Содержание

Во всех задачах можно использовать `std::sort`.

Пример работы с файлами.

Если вы не умеете читать/выводить данные, или открывать файлы, воспользуйтесь примерами. `http://acm.math.spbu.ru/~sk1/algo/sum/`

Пример работы с файлами.

В некоторых задачах большой ввод и вывод. Про ввод-вывод в C++:
`http://acm.math.spbu.ru/~sk1/algo/input-output/cpp_common.html`
Имеет смысл пользоваться супер быстрым вводом-выводом. Две версии:
`http://acm.math.spbu.ru/~sk1/algo/input-output/io_export.cpp.html`
`http://acm.math.spbu.ru/~sk1/algo/input-output/fread_write_export.cpp.html`

Выделение памяти.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу:
`http://acm.math.spbu.ru/~sk1/algo/memory.cpp.html`

Обязательные задачи

Задача 3A. Обмен [0.3 sec, 256 mb]

Пусть все натуральные числа исходно организованы в список в естественном порядке. Разрешается выполнить следующую операцию: $swap(a, b)$. Эта операция возвращает в качестве результата расстояние в текущем списке между числами a и b и меняет их местами.

Задана последовательность операций *swap*. Требуется вывести в выходной файл результат всех этих операций.

Формат входных данных

Первая строка входного файла содержит число n ($1 \leqslant n \leqslant 200\,000$) — количество операций. Каждая из следующих n строк содержит по два числа в диапазоне от 1 до 10^9 — аргументы операций *swap*.

Формат выходных данных

Для каждой операции во входном файле выведите ее результат.

Пример

swap.in	swap.out
4 1 4 1 3 4 5 1 4	3 1 4 2

Замечание

В этой задаче нельзя использовать STL-контейнеры.

Задача 3B. Квадратный корень [0.3 sec, 256 mb]

Дано целое число n от 0 до $2^{64} - 1$. Ваша задача — найти $\lfloor \sqrt{n} \rfloor$.

Формат входных данных

Мультитест. На каждой строке по числу n. Не более 1000 строк.

Формат выходных данных

Для каждого n на отдельной строке ответ на запрос.

Примеры

sqrt.in	sqrt.out
0 1 2 3 4 5	0 1 1 1 2 2

Задача 3C. Поиск [0.5 sec, 256 mb]

В этой задаче нужно уметь выяснять, содержится ли число в последовательности.

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и k ($1 \leqslant n \leqslant 300\,000$, $1 \leqslant k \leqslant 300\,000$). Во второй строке задана последовательность из n отсортированных целых чисел a_1, a_2, ..., a_n, записанных через пробел ($1 \leqslant a_i \leqslant 10^9$). В третьей строке записаны запросы — k целых чисел b_1, b_2, ..., b_k записанных через пробел, в порядке возрастания ($1 \leqslant b_j \leqslant 10^9$).

Формат выходных данных

В выходной файл выведите k строк. В j-ой строке выведите "`YES`", если число b_j содержится в последовательности $\{a_i\}$, и "`NO`" в противном случае.

Примеры

find2.in	find2.out
3 3 2 3 5 1 2 3	NO YES YES
3 4 1 2 2 1 2 4 5	YES YES NO NO

Замечание

В этой задаче можно использовать STL =)

Задача 3D. Линейная сумма [2 sec, 256 mb]

Есть n случайных точек на прямой с координатами от 0 до $2^{32}-1$. У каждой точки есть значение от 0 до $2^{32}-1$. Вам нужно обработать q случайных запросов вида "сумма значений точек, с координатами от l до r включительно".

Формат входных данных

На первой строке числа n, q. ($1 \leqslant n \leqslant 2^{20}, 1 \leqslant q \leqslant 2^{23}$). На второй строке пара целых чисел a, b от 1 до 10^9, используемая в генераторе случайных чисел.

```
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand24() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur >> 8; // число от 0 до 2^24 - 1.
5. }
6. unsigned int nextRand32() {
7.     unsigned int a = nextRand24(), b = nextRand24();
8.     return (a << 8) ^ b; // число от 0 до 2^32 - 1.
9. }
```

Каждая точка генерируется следующим образом:

```
1. value = nextRand32(); // значение точки
2. x = nextRand32(); // координата точки
```

Каждый запрос генерируется следующим образом:

```
1. l = nextRand32();
2. r = nextRand32();
3. if (l > r) swap(l, r); // получили отрезок [l..r]
```

Сперва генерируются точки, затем запросы.

Формат выходных данных

Выведите сумму ответов на все запросы по модулю 2^{32}.

Примеры

linesum.in	linesum.out
5 5 13 239	3950632748

Замечание

```
   p = {value, x}
p[0] = {13, 41645}
p[1] = {7695587, 1253435649}
p[2] = {749170640, 2683600557}
p[3] = {2444595881, 1270561959}
p[4] = {3436107648, 486388002}
```

Задача 3E. Минимум и максимум [0.3 sec, 256 mb]

Пусть есть мультимножество целых чисел (множество с возможными повторениями). Необходимо реализовать структуру данных для их хранения, поддерживающую следующие операции: `GetMin` — извлечение минимума, `GetMax` — извлечение максимума, `Insert(N)` — добавление числа в множество.

Формат входных данных

В первой строке входного файла записано одно целое число N ($1 \leqslant N \leqslant 100\,000$) — число запросов к структуре. Затем в N строках следуют запросы по одному в строке: `GetMin`, `GetMax`, `Insert(A)` — извлечение минимума, максимума и добавление числа A ($1 \leqslant A \leqslant 2^{31} - 1$). Запросы корректны, то есть нет операций извлечения для пустого множества.

Формат выходных данных

Для каждого запроса `GetMin` или `GetMax` выведите то число, которое было извлечено.

Примеры

minmax.in	minmax.out
10 Insert(100) Insert(99) Insert(1) Insert(2) GetMin GetMax Insert(1) GetMin GetMin GetMax	1 100 1 2 99

Замечание

В этой задаче нельзя использовать STL-контейнеры.

Задача 3F. Управление Памятью [0.1 sec, 256 mb]

Недавно школьники изобрели новый язык программированию. Язык называется `D++`. На самом деле не важно, слышали ли вы уже про этот язык, или нет. Важно, что, чтобы запускать программы на языке `D++`, нужна новая операционная система. Новая ОС должна быть достаточно мощной, должна работать быстро и иметь кучу возможностей. Но это всё в будущем. А сейчас нужно... Нет. Не нужно придумывать название для новой ОС. Вам предстоит реализовать первый модуль в новой ОС. Конечно, этот модуль – модуль управления памятью. Давайте обсудим, как он должен работать.

Наша ОС будет выделять память кусками, назовём их *"блоки"*. Блоки нумеруются целыми числами от 1 до N. Когда ОС требуется больше памяти, она обращается к модулю управления памятью. Чтобы обработать такой запрос, модуль должен найти свободный блок памяти с минимальным номером. Можете смело предположить, что памяти достаточно, чтобы обработать все запросы. Определим понятие *"свободный блок"*. В момент первого запроса к памяти все блоки свободны. Занятый блок становится свободным, если к нему нет запросов доступа в течении T минут. Вас может удивить запись "запрос доступа к занятому блоку". Что же это значит? Ответ прост: в любой момент времени модуль управления памятью может получить запрос доступа к занятому блоку памяти. Обработка такого запроса происходит следующим образом: модуль проверяет, правда ли запрашиваемый блок занят. Если да, запрос считается успешным и блок остаётся занятым ещё на T минут. Иначе запрос считается ошибочным. Это всё, что нужно знать. Ваша задача – реализовать алгоритм для $N = 30\,000$ и $T = 10$ минутам.

Формат входных данных

Каждая строка ввода содержит запрос доступа к блоку памяти или запрос на выделение памяти. Запрос выделения памяти имеет форму "`<Time> +`", где `<Time>` – неотрицательное целоые число не более 65 000. Время даётся в секундах. Запрос доступа к памяти имеет форму "`<Time> . <BlockNo>`", где `<Time>` значит то же, что и выше, а `<BlockNo>` – целое число от 1 до N. Всего будет не более 80 000 запросов.

Формат выходных данных

Для каждой строки ввода нужно вывести ровно одну строку с ответом на запрос. На запрос выделения памяти выведите целое число – номер выделенного блока. Как было уже замечено выше, вы можете предположить, что одновременно будет нужно не более N блоков. Для запроса доступа к блоку памяти нужно выести один символ:

- "`+`" если запрос успешен (блок занят)
- "`-`" если запрос ошибочен (блок свободен)

Запросы даны в порядке возрастания времени. Запросы с одинаковым временем должны обрабатываться в том порядке, в котором даны.

Примеры

memory.in	memory.out
1 + 1 + 1 + 2 . 2 2 . 3 3 . 30000 601 . 1 601 . 2 602 . 3 602 + 602 + 1202 . 2	1 2 3 + + - - + - 1 3 -

Замечание

В этой задаче нельзя использовать STL-контейнеры.

Задача 3G. Сложение и вычитание [0.5 sec, 256 mb]

Выведите значение заданного арифметического выражения, состоящего из чисел, скобок и знаков сложения и вычитания.

Формат входных данных

В первой строке входного файла задано выражение, состоящее из чисел, скобок и знаков бинарных операций. Каждое число в выражении это — целое неотрицательное число в промежутке от 0 до 10 000, включительно, записанное без ведущих нулей. Скобки бывают открывающие (‘(’) и закрывающие (‘)’). Операции задаются символами ‘+’ и ‘-’. Гарантируется, что заданное выражение математически корректно, и результаты всех промежуточных операций — целые числа, не превышающие по модулю 10 000. Выражение не содержит каких-либо других символов, в частности, пробелов. Длина выражения не меньше 1 и не больше 1000 символов.

Учтите, что операции при отсутствии скобок выполняются слева направо. Например, выражение $a - b - c$ вычисляется как $(a - b) - c$.

Формат выходных данных

В первой строке выходного файла выведите одно число — значение заданного выражения.

Примеры

evalpm.in	evalpm.out
48-13	35
5-(52+3)	-50

Задача 3H. Длинное выражение [0.3 sec, 256 mb]

Выведите значение заданного арифметического выражения.

Формат входных данных

В первой строке входного файла задано выражение, состоящее из чисел, скобок и знаков бинарных операций. Каждое число в выражении это — целое неотрицательное число в промежутке от 0 до 10 000, включительно, записанное без ведущих нулей. Скобки бывают открывающие (‘(’) и закрывающие (‘)’). Операции задаются символами ‘+’, ‘-’, ‘*’ и ‘/’; знак умножения не может быть опущен. Гарантируется, что заданное выражение математически корректно, и результаты всех промежуточных операций — целые числа, не превышающие по модулю 10^9. Выражение не содержит каких-либо других символов, в частности, пробелов. Длина выражения не меньше 1 и не больше 1 000 000 символов.

Учтите, что операции с одинаковым приоритетом при отсутствии скобок выполняются слева направо. Например, выражение $a + b + c$ вычисляется как $(a + b) + c$.

Формат выходных данных

В первой строке выходного файла выведите одно число — значение заданного выражения.

Примеры

evalhard.in	evalhard.out
40-8/1*3	16
(5+50)/(2+3)	11

Замечание

Подробно решение обсудим в пятницу на лекции...

Дополнительные задачи

Задача 3I. Менеджер памяти [0.3 sec, 256 mb]

Пете поручили написать менеджер памяти для новой стандартной библиотеки языка H++. В распоряжении у менеджера находится массив из N последовательных ячеек памяти, пронумерованных от 1 до N. Задача менеджера – обрабатывать запросы приложений на выделение и освобождение памяти. Запрос на выделение памяти имеет один параметр K. Такой запрос означает, что приложение просит выделить ему K последовательных ячеек памяти. Если в распоряжении менеджера есть хотя бы один свободный блок из K последовательных ячеек, то он обязан в ответ на запрос выделить такой блок. При этом непосредственно перед самой первой ячейкой памяти выделяемого блока не должно располагаться свободной ячейки памяти. После этого выделенные ячейки становятся занятыми и не могут быть использованы для выделения памяти, пока не будут освобождены. Если блока из K последовательных свободных ячеек нет, то запрос отклоняется. Запрос на освобождение памяти имеет один параметр T. Такой запрос означает, что менеджер должен освободить память, выделенную ранее при обработке запроса с порядковым номером T. Запросы нумеруются, начиная с единицы. Гарантируется, что запрос с номером T – запрос на выделение, причем к нему еще не применялось освобождение памяти. Освобожденные ячейки могут снова быть использованы для выделения памяти. Если запрос с номером T был отклонен, то текущий запрос на освобождение памяти игнорируется. Требуется написать менеджер памяти, удовлетворяющий приведенным критериям.

Формат входных данных

Первая строка входного файла содержит числа N и M – количество ячеек памяти и количество запросов соответственно ($1 \leqslant N \leqslant 2^{31} - 1$; $1 \leqslant M \leqslant 10^5$). Каждая из следующих M строк содержит по одному числу: $(i+1)$-я строка входного файла ($1 \leqslant i \leqslant M$) содержит либо положительное число K, если i-й запрос – запрос на выделение с параметром K ($1 \leqslant K \leqslant N$), либо отрицательное число $-T$, если i-й запрос – запрос на освобождение с параметром T ($1 \leqslant T < i$).

Формат выходных данных

Для каждого запроса на выделение памяти выведите в выходной файл результат обработки этого запроса: для успешных запросов выведите номер первой ячейки памяти в выделенном блоке, для отклоненных запросов выведите число -1. Результаты нужно выводить в порядке следования запросов во входном файле.

Примеры

memory.in	memory.out
6 8 2 3 -1 3 3 -5 2 2	1 3 -1 -1 1 -1

Замечание

В этой задаче нельзя использовать STL-контейнеры.

Задача 3J. Минимумы в подматрицах [0.5 sec, 256 mb]

Дана матрица $n \times n$, состоящая из целых чисел. Для каждой её подматрицы размера $L \times L$ найдите минимум в этой подматрице. Подматрицей здесь называется "подпрямоугольник".

Внимание. Решение должно работать за $O(n^2)$.

Формат входных данных

Первая строка входных данных содержит два целых числа n и L ($1 \leqslant L \leqslant n \leqslant 1000$). Далее в n строках идет описание матрицы $n \times n$, по n чисел в каждой строке. Все числа в матрицы целые, от -10^9 до 10^9.

Формат выходных данных

Выведите $n - L + 1$ строку по $n - L + 1$ числу в каждой. j-е число в i-й строке должно быть равно минимуму в подматрице размера $L \times L$ с левым верхнем углом на пересечении i-й строки и j-го столбца исходной матрицы.

Примеры

matrixmin.in	matrixmin.out
1 1 5	5
2 1 2 1 3 4	2 1 3 4
2 2 2 1 3 4	1
4 2 4 5 3 2 1 2 5 4 3 4 2 3 1 3 5 5	1 2 2 1 2 2 1 2 2

Замечание

В этой задаче нельзя использовать структуры данных, которые мы ещё не проходили – деревья отрезков, sparse table, декартовы деревья.

Задача 3K. Интересный разбор выражений [2 sec, 256 mb]

Задача: дано арифметическое выражение, посчитайте значение.

В выражении присутствуют:

- Целые числа из диапазона $[-2^{31}, 2^{31})$.
- Операции:
  `+`, `-`, `*` (сложение, вычитание, умножение),
  `%` (остаток по модулю, не отрицателен),
  `/` (целочисленное деление, остаток неотрицателен),
  `^` (возведение в степень).
- Унарный минус.
- Скобки трех типов: `{ } [ ] ( )`.
- Переменные с целочисленными значениями. Имена переменных — строки из букв латинского алфавита. Регистр важен.
- Функции `sqr` (квадрат числа), `cube` (куб числа), `sign` (знак числа).

Правила разбора выражения:

- Минус: после числа/имени или закрывающей скобки идет бинарный минус. Иначе минус унарный.
- Приоритеты операций: (`+`, `-`) затем (`*`, `%`, `/`) затем (`^`).
- Операции `+`, `-`, `*`, `/` левоассоциативны: `2-3+4 = (2-3)+4`.
- Операция возведения в степень `^` правоассоциативна: `3^3^3 = 3^27`.
- Если после имени идет открывающая скобка — это функция, иначе переменная.

Вычисление выражения: сперва происходит разбиение на лексемы и построение дерева вычислений, затем вычисляется значение выражения с помощью обхода дерево разбора слева направо. Сперва вычисляются значения аргументов операции в порядке слева направо, а когда все аргументы посчитаны, вычисляется значение оператора. Унарные операторы и функции в дереве имеют степень один, бинарные операторы имеют степень два.

Формат входных данных

Первые несколько строк содержат значения переменных в формате ⟨`name`⟩ = ⟨`value`⟩. Эти строки точно корректны. Имена переменных в присваиваниях могут совпадать, используется последнее значение. Последняя строка содержит арифметическое выражение, значение которого нужно посчитать. Арифметическое выражение может содержать синтаксические ошибки, и ошибки, непозволяющие вычислить его значение. Подробнее читайте ниже. Суммарная длина всех строк входного не более 10^6. Во входном файлы используются только допустимые символы.

Формат выходных данных

Если арифметическое выражение некорректно, выведите `Error:` ⟨`ошибка`⟩. Если произошло несколько ошибок, нужно выводить только первую.

Ошибки на этапе разбора выражения на лексемы в порядке приоритета:

- `unmatched bracket` — лишние или не парные скобки.
- `parsing expression` — численные значения и операции не чередуются.
- `undefined name` — имя, которое не соответствует ни переменной, ни функции.
- `too long integer` — используется число не из диапазона $[-2^{31}, 2^{31})$.

Ошибки на этапе вычисления значения выражения, выражение вычисляется обходом дерева слева направо, нас интересует первая ошибка именно в этом порядке:

- `integer overflow` — конечное, или одно из промежуточных значений лежат за пределами диапазона $[-2^{31}, 2^{31})$.
- `dividing by zero` — деление на ноль.
- `negative power is not allowed` — возведение в отрицательную степень.

Если арифметическое выражение задано корректно и может быть корректно вычислено, выведите целое число — значение выражения.

Примеры

exprf.in	exprf.out
x = 2 value = 3 x^value - cube(2+3)	-117
-2+3-(-2+3)+[-100]-{-100}--100+ sqr(-2)+sign(-1)+-2^5	71
x = 2 x^100	Error: integer overflow
()	Error: parsing expression
(2 2 / 0 + 1000000000000	Error: unmatched bracket
(2 2) / 0 + 1000000000000	Error: parsing expression
(2 + x) / 0 + 1000000000000	Error: undefined name
(2 + 2) / 0 + 1000000000000	Error: too long integer
(2 + 2) / 0	Error: dividing by zero
2^(2 - sqr(2))	Error: negative power is not allowed
sqr = 8 sqr = 10 sqr + (sqr + 2 + sqr) + sqr(sqr(sqr)) + sqr	10042