Probleme cu vectori

1. Se dau şirurile de întregi:

x(x[i], i=1,k)

y(y[i], i=1,l)

Să se conceapă un algoritm care determină numărul elementelor diferite din cele două şiruri (din x[1],x[2],...,x[k],y[1],y[2],...,y[l])

*Numărul operaţiilor folosite ar trebui să fie de ordinul k+l

Ideea de rezolvare:

- se identifică valoare minimă (min) și cea maximă (max) din cei doi vectori, parcurgând toate elementele o dată (ordinul k+l)
- se va folosi un vector de frecvență f (f[i], i=1,|max-min|+1)
- se inițializează un contor, cont = 0
- se parcurge vectorul x și y, și pentru fiecare element,
    - dacă f[x[i]-min+1]=0, i=1,k respectiv f[y[j]-min+1]=0, j=1,l atunci f[*indicele corespunzator*] ←1, iar contorul va crește, pentru că s-a identificat un nou element,
    - iar dacă deja este egal cu 1, înseamnă că elementul current nu e unul nou şi nu va fi numărat (ordinul k+l)

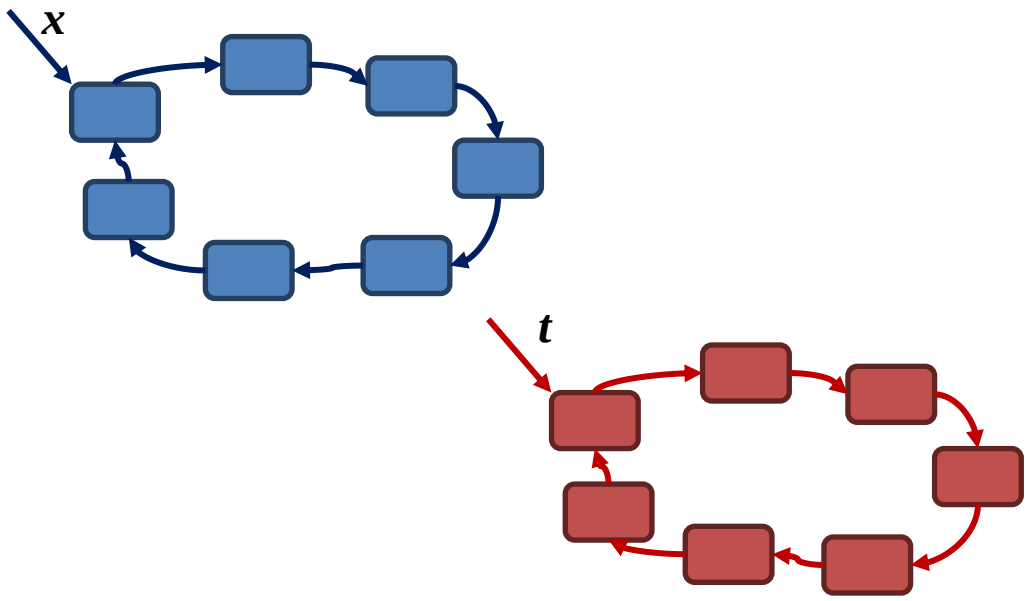
Rezolvarea în pseudocod:

```
Algoritm elemente_distincte (x, y, cont)
    Pre: x(x[i], i=1,k), y(y[i], i=1,l)
    Post: cont – nr. de elemente distincte

    cont←0
    Daca k≠0 Atunci
        min←x[1]
        max←x[1]
```

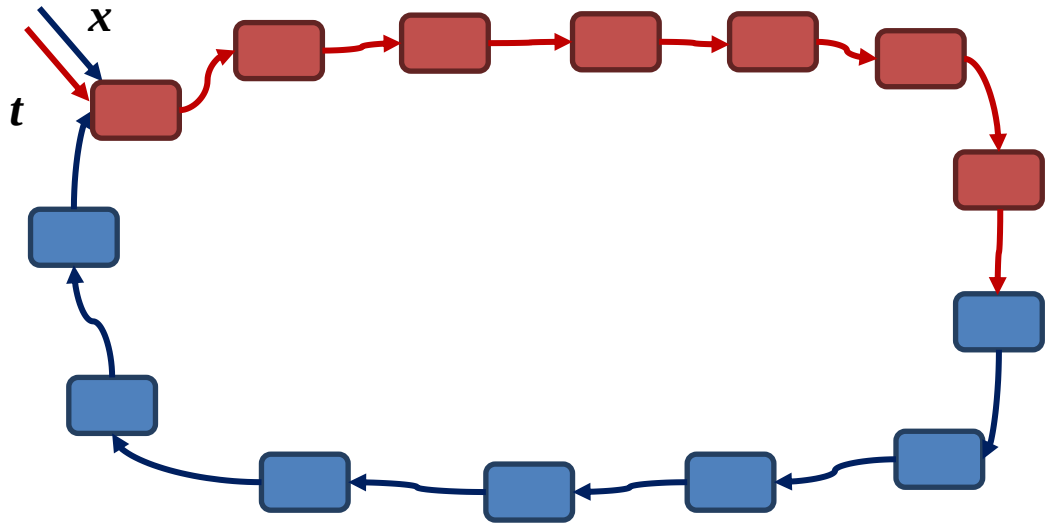
```
        Altfel
            Daca l≠0 Atunci
            min←y[1]
            max←y[1]
    Sf_Daca

    Daca k≠0 sau l≠0
        i=1
        Cat timp i<=k executa
            Daca x[i]<min Atunci min←x[i] Sf_Daca
            Daca x[i]>max Atunci max←x[i] Sf_Daca
            i←i+1
        Sf_Cat timp
        i=1
        Cat timp i<=l executa
            Daca y[i]<min Atunci min←y[i] Sf_Daca
            Daca y[i]>max Atunci max←y[i] Sf_Daca
            i←i+1
        Sf_Cat timp

        Iniţializează tot f cu 0

        Pentru i←1,k Executa
            Daca f[x[i]-min+1]=0 Atunci
                count←count+1
                f[x[i]-min+1]=1
            Sf_Daca
        Sf_Pentru
        Pentru i←1,l Executa
            Daca f[y[i]-min+1]=0 Atunci
                count←count+1
                f[y[i]-min+1]=1
            Sf_Daca
        Sf_Pentru
    Sf_Daca

Sf_Algoritm
```

Probleme cu liste

1.Concepeți și implementați un algoritm care, având două liste circulare distincte, accesibile prin pointerii x și t, inserează elementele din a doua listă (al cărei pointer este t) la începutul primei liste (al cărei pointer este x)

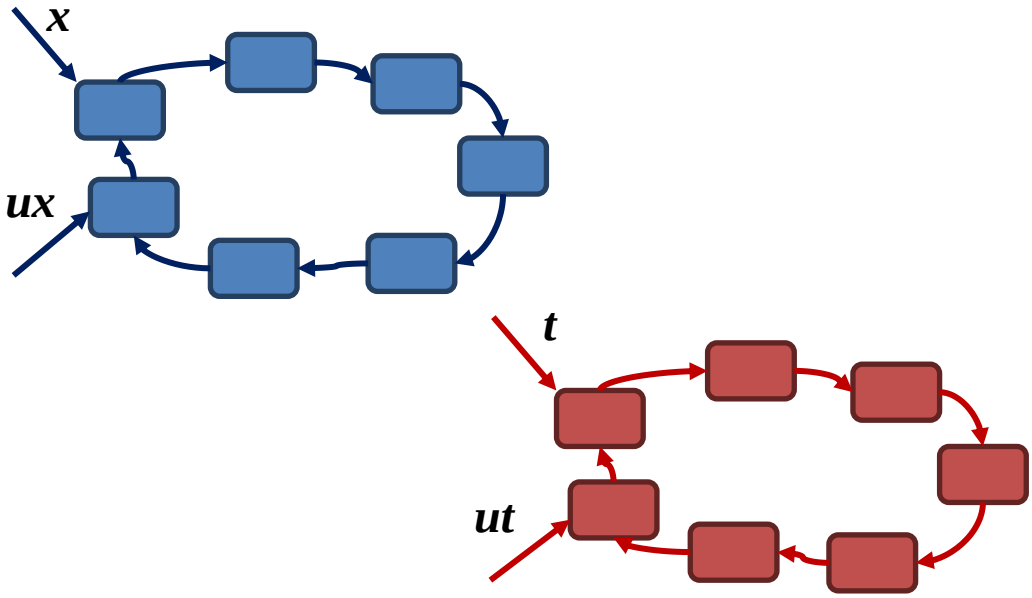


Să devină:

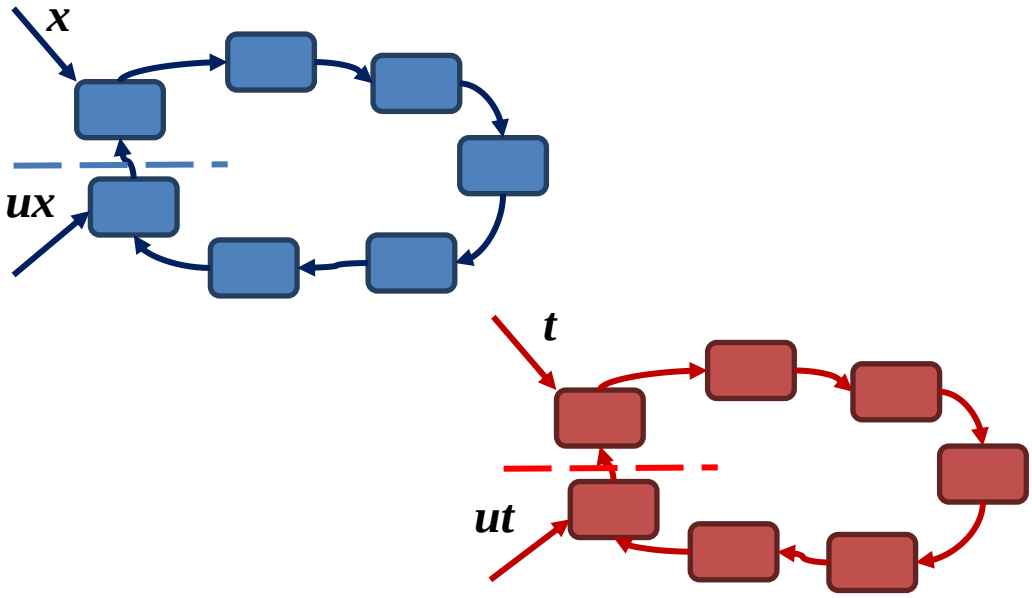


Ideea de rezolvare:

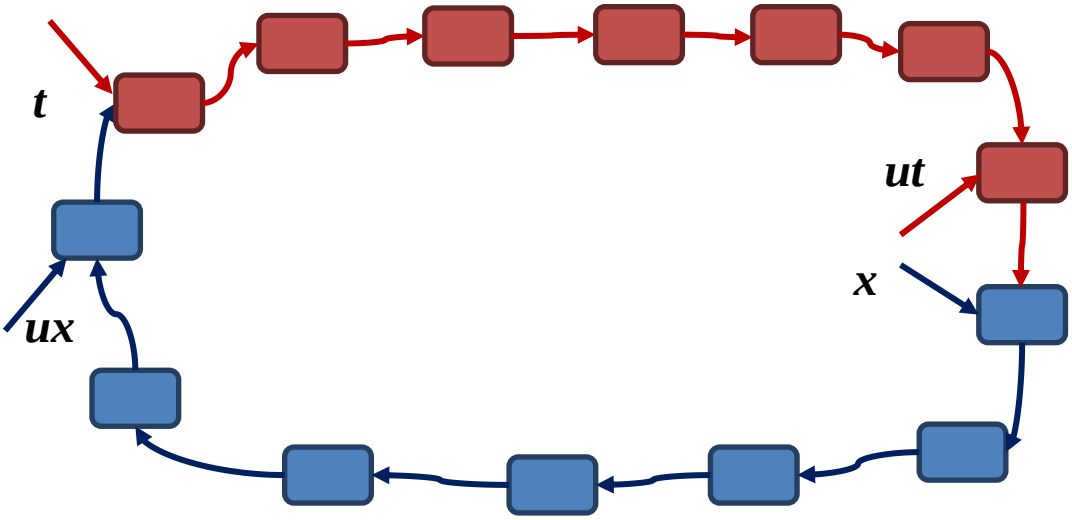
Parcurgem cele 2 liste pentru a obține adresele ultimelor noduri din fiecare listă.



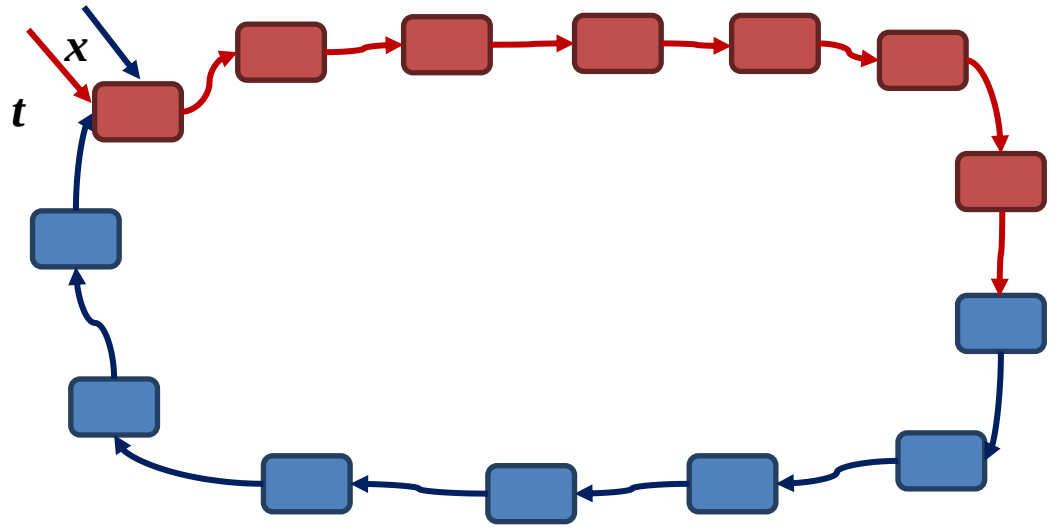
Cele 2 liste circulare se vor desface



ut se va lega de *x*, iar *ux* de *t*



Se actualizează *x*-ul



Rezolvarea în C++:

```
typedef struct node* link;
struct node { itemType item; link next; };
typedef link Node;

void algoritm_unire_liste(Node x,Node t)
{
     //trebuie sa identificam ultimul element din lista
     //x
     Node px=x;
     while (px->next != x)
          px=px->next;

     //trebuie sa identificam ultimul element din lista
     //t
     Node pt=t;
     while (pt->next != t)
          pt=pt->next;

     //legam ut de x
     ut->next=x;
     //legam ux de t
     ux->next=t;

     //actualizam x-ul
     x=t;
}
```

TEMA: termen limită 4.04.2020 (bonus-cine observă erori)

1. Implementaţi şi testaţi algoritmul descris la prima problemă
2. Daţi ideea de rezolvare, analizaţi şi implementaţi din setul 1 şi 2 problema cu numărul rezultat în urma următorului calcul:
   NR = numărul de ordine din catalogul grupei MOD16 + 2

Aveţi grijă de voi, de cei dragi şi cei aflaţi în dificultate!