Pretraga i sortiranje niza (C I C++)

Programski jezik C

Formulacija problema: Neka je dato n elemenata v[0],v[1],..,v[n-1] takvih da
v[0] <= v[1] <=... <=v[n-1].
Za zadati element *x* ustanoviti da li se on pojavljuje u nizu *v*.

Postupak linearnog pretraživanja u uređenoj(sortiranoj) kolekciji

Proveravamo da li se element *x* nalazi u nizu *v* lineranim (sekvencijalnim) pretraživanjem. Redom se upoređuju elementi neopadajućeg niza *v* sa vrednošću od *x*, sve do prvog elementa v[i] za koji važli v[i] >=x.
Tada, ako je:

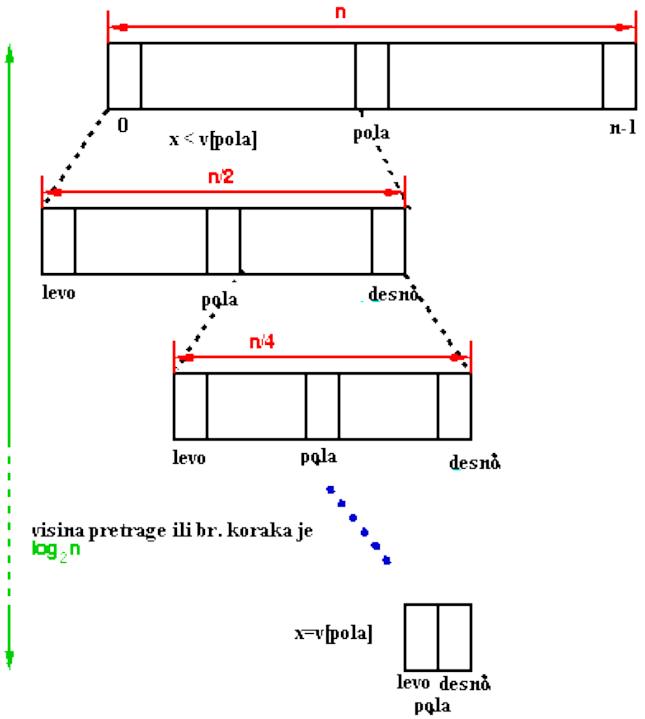
- v[i] = = x, onda je poz=i tražena pozicija
- v[i] > x, onda se x ne nalazi u nizu v

Ako je svaki element niza v manji od vrednosti od x, onda se x ne nalazi u nizu v (izvrsi se n poredjenja).

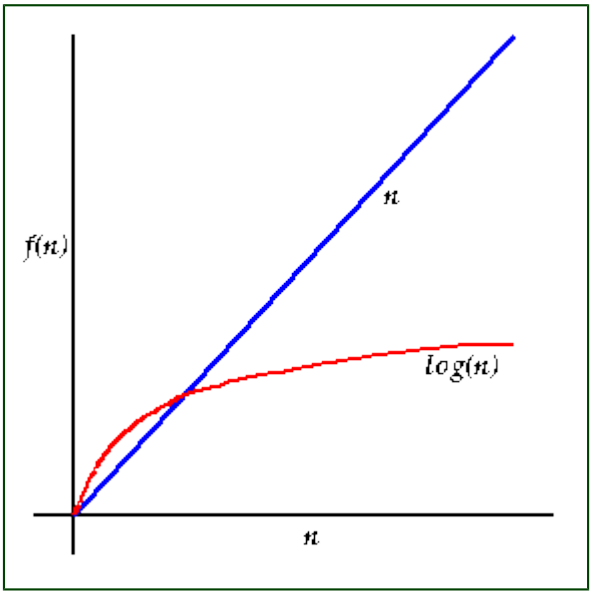
Binarna pretraga

Rešenje:

- 0 <= i <= n-1, tako da v[i]=x (ako je ustanovljeno da se x pojavljuje u nizu v)



Odnos broja poredjenja kod linearne i binarne pretrage



Linearna pretraga u niza karaktera u neuređenoj kolekciji

1. Napisati C program koji sa standardnog ulaza unosi pomocu funkcije scanf nisku sa ne više od 20 karaktera i proverava da li niska sadrzi znak @

```
#include <stdio.h>
#include <string.h>
/* Funkcija proverava da li se dati element x nalazi
   u datom nizu a.
   Funkcija vraca poziciju u nizu na kojoj je x pronadjen
   odnosno -1 ukoliko elementa nema.
*/
int linear_search(char a[], int n, int x)
{
        int i;
        for (i = 0; i<n; i++)
                if (a[i] == x)
                        return i;
                /* nema potrebe za  else!!! */
        return -1;
}

main()
{
        char a[21] ; /*niska sa stdin */
                 int n; /*broj unetih karaktera niske a*/
        int i; /*brojacka promenljiva*/
        printf("Unesite nisku : ");
         scanf("%s",a);
        n = strlen(a);
        i = linear_search(a, n, '@');
        if (i == -1)
                printf("Karakter @ nije nadjen\n");
```

```
        else
                printf("Karakter @  je na poziciji %d\n", i);
}
```

Binarna pretraga niza karaktera

2. Napisati C program koji sa standardnog ulaza unosi pomocu funkcije scanf leksikografski uređenu nisku sa ne više od 20 karaktera i proverava da li niska sadrzi znak @. (ASCII(@) = 64)

```
#include <stdio.h>
#include <string.h>

/* Funkcija proverava da li se element x javlja unutar niske a.
   Funkcija vraca poziciju na kojoj je element nadjen odnosno
   -1 ako ga nema.

   !!!!! VAZNO !!!!!
   Pretpostavka je da je niska a sortirana
*/

int binary_search(char a[], int n, int x)
{
        /* Pretrazujemo interval [l, d] */
        int l = 0;
        int d = n-1;

        /* Sve dok interval [l, d] nije prazan */
        while (l <= d)
        {
                /* Srednja pozicija intervala [l, d] */
                int s = (l+d)/2;

                /* Ispitujemo odnos x i srednjeg elementa a[s]*/
                if (x == a[s])
                        /* Element je pronadjen */
                        return s;
                else if (x < a[s])
                {
                        /* Pretrazujemo interval [l, s-1] */
                        d = s-1;
                }
                else
                {
                        /* Pretrazujemo interval [s+1, d] */
                        l = s+1;
                }
        }
```

```
        /* Element je nadjen */
        return -1;
}

main()
{
        char a[21] ; /*niska sa stdin */
                 int n; /*broj unetih karaktera niske a*/
        int i; /*brojacka promenljiva*/
        printf("Unesite nisku : ");
        scanf("%s",a);

        i = binary_search(a, strlen(a), '@');

        if (i==-1)
                printf("Nema karaktera @\n");
        else
                printf("Pronadjen @ na poziciji %d\n", i);
}
```

3. Napisati C program koji učitava sortirani niz celih brojeva i ceo broj x i poziva implementiranu iterativnu funkcije binarne pretrage ispisuje da li se x pojavljuje u nizu.

```
#include<stdio.h>
#define MAX 10

/* binarno pretrazivanje clana niza a koji je jednak sa x.
   F-ja bin_pret vraca indeks (0..n-1) clana niza a
   koji je jednak vrednosti od x ili -1 ako takav ne postoji */
int bin_pret(int a[], int n, int x);

int main()
{
      int n; /*dimenzija niza */
      int a[MAX];  /*neopadajuci niz celih brojeva
                       sa  ne vise od MAXDUZ elemenata*/
      int x;    /* element koji se trazi */
      int poz;  /* indeks onog clana niza cija je vrednost jednaka
                  vrednosti koja se trazi (ako postoji takav clan niza)*/
       int pom; /* pomocna promenljiva pri kontroli unosa niza u odgivarajucem poretku */
       int i; /* brojacka promenljiva */

     do
       {
          printf("Unesite broj clanova niza izmedju 1 i %d\n", MAX);
          scanf("%d", &n);
       } while (n <1 || n >MAX) ;

      printf("\ta[0]= ");
      scanf("%d", &a[0]);
      pom=a[0];
   /* Unos clanova neopadajuceg niza celih brojeva */
```

```
    for( i=1; i<n; i++)
        {
         printf("\ta[%d]= ", i);
         scanf("%d", &pom);
         if (pom < a[i-1])
          { printf("\nUNOS NE PODRZAVA NEOPADAJUCI POREDAK !!!\n");
            break;
          }
         a[i]=pom;
        }

    if ( i != n) return (-1);

  /* Unos elementa binarne pretrage */
  printf("Unesite element koji se trazi medju clanovima niza\n");
  scanf("%d",&x);

   poz=bin_pret(a,n,x);

  /* Ispis rezultata */
  if(poz == -1)
    printf("\nTrazeni broj se ne nalazi u nizu\n");
  else
    printf("\nBroj %d se nalazi na %d. poziciji unesenog sortiranog niza \n",x,poz+1);

  return 0;

}

int bin_pret(int a[], int n, int x)
{  int gornja, donja, sred ;  /* gornja granica intervala pretrage,
                                 donja  granica intervala pretrage,
                                 sredina intervala */
  donja=0;
  gornja=n-1;
  while (donja <= gornja)  /* traganje se obavlja dok je
                              duzina intervala pretrage >= 0*/
   {
    sred=(donja+gornja)/2; /* polovljenje intervala za pretragu */

    if (x==a[sred]) return sred; /* nadjeno x u nizu a */
    else
      if (x < a[sred]) gornja=sred-1; /*leva polovina tekuceg
                                        intervala je kandidat
                                        za pretragu */
      else donja=sred+1;   /* desna polovina tekuceg intervala je
                              kandidat za novu pretragu */
   }
   return -1;  /* nije nadjeno x u nizu a */
}
```

4. Napisati C program koji učitava sortirani niz celih brojeva i ceo broj x i poziva implementiranu **rekurzivnu (samopozivajuću)** funkcije binarne pretrage ispisuje da li se x pojavljuje u nizu.

```
#include<stdio.h>
#define MAX 10

/* rekurzivno binarno pretrazivanje clana niza a koji je jednak sa x.
   F-ja bin_pret vraca indeks (d..g) clana niza a
   koji je jednak vrednosti od x ili -1 ako takav ne postoji */
int bin_pret(int a[], int d, int g, int x);

int main()
{
     int n; /*dimenzija niza */
     int a[MAX];  /*neopadajuci niz celih brojeva
                    sa  ne vise od MAXDUZ elemenata*/
     int x;    /* element koji se trazi */
     int poz;  /* indeks onog clana niza cija je vrednost jednaka
                vrednosti koja se trazi (ako postoji takav clan niza)*/
     int pom; /* pomocna promenljiva pri kontroli unosa niza u odgivarajucem poretku */
      int i; /* brojacka promenljiva */

    do
     {
        printf("Unesite broj clanova niza izmedju 1 i %d\n", MAX);
        scanf("%d", &n);
      } while (n <1 || n >MAX) ;

     printf("\ta[0]= ");
     scanf("%d", &a[0]);
     pom=a[0];
  /* Unos clanova neopadajuceg niza celih brojeva */
     for( i=1; i<n; i++)
        {
         printf("\ta[%d]= ", i);
         scanf("%d", &pom);
         if (pom < a[i-1])
          { printf("\nUNOS NE PODRZAVA NEOPADAJUCI POREDAK !!!\n");
            break;
          }
         a[i]=pom;
        }

     if ( i != n) return (-1);

  /* Unos elementa binarne pretrage */
  printf("Unesite element koji se trazi medju clanovima niza\n");
  scanf("%d",&x);

   poz=bin_pret(a,0,n-1,x);
```

```
  /* Ispis rezultata */
  if(poz == -1)
    printf("\nTrazeni broj se ne nalazi u nizu!\n");
  else
    printf("\nBroj %d se nalazi na %d poziciji unesenog sortiranig niza \n",x,poz+1);

  return 0;

}

int bin_pret(int a[], int donja, int gornja, int x)
{  int sred ;  /* sredina intervala pretrage*/

  if (donja > gornja)  /* traganje se prekida kada
                          duzina intervala pretrage postane < 0 */
   return -1;

  else
   {
    sred=(donja+gornja)/2; /* polovljenje intervala za pretragu */

    if (x==a[sred]) return sred; /* nadjeno x u nizu a */
    else
      if (x < a[sred]) return bin_pret(a, donja,sred-1, x);
                                    /*leva polovina tekuceg
                                    intervala je kandidat
                                    za pretragu */
      else return bin_pret(a,sred+1,gornja, x);
                  /* desna polovina tekuceg intervala je
                        kandidat za novu pretragu */
   }
}
```

Bibliotečka funkcija bsearch iz <stdlib.h>

void *bsearch(const void *key, const void *base, size_t n, size_t size, int (*cmp)(const void *, const void *))

Funkcija *bsearch* pretražuje *base[0]...base[n-1]* ne bi li našla element jednak sa **key*. Funkcija *cmp* mora vratiti:

1. negativnu vrednost, ako je njen prvi argument (kljuc pretrage) manji od drugog agumenta
2. nulu, ako su jednaki prvi i drugi argument
3. pozitivnu vrednost, ako je prvi argument veci.

Funkcija poređenja dobija pokazivače na elemente. Svi pokazivači su deklarisani kao void * i moraju se ponovo konvertovati u svoj pravi tip (kao u primeru).

Funkcija *bsearch* vraca pokazivač na prvi element jednak sa *key, ili NULL ako on ne postoji.

```
int cmp(const void *p1, const void *p2)
{  int i= *(int *)p1; /* vraca ceo broj na koji se pokazuje */
   int j= *(int *)p2;
      return ( i < j ? -1 : (i==j ? 0: 1) );
}

void main()
{
  int dNiz[]={-1, 3, 8, 26, 28, 29, 30, 35, 45};
  int i=30;
  void *p=bsearch(&i, dNiz, 9, sizeof(int), &cmp);
   if (p==NULL) printf("\nnije nadjen\n"); else printf("\nPronadjen je i\n");
}
```

5. Napisati C program koji učitava ceo broj x i poziva bibliotečku funkcije binarne pretrage ispisuje da li se x pojavljuje u sortiranom nizu koji je zadat u programu eksplicitnim navođenjem članova.

```
#include <stdio.h>
#include <stdlib.h>

/* inicijalizacija niza*/
int niz[]={1, 22, 56, 87, 92, 101};

/* funkcija za poredjenje podataka koju koristi bsearch */
int poredi(const void *a, const void *b)
{
       return   (* (int *)a -* (int *)b);
}


int nadji(int kljuc)   /* trazi element zadatog kljuca u nizu binarnom pretragom
i vraca nula u slucaju da je pretraga
                                        neuspesna, odnosno nenula vrednost u
slucaju pronadjenog elementa*/
{
      int *p;  /*pokazivac rezultata pretrage */
       p= (int *) bsearch(&kljuc, niz, 6, sizeof(int), poredi);
      return (p !=NULL);
}


main()
{  int el;

    printf("Unesite element koji trazite u nizu: ");  scanf("%d",&el);
    printf("\n\nTrazeni element %s postoji u nizu\n", nadji(el)?"":"ne");

    return 0;
}
```

Domaći rad

1.

Kuvar želi da gostima na proslavi rođendana spremi svoj omiljeni specijalitet.

Kuvarov omiljeni specijalitet sastoji se od N sastojaka. Za pripremu jedne porcije potrebna je određena količina svakog sastojka.
Određene količine nekih sastojaka već postoje u kuhinji, ostatak treba kupiti. U prodavnici se prodaju svi sastojci, a svaki sastojak je dostupan u manjem ili većem pakovanju.

Kuvar ima D dinara i želi da ih potroši tako da od ukupne količine sastojaka može da se napravi što više porcija njegovog omiljenog jela.

U prvom redu nalaze se dva prirodna broja N i D, $1 \leq N \leq 100$, $1 \leq D \leq 100\,000$
U svakom od sledećih N redova nalazi se po 6 prirodnih brojeva, koji predstavljaju podatke o jednom sastojku, redom:

- X, $10 \leq X \leq 100$, količina sastojka potrebna za jednu porciju kuvarovog specijaliteta u gramima
- Y, $1 \leq Y \leq 100$, količina sastojka koja već postoji u kuhinji u gramima
- SM, $1 \leq SM < 100$, veličina manjeg pakovanja sastojka u prodavnici u gramima
- CM, $10 \leq CM < 100$, cena manjeg pakovanja sastojka u dinarima
- SV, $SM < SV \leq 100$, veličina većeg pakovanja sastojka u prodavnici u gramima
- CV, $CM < CV \leq 100$, cena većeg pakovanja sastojka u dinarima

U prvom i jedinom redu ispisati najveći mogući broj porcija koje kuvar može da pripremi, ako pametno potroši novac kojim raspolaže.

Primer ulaza	Primer izlaza
2 100 10 8 10 10 13 11 12 20 6 10 17 24	5

Za 99 dinara kuvar će kupiti 3 manja i jedno veće pakovanje prvog sastojka i jedno manje i dva veća pakovanja drugog sastojka ($3 \cdot 10 + 1 \cdot 11 + 1 \cdot 10 + 2 \cdot 24 = 99$).
Kuvar će tada imati na raspolaganju 51 gram (8 + 3 * 10 + 1 * 13) prvog sastojka i 60 grama (20 + 1 * 6 + 2 * 17) drugog sastojka, što je dovoljno za 5 porcija.

2. Zadat je histogram sa N celobrojnih intervala, a[0]…a[N-1], i N celobrojnih visina tih intervala, h[0]…h[N-1]. Potrebno je pronaći stranicu najvećeg kvadrata koji se može upisati u histogram.

U prvom redu nalazi se ceo broj N ($0 < N < 100\,000$).
U narednih N redova se nalazi par celih brojeva a[i] i h[i], $0 < h[i] < 1\,000\,000\,000$,
$0 < a[i] < 1\,000\,000\,000$
Potrebno je ispisati dužinu stranice najvećeg kvadrata koji može da se upiše u histogram

Primer ulaza	Primer izlaza
6 1 3 2 5 1 6 2 1 2 2 1 1	3

Sortiranje

Formulacija problema:Zadatih n elemenata a[0],a[1],..,a[n-1] urediti u neopadajući poredak.
Rešenje: niz različitih indeksa $0<=i_1,i_2,..., i_n <=n-1$ tako da vazi: $a[i_1] <= a[i_2] <=... <=a[i_n]$

```
Bubble sort        Selection sort      Quick sort
```

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.

Programski jezik C i C++

Sortiranje koriscenjem funkcije sort iz biblioteke algorithm.h

1. kreirati C++ projekat u razvojnom okruženju

```
 #include <cstdio>
#include <algorithm>
using namespace std;
#define MAXN 200
int main()
{
int a[MAXN];
int n,i;
scanf("%d", &n);
for (i=0;i<n; ;i++)   scanf("%d", &a[i]);
sort(a, a + n);
for (i=0;i<n; ;i++) printf("%d ", a[i]);
return 0;
}
```

1. Napisati funkciju koji određuje dva najveca elementa niza a i program koji je testira. Elementi niza se unose sa standardnog ulaza sve dok se ne unese 0. Pretpostaviti da niz neće imati više od 32 elemenata.

Primer 1
ulaz 1 2 3 4 5 6 0
izlaz 5 6

Primer 2
ulaz: 3 4 -1 2 3 0
izlaz 3 4
Primer 3
ulaz 4 5 4 5 2 0
izlaz 5 5

2. *Unikat* je član niza koji se pojavljuje tačno jednom u nizu. Napisati program koji učitava dimenziju niza, a potom i niz celih brojeva i nalazi i ispisuje najveći unikat tog niza. Dimenzija niza je n, $3<n<100$. Voditi računa o vremenskoj složenosti programa.

PRIMER 1
ULAZ
5
1 -2 36 9 1

IZLAZ
36

PRIMER 2
ULAZ
16
1 2 3 2 1 -8 3 67 1 13 28 1 67 13 4 6

IZLAZ
28

3. 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 lubenica 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
5
1 2 3 4 5

IZLAZ
1 3 5 4 2

RESENJE:

C

C++

```
#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;
	}

4. Napisati C ili C++ program koji učitava dimenziju niza celih brojeva, a potom i sam niz. Program treba da ispiše najveću razliku između bilo koja dva člana niza.

ULAZ IZLAZ
7 21
3 8 -10 2 11

5. 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:

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

Ulaz.
(Ulazni podaci se učitavaju sa standardnog ulaza.) U prvom redu standardnog ulaza nalazi se prirodan broj n ($1 \le n \le 100000$). 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 8
+-+
1 2 3 4
Objašnjenje.
Jedno rešenje predstavlja raspored brojeva
2+3-1+4

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

Objašnjenje.
Bilo koji raspored brojeva vodi ka optimalnom rešenju.

Primer 3.

standardni ulaz	**standardni izlaz**
4	6

---
3 12 1 2 0
Objašnjenje.
Bilo koji raspored takav da je na prvom mestu broj 12 vodi ka optimalnom rešenju.

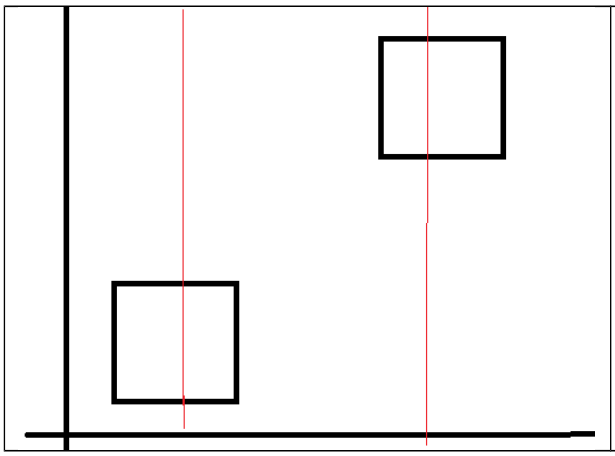
Rešenje:
Prebroji se broj znakova – pri cemu se kod prvog sabirka podrazumeva da je pozitivan, i rezultat se dodeli promenljivoj **cnt**. Sortira se niz **a**, u neopadajući. Koliko je vrednost na promenljivoj **cnt** toliko najmanjih članova niza se oduzme od sume, a ostali članovi (koji su među većim)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

6. Posle dugorocne saradnje na plantažama Jagoda, farmeri Šomi, Dragance i Đurica su se posvađali, te su poželeli da podele plantaže. Pošto je Viskonsin (Država iz koje su naši farmeri) poznata po tome da ima mocvarno zemljište farmeri su pravili plantaže u obliku kvadrata. Da bi se spasili papirologije odlucili su da zemljište podele linijama koje idu od severa ka jugu tako da svako dobije neke od plantaža (i neke delove ostalih). Njihov komšija Zlatevic je uzeo plan u ruke i rekao da je problem lako rešiti. Medutim u meduvremenu farmer Zlatevic se izgubio jureci zlatnu žabu u šumi. Tako da na vama ostaje da pomognete farmerima da reše svoj problem.
Ulazni podaci se ucitavaju sa standardnog ulaza. U prvom redu standardnog ulaza se nalazi ceo broj N (1<= N <=200), koji predstavlja ukupan broj plantaža koje poseduju Šomi, Dragance i Đurica. U svakom od sledecih redova nalaze se tri realna broja X, Y, i A, (0 <= X, Y, A <= 30000). X, Y predstavljaju koordinate pocetka plantaže na mapi (Jugo-Zapadni košak ili donji levi ugao plantaže), dok A predstavlja dužinu strane plantaže.
Na standardni izlaz, u dva reda treba ispisati dva realna broja, sa tacnošcu na dve decimale (tj. sa dve cifre iza decimalne tacke). Oni predstavljaju X koordinatu zamišljenih linija koje dele zemljište.
Primer

Ulaz:	Izlaz:
2	67.67
1 1 100	233.33
200 200 100	

Obrazloženje. U datom primeru jednom od farmera ce pripasti dve trecine prvog kvadrata (plantaže), drugom trecina prvog i trecina drugog, i najzad trecem ce pripasti dve trecine drugog kvadrata. Rešenja su ce u svakom test-primeru biti jedinstvena.



7. Dato je n<=200 disjunktnih kvadrata u ravni. Podeliti ravan (I kvadrant) vertikalnim linijama na 3 oblasti tako da su sume površina kvadrata ili delova kvadrata u dobijenim oblastima budu jednake.
Napomena: Pretpostavka da su kvadrati u I kvadrantu i da su im strane paralelne koordinatnim osama. Svaki kvadrat je zadat kordinatama donjeg levog temena i dužinom stranice. Na standardni izlaz, u dva reda treba ispisati dva realna broja, sa tačnošću na dve decimale. Oni predstavljaju **X** koordinatu zamišljenih linija koje dele prvi kvadrant.

Resenje: Upotreba binarne pretrage
Pri unosu donjeg levog temena kvadrata i stranica, izračuna max (najveća vrednost po x-osi do koje doseže desno teme nekog kvadrata), i ukupna površina svih kvadrata (kv).
Koristimo za određivanje prve i druge vertikalne prave funkciju binpretraga.

Binpretraga za prvu vertikalu za segment [0,max] će postaviti pravu(x1=binpretraga(0,max);), tako da levo od nje je aproksimativno trećina ukupne površine.
Koristimo binarnu pretragu tj. delimo segment [l,r] na pola, m:=(l+r)/2, i izračunamo površinu levo od sredine i desno od koordinatnog početka.
a) Ako je rezultat veći od trećine ukupne površine, onda se poziva funkcija sa parametrima binpretraga(l,m);
b) U suprotnom, poziva se funkcija binpretraga(m,r);
c) Kraj rekurzije je kad se postigne tačnost r-l<=0.001.

```
#include <stdio.h>
#include <math.h>

#define eps 1e-8
#define eps2 1e-4
#define KVADRAT(x) ((x)*(x))

#define MAXN 200
#define MAXX 30000


int n ;
double x [MAXN], y [MAXN], d [MAXN] ;
double ukupno = 0.0 ;

double NadjiPovrsinu(double cx) {
  double povrsina = 0.0 ; int i;
  for (i = 0 ; i < n ; i++)
    if (cx >= x [i])
      if (cx <= x [i] + d [i]) povrsina += d [i] * (cx - x [i]) ;
      else povrsina += KVADRAT(d [i]) ;
  return povrsina ;
}

void binpretraga(double xl, double xr, double povrsina, double *rx) {
    double tekPovrsina;
  while (xl< (xr-eps2)) {
    *rx = (xl + xr) / 2 ;
    tekPovrsina = NadjiPovrsinu(*rx) ;
    if (fabs(tekPovrsina- povrsina)<eps) break ;  //resenje!!!
    if (tekPovrsina < (povrsina-eps)) xl = *rx ;
    else xr = *rx ;
  }
}

int main() {
    int i; double x1, x2 ;
  scanf("%d", &n) ;
  for (i = 0 ; i < n ; i++)
  { scanf("%lf %lf %lf", x + i, y + i, d + i) ;
          ukupno += KVADRAT(d [i]) ;
  }
  binpretraga(0, MAXX, ukupno/3, &x1) ;
  binpretraga(0, MAXX, 2.0*ukupno/3, &x2) ;

  printf("%.2lf\n%.2lf\n", x1, x2) ;
  return 0 ;
}
```

8. 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("%s\n", toString(solve()).c_str());
        return 0;
}
```

9. Data su dva skupa realnih brojeva S1 i S2 i realni broj x. Ustanoviti da li postoje realni brojevi y1 iz S1 i y2 iz S2 takvi da im je zbir jednak x. Vremenska slozenost algoritma treba da bude **O(n log n)**, gde je **n** ukupan broj elemenata u oba skupa.

10. Kutije su numerisane redom od *1* do *n*, gde je *n* paran broj.U *i*-toj kutiji je smeštena količina od a[i] objekata. Konstruisati algoritam koji će spojiti sadržaje po dve kutije, ali tako da maksimalna količina objekata u po dve kutije bude što je moguće manja.

11. http://bee.bubblecup.org/Lectures/osnovni-algoritmi-sortiranja#problemi