# АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
# СУМИ

Задачата изисква ефективен алгоритъм, тъй като броят на числата във всяко от множествата е много голям. Алгоритъм от вида „за всяко $a$ от $S_1$ и за всяко $b$ от $S_2$“ е със сложност $O(N^2)$ и не може да получи висока оценка. Едно възможно по-добро решение е да се сортират елементите на едно от множествата – например второто и след това за всяко $a$ от $S_1$, само ако $x > a$, да се провери с двоично търсене дали сортираният масив $S_2$ съдържа числото $y = x - a$. Ако двоичното търсене завърши успешно, трябва да се огледа „околността“ на намереното $y$ за други такива стойности и да се намери броят на тези стойности, които да се натрупват в предварително анулиран брояч. Сложността на това решение е O($N$.log $N$+ $M$.log $N$)

```
#include <stdio.h>
using namespace std;
#include <algorithm>
#define MAXN 1000001
int s1[MAXN],s2[MAXN],n;

int bin_search(int y)
{ int l=0,r=n-1,k;
  while(l<=r)
  { k=(l+r)/2;
    if(s2[k]==y)return k;
    if(y<s2[k]) r=k-1;
    else l=k+1;
  }
  return -1;
}

int main()
{  int i,j,k,x,br=0;
   scanf("%d %d",&n,&x);
   for(i=0;i<n;i++) scanf("%d",&s1[i]);
   for(i=0;i<n;i++) scanf("%d",&s2[i]);
   sort(s2,s2+n);
   for(i=0;i<n;i++)
     if(x>s1[i])
     {  j=bin_search(x-s1[i]);
        if(j!=-1)
        {  br++;
           k=j-1;while(k>=0&&(s2[k]==x-s1[i])){br++;k--;}
           k=j+1;while(k<n&&(s2[k]==x-s1[i])){br++;k++;}
        }
     }
   printf("%d\n",br);
   return 0;
}
```

*Автор: Красимир Манев*