

27 задачи

(№ 2666) Имеется набор данных, состоящий из положительных целых чисел, каждое из которых не превышает 1000. Требуется найти для этой последовательности контрольное значение – наибольшее число R, удовлетворяющее следующим условиям:

- R – произведение двух различных переданных элементов последовательности («различные» означает, что не рассматриваются квадраты переданных чисел, произведения различных, но равных по величине элементов допускаются);
- R делится на 6.

Вводим наше число  $a_j$ . Затем проверяем его на кратность 6, 3, 2. Если наше число кратно 6, то мы берем для него максимум из префикса -  $a_i$ . Если кратно 2, то берем для него максимум, кратный 3, (если одно число кратно 2, а другое 3, то их произведение будет кратно 6) Аналогично, берем максимум для чисел, кратных 3.

После взятия максимума проверяем то, является ли произведение  $a_i * a_j$  больше нашего имеющегося ответа( учитывая, существует ли взятый максимум) Далее обновляем наши максимумы.

В итоге, в переменной R получим ответ на задачу.

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      int n, max = 0, max6 = 0, max3 = 0, max2 = 0, R = 0, ai, aj, i;
6      ifstream in("input.txt");
7      in >> n;
8      for(i = 0; i < n; i++)
9      {
10         in >> aj;
11         if(aj % 6 == 0)
12         {
13             ai = max;
14         }
15         else
16         {
17             if(aj % 2 == 0)
18             {
19                 ai = max3;
20             }
21             else
22             {
23                 if(aj % 3 == 0)
24                 {
25                     ai = max2;
26                 }
27                 else
28                 {
29                     ai = max6;
30                 }
31             }
32         }
33         if(ai * aj >= R && ai != 0) R = ai * aj;
34         if(aj > max) max = aj;
35         if(aj % 6 == 0 && aj > max6) max6 = aj;
36         if(aj % 3 == 0 && aj > max3) max3 = aj;
37         if(aj % 2 == 0 && aj > max2) max2 = aj;
38     }
39     cout << R;
40 }
```

(№ 2667) Имеется набор данных, состоящий из положительных целых чисел, каждое из которых не превышает 1000. Требуется найти для этой последовательности контрольное значение – наибольшее число  $R$ , удовлетворяющее следующим условиям:

- $R$  – произведение двух различных переданных элементов последовательности («различные» означает, что не рассматриваются квадраты переданных чисел, произведения различных, но равных по величине элементов допускаются);
- $R$  делится на 7 и не делится на 49.

Для начала, вводим наше число  $a_j$ , после этого проверяем его на кратность 49, 7. Если наше число кратно 49, то мы не берем для него максимум из всех взятых чисел (префикс  $a_j$ )  $a_i$  по условию. Если кратно 7, то берем для него максимум, не кратный 7, т.к. если одно число кратно 7, а другое не 7, то их произведение будет кратно 7. Аналогично, берем максимум для чисел не кратных ни 49, ни 7.

После взятия нужного максимума проверяем то, является ли произведение  $a_i$  и  $a_j$  больше нашего промежуточного ответа, при этом, учитывая, существует ли взятый максимум. Далее обновляем наши максимумы в зависимости от кратности введенного  $a_j$ .

В итоге, в переменной  $R$  получим ответ на нашу задачу.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, i, max7 = 0, maxn7 = 0, R = 1, ai, aj;
    cin >> n;
    for(i = 0; i < n; i++)
    {
        cin >> aj;
        if(aj % 49 == 0) continue;
        if(aj % 7 == 0) ai = maxn7;
        else
        {
            ai = max7;
        }
        if(ai != 0 && ai * aj > R) R = ai * aj;
        if(aj % 7 != 0 && aj > maxn7) maxn7 = aj;
        if(aj % 7 == 0 && aj % 49 != 0 && aj > max7) max7 = aj;
    }
    cout << R;
}
```

(№ 2672) (Д.В. Богданов) Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов  $(a_i, a_j)$  этого набора, в которых  $1 \leq i < j \leq N$  и произведение элементов кратно 6.

Для начала, вводим наше число  $a_j$ , после этого проверяем его на кратность 6, если оно кратно, то мы увеличиваем кол-во делителей, кратных 2 ( $del2$ ) и кратных 3 ( $del3$ ), и увеличиваем счетчик( $count$ ) на кол-во введенных до этого числа чисел(префикс), потому что в таком случае пару мы можем составить с каждым из введенных до нашего  $a_j$ .

Далее проверяем, если  $a_j$  кратно 3, то увеличиваем кол-во делителей кратных 3 ( $del3$ ) и увеличиваем счетчик( $count$ ) на кол-во делителей, кратных 2 ( $del2$ ), т.к. в таком условии произведение будет кратно 6. Аналогично делаем с числами, кратными 2.

В итоге, в счетчике ( $count$ ) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, i, count = 0, ost, aj, del2 = 0, del3 = 0, del6 = 0;
    cin >> n;
    for(i = 0; i < n; i++)
    {
        cin >> aj;
        if(aj % 6 == 0)
        {
            del2++;
            del3++;
            count += i;
        }
        else
        {
            if(aj % 3 == 0)
            {
                count += del2;
                del3++;
            }
            if(aj % 2 == 0)
            {
                count += del3;
                del2++;
            }
        }
    }
    cout << count;
}
```

(№ 2674) (Д.В. Богданов) Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов  $(a_i, a_j)$  этого набора, в которых  $1 \leq i < j \leq N$  и сумма элементов кратна 12.

Для начала создадим вектор, в котором будут лежать кол-во чисел в префиксе, имеющих данный остаток при делении на 12. После вводим наше число ( $a_j$ ), и в переменную *ost* записываем его остаток при делении на 12. После этого увеличиваем счетчик(*count*) на кол-во чисел, которые имеют в сумме с этим остатком(*ost*) 12, т. к. при таком условии сумма двух чисел кратна 12. В конце увеличиваем кол-во чисел с данным остатком (*v[ost]*) на 1.

В итоге, в счетчике (*count*) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, count = 0, i, ai, aj, ost;
    cin >> n;
    vector <int> v(12, 0);
    for(i = 0; i < n; i++)
    {
        cin >> aj;
        ost = aj % 12;
        count += v[(12 - ost) % 12];
        v[ost]++;
    }
    cout << count;
}
```

(№ 2676) (А. Жуков) Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов  $(a_i, a_j)$  этого набора, в которых  $1 \leq i < j \leq N$ , сумма элементов нечётна, а произведение делится на 13.

Сначала, вводим наше число  $a_j$  и проверяем его на кратность 13. Если оно кратно, то мы проверяем его на кратность 2 и если оно кратно то в пару ему мы можем поставить ему любое нечетное число на префиксе, потому что тогда сумма элементов будет нечетна, а произведение кратно 13, значит, увеличиваем счетчик count на кол-во всех нечетных чисел (deln2) на префиксе.

После увеличиваем кол-во числе кратных 13 и 2 одновременно (del13k2) и кол-во всех четных чисел на префиксе (del2). Если число нечетное, то делаем все аналогично, но наоборот (del2, del13nk2, deln2).

Аналогичные действия проводим с числом, если оно не кратно 13, с условием того, что del13k2 и del13nk2 мы не трогаем.

В итоге, в счетчике (count) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, i, aj, count = 0, del13k2 = 0, del13nk2 = 0, del2 = 0, deln2 = 0;
    cin >> n;
    for(i = 0; i < n; i++)
    {
        cin >> aj;
        if(aj % 13 == 0)
        {
            if(aj % 2 == 0)
            {
                count += deln2;
                del13k2++;
                del2++;
            }
            else
            {
                count += del2;
                del13nk2++;
                deln2++;
            }
        }
        else
        {
            if(aj % 2 == 0)
            {
                count += del13nk2;
                del2++;
            }
            else
            {
                count += del13k2;
                deln2++;
            }
        }
    }
    cout << count;
}
```



(№ 2668) Имеется набор данных, состоящий из положительных целых чисел, каждое из которых не превышает 1000. Они представляют собой результаты измерений, выполняемых прибором с интервалом 1 минута. Требуется найти для этой последовательности контрольное значение – наименьшую сумму квадратов двух результатов измерений, выполненных с интервалом не менее, чем в 5 минут.

Смысл в том, чтобы расстояние между числом и его префиксом было  $\geq 5$ . Вводим очередь и записываем в нее первые 5 чисел, тогда ее первое число `q.front()` будет входить в префикс.

Дальше вводим `aj` и сравниваем `q.front()` с префиксным минимумом, т.к. сумма квадратов будет минимальна, если сами числа минимальны.

После этого сравниваем сумму квадратов `aj` и `q.front()` с минимальной суммой (`smin`) и заменяем, если сумма меньше. Дальше удаляем первый элемент очереди и в конец кладем текущий `aj`. В итоге, в минимальной сумме (`smin`) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, m = 5, i, smin = 10e10, min = 10001, aj;
    cin >> n;
    queue <int> q;
    for(i = 0; i < 5; i++)
    {
        cin >> aj;
        q.push(aj);
    }
    for(i = 5; i < n; i++)
    {
        cin >> aj;
        if(q.front() < min) min = q.front();
        if(aj * aj + min * min < smin)
        {
            smin = aj * aj + min * min;
        }
        q.pop();
        q.push(aj);
    }
    cout << smin;
}
```

(№ 2669) На спутнике «Восход» установлен прибор, предназначенный для измерения солнечной активности. Каждую минуту прибор передает по каналу связи неотрицательное целое число – количество энергии солнечного излучения, полученной за последнюю минуту, измеренное в условных единицах. Временем, в течение которого происходит передача, можно пренебречь. Необходимо найти в заданной серии показаний прибора минимальное нечётное произведение двух показаний, между моментами передачи которых прошло не менее 6 минут. Если получить такое произведение не удаётся, ответ считается равным «-1».

Смысл в том, чтобы расстояние между числом и его префиксом было  $\geq 6$ . Вводим очередь и записываем в нее первые 6 чисел, тогда ее первое число `q.front()` будет входить в префикс.

Дальше вводим `aj` и сравниваем `q.front()` с префиксным минимумом, т.к. произведение будет минимально, если сами числа минимальны. После этого сравниваем произведение `aj` и `q.front()` с минимальным произведением (`smin`) и заменяем, если сумма меньше.

Дальше проверяем произведение на нечетность, если нам подходит, то заменяем (`ans`).

Дальше удаляем первый элемент очереди и в конец кладем текущий `aj`.

В итоге, в минимальной сумме (`ans`) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, m = 5, i, smin = 10e10, min = 10001, aj, ans;
    cin >> n;
    queue <int> q;
    for(i = 0; i < 6; i++)
    {
        cin >> aj;
        q.push(aj);
    }
    for(i = 6; i < n; i++)
    {
        cin >> aj;
        if(q.front() < min) min = q.front();
        if(aj * min < smin)
        {
            smin = aj * min;
        }
        if(smin % 2 == 1) ans = smin;
        q.pop();
        q.push(aj);
    }
    cout << ans;
}
```



(№ 2673) Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов  $(a_i, a_j)$  этого набора, в которых  $1 \leq i + 7 \leq j \leq N$  и произведение элементов кратно 14.

Смысл в том, чтобы расстояние между числом и его префиксом было  $\geq 7$ . Вводим очередь и записываем в нее первые 7 чисел, тогда ее первое число `q.front()` будет входить в префикс.

Дальше вводим  $a_j$  и проверяем `q.front()` на кратность 14/7/2 и увеличиваем соответствующее кол-во делителей (`del14`, `del2`, `del7`). Это будет кол-во соответствующих делителей на префиксе.

После этого проверяем его на кратность 14, если оно кратно, то мы увеличиваем счетчик (`count`) на кол-во введенных до этого числа чисел(префикс)( $i - 6$  (т.к. расстояние  $\geq 7$ )), потому что в таком случае пару мы можем составить с каждым из введенных до нашего  $a_j$ . Далее проверяем, если  $a_j$  кратно 7, то увеличиваем счетчик(`count`) на кол-во делителей, кратных 2 (`del2`), т.к. в таком условии произведение будет кратно 14. Аналогично делаем с числами, кратными 2.

Для чисел не кратных ни 14, ни 2, ни 7 имеем отдельный `del14` и если  $a_j$  таково, то увеличиваем счетчик(`count`) на `del14`.

Дальше удаляем первый элемент очереди и в конец кладем текущий  $a_j$ .

В итоге, в счетчике (`count`) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, i, aj, count = 0, del7 = 0, del2 = 0, del14 = 0;
    cin >> n;
    queue <int> q;
    for(i = 0; i < 7; i++)
    {
        cin >> aj;
        q.push(aj);
    }
    for(i = 7; i < n; i++)
    {
        cin >> aj;
        if(q.front() % 14 == 0)
        {
            del14++;
            del2++;
            del7++;
        }
        else
        {
            if(q.front() % 7 == 0)
            {
                del7++;
            }
            if(q.front() % 2 == 0)
            {
                del2++;
            }
        }

        if(aj % 14 == 0)
        {
            count += i - 6;
        }
        else
        {
            if(aj % 7 == 0)
            {
                count += del2;
            }
            if(aj % 2 == 0)
            {
                count += del7;
            }
            if(aj % 7 != 0 && aj % 2 != 0)
            {
                count += del14;
            }
        }
        q.pop();
        q.push(aj);
    }
    cout << count;
```

(№ 2675) Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов  $(a_i, a_j)$  этого набора, в которых  $1 \leq i+5 \leq j \leq N$  и сумма элементов кратна 14.

Для начала создадим вектор, в котором будут лежать кол-во чисел в префиксе, имеющих данный остаток при делении на 14. Смысл в том, чтобы расстояние между числом и его префиксом было  $\geq 5$ . Вводим очередь и записываем в нее первые 5 чисел, тогда ее первое число `q.front()` будет входить в префикс.

После вводим наше число (`aj`), и в переменную `ostq` записываем остаток `q.front()` при делении на 14. После этого увеличиваем `v[ostq]` на 1.

Далее в переменную `ostaj` записываем остаток `aj` при делении на 14.

После этого увеличиваем счетчик(`count`) на кол-во чисел, которые имеют в сумме с этим остатком(`ost`) 14, т.к. при таком условии сумма двух чисел кратна 14.

Дальше удаляем первый элемент очереди и в конец кладем текущий `aj`.

В итоге, в счетчике (`count`) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, i, aj, count = 0, ostq, ostaj;
    cin >> n;
    queue<int> q;
    vector<int> v(14, 0);
    for(i = 0; i < 5; i++)
    {
        cin >> aj;
        q.push(aj);
    }
    for(i = 5; i < n; i++)
    {
        cin >> aj;
        ostq = q.front() % 14;
        v[ostq]++;
        ostaj = aj % 14;
        count += v[(14 - ostaj) % 14];
        q.pop();
        q.push(aj);
    }
    cout << count;
}
```

(№ 2677) (А. Жуков) Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов  $(a_i, a_j)$  этого набора, в которых  $1 \leq i+5 \leq j \leq N$ , сумма элементов нечётна, а произведение делится на 13.

Смысл в том, чтобы расстояние между числом и его префиксом было  $\geq 5$ . Вводим очередь и записываем в нее первые 5 чисел, тогда ее первое число `q.front()` будет входить в префикс.

Дальше считываем все остальные числа  $a_j$  и проверяем `q.front()` на кратность 13. Если оно кратно, то мы проверяем его на кратность 2 (чтобы понять зачем см. №2676).

После увеличиваем кол-во числе кратных 13 и 2 одновременно (`del13k2`) и кол-во всех четных чисел на префиксе (`delk2`). Если число нечетное, то делаем все аналогично, но с точностью наоборот (в плане остатков (`delk2`, `del13nk2`, `delnk2`)).

Аналогичные итерации проводим с числом, если оно не кратно 13, с условием того, что `del13k2` и `del13nk2` мы не трогаем.

Дальше смотрим наше введенное  $a_j$ . Если оно кратно, то мы проверяем его на кратность 2 и если оно кратно то в пару ему мы можем поставить ему любое нечетное число на префиксе, потому что тогда сумма элементов будет нечетна, а произведение кратно 13, значит, увеличиваем счетчик `count` на кол-во всех нечетных чисел (`delnk2`) на префиксе. Если число нечетное, то делаем все аналогично, но с точностью наоборот (в плане остатков и делителей).

Аналогичные итерации проводим с числом, если оно не кратно 13.

В итоге, в счетчике (`count`) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, i, aj, count = 0, del13nk2 = 0, del13k2 = 0, delnk2 = 0, delk2 = 0;
    cin >> n;
    queue<int> q;
    for(i = 0; i < 5; i++)
    {
        cin >> aj;
        q.push(aj);
    }
    for(i = 5; i < n; i++)
    {
        cin >> aj;
        if(q.front() % 13 == 0)
        {
            if(q.front() % 2 == 0)
            {
                delk2++;
                del13k2++;
            }
            else
            {
                delnk2++;
                del13nk2++;
            }
        }
        else
        {
            if(q.front() % 2 == 0) delk2++;
            else delnk2++;
        }
        if(aj % 13 == 0)
        {
            if(aj % 2 == 0) count += delnk2;
            else count += delk2;
        }
        else
        {
            if(aj % 2 == 0) count += del13nk2;
            else count += del13k2;
        }
        q.pop();
        q.push(aj);
    }
    cout << count;
}
```

(№ 2678) (А. Жуков) Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов  $(a_i, a_j)$  этого набора, в которых  $1 \leq i < j \leq N$ , сумма элементов нечётна, произведение делится на 13, а **номера элементов в последовательности отличаются МЕНЕЕ, чем на 5**.

Смысл в том, чтобы расстояние между числом и его префиксом было  $< 5$ . Вводим deck, в котором будет находиться наш префикс. Удобно брать его, т.к. в нем мы можем обращаться к элементу по его номеру, к тому же он является двусторонней очередью.

Итак, сначала мы обнуляем все счетчики делителей ( $k2\_13$  - кратные 13 и четные;  $k1\_13$  - кратные 13 и нечетные;  $k2$  - четные;  $k1$  - и нечетные), чтобы заново считать их кол-во на уже данном префиксе.

Дальше, проходя по нашему deck'у (префиксу) увеличиваем соответствующие счетчики, исходя от числа  $dq[j]$ .

После вводим текущее число  $aj$  и проверяем его на нужные нам кратности, после чего увеличиваем наш конечный счетчик (count) на соответствующий счетчик делителей.

После этого проверяем размер нашего deck'а, и если он равен 4, то удаляем из него первый элемент, т.к. он в префикс больше входить не будет.

И добавляем в deck текущее число  $aj$ .

В итоге, в счетчике (count) получаем ответ.

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n, i, aj, k2_13 = 0, k1_13 = 0, k1 = 0, k2 = 0, j, count = 0;
    cin >> n;
    deque<int> dq;
    for(i = 0; i < n; i++)
    {
        k2_13 = 0; k1_13 = 0; k1 = 0; k2 = 0;
        for(j = 0; j < dq.size(); j++)
        {
            if(dq[j] % 26 == 0) k2_13++;
            if(dq[j] % 2 == 0) k2++;
            if(dq[j] % 13 == 0 && dq[j] % 2 == 1) k1_13++;
            if(dq[j] % 2 == 1) k1++;
        }
        cin >> aj;
        if(aj % 26 == 0) count += k1;
        if(aj % 2 == 0 && aj % 13 != 0) count += k1_13;
        if(aj % 13 == 0 && aj % 2 == 1) count += k2;
        if(aj % 2 == 1 && aj % 13 != 0) count += k2_13;
        if(dq.size() == 4) dq.pop_front();
        dq.push_back(aj);
    }
    cout << count;
}
```

Спасибо

