Задача 2-1 (35 баллов). Дано две последовательности целых чисел: $x_1, x_2, \ldots, x_n$ и $y_1, y_2, \ldots, y_k$. Нужно найти их наибольшую общую подпоследовательность, то есть такую последовательность $a_1, a_2, \ldots, a_m$, что $a_1 = x_{i_1} = y_{j_1}, a_2 = x_{i_2} = y_{j_2}, \ldots, a_m = x_{i_m} = y_{j_m}$, $1 \le i_1 < i_2 < \cdots < i_m \le n$, $1 \le j_1 < j_2 < \cdots < j_m \le k$.

В первой строке входа число n, во второй — последовательность x_1, x_2, …, x_n, в третьей — число k, в четвертой — последовательность $y_1, y_2, \ldots, y_k$. Выведите одно число — длину наибольшей общей подпоследовательности.

Ограничения. $1 \le n, k \le 2000$. Числа не превосходят по модулю 10^9.

Пример входа	Пример выхода
5 1 2 3 4 5 5 1 3 5 7 9	3
1 1 1 2	0

Решение.

Задача решается методом динамического программирования. Как всегда в этом методе, мы решим больше задач, чем нужно, но за счет этого нам будет проще получать решения для отдельных подзадач через решения других подзадач.

Найдем для каждых двух префиксов последовательностей x и y длину их наибольшей общей подпоследовательности. Т.е. для каждого $k = 0, 1, \ldots, n$ и $l = 0, 1, \ldots, m$ найдем длину наибольшей подпоследовательности префикса x длины k и префикса y длины l. Если $k = 0$ или $l = 0$, то очевидно, что ответ также равен нулю. Теперь пусть нам нужно вычислить ответ $Len[k][l], k > 0, l > 0$, а мы уже знаем ответы для всех пар $Len[k'][l']$, где либо $k' < k$, либо $k' = k, l' < l$. Рассмотрим числа $x[k-1]$ и $y[l-1]$. Если они совпадают, то ясно, что можно их взять оба в общую подпоследовательность, и она от этого не ухудшится. Поэтому в этом случае $Len[k][l] = Len[k-1][l-1] + 1$. Если же они разные, то как минимум один из них не участвует в наибольшей общей подпоследовательности, поэтому $Len[k][l] = \max(Len[k-1][l], Len[k][l-1])$.

Каждое значение $Len[k][l]$ вычисляется за $O(1)$ через предыдущие, а всех таких значений $O(mn)$, поэтому сложность алгоритма тоже $O(mn)$. Памяти уйдет, если не экономить, тоже $O(mn)$. Можно сэкономить, заметив, что в вычислениях используется всегда только два последних слоя матрицы Len, и тогда хранить только два этих последних слоя. Тогда можно добиться оценки памяти $O(n)$ или $O(m)$ или даже $O(\min(m, n))$.

Если бы в задаче требовалось не только вычислить длину наибольшей общей подпоследовательности, но и саму эту подпоследовательность, то в данном методе пришлось бы уже обязательно использовать $O(mn)$ памяти. По полной матрице Len можно восстановить и саму подпоследовательность: становимся в позицию $Len[n][m]$, и начинаем двигаться по матрице, по ходу дела выдавая подпоследовательность с конца. Если в текущей позиции $Len[k][l] = Len[k-1][l-1] + 1$, то выдаем число $x[k] = y[l]$ и переходим к $Len[k-1][l-1]$, иначе переходим либо к $Len[k-1][l]$, либо к $Len[k][l-1]$, к тому из них, которое равно $Len[k][l]$. Когда дойдем до позиции, где $k = 0$ или $l = 0$, останавливаемся.

Затем останется только перевернуть последовательность.

Задача 2-2 (40 баллов). Вы хотите набрать футбольную команду. У каждого игрока своя эффектиность, она описывается одним целым числом. Чем больше число, тем больше эффективность футболиста. Обязательным условием для любой команды является сплоченность. Если один из игроков играет сильно лучше всех остальных, его будут недолюбливать, и команда распадется. Поэтому эффективность любого игрока команды не должна превышать сумму эффективностей любых двух других игроков. Ваша задача — набрать команду, которая будет удовлетворять условию сплоченности, и при этом иметь наибольшую суммарную эффективность.

В первой строке входа задано число n ($1 \le n \le 100000$). Во второй строке — эффективности каждого из n игроков — положительные целые числа, не превосходящие $2^{31}-1$. Выведите две строки. В первую запишите наибольшую возможную сумму эффективностей игроков в команде. Во вторую строку выведите номера всех игроков, которых нужно взять в команду, в порядке возрастания. Игроки пронумерованы от 1 до n в том порядке, в котором заданы их эффективности на входе. Если существует несколько способов набрать команду максимальной эффективности, выведите любой из них.

Пример входа	Пример выхода
5 3 2 5 4 1	14 1 2 3 4
5 1 2 4 8 16	24 4 5

Решение.

Отсортируем всех игроков в порядке возрастания эффективности. Докажем, что в получившемся массиве оптимальная команда занимает целый непрерывный отрезок. Действительно, зафиксируем двух наиболее слабых игроков в оптимальной команде, пусть их индексы i и j, $i < j$. Если $j \neq i+1$, то можно игрока i заменить на игрока $j-1$ с большей эффективностью — противоречие с оптимальностью команды. Значит, $j = i+1$. Кроме того, если k — индекс наиболее эффективного игрока в команде, то всех игроков от i до k можно взять в команду без нарушения условия задачи, а значит мы их обязательно возьмем, т.к. все эффективности положительны, и сумма только увеличивается, если мы берем дополнительных игроков.

Это дает нам возможность решить оставшуюся часть задачи за $O(n)$. Вся команда однозначно определяется своим самым слабым игроком: если это игрок с индексом i, то все игроки, начиная с i-го и до последнего, не сильнее чем сила i-го плюс сила $i+1$-го обязаны быть в команде, и только они. Будем перебирать самого слабого игрока по i от 0 до $n-1$. При этом когда мы сдвигаем i вправо, индекс самого сильного игрока в команде тоже может сдвигаться только вправо. Для $i = 0$ линейным поиском найдем наибольшее такое k, что $Strength[k] <= Strength[0] + Strength[1]$, и посчитаем эффективность команды из всех игроков от 0 до k. Далее в цикле будем сдвигать i на один вправо, вычитать $Strength[i-1]$ из суммарной силы команды, а затем сдвигать по одному вправо k, пока выполнено условие $k+1 < n$ и $Strength[k+1] <= Strength[i] + Strength[i+1]$, и прибавлять $Strength[k]$ к суммарной силе команды. Если сила текущей команды больше текущего максимума, то запоминаем новый максимум. Это будет работать за $O(n)$, т.к.

оба индекса — i и k — двигаются только вправо, и общий пройденный каждым из них путь равен n.

Общая сложность решения $O(n \log n)$, т.е. столько уходит на сортировку, а далее решение линейное. Затраты памяти линейны.

Задача 2-3 (65 баллов). На плоскости даны n точек с координатами $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Необходимо накрыть по крайней мере k из них кругом с центром на оси Ox. Найдите наименьший возможный радиус такого круга с точностью до 10^{-3}.

В первой строке входа даны числа n и k, $1 \le k \le n \le 10000$. В следующих n строках — по два целых числа, координаты соотетствующей точки. Координаты не превосходят по модулю 1000. Выведите радиус круга ровно с 6 знаками после запятой. Ваш ответ должен отличаться от правильного не более, чем на 10^{-3}, вывести нужно при этом 6 знаков, чтобы избежать ошибок из-за округления.

Пример входа	Пример выхода
3 3 0 5 3 4 -4 -3	5.000000
3 2 0 1 2 1 1 100	1.414214

Решение.

Будем искать ответ к задаче бинарным поиском. Заметим, что если R — искомый радиус, то для любого $r < R$ не существует круга радиуса r с центром на Ox, покрывающего хотя бы k точек из нашего множества, а для любого $r >= R$ такой круг существует. Поэтому если бы мы научились определять для фиксированного r, существует ли такой круг, то мы бы дальше просто применили бинарный поиск по r и нашли бы искомый радиус с любой заданной наперед точностью.

Как же нам узнать, существует ли такой круг? Для каждой из точек нашего множества существует геометрическое место точек на прямой Ox, которые могут являться центром круга радиуса r, накрывающего нашу точку. А именно, это ГМТ представляет собой либо отрезок, либо пустое множество. Если ордината y нашей точки больше r, то это пустое множество, иначе это отрезок с концами $[x - \sqrt{r^2 - y^2}, x + \sqrt{r^2 - y^2}]$. Соответственно, задача сводится к следующей: существует ли точка на прямой Ox, которая покрыта хотя бы k отрезками из данных отрезков (для каждой из наших точек составляем отрезок на Ox из возможных центров круга, накрывающего нашу точку; если точка на Ox принадлежит l таким отрезкам, то круг с центром в ней радиуса r накрывает ровно l наших точек).

Вместо этого мы решим следующую задачу: дано n отрезков на прямой. Определить, каким наибольшим количеством отрезков из этих накрыта какая-либо точка прямой. Понятно, что искомой точкой является один из концов отрезков. Отсортируем все концы отрезков по координате. В случае равной координаты первыми идут все левые

концы, а затем все правые. Затем будем двигаться по массиву с концами слева направо (мы помним, является ли очередная точка левым или правым концом отрезка). Если мы встречаем левый конец, значит текущую точку начинает покрывать на один отрезок больше. Если встречаем правый конец, то мы "выходим" из очередного отрезка. Соответственно, будем поддерживать текущее количество отрезков, накрывающих нашу точку. Изначально это ноль, если встречаем левый конец прибавляем единицу, если встречаем правый — вычитаем единицу. Запомним максимальное из промежуточных значений — это и будет ответ. А точка, в которой он достигнут, накрыта наибольшим количеством отрезков. Здесь важно, что в случае равной координаты сначала идут левые концы, а потом уже правые, т.к. правые концы отрезков должны учитываться, когда мы добавляем отрезки с левым концом в этой точке. Соответственно, эту задачу мы решили за $O(n \log n)$, причем все, кроме сортировки, делается за линейное время. Если найдем точку, накрытую наибольшим числом отрезков, то сможем определить, есть ли точка, накрытая хотя бы k отрезками, а значит решим подзадачу для бинарного поиска.

Общая сложность алгоритма таким образом составляет $O(n \log n \log \frac{MaxR}{\epsilon})$, где $MaxR$ — максимально возможное значение радиуса, исходя из ограничений на координаты точек, а ϵ — требуемая в задаче точность.

Есть и другое решение.

Рассмотрим оптимальный круг. Докажем, что на его границе либо лежат хотя бы две из наших точек, либо лежит хотя бы одна точка из наших, абсцисса которой совпадает с абсциссой центра круга. Действительно, если на границе нет точек, то очевидно, что круг можно немножко уменьшить, и он все еще будет содержать k точек. Если на границе ровно одна точка, и она не находится ровно над или под центром круга, то можно немного сдвинуть круг в сторону этой точки, а затем еще немного уменьшить, и круг все еще будет содержать k точек. Т.к. круг оптимален, то его нельзя уменьшить, поэтому выполнено указанное условие.

Тогда искомый круг можно найти перебором всех пар точек, лежащих на окружности, определяющих круг с центром на Ox однозначно, и всех точек, которые лежат на окружности непосредственно над или под центром круга, этим круг тоже определяется однозначно. Всего таких кругов $O(n^2)$. Каждый из них нужно проверить: лежит ли в нем хотя бы k точек, для этого придется произвести $O(n)$ действий. Общая сложность алгоритма получается $O(n^3)$, что сильно медленнее первого решения, но зато находит точное значение радиуса и координаты центра (по крайней мере, квадрат радиуса — рациональное число, если изначально координаты всех точек целые, поэтому квадрат радиуса можно найти абсолютно точно).

Задача 2-4 (35 баллов). Вам дано n упорядоченных по возрастанию последовательностей целых чисел, каждая из которых имеет длину m. Необходимо слить их в одну упорядоченную по возрастанию последовательность целых чисел длины mn. Сложность алгоритма не должна превышать $O(mn \log n)$, затраты памяти — $O(mn)$.

В первой строке входа два целых числа n, m ($1 \le m, n \le 1000$). В следующих n строках по m целых чисел в каждой, числа в каждой строке упорядочены по возрастанию. Числа не превосходят по модулю 10^9. Выведите все mn чисел, упорядоченных по возрастанию.

Пример входа	Пример выхода
2 5 1 3 5 7 9 2 4 6 8 10	1 2 3 4 5 6 7 8 9 10
4 2 1 4 2 8 3 7 5 6	1 2 3 4 5 6 7 8

Решение.

Слияние двух последовательностей длины k мы можем произвести за $O(k)$ операций. Будем сливать исходные последовательности постепенно, как будто мы на самом деле производим сортировку слиянием. На первом шаге сольем все пары последовательностей и получим набор последовательностей длины $2m$ (возможно, еще останется одна последовательность длины m), затем сольем пары получившихся последовательностей и т.д., пока не останется единственная последовательность. На каждом этапе суммарная длина всех последовательностей остается равной mn, а значит на каждый этап мы тратим $O(mn)$ операций. Всего этапов будет $O(\log n)$, поэтому общая сложность алгоритма $O(mn \log n)$. Затраты памяти $O(mn)$. Можно либо использовать дополнительные $O(mn)$ памяти для всех операций, либо проводить слияние на месте, и тогда дополнительная память не нужна.

Второе решение опирается на кучи. К тому моменту, как эта задача была дана, кучи мы еще не проходили, поэтому такое решение не ожидалось, однако оно более стандартное, и некоторые люди прислали именно его. Создадим кучу, изначально добавим в нее первые элементы всех n последовательностей. Будем последовательно доставать наименьший элемент из кучи, определять, из какой из последовательностей этот элемент, а затем добавлять в кучу следующий элемент этой последовательности, если в ней еще остались элементы. Будем повторять процесс, пока не закончатся все элементы. Очевидно, в этом случае мы выдадим упорядоченный набор всех данных чисел. Размер кучи всегда будет равен n или $n - 1$, поэтому каждая операция занимает $O(\log n)$ времени, а всего операций mn, поэтому общая сложность опять же $O(mn \log n)$. Затраты памяти $O(mn)$, дополнительной памяти нужно $O(n)$.