Университетские субботы-2021. Информатика.

Юлия Борисовна Буркатовская, ОИТ

Обработка больших данных

Задания с сайтов

https://inf-ege.sdamgia.ru

https://kpolyakov.spb.ru/school/ege.htm

Задание 27. Программирование

Типы заданий

• Поиск пары различных значений по заданным условиям
• Поиск пары значений с ограничениями по расстоянию
• Выбор подмножества чисел с условиями
• Подсчет количества пар чисел с условиями
• Подсчет количества различных чисел

Эффективное решение:

1. По времени: алгоритм линейный относительно размерности входных данных.
2. По памяти: практически это означает, что все данные в массиве или списке хранить не нужно.

https://inf-ege.sdamgia.ru

Пример 1. Задание 27 № 27891

Последовательность натуральных чисел характеризуется числом X — наибольшим числом, кратным 14 и являющимся произведением двух элементов последовательности с различными номерами. Гарантируется, что хотя бы одно такое произведение в последовательности есть.

Входные данные.

Даны два входных файла (файл *A* и файл *B*), каждый из которых содержит в первой строке количество чисел N ($1 \le N \le 100000$). В каждой из последующих N строк записано одно натуральное число, не превышающее 1000.

Пример организации исходных данных во входном файле:

5
40
1000
7
28
55

Пример выходных данных для приведённого выше примера входных данных:

28000

В ответе укажите два числа: сначала значение искомого произведения для файла *A*, затем для файла *B*.

Неэффективное решение. Для файла A подходят переборные алгоритмы с сохранением всех данных в массив и просмотром всех пар элементов.

Идея

- Перебор всех пар, поиск максимума с условиями.
- Каждый элемент образует пару со всеми элементами, большими по номеру. Сохраняем все данные в массив A, рассматриваем пары$(A[i], A[j])$, для которых $i < j$.
- Для предотвращения выхода за границу массива требуется выполнение условия

$$0 \le i < j \le n-1,$$
$$i = 0, \dots, n-2$$
$$j = i+1, \dots, n-1$$

- Условие – при поиске максимума в обновлении максимума участвуют только пары, произведение элементов в которых делится на 14.

Основные ошибки

- Выход за границу массива
- Неверная инициализация максимума (минимума)
- Неверно перебираются пары (например, рассматриваются пары $(A[i], A[i])$.

Pascal

```
var pmax,p,j,i,n: longint;
A:array[0..9999] of longint;
f:text;
begin
pmax:=0;
  assign(f,'A.txt');
  reset(f);
readln(f,n);
  for i:=0 to n-1 do
readln(f,A[i]);
  for i:=0 to n-2 do
    for j:=i+1 to n-1 do
    begin
      p:=A[i]*A[j];
      if (p mod 14 = 0) then
        if (p>pmax) then
          pmax:=p;
    end;
writeln(pmax);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{ long pmax=0,p,j,i,n;
  ifstream f("A.dat");
  f>> n;
  int A[n];
  for(i=0;i<n;i++)
    f>>A[i];
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
    {
    p=A[i]*A[j];
      if((p%14==0)&&(p>pmax))
         pmax=p;
    }
cout<<pmax;
return 0;
}
```

Python

```
f=open('A.dat','r')
n=int(f.readline())
A=[]
for I in range(n):
x=int(f.readline())
A.append(x)
pmax=0
for i in range(n-1):
    for j in range(i+1,n):
        p=A[i]*A[j]
        if p%14==0 and p>pmax:
            pmax=p
print(pmax)
f.close()
```

Эффективное решение.

- Максимальное произведение состоит из максимальных множителей.
- Произведение xy кратно 14 в двух случаях:
  - x кратно 14, y – любое число (или наоборот);
  - x кратно 7, y кратно 2 (или наоборот).

Идея 1. Пусть к нам пришло очередное число, скажем, x. Мы можем умножать его на числа, пришедшие до него. При этом:

- Если x кратно 14, его можно умножить на любое число;
- Если x кратно 7, но не кратно 2, его можно умножать на число, кратное 2;
- Если x кратно 2, но не кратно 7, его можно умножать на число, кратное 7;
- Если x не кратно ни 2, ни 7, его можно умножать на число, кратное 14;
- Для получения максимального произведения x умножаем на максимальное из подходящих чисел.
- Все предыдущие числа хранить не нужно, только максимальные.
- Если полученное произведение больше текущего максимального произведения, обновляем максимальное произведение.
- После обновления произведения обновляем максимумы в зависимости от вида x.

Смысл переменных:

- x −текущее число
- p −текущее произведение
- $pmax$ −максимальное произведение
- $max14$ −максимальное число, кратное 14
- $max7$ −максимальное число, кратное 7
- $max2$ −максимальное число, кратное 2
- $maxa$ −максимальное число, без условий

x	p	pmax	max14	max7	max2	maxa	произведение	обновляем
	0	0	0	0	0	0		
7	0	0	0	7	0	7	p=x*max2	max7,maxa
14	98	98	14	14	14	14	p=x*maxa	все
11	154	154	14	14	14	14	p=x*max14	maxa
20	280	280	14	14	20	20	p=x*max7	max2,maxa
7	140	280	14	14	20	20	p=x*max2	max7,maxa
21	420	420	14	21	20	21	p=x*max2	max7,maxa

Основные ошибки:

- Обновление произведения после обновления максимумов, это позволяет получить произведение числа на него же
- Путаница в условных операторах

Приведенные программы не самые короткие, но имеют достаточно прозрачный код. При опыте в программировании можно, конечно, написать все короче.

Pascal

```
var pmax,p,i,n: longint;
x,max14,max7,max2,maxa:integer;
f:text;
begin
pmax:=0;
  assign(f,'A.txt');
  reset(f);
pmax:=0;
  max14:=0;
  max7:=0;
  max2:=0;
maxa:=0;
readln(f,n);
  for i:=0 to n-1 do begin
readln(f,x);
  if(x mod 14 = 0) then begin
      p:=x*maxa;
      if(x>max14) then max14:=x;
      if(x>max7) then max7:=x;
      if(x>max2) then max2:=x;
    end;
    if(x mod 7 = 0) then
      if(x mod 2>0) then begin
        p:=x*max2;
        if(x>max7) then max7:=x;
      end;
    if(x mod 2 = 0) then
      if(x mod 7>0) then begin
        p:=x*max7;
        if(x>max2) then max7:=x;
      end;
    if(x mod 7> 0) then
      if(x mod 2>0) then
        p:=x*max14;
    if(x>maxa) then maxa:=x;
    if(p>pmax) then pmax:=p;
end;
writeln(pmax);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
intx,max14,max7,max2,maxa;
  longp,pmax,n,i;
ifstream f("A.txt");
max14=max7=max2=maxa=pmax=0;
  f>>n;
for(i=0;i<n;i++)
{
    f>>x;
    if(x%14==0)
{
        p=x*maxa;
        if(x>max14) max14=x;
        if(x>max7) max7=x;
        if(x>max2) max2=x;
}
    if((x%7==0)&&(x%2!=0))
{
p=x*max2;
if(x>max7) max7=x;
}
    if((x%2==0)&&(x%7!=0))
{
p=x*max7;
if(x>max2) max2=x;
}
    if((x%2!=0)&&(x%7!=0))
p=x*max14;
if(x>maxa) maxa=x;
if(p>pmax) pmax=p;
}
cout<<pmax;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
max14=0
max7=0
max2=0
maxa=0
pmax=0
for i in range(n):
    x=int(f.readline())
    if(x%14==0):
        p=x*maxa
        if(max14<x):
            max14=x
        if(max7<x):
            max7=x
        if(max2<x):
            max2=x
    if (x%7==0) and (x%2!=0):
        p=x*max2
        if(max7<x):
            max7=x
    if (x%7!=0) and (x%2==0):
        p=x*max7
        if(max2<x):
            max2=x
    if (x%7!=0) and (x%2!=0):
        p=x*max14
    if(maxa<x):
        maxa=x
    if(p>pmax):
        pmax=p
print(pmax)
f.close()
```

Чуть более сложно, по-другому используется условный оператор. Приведен только основной цикл программы.

Pascal

```
for i:=0 to n-1 do begin
readln(f,x);
  if(x mod 14 = 0) then begin
      p:=x*maxa;
      if(x>max14) then max14:=x;
      if(x>max7) then max7:=x;
      if(x>max2) then max2:=x;
    end
    else if(x mod 7 = 0) then
      begin
        p:=x*max2;
        if(x>max7) then max7:=x;
      end
    else if(x mod 2 = 0) then
      begin
        p:=x*max7;
        if(x>max2) then max7:=x;
      end
    else
        p:=x*max14;
    if(x>maxa) then maxa:=x;
    if(p>pmax) then pmax:=p;
  end;
```

C++

```
for(i=0;i<n;i++)
{ f>>x;
    if(x%14==0)
{
        p=x*maxa;
        if(x>max14) max14=x;
        if(x>max7) max7=x;
        if(x>max2) max2=x;
}
    else if(x%7==0)
{
p=x*max2;
if(x>max7) max7=x;
}
    else if(x%2==0)
{
p=x*max7;
if(x>max2) max2=x;
}
else p=x*max14;
if(x>maxa) maxa=x;
if(p>pmax) pmax=p;
}
```

Python

```
for i in range(n):
    a=int(f.readline())
    if(a%14==0):
        p=a*maxa
        if(max14<a):
            max14=a
        if(max7<a):
            max7=a
        if(max2<a):
            max2=a
    elif (a%7==0):
        p=a*max2
        if(max7<a):
            max7=a
    elif (a%2==0):
        p=a*max7
        if(max2<a):
            max2=a
    else:
        p=a*max14
    if(maxa<a):
        maxa=a
    if(p>pmax):
        pmax=p
```

Идея 2. Максимальное произведение подсчитаем в конце. Это произведение максимальных множителей следующего вида:

- Один множитель кратен 14, второй – любой (эти числа должны прийти в разное время!);
- Один множитель кратен 2, но не кратен 7, второй кратен 7, но не кратен 2 (они разные по определению).

Смысл переменных:

- x −текущее число
- $max14$ −максимальное число, кратное 14
- $max7$ −максимальное число, кратное 7, но не кратное 2;
- $max2$ −максимальное число, кратное 2, но не кратное 7;
- $maxa$ −максимальное число, без условий, но оно не то же самое, что текущее $max14$. Они могут быть равны, но тогда эти равные числа должны были прийти в разное время, и они максимальны среди чисел, поступивших до текущего момента.

Итоговое максимальное произведение находится как

$$pmax = \max\{max14 * maxa, max2 * max7\}$$

x	max14	max7	max2	maxa	примечание
	0	0	0	0	
7	0	7	0	7	Обновляем max7, maxa
14	14	7	0	7	Обновляем max14
11	14	7	0	11	Обновляем maxa
20	14	7	20	20	Обновляем max2, maxa
7	14	7	20	20	
21	14	21	20	21	Обновляем max7, maxa
28	28	21	20	21	Обновляем max14, maxa нет(текущее max14<maxa)
70	70	21	20	28	Обновляем max14, maxa (текущее max14>maxa)
42	70	21	20	28	Обновляем maxa

Если пришло число, кратное 14, и оно больше текущего $max14$, сравниваем $max14$ и $maxa$. Если $maxa < max14$, обновляем $maxa$, а уже потом $max14$, и обнуляем x. Последнее действие делаем потому, что, если это не сделать, то в конце цикла при обновлении $maxa$ этой переменной присвоится значение x. Можно использовать другие пути: прерывать цикл командой break, обновлять $maxa$ при любых условиях на x внутри условного оператора, и т.д.

Основные ошибки:

- Неверное обновление $max14$ и $maxa$, их нельзя заменять одновременно на одно и то же число.
- Путаница в условных операторах.

На мой взгляд, предыдущая идея проще в реализации.

Pascal

```
varpmax,i,n: longint;
x,max14,max7,max2,maxa:integer;
f:text;
begin
pmax:=0;
  assign(f,'A.txt');
  reset(f);
  max14:=0;
  max7:=0;
  max2:=0;
maxa:=0;
readln(f,n);
  for i:=0 to n-1 do begin
readln(f,x);
  if(x mod 14 = 0) then
      if(x>max14) then begin
        if(max14>maxa) then
maxa:=max14;
max14:=x;
        x:=0;
    end;
    if(x mod 7 = 0) then
      if(x mod 2>0) then
        if(x>max7) then max7:=x;
    if(x mod 2 = 0) then
      if(x mod 7>0) then
        if(x>max2) then max7:=x;
    if(x>maxa) then maxa:=x;
  end;
pmax:=max14*maxa;
  if(max2*max7>pmax) then
pmax:=max2*max7;
writeln(pmax);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
intx,max14,max7,max2,maxa;
  longpmax,n,i;
ifstream f("A.txt");
max14=max7=max2=maxa=0;
  f>>n;
for(i=0;i<n;i++)
{
    f>>x;
    if(x%14==0)
{
        if(x>max14)
        {
        if(maxa<max14)
        maxa=max14;
        max14=x;
        x=0;
        }
}
    if((x%7==0)&&(x%2!=0))
if(x>max7) max7=x;
    if((x%2==0)&&(x%7!=0))
if(x>max2) max2=x;
if(x>maxa) maxa=x;
}
pmax=max14*maxa;
  if(max2*max7>pmax)
pmax=max2*max7;
cout<<pmax;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
max14=0
max7=0
max2=0
maxa=0
for i in range(n):
    x=int(f.readline())
    if(x%14==0):
        if(max14<x):
            if(maxa<max14):
                maxa=max14
             max14=x
             x=0
    if (x%7==0) and (x%2!=0):
        if(max7<x):
            max7=x
    if (x%7!=0) and (x%2==0):
         if(max2<x):
             max2=x
    if(maxa<x):
        maxa=x
    pmax=max14*maxa
    if max7*max2>pmax:
        pmax=max7*max2
print(pmax)
f.close()
```

Пример 2.Задание 27 № 27989

На вход программы поступает последовательность из N целых положительных чисел, все числа в последовательности различны. Рассматриваются все пары различных элементов последовательности (элементы пары не обязаны стоять в последовательности рядом, порядок элементов в паре не важен). Необходимо определить количество пар, для которых произведение элементов делится на 26.

В первой строке входных данных задаётся количество чисел N ($1 \le N \le 1000$). В каждой из последующих N строк записано одно целое положительное число, не превышающее 10 000. В качестве результата программа должна напечатать одно число: количество пар, в которых произведение элементов кратно 26.

Входные данные.

Файл A

Файл B

Даны два входных файла (файл *A* и файл *B*), каждый из которых содержит в первой строке количество пар *N* ($1 \le N \le 100000$). В каждой из последующих *N* строк записано одно натуральное число, не превышающее 1000.

Пример организации исходных данных во входном файле:

4
2
6
13
39

Пример выходных данных для приведённого выше примера входных данных:

4

В ответе укажите два числа: сначала значение искомой суммы для файла *A*, затем для файла *B*.

Неэффективное решение.Перебираем все пары по тем же принципам, что и в предыдущей задаче; увеличиваем счетчик каждый раз, когда появляется пара с нужными условиями.

Pascal

```
vark,p,j,i,n: longint;
A:array[0..9999] of longint;
f:text;
begin
k:=0;
  assign(f,'A.txt');
  reset(f);
  readln(f,n);
  for i:=0 to n-1 do
    readln(f,A[i]);
  for i:=0 to n-2 do
    for j:=i+1 to n-1 do
    begin
      p:=A[i]*A[j];
      if (p mod 14 = 0) then
        k:=k+1;
    end;
writeln(k);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{ long k=0,j,i,n;
ifstream f("A.txt");
  f>>n;
int A[n];
  for(i=0;i<n;i++)
    f>>A[i];
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
      if((A[i]*A[j])%26==0)
        k++;
    }
cout<<k;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
A=[]
for i in range(n):
x=int(f.readline())
A.append(x)
k=0
for i in range(n-1):
    for j in range(i+1,n):
        if (A[i]*A[j])%26==0:
            k+=1
print(k)
f.close()
```

Эффективное решение.

- Произведение xy кратно 26 в двух случаях:
  - x кратно 26, y – любое число (или наоборот);
  - x кратно 13, y кратно 2 (или наоборот).

Идея 1. Пусть к нам пришло очередное число, скажем, x. Мы можем умножать его на числа, пришедшие до него. При этом:

- Если x кратно 26, его можно умножить на любое число, то есть, оно образует пару со всеми пришедшими до него числами;
- Если x кратно 13, но не кратно 2, его можно умножать на число, кратное 2, образуется столько пар, сколько на данный момент есть чисел, кратных 2;
- Если x кратно 2, но не кратно 13, его можно умножать на число, кратное 13, образуется столько пар, сколько на данный момент есть чисел, кратных 13;
- Если x не кратно ни 2, ни 13, его можно умножать на число, кратное 26, образуется столько пар, сколько на данный момент есть чисел, кратных 26;
- В специальный счетчик добавляем найденное количество пар, затем обновляем счетчики чисел с различными условиями.

Смысл переменных:

- i −номер числа
- x −текущее число
- $count$ −счетчик пар
- $k26$ −количество чисел, кратных 26
- $k13$ −количество чисел, кратных 13
- $k2$ −количество чисел, кратных 2

i	x	count	k26	k13	k2	количество пар	с чем образует пару	обновляем
		0	0	0	0			
0	13	0	0	1	0	k2=0		k13
1	20	1	0	1	1	k13=1	13	k2
2	12	2	0	1	2	k13=1	13	k2
3	26	5	1	2	3	i=3	13,20,12	k26,k13,k2
4	11	6	1	2	3	k26=1	26	
5	52	11	2	3	4	i=5	13,20,12,26,11	k26,k13,k2

Основные ошибки:

- Обновление счетчика пар после обновления счетчиков чисел
- Неверная организация счетчиков
- Путаница в условных операторах

Pascal

```
var count,k26,k13,k2,i,n: longint;
x:integer;
f:text;
begin
  assign(f,'A.txt');
  reset(f);
count:=0;
k26:=0;
k13:=0;
k2:=0;
readln(f,n);
  for i:=0 to n-1 do begin
readln(f,x);
  if(x mod 26 = 0) then begin
count:=count+i;
      k26:=k26+1;
      k13:=k13+1;
      k2:=k2+1;
    end
else if(x mod 13 = 0) then
begin
      count:=count+k2;
      k13:=k13+1;
    end
else if(x mod 2 = 0) then
    begin
      count:=count+k13;
      k2:=k2+1;
    end
    else count:=count+k26;
  end;
writeln(count);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
intx;
  long k26,k13,k2,n,i,count;
ifstream f("A.txt");
  f>>n;
  count=k26=k13=k2=0;
for(i=0;i<n;i++)
{
    f>>x;
    if(x%26==0)
{
        count+=i;
        k26++;
        k13++;
        k2++;
}
else if(x%13==0)
{
count+=k2;
      k13++;
}
else if(x%2==0)
{
      count+=k13;
      k2++;
}
    else count+=k26;
  }
cout<<count;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
k26=k13=k2=count=0
for i in range(n):
    x=int(f.readline())
    if(x%26==0):
        count+=i
        k26+=1
        k13+=1
        k2+=1
    elif (x%13==0):
        count+=k2
        k13+=1
    elif (x%2==0):
        count+=k13
        k2+=1
    else:
        count+=k26
print(count)
f.close()
```

Идея 2. Подсчитываем количество чисел, кратных 26, кратных только 13, кратных только 2, не кратных ни 2, ни 13. Общее количество пар подсчитываем в конце.

- Числа, кратные 26, образуют пары с числами из всех групп, в том числе друг с другом
- Числа, кратные только 13, образуют также пары с числами, кратными только 2

Смысл переменных:

- i −номер числа
- x −текущее число
- $count$ −счетчик пар
- $k26$ −количество чисел, кратных 26
- $k13$ −количество чисел, кратных 13 и не кратных 2
- $k2$ −количество чисел, кратных 2 и не кратных 13

Общее количество пар:

$$k13 * k2 + k26 * (n - k26) + k26 * (k26 - 1)/2$$

- $k13 * k2$ числа, кратные только 13, с числами, кратными только 2
- $k26 * (n - k26)$ числа, кратные 26, с числами, не кратными 26
- $k26 * (k26 - 1)/2$ числа, кратные 26, друг с другом

Поясним последнее слагаемое. Пусть у нас есть 4 числа, кратных 26.

- Первое число образует пары со вторым, третьим и четвертым (три пары)
- Второе число образует пары с третьим и четвертым, с первым уже рассмотрели (две пары)
- Третье число образует пару только с четвертым (одна пара).

В общем виде получаем формулу суммы арифметической прогрессии

$$(k26 - 1) + (k26 - 2) + \cdots + 1 = k26 * (k26 - 1)/2$$

Основные ошибки:

- Общая формула
- Неверная организация счетчиков (либо, как предложено, каждое число учитываем в не более чем одном счетчике, либо общая формула будет другой).
- Путаница в условных операторах

Pascal

```
varcount,k26,k13,k2,i,n: longint;
x:integer;
f:text;
begin
  assign(f,'A.txt');
  reset(f);
  k26:=0;
  k13:=0;
  k2:=0;
readln(f,n);
  for i:=0 to n-1 do begin
readln(f,x);
  if(x mod 26 = 0) then
      k26:=k26+1
    else if(x mod 13 = 0) then
      k13:=k13+1
    else if(x mod 2 = 0) then
      k2:=k2+1;
  end;
  count:=k2*k13+k26*(n-k26);
  count:=count+k26*(k26-1)div 2;
writeln(count);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
intx;
  long k26,k13,k2,n,i,count;
ifstream f("A.txt");
  f>>n;
  k26=k13=k2=0;
for(i=0;i<n;i++)
{
    f>>x;
    if(x%26==0)k26++;
    else if(x%13==0) k13++;
    else if(x%2==0) k2++;
  }
  count=k2*k13+k26*(n-k26);
  count+=k26*(k26-1)/2;
cout<< count;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
k26=k13=k2=0
for i in range(n):
    x=int(f.readline())
    if(x%26==0):
        k26+=1
    elif (x%13==0):
        k13+=1
    elif (x%2==0):
        k2+=1
count=k2*k13+k26*(n-k26)
count+=k26*(k26-1)//2
print(count)
f.close()
```

Пример 3. Задание 27 № 28128

По каналу связи передавались положительные целые числа, не превышающие 1000 — результаты измерений, полученных в ходе эксперимента (количество измерений N известно заранее, гарантируется, что $N>2$). После окончания эксперимента передаётся контрольное значение — наибольшее число R, удовлетворяющее следующим условиям.

1. R — сумма двух различных переданных элементов последовательности («различные» означает, что нельзя просто удваивать переданные числа, суммы различных, но равных по величине элементов допускаются).

2. R кратно 3.

3. Если в последовательности нет двух чисел, сумма которых кратна 3, контрольное значение считается равным 1.

В результате помех при передаче как сами числа, так и контрольное значение могут быть искажены.

Входные данные.

Файл A
Файл B

На вход программе в первой строке подаётся количество чисел N ($N>2$). В каждой из последующих N строк записано одно натуральное число, не превышающее 10 000.

Пример организации исходных данных во входном файле:

```
6
100
8
33
145
19
84
```

Пример выходных данных для приведённого выше примера входных данных:

```
153
```

В ответе укажите два числа: сначала контрольное для значений из файла *A*, затем из файла *B*.

Неэффективное решение. Для файла A подходят переборные алгоритмы с сохранением всех данных в массив и просмотром всех пар элементов, как в предыдущих задачах.

Pascal

```
var smax,s,j,i,n: longint;
A:array[0..9999] of longint;
f:text;
begin
smax:=1;
  assign(f,'A.txt');
  reset(f);
readln(f,n);
  for i:=0 to n-1 do
readln(f,A[i]);
  for i:=0 to n-2 do
    for j:=i+1 to n-1 do
    begin
      s:=A[i]+A[j];
      if (s mod 3 = 0) then
        if (s>smax) then
           smax:=s;
    end;
writeln(smax);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{ long j,i,n;
ifstream f("A.dat");
  f>> n;
int A[n],smax,s;
  for(i=0;i<n;i++)
    f>>A[i];
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
    {
      s=A[i]+A[j];
      if((s%3==0)&&(s>smax))
          smax=s;
    }
cout<<smax;
return 0;
}
```

Python

```
f=open('A.dat','r')
n=int(f.readline())
A=[]
for i in range(n):
x=int(f.readline())
A.append(x)
smax=1
for i in range(n-1):
    for j in range(i+1,n):
        s=A[i]+A[j]
        if s%3==0 and s>smax:
            smax=s
print(smax)
f.close()
```

Эффективное решение.

- Максимальная сумма состоит из максимальных слагаемых.
- Сумма $x + y$ кратна 3, если сумма их остатков от деления на 3 кратна 3 (равна 3 или 0). Примеры: 3+9, 11+1.

Идея 1. Пусть к нам пришло очередное число, скажем, x. Мы можем складывать его с числами, пришедшие до него. При этом:

- Если x кратно 3, его можно сложить лишь с числом, кратным 3 (остатки 0 и 0);
- Если x при делении на 3 имеет остаток 1, то его можно складывать с числами, которые при делении на 3 имеют остаток 2;
- Если x при делении на 3 имеет остаток 2, то его можно складывать с числами, которые при делении на 3 имеют остаток 1;
- Обобщим все сказанное: если при делении на 3 x имеет остаток p, то второе слагаемое имеет остаток q, вычисляемый по формуле (mod– операция взятия остатка):

$$q = (3 - p)\ mod\ 3$$

- Для получения максимальной суммы x складываем с максимальным из подходящих чисел (**если таковое есть**).
- Все предыдущие числа хранить не нужно, только максимальные.
- Если полученная сумма больше текущей максимальной суммы, обновляем максимальную сумму.
- После обновления суммы обновляем максимумы в зависимости от вида x.

Смысл переменных:

- x −текущее число
- p −остаток от деления x на 3
- q −остаток от деления на 3 числа, с которым складываем x (это $m[q]$)
- s −текущая сумма
- $smax$ −максимальная сумма
- $m[0]$ −максимальное число, кратное 3
- $m[1]$ −максимальное число с остатком 1
- $m[2]$ −максимальное число с остатком 2

x	p	q	s	smax	m[0]	m[1]	m[2]	примечание	сумма	обновляем
	0	0	0	1	0	0	0			
7	1	2	0	1	0	7	0	m[2]=0		m[1]
6	0	0	0	1	6	7	0	m[0]=0		m[0]
3	0	0	9	9	6	7	0	m[0]>0	x+m[0]	
20	2	1	27	27	6	7	20	m[1]>0	x+m[1]	m[2]
22	1	2	42	42	6	22	20	m[2]>0	x+m[2]	m[1]
2	2	1	24	42	6	22	20	m[1]>0	x+m[1]	

Основные ошибки:

- Обновление суммы после обновления максимумов, это позволяет получить сумму числа с ним же
- Путаница в условных операторах
- Не учитывают, что должно быть число, с которым можно сложить текущее, то есть надо проверять условие $m[q] > 0$ перед тем, как считать сумму $x + m[q]$. При поиске максимального произведения такой проблемы не возникает, поскольку максимальные числа инициализируются нулями, и произведение получается 0.При поиске минимального произведения проблема та же.

Pascal

```
vars,smax,i,n: longint;
x,p,q:integer;
m:array[0..2] of integer;
f:text;
begin
  assign(f,'A.txt');
  reset(f);
  for i:=0 to 2 do m[i]:=0;
smax:=1;
readln(f,n);
  for i:=0 to n-1 do begin
    readln(f,x);
    p:=x mod 3;
    q:=(3-p) mod 3;
    if(m[q]>0) then
    begin
      s:=x+m[q];
      if(s>smax) then smax:=s;
    end;
    if(x>m[p]) then m[p]:=x;
  end;
writeln(smax);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
intx,m[3],p,q;
  long s,smax=1,i,n;
ifstream f("A.txt");
  f>>n;
for(i=0;i<3;i++) m[i]=0;
for(i=0;i<n;i++)
{
    f>>x;
    p=x%3;
    q=(3-p)%3;
    if(m[q]>0)
    {
      s=x+m[q];
      if(s>smax) smax=s;
    }
if(x>m[p]) m[p]=x;
  }
cout<<smax;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
m=[0]*3
smax=1
for i in range(n):
    x=int(f.readline())
    p=x%3
    q=(3-p)%3
    if m[q]>0:
        s=x+m[q]
        if s>smax:
            smax=s
    if x>m[p]:
        m[p]=x
print(smax)
f.close()
```

Еще раз обратим внимание на необходимость проверки $m[q] > 0$. Простой пример. Так работает наш алгоритм.

x	p	q	s	smax	m[0]	m[1]	m[2]	примечание	сумма	обновляем
	0	0	0	1	0	0	0			
1	1	2	0	1	0	1	0	m[2]=0		m[1]
2	2	1	3	3	0	1	2	m[1]>0	x+m[1]	m[2]
300	0	0	0	3	300	1	2	m[0]=0		

Если не проверяем условие, просто складываем$x + m[q]$.

x	p	q	s	smax	m[0]	m[1]	m[2]	примечание	сумма	обновляем
	0	0	0	1	0	0	0			
1	1	2	1	1	0	1	0		x+m[2]	m[1]
2	2	1	3	3	0	1	2		x+m[1]	m[2]
300	0	0	300	300	300	1	2		x+m[0]	m[0]

Идея 2. Храним максимальные числа, нужно два числа, кратных 3, одно число с остатком 1, одно число с остатком 2. Максимальная сумма получается либо как сумма слагаемых с остатками 1 и 2 (если оба не нули!), либо как сумма двух чисел, кратных 3 (тоже должно быть два таких ненулевых числа!).
Здесь нужно уметь искать второй максимум (в прошлой задаче было подобное).
Второй максимум − максимальное число в последовательности, из которой убрали максимальное число. Например, в последовательности $\{2,15,4,7\}$ первый максимум равен 15, второй равен 7.
Рассмотрим на примере, сразу для нашей задачи.

- x −текущее число
- p −остаток от деления x на 3
- $m0a$ −первое максимальное число, кратное 3
- $m0b$ −второе максимальное число, кратное 3
- $m1$ −максимальное числос остатком 1
- $m2$ −максимальное число с остатком 2

x	p	m0a	m0b	m1	m2	примечание	обновляем
	0	0	0	1	0		
97	1	0	0	97	0		m1
6	0	6	0	97	0	x>m0a	m0a
3	0	6	3	97	0	x<m0a,x>m0b	m0b
21	0	21	6	97	0	x>m0a, m0a>m0b	m0a,m0b
3	0	21	6	97	0	x<m0b	

Ответ здесь $m0a + m0b = 21 + 6 = 27$. Поскольку $m2 = 0$,сумму $m1 + m2$не рассматриваем.

Итак, поиск второго максимума. Пусть очередное число x кратно 3. Возможны три случая:

- $x > m0a$,тогда полагаем $m0b = m0a, m0a = x$ (здесь учтено, что $m0a \geq m0b$ всегда);
- $m0a \geq x > m0b$, тогда полагаем $m0b = x$;
- $m0b \geq x$,тогда ничего не обновляем.

Основные ошибки:

- Неверный поиск второго максимума.
- Забывают учитывать, что оба суммируемых числа должны быть ненулевыми.

Pascal

```
var smax,i,n: longint;
x,p,m1,m2,m0a,m0b:integer;
f:text;
begin
  assign(f,'A.txt');
  reset(f);
  m1:=0;
  m2:=0;
  m0a:=0;
  m0b:=0;
smax:=1;
readln(f,n);
  for i:=0 to n-1 do begin
readln(f,x);
p:=x mod 3;
    if (p=1)and(x>m1) then m1:=x;
    if (p=2)and(x>m2) then m2:=x;
    if (p=0) then
      if (x>m0a) then begin
        m0b:=m0a;
        m0a:=x;
      end
    else if x>m0b then
      m0b:=x;
  end;
if (m1*m2>0) then
smax:=m1+m2;
if (m0b>0) and (m0a+m0b>smax) then
smax:=m0a+m0b;
writeln(smax);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
int x,m1,m2,m0a,m0b,p;
  long smax=1,i,n;
ifstream f("A.txt");
  f>>n;
m0a=m0b=m1=m2=0;
for(i=0;i<n;i++)
{
    f>>x;
    p=x%3;
    if((p==1)&&(x>m1)) m1=x;
    if((p==2)&&(x>m2)) m2=x;
    if(p==0)
      if(x>m0a)
      {
        m0b=m0a;
        m0a=x;
      }
      else if(x>m0b) m0b=x;
  }
  if(m1*m2>0) smax=m1+m2;
  if((m0b>0)&&(m0a+m0b>smax))
smax= m0a+m0b;
cout<<smax;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
m0a=m0b=m1=m2=0
smax=1
for i in range(n):
    x=int(f.readline())
    p=x%3
    if p==1 and x>m1:
        m1=x
    if p==2 and x>m2:
        m2=x
    if p==0:
        if x>m0a:
            m0b=m0a
            m0a=x
        elif x>m0b:
            m0b=x
if m1*m2>0:
    smax=m1+m2
if m0b>0 and m0a+m0b>smax:
    smax=m0a+m0b
print(smax)
f.close()
```

Пример 4.Задание 27 № 28129

На вход программы поступает последовательность из *N* натуральных чисел. Рассматриваются все пары различных элементов последовательности, у которых различные остатки от деления на $d = 160$ и хотя бы одно из чисел делится на $m = 7$. Среди таких пар, необходимо найти и вывести пару с максимальной суммой элементов.

Входные данные.

Файл A
Файл B

В первой строке входных данных задаётся количество чисел *N* ($1 \le N \le 1000$). В каждой из последующих N строк записано одно натуральное число, не превышающее 10 000. В качестве результата программа должна напечатать элементы искомой пары. Если среди найденных пар максимальную сумму имеют несколько, то можно напечатать любую из них. Если таких пар нет, то вывести два нуля.

Пример организации исходных данных во входном файле:

4
168
7
320
328

Пример выходных данных для приведённого выше примера входных данных:

168 320

В ответе укажите четыре числа: сначала значение искомой пары для файла *A* (два числа через пробел по возрастанию), затем для файла *B* (два числа через пробел по возрастанию).

Неэффективное решение строится по тем же принципам, что и раньше, и здесь не приводится.

Эффективное решение.

Идея 1. Пусть к нам пришло очередное число, скажем, x. Мы можем складывать его с числами, пришедшие до него. При этом:

- Если x кратно 7, его можно сложить с любым числом, остаток которого от деления на 160 не совпадает с остатком от деления x на 160;
- Если x не кратно 7, его можно сложить лишь с числом, кратным 7, остаток которого от деления на 160 не совпадает с остатком от деления x на 160;
- Для получения максимальной суммы x складываем с максимальным из подходящих чисел (**если таковое есть**).
- Все предыдущие числа хранить не нужно, только максимальные.
- Если полученная сумма больше текущей максимальной суммы, обновляем максимальную сумму и запоминаем ее слагаемые.
- После обновления суммы обновляем максимумы в зависимости от вида x.

Смысл переменных:

- x −текущее число
- p −остаток от деления x на 3
- s −текущая сумма
- $smax$ −максимальная сумма
- $y[0..159]$ −максимальные числа с соответствующими остатками от деления на 160
- $z7[0..159]$ −максимальные числа, кратные 7, с соответствующими остатками от деления на 160

Pascal

```
var s1,s2,smax,i,j,n: longint;
x,p,t:integer;
y,z7:array[0..159] of integer;
f:text;
begin
  assign(f,'A.txt');
  reset(f);
  for i:=0 to 159 do begin
    y[i]:=0;
    z7[i]:=0;
  end;
  s1:=0;
  s2:=0;
  smax:=0;
  readln(f,n);
  for i:=0 to n-1 do begin
    readln(f,x);
    p:=x mod 160;
    t:=0;
    if(x mod 7 =0) then begin
      for j:=0 to 159 do
       if(j<>p)and(y[j]>t) then
        t:=y[j];
      if(x>z7[p]) then
        z7[p]:=x;
    end
    else
     for j:=0 to 159 do
      if(j<>p)and(z7[j]>t) then
        t:=z7[j];
    if((t>0)and(x+t>smax)) then
    begin
      smax:=x+t;
      s1:=x;
      s2:=t;
    end;
    if(x>y[p]) then
      y[p]:=x;
  end;
writeln(s1,' ',s2);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
  int x,y[160],z7[160],p;
  long s1,s2,t,smax,i,j,n;
  ifstream f("A.txt");
  f>>n;
  for(i=0;i<160;i++)
    y[i]=z7[i]=0;
  s1=s2=smax=0;
  for(i=0;i<n;i++)
  {
    f>>x;
    p=x%160;
    t=0;
    if(x%7==0)
    {
      for(j=0;j<160;j++)
        if((j!=p)&&(y[j]>t))
          t=y[j];
      if(x>z7[p]) z7[p]=x;
    }
    else
      for(j=0;j<160;j++)
        if((j!=p)&&(z7[j]>t))
          t=z7[j];
    if((t>0)&&(x+t>smax))
    {
      smax=x+t;
      s1=x;
      s2=t;
    }
    if(x>y[p])
      y[p]=x;
  }
cout<<s1<<' '<<s2;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
y=[0]*160
z7=[0]*160
smax=0
s1=s2=0
for i in range(n):
    x=int(f.readline())
    p=x%160
    t=0
    if x%7==0:
        for j in range(160):
            if j!=p and y[j]>t:
                t=y[j]
        if x>z7[p]:
            z7[p]=x
    else:
        for j in range(160):
            if j!=p and z7[j]>t:
                t=z7[j]
    if t>0 and x+t>smax:
        smax=x+t
        s1=x
        s2=t
    if x>y[p]:
        y[p]=x
print(s1,s2)
f.close()
```

Пример 5. Задание 27 № 27424
Имеется набор данных, состоящий из пар положительных целых чисел. Необходимо выбрать из каждой пары ровно одно число так, чтобы сумма всех выбранных чисел не делилась на 3 и при этом была максимально возможной. Гарантируется, что искомую сумму получить можно.
Программа должна напечатать одно число — максимально возможную сумму, соответствующую условиям задачи.
Входные данные.

Файл A
Файл B

Даны два входных файла (файл *A* и файл *B*), каждый из которых содержит в первой строке количество пар N ($1 \le N \le 100000$). Каждая из следующих N строк содержит два натуральных числа, не превышающих 10 000.
Пример организации исходных данных во входном файле:
6
1 3
5 12
6 9
5 4
3 3
1 1
Для указанных входных данных значением искомой суммы должно быть число 32.
В ответе укажите два числа: сначала значение искомой суммы для файла *A*, затем для файла *B*.

Эффективное решение. В этой задаче оно проще неэффективного.
Идея:

- Максимальная сумма состоит из максимальных слагаемых
- Сказать, будет ли сумма кратна трем, нельзя, пока не посчитана вся сумма
- Будем составлять сумму из максимальных слагаемых, вдруг нам повезет
- Если не повезет, нам нужно минимально уменьшить сумму, для этого достаточно изменить свой выбор в одной паре. Мы уберем из суммы одно число и добавим другое, то есть, вычтем из суммы разность данных чисел. Это разность не должна быть кратной 3 и должна быть минимальной.

Пример.

- a, b –текущие числа
- sum –текущая сумма
- d –минимальная разность между числами, не кратная 3

a	b	max(a,b)	sum	d=|a-b|	dmin	примечание	обновляем
			0		10000		
1	3	3	3	2	2	|a-b|<d, |a-b| mod 3 >0	d
5	12	12	15	7	2	|a-b|>d	
6	5	6	21	1	1	|a-b|<d, |a-b| mod 3 >0	d
5	4	5	26	1	1	|a-b|=d	
4	4	4	30	0	1	|a-b|<d, |a-b| mod 3 =0	

Здесь ответ $sum - dmin = 30 - 1 = 29$.

C++	Python

#include<iostream>
#include<fstream>
using namespace std;
int main()
{
  ifstream f("A.txt");
  long n,a,b,i,d,dmin=10000,sum=0;
  f >> n;
  for(i=0;i<n;i++)
  {
  f >> a >> b;
  if (a>b)
  {
  sum+=a;
  d=a-b;
  }
  else
  {
  sum+=b;
  d=b-a;
  }
  if ((d%3!=0)&&(d<dmin))
  dmin=d;
  }
  if (sum%3==0)
  sum-=dmin;
  cout << sum;
  return 0;
}
	f=open('A.txt','r')
	n=int(f.readline())
	sum=0
	dmin=10000
	for i in range(n):
	  a,b = map(int,f.readline().split())
	  if a>b:
	  sum+=a
	  d=a-b
	  else:
	  sum+=b
	  d=b-a
	  if d%3!=0 and d<dmin:
	  dmin=d
	if sum%3==0:
	  sum-=dmin
	print(sum)
	f.close()

https://kpolyakov.spb.ru/school/ege.htm

Пример 6. (Задача 2). Имеется набор данных, состоящий из пар положительных целых чисел. Необходимо выбрать из каждой пары ровно одно число так, чтобы сумма всех выбранных чисел делилась на 3 и при этом была максимально возможной. Гарантируется, что искомую сумму получить можно. Программа должна напечатать одно число – максимально возможную сумму, соответствующую условиям задачи.
Входные данные: Даны два входных файла: файл A (**27-2a.txt**) и файл B (**27-2b.txt**), каждый из которых содержит в первой строке количество пар N ($1 \le N \le 100000$). Каждая из следующих N строк содержит два натуральных числа, не превышающих 10 000.
Пример входного файла:
6
1 3
5 11
6 9
5 4
3 3
1 1
Для указанных входных данных значением искомой суммы должно быть число 30.
В ответе укажите два числа: сначала значение искомой суммы для файла A, затем для файла B.

Решение. Задача сразу усложняется, хотя из условия пропала только частица «не». Если сумма не делится на 3, то она при делении на 3 дает какой-либо остаток, 1 или 2. Чтобы она делилась на 3, нужно вычесть из нее число с тем же остатком (в прошлой задаче вычиталось число, не кратное 3, это более слабое условие).
Более того, возможно, нам придется сделать замены не в одной паре. Рассмотрим это на примере.
В левой таблице сумма составляется из максимальных чисел в паре, они выделены цветом. В правой таблице выделены числа (s), которые надо было выбрать, чтобы получить решение.

a	b	max(a,b)	sum
			0
1	3	3	3
5	12	12	15
6	4	6	21
15	4	15	36
4	4	4	40

a	b	s	sum
			0
1	3	1	1
5	12	12	13
6	4	4	17
15	4	15	32
4	4	4	36

К сожалению, заранее нельзя сказать, какие числа придется заменить. Поэтому мы будем хранить массив $dmin[]$, в нем будут разности с соответствующими остатками, но они могут складываться из сумм разностей в нескольких парах. В предыдущем примере нам надо было уменьшить 40 на величину с остатком 1, и она сложилась из замен в двух парах (2+2). В итоге из 40 мы вычли 4. Можно было заменить выбор в одной паре, 12 на 5, но тогда результат был бы хуже (33).

a	b	max(a,b)	sum	d=|a-b|	dmin[1]	dmin[2]	примечание	обновляем
			0		10000	10000		
1	3	3	3	2	10000	2	|a-b|<dmin[2], |a-b| mod 3 =2	dmin[2]
5	12	12	15	7	7	2	|a-b|<dmin[2], |a-b| mod 3 =2	dmin[1]
6	4	6	21	2	4	2	|a-b|+dmin[2]<dmin[1], |a-b| mod 3 = 2 (2+2) mod 3 = 1	dmin[1]
15	4	15	36	11	4	2		
4	4	4	40	0	4	2		

Математика.

- Если $|a-b|\ mod\ 3\ =\ p$, и $|a-b| < dmin[p]$, то можно обновить $dmin[p]$
- Если $(p+k)\ mod\ 3 = r$, и $|a-b| + dmin[k] < dmin[r]$, то можно обновить еще и $dmin[r]$

- Поскольку все элементы массива обновляются друг через друга, и это достаточно сложная структура, то лучше завести еще один массив $d[]$. В него записываем новые значения минимальных разностей, затем, когда все посчитано, копируем все в основной массив.
- $d[0] = dmin[0] = 0$ для общности, это позволяет все пересчитать за один цикл по k. При $k = 0$ имеем

```
if(dmin[k]+dd<dmin[(k+p)%3]):
        d[(k+p)%3]=dmin[k]+dd
```

поскольку `dmin[k]=dmin[0]=0,` практически код аналогичен

```
if(dd<dmin[p]):
        d[p]=dd
```

то есть происходит обновление $dmin[p]$. При $k > 0$ обновляются $dmin[(k + p)\%3]$.

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
  ifstream f("A.txt");
  long n,a,b,i,k,dmin[3],d[3],dd,sum=0,p;
  d[0]=dmin[0]=0;
  d[1]=d[2]=dmin[1]=dmin[2]=10000;
    f >> n;
    for(i=0;i<n;i++)
    {
        f >> a >> b;
        if (a>b)
        {
            sum+=a;
            dd=a-b;
        }
        else
        {
            sum+=b;
            dd=b-a;
        }
        p=dd%3;
        for(k=0;k<3;k++)
        if(dmin[k]+dd<dmin[(k+p)%3])
            d[(k+p)%3]=dmin[k]+dd;
        for(k=0;k<3;k++)
         dmin[k]=d[k];
    }
    sum-=dmin[sum%3];
    cout << sum;
    return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
dmin=[10000]*3
d=[10000]*3
d[0]=0
dmin[0]=0
sum=0
print(n)
for i in range(n):
    a,b = map(int,f.readline().split())
    if a<b:
        a,b=b,a
    sum+=a
    dd=a-b
    p=dd%3
    for k in range(3):
        if(dmin[k]+dd<dmin[(k+p)%3]):
            d[(k+p)%3]=dmin[k]+dd
    for k in range(3):
        dmin[k]=d[k]
sum-=dmin[sum%3]
print(sum)
f.close()
```

Пример 7. Задача 13. Имеется набор данных, состоящий из положительных целых чисел. Необходимо определить количество пар элементов (a_i, a_j) этого набора, в которых индексы элементов последовательности различаются не меньше, чем на 7, и произведение элементов кратно 14.

Входные данные: Даны два входных файла: файл A (**27-13a.txt**) и файл B (**27-13b.txt**), каждый из которых содержит в первой строке количество чисел N ($1 \le N \le 100000$). Каждая из следующих N строк содержит натуральное число, не превышающее 1000.

Пример входного файла:

9
7
5
6
12
5
11
8
16
14

Для указанных входных данных количество подходящих пар должно быть равно 3. В приведённом наборе имеются три подходящие пары (7, 16), (7, 14), (5, 14), произведение элементов которых кратно 14, а индексы элементов последовательности различаются не меньше, чем на 7.

В ответе укажите два числа: сначала количество подходящих пар для файла A, затем для файла B.

Неэффективное решение организуется перебором пар. Здесь стоит обратить внимание на цикл, каким образом меняются индексы.

$$0 \le i < n,$$
$$0 \le j < n,$$
$$i + 7 \le j$$

Отсюда получаем

$$0 \le i < n - 7,$$
$$i + 7 \le j < n.$$

Pascal

```
var count,p,j,i,n: longint;
A:array[0..9999] of longint;
f:text;
begin
count:=0;
  assign(f,'A.txt');
  reset(f);
  readln(f,n);
  for i:=0 to n-1 do
  readln(f,A[i]);
  for i:=0 to n-8 do
    for j:=i+7 to n-1 do
    begin
      p:=A[i]*A[j];
      if (p mod 14 = 0) then
        count:=count+1
    end;
writeln(count);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{ long count=0,p,j,i,n;
ifstream f("A.dat");
  f>> n;
  int A[n];
  for(i=0;i<n;i++)
    f>>A[i];
  for(i=0;i<n-7;i++)
    for(j=i+7;j<n;j++)
    {
    p=A[i]*A[j];
      if((p%14==0)&&(p>pmax))
         count++;
    }
cout<<pmax;
return 0;
}
```

Python

```
f=open('A.dat','r')
n=int(f.readline())
A=[]
for i in range(n):
x=int(f.readline())
A.append(x)
count=0
for i in range(n-7):
    for j in range(i+7,n):
        p=A[i]*A[j]
        if p%14==0 and p>pmax:
            count+=1
print(count)
f.close()
```

Эффективное решение. Пусть к нам пришло очередное число, скажем, x. Мы можем умножать его на числа, пришедшие до него, но чтобы расстояние было не меньше 7. При этом:

- Если x кратно 14, его можно умножить на любое число, то есть, оно образует пару со всеми пришедшими до него числами;
- Если x кратно 7, но не кратно 2, его можно умножать на число, кратное 2, образуется столько пар, сколько на данный момент есть чисел, кратных 2;
- Если x кратно 2, но не кратно 7, его можно умножать на число, кратное 7, образуется столько пар, сколько на данный момент есть чисел, кратных 7;
- Если x не кратно ни 2, ни 7, его можно умножать на число, кратное 14, образуется столько пар, сколько на данный момент есть чисел, кратных 14;
- В специальный счетчик добавляем найденное количество пар, затем обновляем счетчики чисел с различными условиями;
- Для обеспечения условия на расстояние заведем массив $buff$ размерности 7, в котором будем хранить 7 чисел, пришедших непосредственно перед x (буферный массив). Эти числа мы пока не можем использовать;
- Когда придет следующее число, мы достаем из буфера число, которое пришло раньше всех, и обновляем с его помощью счетчики;
- Первые 7 чисел просто считываем в массив.

Смысл переменных:

- i −номер числа
- x −текущее число
- $count$ −счетчик пар
- $k14$ −количество чисел, кратных 14
- $k7$ −количество чисел, кратных 7
- $k2$ −количество чисел, кратных 2
- $buff$ −буферный массив

i	x	count	k14	k7	k2	buff	с чем образует пару	обновляем
		0	0	0	0			
0	7	0	0	0	0	7		
1	5	0	0	0	0	7,5		
2	6	0	0	0	0	7,5,6		
3	12	0	0	0	0	7,5,6,12		
4	5	0	0	0	0	7,5,6,12,5		
5	11	0	0	0	0	7,5,6,12,5,11		
6	8	0	0	0	0	7,5,6,12,5,11,8		
7	16	1	0	1	0	5,6,12,5,11,8,16	7(+k7)	k7
8	14	3	0	1	0	6,12,5,11,8,16,14	7,5(+(i-6))	
9	4	4	0	1	1	12,5,11,8,16,14,4	7(+k7)	k2

В первой программе из буферного массива каждый раз извлекается элемент $buff[0]$, затем массив сдвигается, и число x добавляется в конец. Во второй программе используется идея циклического массива, извлекается элемент $buff[i\%7]$, затем на его место записывается x. В этом случае сдвиг массива не нужен, программа работает быстрее.

Pascal

```
var count,k14,k7,k2,i,j,n: longint;
x,y:integer;
buff:array[0..6] of integer;
f:text;
begin
  assign(f,'A.txt');
  reset(f);
  count:=0;
  k14:=0;
  k7:=0;
  k2:=0;
  readln(f,n);
  for i:=0 to 6 do begin
    readln(f,buff[i]);
  for i:=0 to n-8 do begin
    readln(f,x);
    y:=buff[0];
    if(y mod 14 = 0) then
      k14:=k14+1;
    if(y mod 7 = 0) then
      k7:=k7+1;
    if(y mod 2 = 0) then
      k2:=k2+1;
    if(x mod 14 = 0) then
      count:=count+i+1;
    else if(x mod 7 = 0) then
      count:=count+k2;
    else if(x mod 2 = 0) then
      count:=count+k7;
    else count:=count+k14;
    for j:=0 to 5 do
      buff[j]:=buff[j+1];
    buff[6]:=x;
  end;
writeln(count);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
  int x,y,buff[7];
  long k14,k7,k2,n,i,j,count;
  ifstream f("A.txt");
  f>>n;
  count=k14=k7=k2=0;
  for(i=0;i<n;i++)
    f>>buff[i];
  for(i=0;i<n-7;i++)
  {
   f>>x;
   y=buff[0];
   if(y%14==0) k14++;
   if(y%7==0) k7++;
   if(y%2==0) k2++;
   if(x%14==0) count+=(i+1);
   else if (x%7==0) count+=k2;
   else if (x%2==0) count+=k7;
   else count+=k14;
   for (j=0;j<6;j++)
        buff[j]=buff[j+1];
    buff[6]=x;
  }
cout<<count;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
k14=k7=k2=count=0
buff=[]
for i in range(7):
    x=int(f.readline())
    buff.append(x)
for i in range(n-7):
    x=int(f.readline())
    y=buff[0]
    if(y%14==0):
        k14+=1
    if(y%7==0):
        k7+=1
    if(y%2==0):
        k2+=1
    if(x%14==0):
        count+=(i+1);
    elif (x%7==0):
        count+=k2
    elif (x%2==0):
        count+=k7
    else:
        count+=k14
    for j in range(6):
        buff[j]=buff[j+1]
    buff.append(x)
print(count)
f.close()
```

Pascal

```
var count,k14,k7,k2,i,j,n: longint;
x,y:integer;
buff:array[0..6] of integer;
f:text;
begin
  assign(f,'A.txt');
  reset(f);
  count:=0;
  k14:=0;
  k7:=0;
  k2:=0;
  readln(f,n);
  for i:=0 to 6 do begin
    readln(f,buff[i]);
  for i:=0 to n-8 do begin
    readln(f,x);
    y:=buff[i%7];
    if(y mod 14 = 0) then
      k14:=k14+1;
    if(y mod 7 = 0) then
      k7:=k7+1;
    if(y mod 2 = 0) then
      k2:=k2+1;
    if(x mod 14 = 0) then
      count:=count+i+1;
    else if(x mod 7 = 0) then
      count:=count+k2;
    else if(x mod 2 = 0) then
      count:=count+k7;
    else count:=count+k14;
    buff[i%7]:=x;
  end;
writeln(count);
close(f);
end.
```

C++

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
  int x,y,buff[7];
  long k14,k7,k2,n,i,j,count;
  ifstream f("A.txt");
  f>>n;
  count=k14=k7=k2=0;
  for(i=0;i<n;i++)
    f>>buff[i];
  for(i=0;i<n-7;i++)
  {
   f>>x;
   y=buff[i%7];
   if(y%14==0) k14++;
   if(y%7==0) k7++;
   if(y%2==0) k2++;
   if(x%14==0) count+=(i+1);
   else if (x%7==0) count+=k2;
   else if (x%2==0) count+=k7;
   else count+=k14;
   buff[i%7]=x;
  }
cout<<count;
return 0;
}
```

Python

```
f=open('A.txt','r')
n=int(f.readline())
k14=k7=k2=count=0
buff=[]
for i in range(7):
    x=int(f.readline())
    buff.append(x)
for i in range(n-7):
    x=int(f.readline())
    y=buff[i%7]
    if(y%14==0):
        k14+=1
    if(y%7==0):
        k7+=1
    if(y%2==0):
        k2+=1
    if(x%14==0):
        count+=(i+1);
    elif (x%7==0):
        count+=k2
    elif (x%2==0):
        count+=k7
    else:
        count+=k14
    buff[i%7]=x
print(count)
f.close()
```