

Задача D2. ДВОЙКИ

Пояснение към решението

Бавно решение с изчерпващо търсене – файл pair_31p.cpp

Програмата прочита числата от дадената редица в масива `a[]`. След това се извършва изчерпващо търсене, където резултатът се пресмята в променливата `s`:

```
long long int c=0;
for(int i=0;i<n;i++)
for(int j=i+1;j<n;j++)
    if(a[i]+a[j]==s) c++;
```

По-добро решение с изчерпващо търсене – файл pair_38p.cpp

Използваме, че в дадената редица няма повтарящите се числа. Затова, след намиране във вътрешния цикъл на двойка елементи, чиято сума е равна на `s`, излизаме от този цикъл, понеже не е възможно за друга стойност на индекса `j` да има двойка елементи със същата сума:

```
long long int c=0;
for(int i=0;i<n;i++)
for(int j=i+1;j<n;j++)
    if(a[i]+a[j]==s) {c++; break;}
```

По-бързо решение с двоен цикъл – файл pair_69p.cpp

Сортираме масива в растящ ред. Тогава при намиране във вътрешния цикъл на двойка елементи, чиято сума е по-голяма от `s`, излизаме от този цикъл, понеже за всички следващи стойности на `j` тази сума ще става още по-голяма и не е възможно да стане равна на `s`:

```
sort(a,a+n);
long long int c=0;
for(int i=0;i<n;i++)
for(int j=i+1;j<n;j++)
{
    if(a[i]+a[j]==s) c++;
    else if(a[i]+a[j]>s) break;
}
```

Бързо решение с единичен цикъл – файл pair_alt.cpp за 100 т.

Използва се методът на двете показалки. След сортиране на елементите от дадената редица, полагаме `int L=0` за лява показалка, която в началото сочи към индекс 0 на масива `a[]` и `int R=n-1` за дясна показалка, която в началото сочи към последния елемент на масива `a[]`, който има индекс `n-1`. При работата на цикъла `while (L<R)` показалката `L` се движи надясно, а показалката `R` – наляво и това продължава докато `L<R`:

```

while (L < R)
{
    if (a[L] + a[R] < s) L++;
    else if (a[L] + a[R] > s) R--;
    else {res++; L++; R--;}
}

```

Показалката L сочи към по-малките числа от масива, а показалката R – към по-големите. Постепенно L започва да сочи към все по-големи числа, а R – към все по-малки. При стъките на цикъла, когато сумата $a[L] + a[R]$ е по-малка от s , увеличаваме L , а когато тази сума е по-голяма от s – намаляваме R . Когато се получи, че сумата е равна на s , увеличава брояча res и след излизане от цикъла, програмата отпечатва стойността на този брояч.

Най-бързо решение с единичен цикъл – файл `pair.cpp` за 100 т.

Използваме спомагателен масив $b[k]$, в който маркираме с единица елементите му за тези негови индекси $k > 0$ за които $k = s - a[i]$ за някой индекс i :

```

for(int i=0; i<n; i++)
    if(s-a[i]>0) b[s-a[i]]=1;

```

Така, ако за някой елемент на редицата $a[j]$ имаме, че $b[a[j]] == 1$, това означава че за друг елемент от редицата $a[i]$ е вярно, че $a[j] == s - a[i]$ и за този друг елемент също е вярно, че $a[i] == s - a[j]$, т.е. намерили сме два елемента от редицата (а именно $a[j]$ и $a[i] == s - a[j]$), за които $a[i] + a[j] == s$. Програмата преброява всички такива случаи чрез:

```

long long int c=0;
for(int j=0; j<n; j++)
    if(b[a[j]]==1)
        if(s-a[j]<a[j]) c++;

```

Броячът c се увеличава само когато $s - a[j] < a[j]$, т.е. когато $a[i] < a[j]$, за да се избегне двойното броене (да не се броят два пъти двойките $(a[i], a[j])$ и $(a[j], a[i])$), и да не се брои случаят, когато $a[i] == a[j]$.

Емил Келеведжиев