НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА
Областен кръг, 10 март 2007 г.
Тема за група А (12 клас)

Задача А1. ПОДЯЛБА

Двама братя наследили N парцела земя с лица S_1, S_2,…, S_N, съответно, и решили да си ги поделят „по братски". Помогнете им да направят това. Напишете програма **SPLIT**, която разбива множеството от парцели на две подмножества така, че ако A е сумата от лицата на парцелите в първото подмножество, а B – на парцелите във второто, то абсолютната стойност на разликата $A - B$ да е минимална.

Вход

На първия ред на стандартния вход ще бъде зададено числото N ($2 \le N \le 512$). На втория ред, разделени с по един интервал, ще бъдат зададени числата S_1, S_2,…, S_N ($S_1+S_2+\ldots+S_N \le 200\,000$).

Изход

На първия ред на стандартния изход програмата трябва да изведе броя на парчетата земя в едно от двете подмножества, при което се получава най-добро разделяне. На втория ред програмата трябва да изведе, разделени с по един интервал, поредните номера на парчетата земя в това множество (поредните номера се определят от реда, по който са били зададени лицата на входа).

ПРИМЕР

Вход	Изход
5 1 9 5 3 8	3 1 2 4

РЕШЕНИЕ

Задачата за „братска подялба" е типична за използването на схемата „динамично оптимиране" (ДП). Всъщност тази задача е *NP* – пълна и в общия случай не е известно друго решение освен пълно изчерпване. Възможността да се приложи ДП в този случай се дължи на две особености. На първо място е фактът, че стойностите на зададените лица са цели числа. При нецели стойности на лицата подходът не може да бъде приложен. На второ място, това е ограничението за големината на сумата $S = S_1+S_2+\ldots+S_N \le 200\,000$. Както ще видим, предлаганото решение използва двумерен масив от байтове с размер N на $\lfloor S/2 \rfloor$ и при зададените стойности ще е необходима памет от 512.200000/2 байта, т.е. около 50MB.

Задачата е упростен вариант на „задачата за раницата" и решението много прилича на решението на тази популярна задача. Целта е да намерим число, колкото може по близко до $\lfloor S/2 \rfloor$ (размерът на раницата в този случай), което може да се представи като сума на някои от зададените числа. Нека $T_{i,j}$ е елемент на матрица с размери $N + 1$ на $\lfloor S/2 \rfloor + 1$, i=0,1,…N, а j = 0,1,…, $\lfloor S/2 \rfloor$. Стойностите на $T_{i,j}$ ще са булеви. $T_{i,j}$=1, ако j може да се представи като сума на някои от числата S_1, S_2,…, S_i и $T_{i,j}$=0, ако това не е възможно. При предположението, че лицата са положителни числа и $N \ge 2$, $T_{0,j}=T_{i,0}=0$. В общия случай, за стойността на $T_{i,j}$ ще имаме $T_{i,j}$ = 1 ако

а) $T_{i-1,j} = 1$, защото ако j може да се представи като сума на някои от числата $S_1, S_2,\ldots, S_{i-1}$, то j може да се представи и като сума на някои от числата $S_1, S_2,\ldots, S_i$);
или

б) ако $j - S_i \geq 0$ и $T_{i-1,j-Si} = 1$ защото ако $j - S_i$ може да се представи като сума на някои от числата $S_1, S_2,\ldots, S_{i-1}$, то j може да се представи като към тази сума се добави числото S_i.

Ако по време на пресмятанията получим стойност 1 за някой от елементите в реда с номер R и стълба с номер $C = \lfloor S/2 \rfloor$, това ще означава че сме намерили решение на задачата и може да прекратим попълването на таблицата. Ако това не се случи, за намиране на решението трябва да претърсим стълбовете, започвайки от този с номер $\lfloor S/2 \rfloor - 1$ и вървейки назад към началото на таблицата, до намиране на елемент $T_{R,C} = 1$. За да получим исканията списък с номера на парцелите, трябва да направим „обратен ход" по съдържанието на таблицата. Ако $T_{R,C} = 1$, защото $T_{R-1,C} = 1$, тогава просто намаляваме R с 1, а ако $T_{R,C} = 1$, защото $T_{R-1,C-Sr} = 1$, тогава запомняме R като номер на парцел от търсеното множество и намаляваме R с 1, а C с S_R.

```
#include <stdio.h>
#define MAXN 513
#define MAXS 100001
char t[MAXN][MAXS]={0};int s[MAXN],s1[MAXN];
int main()
{
  int i,j,N,sum=0,r=0,c=0,sum1=0,SS;
  scanf("%d",&N);
  for(i=1;i<=N;i++) {scanf("%d",&s[i]);sum+=s[i];t[i][0]=1;}
  t[1][s[1]]=1;SS=sum;sum=sum/2;
// "прав ход" на ДП
  for(i=2;i<=N;i++)
  {
     for(j=1;j<=sum;j++)
        if(t[i-1][j]==1||s[i]<=j&&t[i-1][j-s[i]]==1) t[i][j]=1;
     if(t[i][sum]==1) {r=i;c=sum;break;}
 }
// търсене на оптимума
 if(r==0)
    for(j=sum-1;j>0;j--)
    {
       for(i=1;i<=N;i++)
          if(t[i][j]==1) { r=i;c=j;break;}
       if(r!=0) break;
    }
// "обратен ход" на ДП
  j=0;
  while(c>0)
  {
     if(t[r-1][c]==1)r--;
     else {s1[++j]=r;c-=s[r--];}
  }
  printf("%d\n",j);
  for(i=1;i<=j;i++)
  {  printf("%d",s1[i]);
     if(i<j) printf(" ");
     else printf("\n");
  }
  return 0;
}
```