Методика подготовки к сдаче ЕГЭ по информатике (Задача 27)

В задании 27 встречаются несколько типов задач. Рассмотрим их подробнее.

Тип задач 1 (делимость).

Вариант 1 по задачнику.

Компьютер наземной станции слежения получает от объектов – самолетов, находящихся в зоне ее работы, идентификационные сигналы, представляющие собой последовательность из N целых положительных чисел. Каждый объект направляет на наземную станцию уникальное число, т.е. все числа в получаемой станцией последовательности различны. Обработка сигнала представляет собой рассмотрение всех пар различных элементов последовательности, при этом элементы пары не обязаны быть переданы непосредственно друг за другом, порядок элементов в паре не важен. Считается, что возникла критическая ситуация, если произведение элементов некоторой пары кратно 58. Необходимо определить общее количество возникших критических ситуаций.

Описание входных и выходных данных.

В первой строке входных данных задается количество чисел N (1<= N <= 1000). В каждой из последующих N строк записано одно целое положительное число, не превышающее 10 000.

В качестве результата программа должна напечатать одно число: общее количество возникших критических ситуаций.

Решение.

Следует учитывать, что в задании 27 возможно несколько вариантов решения задачи и их оценивание при правильном решении варьируется от 2 до 4 баллов. От чего зависит результат? От эффективности работы программы. Простейшая программа, которая сохраняет все входные данные в массиве и потом путем обхода этого массива находящая решение оценивается в 2 балла. Если же полученное решение такого, что при росте количества данных, время выполнения и необходимая память растут линейно, то за такую программу экзаменуемый получает 4 балла.

Вернемся к конкретной задаче и сначала рассмотрим самое простое решение.

Алгоритм программы прост.

1. Ввод и сохранение всего набора данных в массиве.
2. С помощью двух циклов организовываем обход всех пар чисел, находящихся в массиве. Если их произведение делится на 58, то увеличиваем счетчик критических ситуаций.
3. Вывод полученных результатов.

Приведем пример программ, построенных по данному алгоритму, на нескольких языках программирования.

C++

```
#include <iostream>
using namespace std;
int main(){
	long int mas[1000];
	long int N, cryt, temp;
	cryt = 0;
	cin >> N;
	for (int i=0;i<N;i++) cin >> mas[i];
	for (i=0;i<N-1;i++){
		for (int j=i+1;j<N;j++){
			temp = mas[i]*mas[j];
			if (temp%58 == 0) cryt++;
		}
	}
	cout << cryt << endl;
}
```

Python

```
cryt = 0
mas = []
N = int(input())
for i in range(N):
```

```
        mas[i] = int(input())
for i in range(N-1):
        for j in range(i+1,N):
                temp = mas[i]*mas[j]
                if temp%58 == 0:
                        cryt++;

print(cryt)
```

Теперь рассмотрим вариант решения, эффективного по времени и памяти, на 4 балла.
Когда произведение двух чисел делится на 58?

1. Когда оба сомножителя делятся на 58.
2. Только один сомножитель делится на 58.
3. Одно число делится на 2, другое на 29.

В процессе ввода данных определяем:
- количество чисел, делящихся на 58 – n58;
- количество чисел, делящихся на 2, но не делящихся на 58 – n2;
- количество чисел, кратных 29, но не делящихся на 58 – n29.

После ввода всех данных и вычисления вышеприведенных переменных получаем, что количество пар в группе 1 будет равно n58*(n58-1)/2. В группе 2: n58*(N-n58), где N – общее количество элементов. В группе 3: n2*n29.
Так как эти группы не пересекаются, то ответом будет сумма полученных значений.
Стоит отметить, что данный алгоритм верен только в случае если число, ответственное за определение критической ситуации, является произведением двух простых чисел.
Примеры решения на нескольких языках программирования.

C++

```
#include <iostream>
using namespace std;
int main(){
        long int N, n58, n29, n2, temp, cryt;
        n58 = 0;
        n29 = 0;
        n2 = 0;
        cin >> N;
        for (int i=0;i<N;i++){
                cin >> temp;
                if (temp%58 == 0) n58++;
                else if (temp%29 ==0) n29++;
                else if (temp%2 == 0) n2++;
        }
        crypt = (n58*(n58-1))/2 + n58*(N-n58) +n29*n2;
        cout << crypt << endl;
}
```

Python

```
n58 = 0
n29 = 0
n2 = 0
N = int(input())
for i in range(N):
        a = int(input())
        if temp%58 == 0:
                n58++
        elif temp%29 ==0:
                n29++
        elif temp%2 == 0:
                n2++
crypt = (n58*(n58-1))//2 + n58*(N-n58) +n29*n2
```

print(crypt)

Тип задач №2.

Следующий тип задач показан в варианте номер 3 задачника. Здесь мы видим, что изначально дано два разных условия задачи.

А. Имеется набор данных, состоящий из 6 пар положительных целых чисел. Необходимо выбрать из каждой пары одно число так, чтобы сумма всех выбранных чисел не делилась на 5 и при этом была максимально возможной. Если получить требуемую сумму невозможно, в качестве ответа нужно выдать 0.

Б. Имеется набор данных, состоящий из пар положительных целых чисел. Необходимо выбрать из каждой пары одно число так, чтобы сумма всех выбранных чисел не делилась на 5 и при этом была максимально возможной. Если получить требуемую сумму невозможно, в качестве ответа нужно выдать 0.

План решения задачи А достаточно прост.

1. Ввод данных и сохранение их в массиве.
2. С помощью циклов организация полного перебора и поиск нужного значения.
3. Вывод результатов на печать.

Примеры программ на разных языках программирования.

C++

```
#include <iostream>
using namespace std;
int main(){
	long int mas[6][2];
	long int max, temp;
	max = 0;
	for (int i=0;i<6;i++) cin >> mas[i][0] >> mas[i][1];
	for (int i1=0;i1<2;i1++)
	for (int i2=0;i2<2;i2++)
	for (int i3=0;i3<2;i3++)
	for (int i4=0;i4<2;i4++)
	for (int i5=0;i5<2;i5++)
	for (int i6=0;i6<2;i6++){
		temp = mas[0][i1] + mas[1][i2] + mas[2][i3] + mas[3][i4] + mas[4][i5] + mas[5][i6];
		if (temp%5 != 0 && max<temp) max = temp;
	}
	cout << max << endl;
}
```

Python

```
max = 0
mas = []
for i in range(6):
	mas.append([])
	mas[i][0],mas[i][1] = map(int, input().split())
for i1 in range(2):
	for i2 in range(2):
		for i3 in range(2):
			for i4 in range(2):
				for i5 in range(2):
					for i6 in range(2):
						temp = mas[0][i1] + mas[1][i2] + mas[2][i3] + mas[3][i4] +
						mas[4][i5] + mas[5][i6]
						if temp%5 != 0 && max<temp:
							max = temp;
print(max)
```

Задача Б требует другого подхода. После считывания каждой пары мы определяем максимальное и минимальное значения. Максимальные величины из разных пар суммируем. В результате получаем максимально возможную сумму. Но, возможен вариант, когда полученное число будет кратно 5. Для решения этой проблемы среди пар числа которых имеют разный остаток от деления на 5 необходимо найти минимальную разность. Тогда если мы отнимем от результата эту величину, то получим максимальную сумму не кратную 5.
Ниже следуют примеры на нескольких языках программирования.

```
#include <iostream>
using namespace std;
int main(){
	long int max, max_t, min_t, min_d, temp;
	int N;
	max = 0;
	min_d = 0;
	cin >> N;
	for (int i=0;i<N;i++){
		cin >> max_t >> min_t;
		if (min_t > max_t){
			temp = max_t;
			max_t = min_t;
			main_t = temp;
		}
		max += max_t;
		temp = max_t – main_t;
		if (temp%5 != 0){
			if (min_d == 0 || min_d > temp) min_d = temp;
		}
	}
	If (max %5 != 0) cout << max << endl;
	else if (min_d >0){
		max -= min_d;
		cout << max << endl;
	} else cout << 0 << endl;
}
```

Python

```
max = 0
N = int(input())
min_d = 0
for i in range(N):
	max_t, min_t = map(int, input().split())
	if max_t<min_t:
		temp=max_t
		max_t=min_t
		main_t=temp
	max += max_t
	temp = max_t – main_t;
	if temp%5 != 0 && (min_d == 0 || min_d>temp)):
		min_d = temp
if max%5==0:
	if min_d>0:
		max -= min_d
	else:
		max = 0;
print(max)
```

Тип задач № 3.

Вариант 6 по задачнику. Является усложнением задачи 2 типа.

Дано.

А. Имеется набор данных, состоящий из 5 троек положительных целых чисел. Необходимо выбрать из каждой тройки ровно одно число так, чтобы сумма квадратов всех выбранных чисел была четной и при этом минимально возможной. Если получить требуемую сумму невозможно, в качестве ответа выдать 0.

Б. Имеется набор данных, состоящий из троек положительных целых чисел. Необходимо выбрать из каждой тройки ровно одно число так, чтобы сумма квадратов всех выбранных чисел была четной и при этом минимально возможной. Если получить требуемую сумму невозможно, в качестве ответа выдать 0.

Решение.

Задача А решается аналогично задаче А второго типа.

Примеры решения.

C++

```
#include <iostream>
using namespace std;
int main(){
	int mas[5][3];
	long int min, temp;
	min =0;
	for (int i=0;i<5;i++) cin >> mas[i][0] >> mas[i][1] >> mas[i][2];
	for (int i1=0;i1<3;i1++)
	for (int i2=0;i2<3;i2++)
	for (int i3=0;i3<3;i3++)
	for (int i4=0;i4<3;i4++)
	for (int i5=0;i5<3;i5++){
		temp = mas[0][i1]*mas[0][i1] + mas[1][i2]* mas[1][i2] + mas[2][i3]* mas[2][i3] + mas[3][i4] +
mas[4][i5]* mas[4][i5];
		if (temp%2 == 0){
			if (min ==0) min = temp;
			if (min > temp) min = temp;
		}
	}
	cout << min << endl;
}
```

Python

```
min = 0
mas = []
for i in range(5):
	mas.append([])
	mas[i][0],mas[i][1],mas[i][2] = map(int, input().split())
for i1 in range(3):
	for i2 in range(3):
		for i3 in range(3):
			for i4 in range(3):
				for i5 in range(3):
					temp = mas[0][i1] + mas[1][i2] + mas[2][i3] + mas[3][i4]+ mas[4][i5]
					if temp%2 == 0 && max<temp:
						max = temp;
print(max)
```

Задача Б.

За основу необходимо взять решение задачи Б 2 типа с некоторыми изменениями.

1. Вместо определения максимального и минимального чисел производится сортировка.
2. При поиске минимума для коррекции результата необходимо учитывать, что мы имеем не пару, а три числа.

Примеры программ.

C++

```
#include <stdlib.h>
#include <iostream>
function comp_up(const void * a, const void * b){
	return(*(long int*) a - *(long int*) b);
}
using namespace std;
int main(){
	long int mas[3];
	int N;
	long int min, temp, min_d;
	min = 0;
	min_d = 0;
	cin >> N;
	for (int i=0;i<N;i++){
		cin >> mas[0] >> mas[1] >> mas[2];
		qsort(mas, 3, sizeof(long int), comp_up);
		min += mas[0]*mas[0];
		temp = mas[1]*mas[1]-mas[0]*mas[0];
		if (temp%2 != 0){
			if (min_d == 0 || min_d > temp) min_d = temp;
		} else{
			temp = mas[2]*mas[2]-mas[1]*mas[1];
			if ((temp%2 !=0)&&(min_d == 0 || min_d > temp)) min_d = temp;
		}
	}
	If (min%2 != 0){
		If (min_d>0) min +=min_d; else min = 0;
	}
	cout << min << endl;
}
```

Python

```
min = 0
min_d = 0;
mas = []
N = int(input())
for i in range(N):
	mas[0], mas[1], mas[2] = map(int, input().split())
	mas.sort()
	min += mas[0]*mas[0]
	temp = mas[1]*mas[1]-mas[0]*mas[0]
	if temp%2 != 0:
		if min_d == 0 || min_d > temp:
			min_d = temp
	 else:
		temp = mas[2]*mas[2]-mas[1]*mas[1]
		if (temp%2 !=0)&&(min_d == 0 || min_d > temp):
			min_d = temp
If min%2 != 0:
	If min_d>0:
		min +=min_d
	else:
		min = 0;
print(min)
```

Тип задач №4 (очередь).

Вариант 13 по задачнику.

Датчик передает каждую секунду по каналу связи неотрицательное целое число, не превосходящее 1000, - текущий результат измерений. Временем, в течение которого происходит передача, можно пренебречь.

Необходимо найти в заданной серии показаний датчика минимальное четное произведение двух показаний, между моментами передачи которых прошло не менее 15 секунд. Если получить такое произведение не удается, ответ считается равным -1. Общее количество показаний датчика в серии не превышает 10 000.

А. Напишите программу, в которой входные данные будут запоминаться в массиве, после чего будут проверены все возможные пары элементов.

Б. Напишите программу, которая будет эффективной по времени и по памяти.

Решение.

Задача А.

Алгоритм решения задачи.

1. Считать данные и поместить их в массив.
2. С помощью циклов проверить все пары значений.
3. Вывести результат.

Примеры программ.

C++

```
#include <iostream>
using namespace std;
int main(){
	int mas[10000], N;
	long int min, temp;
	min = -1;
	cin >> N;
	for (int i=0;i<N;i++) cin >> mas[i];
	for (i=0;i<N-15;i++){
		for (int j=i+15;j<N;j++){
			temp = mas[i]*mas[j];
			if ((temp%2==0)&&(min==-1 || min>temp)) min = temp;
		}
	}
	cout << min << endl;
}
```

Python

```
mas = []
min = -1
N = int(input())
for i in range(15):
	mas[i] = int(input())
for i in range(N-15):
	for j in range(i+15,N):
		temp = mas[i]*mas[j]
		if (temp%2==0)&&(min==-1 || min>temp):
			min = temp
print(min)
```

Задача Б.

Для решения этой задачи необходимо организовать хранение и обработку временно неиспользуемых значений. Лучше всего для этого подходит очередь или FIFO (First In First Out). Далеко не во всех языках программирования работа с такой структурой реализована на уровне стандартной библиотеки. Алгоритм решения задачи достаточно прост.

Нам будут необходимы две вспомогательные переменные. Одна из них будет хранить минимальное четное значение вне 15 секундного диапазона, другая – просто минимальное значение вне 15 сек диапазона.

1. Сначала наполняем очередь первыми 15 значениями.
2. Вынимаем из очереди первый зашедший элемент и используем его для определения минимального значения вне 15 сек зоны и, если элемент четный - то и минимального четного вне 15 сек зоны.
3. Считываем новое значение.
4. Если оно четное, то перемножаем его с минимальным значением вне 15 сек зоны, иначе с минимальным четным значением вне 15 сек зоны.
5. Полученное значение сравниваем с минимальным четным значением. При необходимости обновляем минимум.
6. Помещаем ранее считанное значение в конец очереди.
7. Если еще есть значения, то переходим к пункту 2.
8. Выводим результат на печать.

Примеры программ.

C++

```
#include <iostream>
#include <queue>
using namespace std;
int main(){
	queue <int> mas;
	int N, a;
	long int min, min_out15_even, minn_out15, temp;
	cin >> N;
	for (int i=0;i<15;i++){
		cin >> a;
		mas.push(a);
	}
	min = -1;
	min_out15_even = 0;
	min_out15 = 0;
	for (i=0;i<N-15;i++){
		a = mas.pop();
		if ((a%2==0)&&(min_out15_even==0 || a<min_out15_even)) min_out15_even = a;
		if (min_out15==0 || a<min_out15) min_out15 = a;
		cin >> a;
		mas.push(a);
		if (a%2==0) temp=min_out15*a;
		else temp=min_out15_even*a;
		if (min==-1 || min>temp) min = temp;
	}
	cout << min <<endl;
}
```

Python

```
from queue import Queue
mas = Queue()
N = int(input())
for i in range(15):
	a = int(input())
	mas.put(a)
min = -1
min_out15_even = 0
min_out15 = 0
for i in range(N-15):
	a = mas.get()
	if (a%2==0)&&(min_out15_even==0 || a<min_out15_even):
```

```
            min_out15_even = a
        if min_out15==0 || a<min_out15:
            min_out15 = a
        a = int(input())
        mas.put(a)
        if a%2==0:
            temp=min_out15*a
        else:
            temp=min_out15_even*a
        if min==-1 || min>temp:
            min = temp
print(min)
```

Тип задач №5 (исключения).

Вариант 17 в задачнике.

В химической лаборатории для большого количества органических молекул измеряется количество входящих в молекулу атомов углерода – целое неотрицательное число, которое будем называть С-индексом молекулы. Исследуемых молекул может быть очень много, но не может быть меньше трех. С-индексы во всех молекулах различны.

При обработке результатов отбирается так называемое основное множество С-индексов. Это непустое подмножество всевозможных С-индексов такое, что сумма значений С-индексов у него четна и максимальна среди всех возможных непустых подмножеств с четной суммой. Если таких подмножеств несколько, то из них выбирается то подмножество, которое содержит наименьшее количество элементов.

Требуется написать эффективную по времени и памяти программу.

При решении данного типа задач в первую очередь необходимо определить какие из элементов не должны отображаться в общем списке.

В данном примере не должен отображаться элемент с нулевым С-индексом и элемент с минимальным нечетным С-индексом если количество нечетных С-индексов нечетно.

Таким образом алгоритм решения этой задачи будет следующий.

1. Считываем данные, попутно определяя:
   а) номер элемента с нулевым С-индексом;
   б) количество элементов с нечетным С-индексом ;
   в) номер элемента с минимальным нечетным С-индексом.
2. Выводим номера всех элементов от 1 до N за исключением номеров вышеуказанных элементов.

Примеры программ.

```
#include <iostream>
using namespace std;
int main(){
        int a, i_0, i_min_odd, min_odd, odd_cnt;
        min_odd = -1;
        i_min_odd = 0;
        i_0 = 0;
        odd_cnt = 0;
        cin >> N;
        for (int i=1;i<=N;i++){
                cin >> a;
                if (a==0) i_0=i;
                if (a%2 != 0){
                        odd_cnt++;
                        if (min_odd==-1 || min_odd > a){
                                min_odd = a;
                                i_min_odd = i;
                        }
                }
        }
        for (int i=1;i<=N;i++){
```

```
                if (i != i_0 && !(cnt_odd%2 !=0 && i_min_odd==i)) cout << i << " ";
        }
}
```

Python

```
min_odd = -1
i_min_odd = 0
i_0 = 0
odd_cnt = 0
N = int(input())
for i in range(1,N+1):
        a = int(input())
        if a==0:
                i_0=i;
        if a%2 != 0:
                odd_cnt++
                if min_odd==-1 || min_odd > a:
                        min_odd = a
                        i_min_odd = i
for i in range(1,N+1):
        if i != i_0 && !(cnt_odd%2 !=0 && i_min_odd==i):
                print(I, " ")
```