АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
ПОДСЛУШВАНЕ

Идея:

(автори Момчил Иванов)

В задачата е зададена редица от N (четно) числа, в която има (N-2)/2 двойки еднакви числа и още две различни числа. Искаме да намерим кои са двете числа със сложност O(N).

Тривиалното решение на задачата е да сортираме редицата и така да намерим числата със сложност O(N). Затова N трябва да има големи стойности и времето за изпълнение на задачата да е малко.

Предлагано решение:

Намираме xor-а на всички числа - X, естествено, със сложност O(N). Очевидно X е равно на xor-а на двете търсени числа.

Знаем, че X е различно от 0, тъй като двете числа, които търсим са различни. Нека (X&(1<<j)) != 0, т.е. битът на позиция j е равен на 1.

Обхождаме отново всички числа и xor-ваме с X всяко число от редицата, за което бит j също е равен на 1. В края на операцията ще сме получили резултат Y, който ще е равен на едното от двете числа. Така двете числа, които търсим са Y и X^Y.

Как да се освободим от изискването за памет:

(автор Павлин Пеев)

Отделяме масив, във всеки елемент на който съответно натрупваме изключващото или (xor) на числата с нулев бит 1, първи бит 1 и т. н. Ще използваме 63 променливи от тип long long, а цялата работа ще се върши по време на четенето. Няма да е нужно да се помнят входните данни и да се сканират отново.

Задачата има съвсем проста за писане и сравнима по време (в дадените граници) реализация със сложност N.log(N) с помощта, например, на динамичната структура set от STL. Това решение, обаче, се вмества в рамките на зададената памет само за някои примери. Могат да се използват и други подобни (да речем, побитови) идеи за отразяване на срещането на дадено число, особено ако самите елементи не са много големи. Такава реализация обаче, освен, че решава частично въпроса, е доста по-сложна за писане и има опасност от грешки.

При по-големи входни файлове, четенето с инструментите на стандартната входно-изходна библиотека, вместо с поточната, е силно препоръчително за ускоряване на крайната програма.

Реализация:

```
#include <stdio.h>
long N;
long long M=0,K[63]={0};
int main()
{long long a=1,b,mask[63]={0};
 long i,j;
 for (i=0;i<63;i++){mask[i]=a;a<<=1;}
 scanf("%ld",&N);
 for (i=0;i<N;i++)
 {scanf("%I64d",&a);
  M^=a;
  for (j=0;j<63 && a>=mask[j];j++)if (a & mask[j]) K[j]^=a;
 }
 for (i=0;i<63;i++) if (M & mask[i]) break;
 a=K[i];
 b=M^a;
 if (a>b) {a=M;b=a^b;a=a^b;}
 printf("%I64d %I64d\n",a,b);
 return 0;
}
```