6114bf18c80b8 059cace124e9297b 10 7 4c373cdb0102026b a8b5ef27370796de
output.txt
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 8 14 10 6 0 7 4 12 2 11 1 3 13 9 5 2 12 6 1 0 9 4 8 5 3 10 7 14 13 11 11 4 1 0 6 14 9 3 12 8 10 2 7 5 13 5 1 0 4 2 7 9 3 6 8

Комментарий

В данной задаче бессовестно жёсткое ограничение по времени. Скорее всего никто не сможет уложиться в TL — тогда задача перейдёт в разряд соревнования: кто сможет написать самое быстрое решение.

Внимание: не пытайтесь применять многопоточность! В nsuts замеряется суммарное процессорное время по всем потокам, поэтому многопоточность не поможет.