Техника двух указателей

11244. Сумма двух
11246. Сумма трех
9631. Соревнования последовательностей
9632. Лучшая команда
2910. ABCDEF

11244. Сумма двух

Задан массив A, отсортированный по возрастанию и содержащий n целых чисел. Определите, существует ли в нем такая пара чисел (A_i, A_j), $i < j$, сумма которых равна x.

Вход. Первая строка содержит два целых числа n ($n \le 10^5$) и x ($x \le 10^6$). Вторая строка содержит n целых неотрицательных чисел, каждое из которых не больше 10^6.

Выход. Выведите “YES” если такая пара элементов существует, и “NO” иначе.

Пример входа 1
```
10 13
1 3 5 6 8 10 11 11 11 16
```

Пример выхода 1
```
YES
```

Пример входа 2
```
8 61
5 5 8 12 16 21 44 50
```

Пример выхода 2
```
NO
```

Анализ алгоритма

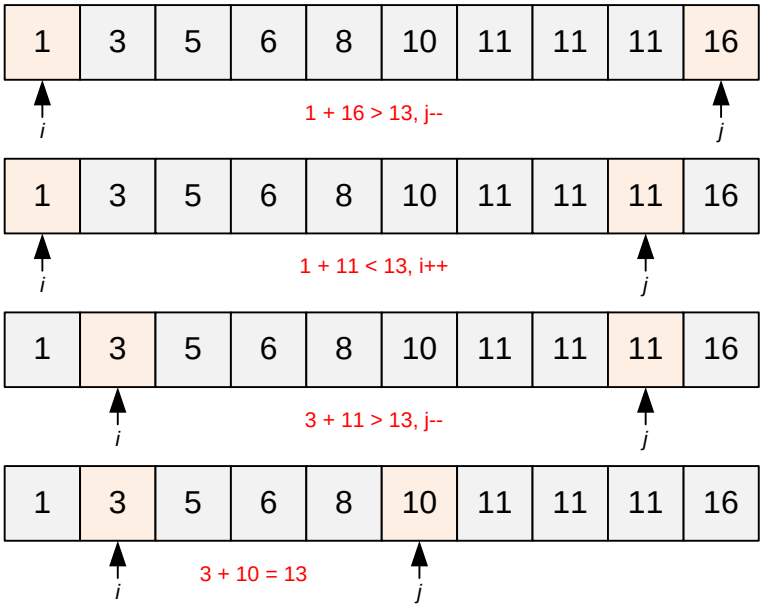
Пусть v – входной массив чисел. Инициализируем указатели $i = 0$ на начало массива, $j = n - 1$ на конец массива. Массив по условию задачи уже отсортирован.

Пока указатели i и j не встретятся, совершаем следующие действия:

- Если v[i] + v[j] = x, то искомая пара элементов найдена, выводим ее и завершаем программу.
- Если v[i] + v[j] < x, то двигаем указатель i на одну позицию вправо. В этом случае сумма v[i] + v[j] будет увеличиваться;
- Если v[i] + v[j] > x, то двигаем указатель j на одну позицию влево. В этом случае сумма v[i] + v[j] будет уменьшаться;

Пример

Рассмотрим первый тест. Инициализируем указатели. Промоделируем работу алгоритма. Значение $x = 13$, ищем два числа с суммой 13.



Реализация алгоритма

Читаем входные данные.

```
scanf("%d %d", &n, &x);
v.resize(n);
for (i = 0; i < n; i++)
  scanf("%d", &v[i]);
```

Инициализируем указатели.

```
int i = 0, j = v.size() - 1;
```

Указатель i двигаем вперед, указатель j двигаем назад.

```
while (i < j)
{
```

Если v[i] + v[j] = x, то искомая пара элементов найдена, выводим ее и завершаем программу.

```
  if (v[i] + v[j] == x)
  {
    printf("YES\n");
    return 0;
  }
```

Если v[i] + v[j] < x, то двигаем указатель i на одну позицию вправо;
Если v[i] + v[j] > x, то двигаем указатель j на одну позицию влево;

```
  if (v[i] + v[j] < x) i++; else j--;
}
```

Если искомая пара не найдена, выводим “NO”.

```
printf("NO\n");
```

11246. Сумма трех

Задан массив целых чисел A и целое число x. Найдите такую тройку чисел (A_i, A_j, A_k) в массиве, сумма которых равна x. Все индексы i, j, k должны быть различны.

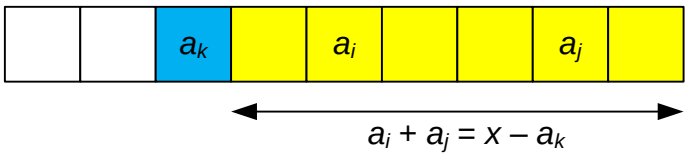
Вход. Первая строка содержит размер массива n ($n \le 10^4$) и значение x ($|x| \le 10^9$). Вторая строка содержит n целых чисел, каждое из которых по модулю не больше 10^9.

Выход. Если требуемая тройка чисел существует, то выведите ее в любом порядке. Если существует несколько троек, выведите любую. Если искомой тройки не существует, выведите -1.

Пример входа	**Пример выхода**
8 19 20 3 5 1 15 7 17 12	1 3 15

Анализ алгоритма

Пусть a_0, a_1, …, a_{n-1} – входной массив. Переберем все возможные индексы k = 0, 1, …, n – 3. Далее для каждого значения k найдем такую пару индексов (i, j) что $i > k$ и сумма $a_i + a_j = x - a_k$. Это можно сделать на отсортированном массиве с помощью техники двух указателей за линейное время.



Реализация алгоритма

Читаем входные данные.

```
cin >> n >> x;
v.resize(n);
for (i = 0; i < n; i++)
  cin >> v[i];
```

Сортируем входной массив.

```
sort(v.begin(), v.end());
```

Перебираем значения $k = 0, 1, \ldots, n - 3$.

```
for (k = 0; k < n - 2; k++)
{
```

Ищем пару индексов (i, j) что $i > k$ и сумма $v_i + v_j = x - v_k$. Инициализируем индексы i и j.

```
  i = k + 1; j = n - 1;
  s = x - v[k];
```

Используем технику двух указателей для поиска искомой пары (i, j).

```
  while (i < j)
  {
    if (v[i] + v[j] == s)
    {
```

Если пара (i, j) найдена, то выводим искомую тройку чисел (v_k, v_i, v_j).

```
      printf("%d %d %d\n", v[k], v[i], v[j]);
      return 0;
    }
    if (v[i] + v[j] < s) i++; else j--;
  }
}
```

Если тройка чисел не найдена, выводим -1.

```
cout << -1 << endl;
```

9631. Соревнования последовательностей

Завтра Зия примет участие в соревновании последовательностей. Число $x \geq 0$ называется вершиной некоторой последовательности, если последовательность 1, 2, 3, ..., $x - 1$, x, $x - 1$, ..., 3, 2, 1 является подпоследовательностью данной последовательности. Силой каждой последовательности считается ее наибольшая вершина. Завтра все студенты пойдут на соревнование и победителем станет обладатель сильнейшей последовательности. Зия имеет последовательность a_1, a_2, a_3, ..., a_n. Он хочет захватить систему оценки соревнования и удалить из нее последовательности с большей силой чем у него самого. Однако, Зия не знает силу собственной последовательности, но очень хочет победить. Помогите ему посчитать силу собственной последовательности.

Вход. В первой строке задано количество n ($1 \leq n \leq 10^5$) чисел в последовательности Зии. В следующей строке записаны n целых чисел a_i ($1 \leq a_i \leq 10^5$) – элементы последовательности.

Выход. Выведите одно число – силу данной последовательности.

Пример входа 1
```
2
2 10
```

Пример выхода 1
```
0
```

Пример входа 2
```
3
1 2 3
```

Пример выхода 2
```
1
```

Пример входа 3
```
5
1 10 2 3 1
```

Пример выхода 3
```
2
```

Анализ алгоритма

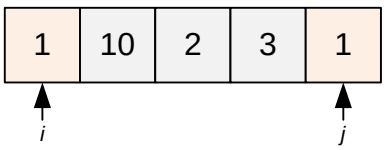
Пусть v – входной массив чисел. Инициализируем указатели $i = 0$ на начало массива, $j = n - 1$ на конец массива. В переменной k будем подсчитывать силу последовательности. Изначально установим $k = 1$.

Пока указатели i и j не встретятся, совершаем следующие действия:

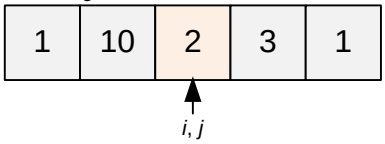
- Двигаем указатель i на одну позицию вправо если он не указывает на число k;
- Двигаем указатель j на одну позицию влево если он не указывает на число k;
- Если оба указателя указывают на число k, то увеличиваем k на единицу и сдвигаем каждый из указателей на одну позицию в соответствующем направлении;

Пример

Рассмотрим второй тест. Инициализируем указатели. Двигаем указатель i вправо, а j влево пока они не будут указывать на число 1.



Двигаем указатель i вправо, а j влево пока они не будут указывать на число 2.



Указатели встретились, останавливаем алгоритм. Ответом будет значение $k = 2$.

Реализация алгоритма

Читаем входные данные.

```
scanf("%d", &n);
```

```
v.resize(n);
for (i = 0; i < n; i++)
  scanf("%d", &v[i]);
```

Установим указатели на начало и конец массива.

```
int i = 0, j = v.size() - 1;
```

В переменной *k* подсчитываем силу последовательности.

```
k = 1;
while (i <= j)
{
```

Указатель *i* двигаем вправо пока не встретится число *k*.

```
  if (v[i] != k) i++;
```

Указатель *j* двигаем влево пока не встретится число *k*.

```
  if (v[j] != k) j--;
```

Если оба указателя указывают на число *k*, то увеличиваем *k* на единицу.

```
  if (v[i] == k && v[j] == k) k++;
}
```

Выводим ответ.

```
printf("%d\n", k - 1);
```

9632. Лучшая команда

Сегодня собрались n программистов. Каждый программист имеет рейтинг, показывающий его силу. Рейтинг – это целое число от 0 до 10^9. Ваш рейтинг как программиста равен m. Со всех собранных сегодня программистов Вы хотите выбрать двух в свою команду. Их следует выбрать так, чтобы сумма их рейтингов была максимальной, однако чтобы эта сумма не превосходила Ваш рейтинг, поскольку Вы хотите быть главой этой команды.

Вход. В первой строке заданы два целых числа: n ($2 \le n \le 10^5$) – количество программистов и m ($0 \le m \le 10^9$) – Ваш рейтинг. Во второй строке записаны n целых чисел $r_1, r_2, ..., r_n$ ($0 \le r_i \le 10^9$) – рейтинги программистов.

Выход. Выведите одно целое число – сумму рейтингов выбранных программистов или -1 если таких двух человек найти невозможно.

Пример входа 1
```
5 8
5 3 4 6 5
```

Пример выхода 1
```
8
```

Пример входа 2
```
7 19
8 4 25 1 20 5 12
```

Пример выхода 2
```
17
```

Пример входа 3
```
4 76
38 41 39 40
```

Пример выхода 3
```
-1
```

Анализ алгоритма

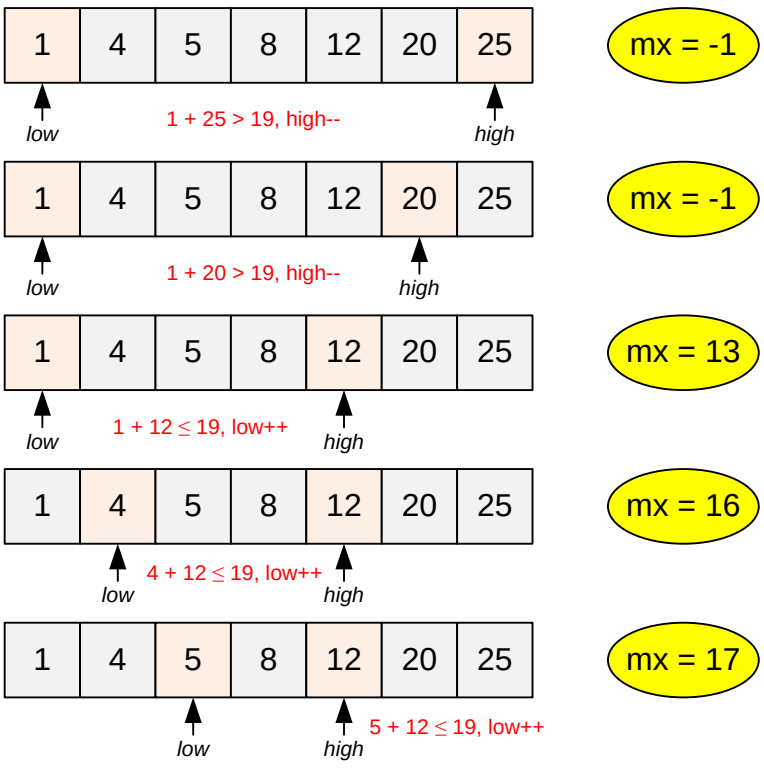
Отсортируем рейтинги программистов в массиве v. Поиск двух программистов с максимальным суммарным рейтингом будем искать при помощи техники двух указателей.

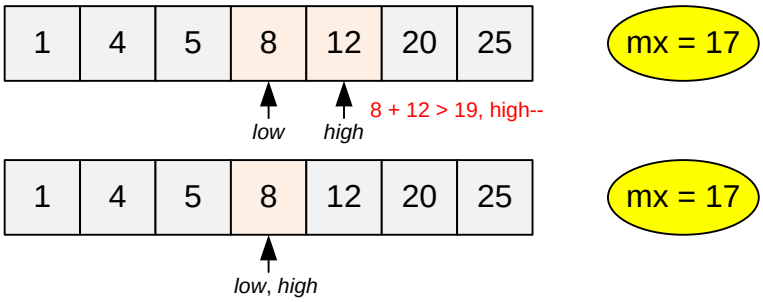
Инициализируем указатель *low* = 0 на начало массива, указатель *high* = *n* – 1 на конец массива.

Среди всех возможных сумм v[*low*] + v[*high*], не больших *m*, находим максимум. Если v[*low*] + v[*high*] ≤ *m*, то двигаем вперед *low*. Иначе двигаем назад *high*. Процесс передвижения указателей продолжаем, пока они не встретятся.

Пример

Рассмотрим второй тест. Отсортируем числа. Инициализируем указатели. Промоделируем работу алгоритма. Здесь *m* = 19: ищем два числа с максимальной суммой, не большей 19.





Реализация алгоритма
Читаем входные данные.

```
scanf("%d %d", &n, &m);
v.resize(n);
for (i = 0; i < n; i++)
  scanf("%d", &v[i]);
```

Сортируем рейтинги программистов.

```
sort(v.begin(), v.end());
```

Инициализируем указатели. В переменной *mx* вычисляем максимальную сумму двух элементов.

```
int mx = -1, low = 0, high = n - 1;
```

Указатель *low* двигаем вперед, указатель *high* двигаем назад.

```
while (low < high)
{
```

Среди всех возможных сумм v[*low*] + v[*high*], не больших *m*, находим максимум. Если v[*low*] + v[*high*] ≤ *m*, то двигаем вперед *low*. Иначе двигаем назад *high*.

```
  if (v[low] + v[high] <= m)
  {
    mx = max(mx, v[low] + v[high]);
    low++;
  }
  else high--;
}
```

Выводим ответ.

```
printf("%d\n", mx);
```

2910. ABCDEF

Имеется множество S целых чисел от -30000 до 30000 (включительно).

Найти общее количество шестерок $(a, b, c, d, e, f) : a, b, c, d, e, f \in S, d \neq 0$, удовлетворяющих равенству:

$$(a * b + c) / d - e = f$$

Вход. Первая строка содержит целое число n ($1 \leq n \leq 100$) – размер множества S. Элементы множества S заданы во второй строке. Все числа множества S различны.

Выход. Вывести общее количество искомых шестерок.

Пример входа	**Пример выхода**
3 5 7 10	10

Анализ алгоритма

Перепишем равенство в виде $a * b + c = (f + e) * d$. Занесем в массив m1 все возможные значения выражения $a * b + c$ ($a, b, c \in S$). Занесем в массив m2 все возможные значения выражения $(f + e) * d$ ($f, e, d \in S$). Отсортируем массивы m1 и m2. Найдем общие числа в двух массивах. Пусть число x содержится в m1 p раз, а в m2 q раз. Тогда количество искомых шестерок следует увеличить на $p * q$.

Реализация алгоритма

В массиве s храним элементы множества S. Объявим массивы m1 и m2. Поскольку имеется не более 100 различных значений для всех переменных, то различных значений $a * b + c$ и $(f + e) * d$ не более 100^3.

```
#define MAX 100
int s[101];
int m1[MAX*MAX*MAX+10], m2[MAX*MAX*MAX+10];
```

Основная часть программы. Читаем элементы множества S в массив s.

```
scanf("%d",&n);
for(i = 0; i < n; i++) scanf("%d",&s[i]);
```

Генерируем все значения выражения $a * b + c$ и заносим их в массив m1. Сортируем их.

```
for(i = a = 0; a < n; a++)
for(b = 0; b < n; b++)
for(c = 0; c < n; c++)
  m1[i++] = s[a] * s[b] + s[c];
sort(m1,m1+i);
```

Генерируем все значения выражения (*f* + *e*) * *d* и заносим их в массив m2. Помним, что значение *d* не равно 0. Сортируем их.

```
for(j = d = 0; d < n; d++)
if (s[d] != 0)
  for(e = 0; e < n; e++)
  for(f = 0; f < n; f++)
    m2[j++] = s[d] * (s[e] + s[f]);
sort(m2,m2+j);
```

Массив m1 содержит *i* чисел. Массив m2 содержит *j* чисел. Индекс *pi* движется по ячейкам массива m1, индекс *pj* движется по ячейкам массива m2. За время O(*i* + *j*) находим общие числа массивов и количество их вхождений в m1 и m2. Если некоторое число содержится в m1 *qi* – *pi* раз, а в m2 *qj* – *pj* раз, то количество искомых шестерок *res* увеличиваем на (*qi* – *pi*) * (*qj* – *pj*).

```
for(res = pi = pj = 0; pi < i && pj < j;)
{
  if (m1[pi] < m2[pj]) pi++; else
  if (m1[pi] > m2[pj]) pj++; else
  {
    qi = pi; qj = pj;
    while((m1[qi] == m1[pi]) && (qi < i)) qi++;
    while((m2[qj] == m2[pj]) && (qj < j)) qj++;
    res += (qi - pi) * (qj - pj);
    if ((qi == i) || (qj == j)) break;
    pi = qi; pj = qj;
  }
}
```

Выводим найденное количество шестерок *res*.

```
printf("%d\n",res);
```