АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
БРОЙ НА ИНВЕРСИИТЕ

Директната реализация, в която проверяваме за всички двойки индекси i<j, дали a[i] > a[j] има сложност $O(n^2)$ и е прекалено бавна за големи стойности на n.

Друга реализация, пак със сложност $O(n^2)$, но показваща добро поведение при почти подреден масив е основана на метода за сортиране чрез просто вмъкване. При вмъкването на елемента $x = a[i]$ в сортираната подредица $a[1]$, $a[2]$, …, $a[i-1]$, той заема позиция $j+1$, като е прескочил $i–j–1$ елемента, защото е бил по-голям от тях и следователно е образувал с тях инверсии в първоначалната пермутация.

```
long long cnt = 0;
for(int i=1; i<n; i++)
{ int x = a[i];
  int j=i-1;
  while(j>=0 && a[j]>x)
  { a[j+1]=a[j]; j--; }
  a[j+1]=x ;
  cnt = cnt + (i-j-1); // a[i] има (i-j-1) инверсии
                       // с елементите преди него
}
cout << cnt << endl;
```

За получаването на решение със сложност $O(n \log n)$ приспособяваме метода сортиране чрез сливане. Да предположим, че подмасивите от позиция p до позиция $m–1$ и от позиция m до позиция $q–1$ вече са подредени по големина и предстои сливането им. Искаме да преброим колко са инверсиите в целия интервал $[p,q-1)$. Ако има инверсия между два елемента $b[i]$ и $b[j]$, това може да се случи само, ако $p \le i < m$, $m \le j < q$. При това, ако $b[j]$ образува инверсия с $b[i]$, т.е. $b[i] > b[j]$, елементът на позиция j ще образува инверсии и с всички елементи $b[i+1]$, $b[i+2]$, …, $b[m–1]$, защото те са по-големи от $b[i]$. Общо броят на елементите, с които $b[j]$ образува инверсии е $m– i$, където i е първият индекс, за който $b[i] > b[j]$.

```
// merge [p,m) and [m,q)
for(int i=p; i<q; i++) b[i] = a[i];
int i=p, j=m, k=p;
while(i<m && j<q)
{ if(b[i]<b[j]) { a[k]=b[i]; i++; k++; }
  else {cnt=cnt+m-i; a[k]=b[j]; j++; k++; }
}
```

Автор: Стоян Капралов