The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; ***premature optimization is the root of all evil (or at least most of it) in programming****.* DONALD KNUTH - **Computer Programming as an Art**

Sortiranje niza (C i C++)

Sortiranje (uređivanje) niza podataka ne mora da se obavlja samo nad podacima numeričkog tipa, već se mogu sortirati i niske, datumi, ...
Poredak sortiranja može biti strogo rastući, neopadajući, strogo opadajući, nerastući.
Poznat je veći broj algoritama za sortiranje. Oni se među sobom razlikuju po složenosti, efikasnosti, potrebama za memorijskim prostorom. Ako algoritam za sortiranje ne koristi dodatni memorijski prostor (sem prostora u koji je smešten niz), onda je to tzv. algoritam sortiranja u mestu.

Sortiranje u rastućem poretku koriscenjem funkcije sort iz biblioteke algorithm

1. Kreirati sledeći C++ projekat u razvojnom okruženju

```
 #include <iostream>
#include <algorithm>
using namespace std;
#define MAXN 200
int main()
{
   int a[MAXN];
   int n,i;
// iskljucujemo sinhronizaciju da bi ucitavanje teklo brze
   ios::sync_with_stdio(false);
   cin >>n;


   for (i=0; i<n; i++)   cin >> a[i];
   sort(a, a + n);
   for (i=0; i<n;  i++) cout << a[i] << " ";
   return 0;
}
```

2. **Sortiranje u opadajućem poretku koriscenjem funkcije sort iz biblioteke algorithm i korisničke funkcije poredi**

```
#include <iostream>
#include <algorithm>
using namespace std;
```

```
#define MAXN 200
bool poredi (int i,int j) { return (i>j); }
int main()
{
   int a[MAXN];
   int n,i;
// iskljucujemo sinhronizaciju da bi ucitavanje teklo brze
   ios::sync_with_stdio(false);
   cin >>n;

   for (i=0; i<n; i++)   cin >> a[i];
   sort(a, a + n, poredi);
   for (i=0; i<n;  i++) cout << a[i] << " ";
   return 0;
}
```

3. Obrtanje niza korišćenjem funkcije reverse iz biblioteke algorithm

```
#include <iostream>
#include <algorithm>
using namespace std;
#define MAXN 200

int main()
{
   int a[MAXN];
   int n,i;
// iskljucujemo sinhronizaciju da bi ucitavanje teklo brze
   ios::sync_with_stdio(false);
   cin >>n;

   for (i=0; i<n; i++)   cin >> a[i];
   reverse(a, a + n);
   for (i=0; i<n;  i++) cout << a[i] << " ";
   return 0;
}
```

4. **Sortiranje u delimično rastućem poretku korišćenjem funkcije partial_sort iz biblioteke algorithm**

```
#include <iostream>
#include <algorithm>
using namespace std;
#define MAXN 200

int main()
{
   int a[MAXN], b[]={8,7,6,5,4,3,2,1,0};
   int n,i;
// iskljucujemo sinhronizaciju da bi ucitavanje teklo brze
   ios::sync_with_stdio(false);

   partial_sort(b, b+5, b + 9);
   for (i=0; i<9;  i++) cout << b[i] << " ";
   cout <<"\nUnesite broj clanova niza a ";
   cin >>n;
   cout <<"\nUnesite clanove niza a ";
   for (i=0; i<n; i++)   cin >> a[i];
   partial_sort(a, a+n/2, a + n);
   for (i=0; i<n;  i++) cout << a[i] << " ";
   return 0;
}
```

5. **Sortiranje u delimično opadajućem poretku korišćenjem funkcije partial_sort iz biblioteke algorithm**

```
#include <iostream>
#include <algorithm>
using namespace std;
#define MAXN 200
bool poredi (int i,int j) { return (i>j); }
int main()
{
   int a[MAXN], b[]={8,7,6,5,4,3,2,1,0};
   int n,i;
// iskljucujemo sinhronizaciju da bi ucitavanje teklo brze
   ios::sync_with_stdio(false);

   partial_sort(b, b+5, b + 9, poredi);
   for (i=0; i<9;  i++) cout << b[i] << " ";
   cout <<"\nUnesite broj clanova niza a ";
   cin >>n;
   cout <<"\nUnesite clanove niza a ";
   for (i=0; i<n; i++)   cin >> a[i];
   partial_sort(a, a+n/2, a + n, poredi);
   for (i=0; i<n;  i++) cout << a[i] << " ";
   return 0;
}
```

6. **Poznati prodavac jabuka Steva izabrao je n jabuka koje će odneti na pijacu na prodaju. Kada je stigao na pijacu, uvideo je da toga jutra je konkurencija dosta jaka. Mudri Steva je odlučio da svoju ponudu pospeši tako što će jabuke na svojoj tezgi aranžirati na specijalan način: najlakša jabuka će stajati krajnje levo, sledeća najlakša jabuka će stajati u krajnje desno na tezgi. Ovaj proces se nastavlja sve dok Steva ne postavi sve jabuke, ali tako da najteža jabuka bude u sredini.**

Napišite programa, koji prikazuje raspored Stevinih jabuka.
U prvoj liniji standardnog ulaza će biti predstavljen broj jabuka- *n* (3<n<50). Nakon toga sledi *n* pozitivnih celih brojeva (ne većih od 100), u kojima je izražena masa svake jabuke u kilogramima. Na standardni izlaz ispisati u jednoj liniji masu jabuka posle aranžiranja.

PRIMER

ULAZ	IZLAZ
5 1 2 3 4 5	1 3 5 4 2

C++ REŠENJE:

```
#include<iostream>
#include<algorithm>
using namespace std;
int main ()
{
   int n,a[100],b[100];
   int i,j,z=0;
   cin>>n;
   for ( i=0; i<n; i++) cin>>a[i];

   sort(a,a+n);
   i--;

   for ( j=0; j<n; j++)
   {
    if(j%2!=0)
       {
         b[i]=a[j];i--;
       }
    else
    {b[z]=a[j];z++;}
  }
   for(i=0;i<n;i++)cout<<b[i]<<" ";
   cout<<endl;
  return 0;
}
```

7. Dat je niz znakova $(s_1, ..., s_n)$ dužine n. Znak s_i je + ili -. Dat je niz od $n + 1$ brojeva $(a_1, ..., a_{n+1})$.

Treba rasporediti brojeve u dati niz znakova tako da vrednost dobijenog izraz bude maksimalna. Formalno, treba naći vrednost *val* koja je definisana na sledeći način:

$$val = \max\{a_{p(1)}\ s_1 \ldots a_{p(n)}\ s_n\ a_{p(n+1)} \mid p \text{ je permutacija brojeva od 1 do } n + 1\}$$

Ulaz.
(Ulazni podaci se učitavaju sa standardnog ulaza.) U prvom redu standardnog ulaza nalazi se

prirodan broj n ($1 \leq n \leq 100000$). U drugom redu nalazi se n znakova, redom od s_1 dos_n. Svaki znak je karakter '+' ili '-'. Znakovi nisu odvojeni razmakom. U trećem redu se nalazi $n + 1$ brojeva, brojevi od a_1 do a_{n+1}, odvojenih razmakom. Svaki od tih brojeva je iz intervala [0, 1000000].

Izlaz.

(Izlazne podatke ispisati na standardan izlaz.) U prvi i jedini red standardnog izlaza ispisati traženu vrednost, odnosno vrednost *val*.

Primer 1.

standardni ulaz	**standardni izlaz**
3 +-+ 1 2 3 4	8

Objašnjenje.

Jedno rešenje predstavlja raspored brojeva
2+3-1+4

Primer 2.

standardni ulaz	**standardni izlaz**
2 ++ 1 3 2	6

Objašnjenje.

Bilo koji raspored brojeva vodi ka optimalnom rešenju.

Primer 3.

standardni ulaz	**standardni izlaz**
4 ---- 3 12 1 2 0	6

Objašnjenje.

Bilo koji raspored takav da je na prvom mestu broj 12 vodi ka optimalnom rešenju.

Rešenje:

Prebroji se broj znakova – pri cemu se kod prvog sabirka podrazumeva da je pozitivan, i rezultat se dodeli promenljivoj **cnt**. Sortira se niz **a**, u neopadajući. Koliko je vrednost na promenljivoj **cnt** toliko najmanjih članova niza se oduzme od sume, a ostali članovi (koji su među većim) se saberu.

```
/* C++ projekat */
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#define ffor(_a,_f,_t) for(int _a=(_f),__t=(_t);_a<__t;_a++)
#define FOR(__i, __n) ffor (__i, 0, __n)

using namespace std;
 const int MAXN = 100001;
 int a[MAXN];
 char str[MAXN + 10];

int main(){
  int n;
```

```
  scanf("%d", &n);
  scanf("%s", str);
  FOR (i, n + 1)
    scanf("%d", &a[i]);

  int cnt = 0;
  FOR (i, n)
    cnt += str[i] == '-';
  sort(a, a + n + 1);
  long long ret = 0LL;
  FOR (i, n + 1)
    if (i < cnt)
      ret -= a[i];
    else
      ret += a[i];

  printf("%lld\n", ret);
  return 0;
}
```

8.
Sortiraj Stabilno

- Vremensko ograničenje: 1 s
- Memorijsko ograničenje: 1000 mb

Dat je niz parova celih brojeva <A[i], B[i]>. Kazemo da je par <A[i], B[i]> manji od <A[j], B[j]> ukoliko vazi jedan od sledeca dva uslova:
1. A[i] < A[j]
2. A[i] = A[j], i < j

Mozemo primetiti da ne postoje dva elemente u nizu koja su jednaka. Vas zadatak je da ispisete ovaj niz parova celih brojeva u rastucem redosledu.

Input

Prvi red standardnog ulaza sadrzi prirodan broj N (1 <= N <= 100 000) koji predstavlja broj elemenata niza A i B. Drugi red sadrzi N prirodnih brojeva, razdvojenih jednim znakom razmaka, koji predstavljaju elemente niza A. Naredni red sadrzi N prirodnih brojeva, razdvojenih jednim znakom razmaka, koji predstavljaju elemente niza B. (-10^9 <= A[i], B[i] <= 10^9)

Output

U prvi red standarnog izlaza ispisati niz A posle sortiranja. U sledeci red ispisati niz B posle sortiranja.

Example

Example Input	Example Output
5 1 -1 1 92 -55 4 5 1 -1 2	 -55 -1 1 1 92 2 5 4 1 -1

Rešenje: Nama je potrebno da u ovom zadatku iskoriristimo stabilnost algoritma sortiranja. U biblioteci STL postoji implementacija stabilnog algoritma za sortiranje stable_sort koji garantuje da će elementi sa istom vrednosti biti poredjani onim redosledom u kom su bili pre sortiranja. Detaljan opis STL biblioteke http://www.cplusplus.com/reference/algorithm/

```
#include <cstdio>
#include <algorithm>
#define MAXN 100000
using namespace std;

struct s {
   int a, b;
} niz[MAXN];

bool cmp(s x, s y) {
   return x.a < y.a;
}

int main() {
   int n;
   scanf("%d", &n);

   for (int i = 0; i < n; i++) scanf("%d", &niz[i].a);
   for (int i = 0; i < n; i++) scanf("%d", &niz[i].b);

     stable_sort(niz, niz+n, cmp);

   for (int i = 0; i < n-1; i++) printf("%d ", niz[i].a);
   printf("%d\n", niz[n-1].a);
   for (int i = 0; i < n; i++) printf("%d ", niz[i].b);
   return 0;
}
```

9. Dat je ceo broj s i niz različitih celih brojeva. Napisati program kojim se određuje broj parova u nizu koji imaju zbir jednak datom broju s. U prvoj liniji standardnog ulaza nalazi se ceo broj s, u drugoj liniji nalazi se broj elemenata niza n(1 <n < 1000), a u sledećih n linija nalazi se redom elementi niza.

ULAZ

```
5
6
1
4
3
6
-1
5
```

IZLAZ

2

RESENJE 1

Zadatak možemo rešiti analizirajući sumu elemenata svakog para u nizu a.
Različiti parovi niza a su (a[i], a[j]) gde i uzima vrednosti od 0 do n-2, a j uzima vrednosti od i + 1 do n - 1.

```
#include <iostream>
#include<algorithm>
using namespace std;
int traziX(int x, int a[], int l, int d)
{
   while (l <= d)
   {
      int s = l + (d - l) / 2;
      if (a[s] == x)
         return s;
      if (x > a[s])
         l = s + 1;
      else
         d = s - 1;
   }
   return -1;
}

int main() {
  int s,n;
  cin>>s>>n;
  int a[1000];
  for (int i = 0; i < n; i++)
  {
     cin>>a[i];
  }
  sort(a,a+n);
  int brojParova = 0;
  for (int i = 0; i < n - 1; i++)
  {
     int p = traziX(s - a[i], a, i + 1, n-1);
     if (p != -1)
     {
        brojParova++;
     }
  }
  cout<<brojParova<<endl;
  return 0;
}
```

RESENJE2

```
#include <iostream>
#include<algorithm>
using namespace std;
int main() {
  int s,n;
  cin>>s>>n;
```

```
  int a[1000];
  for (int i = 0; i < n; i++)
  {
     cin>>a[i];
  }
  sort(a,a+n);
  int brojParova = 0;
  int levo = 0, desno = n - 1;
  while (levo < desno)
  {
     if (a[levo] + a[desno] == s)
     {
        brojParova++;
        levo++;
        desno--;
     }
     else if (a[levo] + a[desno] > s)
          {
             desno--;
          }
          else
          {
             levo++;
          }
   }
   cout<<brojParova<<endl;
   return 0;
}
```

RESENJE3

```
#include <iostream>

using namespace std;

int main() {
  int s,n;
  cin>>s>>n;
  int a[1000];
  for (int i = 0; i < n; i++)
  {
     cin>>a[i];
  }
  int brojParova = 0;
  for (int i = 0; i < n-1; i++)
     for(int j=i+1;j<n;j++)
     {
        if (a[i] + a[j] == s)
           brojParova++;
     }
  cout<<brojParova<<endl;
  return 0;
```

}

10. Dato je N celih brojeva izmedju -10000 i 10000. Napisite program koji pronalazi koliko se moze formirati (neuredjenih) trojki brojeva cija je suma 0.

Test primer:
Ulaz
10
2 -5 2 3 -4 7 -4 0 1 -6
Izlaz
6
Objasnjenje:
Moguce trojke su: (2, -5, 3), (2, 2, -4), (2, 2, -4), (-5, 2, 3), (3, -4, 1), (3, -4, 1).
Uocite da broj -4 je dupliran na ulazu, te se neke od trojki zato ponavljaju.

Resenje:

```
#include <cstdio>
#include <string>
#include <algorithm>

#define MAX 10002

using namespace std;

int a[MAX], n;

string toString(long long num)
{
	if (num <= 0) return "0";
	string ret;
	while (num) {ret.push_back(num % 10 + 48); num /= 10;}
	reverse(ret.begin(), ret.end());
	return ret;
}

int binarySearch(int leftPos, int rightPos, int num)
{
	if (a[rightPos] < num) return 0;

	int left, right, mid;
	int lower = rightPos + 1, upper = leftPos - 1;

	// pretraga sleva u intervalu
	left = leftPos; right = rightPos;
	while (left <= right)
	{
		mid = (left + right) / 2;
		if (a[mid] == num) lower = min(lower, mid);
		if (a[mid] < num) left = mid + 1;
		else right = mid - 1;
```

```
        }
        if (lower > rightPos) return 0;

        // pretraga sdesna u intervalu
        left = leftPos; right = rightPos;
        while (left <= right)
        {
                mid = (left + right) / 2;
                if (a[mid] == num) upper = max(upper, mid);
                if (a[mid] <= num) left = mid + 1;
                else right = mid - 1;
        }
        return max(0, upper - lower + 1);
}

long long solve()
{
        sort(a, a + n);
        long long ans = 0;
        for (int i = 0; i < n; i++)
        {
                if (a[i] > 0) break;
                for (int c = i + 1; c < n; c++)
                {
                        if (a[i] + a[c] > 0) break;
                        ans += binarySearch(c + 1, n - 1, -a[i] - a[c]);
                }
        }
        return ans;
}

int main(void)
{

        scanf("%d", &n);
        for (int i = 0; i < n; i++)
                scanf("%d", &a[i]);
       printf("%I64d\n", solve());
//      printf("%s\n", toString(solve()).c_str());
        return 0;
}
```