VECTORI

1. Inserare pe poziția k; eliminare secvenţă între indicii [x,y] din vector; poziţii simetrice faţă de mijloc; parcurgere circulară la stânga/dreapta cu k poziţii

Inserare pe poziţia k

```
for (i = n; i >= k; i--)
      a[i + 1] = a[i];
a[k] = x;
n++;
```

Eliminare de pe poziţia k

```
for (i = k; i < n; i++)
      a[i] = a[i + 1];
n--;
```

Eliminare secvenţă între indicii x şi y

```
Lg = y – x + 1;
for (i = y + 1; i <= n; i++)
      a[i – Lg] = a[i];
n = n - Lg;
```

Prelucrare elemente de pe pozitiii simetrice fata de mijlocul sirului

```
for (i = 1; i <= n / 2; i++)
      prelucrare (a[i], a[n - i + 1]);
```

sau

```
st = 1; dr = n;
while (st < dr)
{
      Prelucrare (a[st], a[dr]);
      st++; dr--;
}
```

Parcurgere circulara la dreapta cu pasul K, K < n

```
i = 1;
while (cond)
      {
            Prelucrare (a[i]);
            i = i + k;
            if (i > n) i = i % n;
      }
```

Parcurgere circulara la stanga cu pasul K, K < n

```
i = 1;
while (cond)
      {
            Prelucrare (a[i]);
            i = i - k;
            if (i < 1) i = i + n;
      }
```

Secvenţa de lungime K de sumă maximă O(n)

```
S = Smax = 0;
for (i = 1; i <= k; i++)
      S = S + a[i];
Smax = S;
for (i = k + 1; i <= n; i++)
{     S = S + a[i] – a[i – k];
      if (S > Smax) Smax = S;
}
```

2. **Secvenţa de sumă maximă - O(n)**

```
int s=0, smax=0, i, pmax=1, p=1;
for (i=1; i<=n; i++)
{
      if (s+a[i]>=a[i]) s+=a[i];
          else s = a[i], p = i;
      if (s > smax) smax=s, pmax = p;
}
```

3. **Sortare STL**

```
#include <algorithm>
....
int a[NMax+1], n;
....
int main()
{
      .....
      stable_sort(a+1, a+n+1);      //sorteaza crescator vect a cu elemente
                                    //citite de la 1 la n
}
```

4. **Element majoritar (cel putin n/2+1 apariții în șirul a1, a2, ...an) - O(n)**

```
int i, sanse = 1, maj=a[1], nr = 0;
for (i = 2; i <= n; i++)
      if (a[i] == maj) sanse++;
     else {
            sanse--;
            if (sanse<0) maj = a[i], sanse = 1;
            }
//validare majoritar
for (i = 1; i <= n && nr < n/2+1; i++)
      if (a[i] == maj) nr++;
if (nr >= n/2 + 1) prelucrare(maj);
```

5. **Vector de frecvenţă**

```
int i, fr[valmax+1], x;
fin >> n;
for (i = 2; i <= n; i++)
{     fin >> x;
      fr[x]++;
}
for (i = 1; i <= valmax; i++)
      if (fr[i] > 0) prelucrare (i);
```

ex: verificarea daca numerele a si b sunt formate din aceleasi cifre dar in alta ordine

```
int i, fra[10], frb[10], a, b, ok;
fin >> a >> b;
if (a == 0) fra[0]++;
while (a > 0)
{     fra[a % 10]++; a = a/10;}
```

```
if (b == 0) frb[0]++;
while (b > 0)
{     frb[b % 10]++; b = b/10;}

ok = 1;     //pp. ca a si b contin aceleasi cifre
for (i = 0; i <= 9; i++)
      if (fra[i] != frb[i]) {ok = 0; break;}
if (ok == 0) cout << "aceleasi cifre";
else cout << "nu";
```

6. **Ciurul lui Eratostene**

```
char ciur[dimmax+1];
int i, j;
........
ciur[0] = ciur[1] = 1;
for (i = 2; i * i <= dimmax; i++)
      if (ciur[i] == 0)
            for (j = i * i; j <= dimmax; j = j + i)
                  ciur[j] = 1;
```

7. **Vectori ordonaţi**
   a. Căutare binară în șir strict crescător – $O(\log_2 n)$

      Varianta clasica

```
int caut_bin(int val)
{ int st = 1, dr = n, mij;
  while(st <= dr)
  {
      mij = (st + dr)/2;
      if (a[mij] == val) return mij;
      if (a[mij] < val) st = mij + 1;
      else dr = mij - 1;
  }
  return 0;
}
```

   b. Interclasare O(n+m)

```
int i = 1, j = 1, k = 0;
while (i <= n && j <= m)
   if (a[i] < b[j]) c[++k] = a[i++];
   else c[++k] = b[j++];
while (i <= n) c[++k] = a[i++];
while (j <= m) c[++k] = b[j++];
```

8. Principiul cutiei lui Dirichlet O(N)

Determinarea unei submulțimi a elementelor `{a1, a2, ...aN}` cu proprietatea că suma elementelor submulțimii este divizibilă cu `N`.

Se calculeaza sumele `partiale S1=a1, S2=a1+a2, ...., Si=a1+a2+...ai` si se construieste vectorul de resturi `R(N)`, `R[k]=i`, unde `Si % N = k`, adica indicele pana la care se aduna elementele a caror suma da restul K la impartirea la N

Caz 1. Daca `Si % N = 0` atunci am gasit submultimea cautata `a1, a2,..., ai`

Caz 2. Daca `Si % N !=0` atunci

a. Daca `R[Si%N] = 0`, `Si` este prima submultime care da restul `Si%N` si `R[Si%N] = i`

b. Daca `R[Si%N] = k` atunci submultimea `Sk= a1 + a2 + ... ak` calculata anterior a produs acelasi rest la impartirea la `N` ca si `Si`. Astfel submultimea cautata este a_{k+1} a_{k+2}, ... a_i care produce suma `Si – Sk` ce este divizibila cu `N`

9. Secvenţa de lungime maximă de elemente cu o anumită proprietate

```
int lg, lgmax, i, p, pmax;
int a[dimmax+1], n;
..........
int main() {
p = pmax = 1;
lg = lgmax = 1;
for (i = 2; i <= n; i++)
      if (a[i] are proprietatea secventei)
            lg++;
      else {
            if (lg>lgmax)
            {
                  lgmax = lg;pmax = p;
            }
            lg = 1; p = i;
      }
if (lg>lgmax)
{     lgmax = lg;      pmax = p;}
```

10. Şmenul lui Mars:

Se da un vector *A* de *N* elemente pe care se fac *M* astfel de operatii:

`ADUNA(st, dr, x)` - toate elementele cu indicii intre *st* si *dr* ($0 \le st \le dr < N$) isi cresc valoarea cu *x*. La sfarsit trebuie sa se afiseze vectorul rezultat.

Construim vectorul *B* de *N+1* elemente, cu proprietatea ca $A_i = B_0 + B_1 + ... B_i$. Astfel, o operatie

`ADUNA(st, dr, x)` devine: `B[st] = B[st] + x; B[dr + 1] = B[dr+1] – x;`
(inceputul intervalului pe care se creste valoarea cu x, adica st, se marcheaza in B prin cresterea valorii B[st] cu x iar sfarsitul intervalului, adica dr, se marcheaza in B prin scaderea valorii cu x)

Aplicatie: se da un sir de N intervale de forma [xi, yi] si un sir de M numere naturale, sa se determine pentru fiecare numar in cate dintre intervalele date se gaseste.

Se aplica smenul lui Mars pentru x = 1 si valoarea Ai rezultata dupa adunare ($A_i = B_0 + B_1 + ... B_i$) reprezinta numarul de intervale care il contin pe i