
Algoritmi și structuri de date (I). Seminar 3: Descrierea în pseudocod a algoritmilor (2).

- Prelucrări asupra tablourilor uni și bi-dimensionale.
 - Execuția pas cu pas și identificarea erorilor.
 - Estimarea timpului de execuție (numărul de execuții ale operației dominante)
-

Problema 1 Se consideră un tablou $x[1..n]$ cu elemente de tip întreg și se pune problema să se decidă dacă toate elementele tabloului au aceeași valoare sau nu. (a) Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentați printr-un contraexemplu. (b) Stabiliți operația dominantă și estimați timpul de execuție a fiecăruia dintre variantele corecte de algoritmi.

```
identic1( $x[1..n]$ )
for  $i = 1, n - 1$  do
  if  $x[i] == x[i + 1]$  then
    return True
  else
    return False
  end if
end for
```

```
identic2( $x[1..n]$ )
for  $i = 1, n - 1$  do
  if  $x[i]! = x[i + 1]$  then
    return True
  else
    return False
  end if
end for
```

```
identic3( $x[1..n]$ )
for  $i = 1, n - 1$  do
  if  $x[i]! = x[i + 1]$  then
    return False
  end if
end for
return True
```

```
identic4( $x[1..n]$ )
 $i = 1$ 
while  $i < n$  do
  if  $x[i]! = x[i + 1]$  then
    return False
   $i = i + 1$ 
end if
end while
return True
```

```
identic5( $x[1..n]$ )
 $i = 0$ 
while  $i < n$  do
   $i = i + 1$ 
  if  $x[i - 1]! = x[i]$  then
    return False
  end if
end while
return True
```

```
identic6( $x[1..n]$ )
 $i = 1$ 
while  $i < n$  do
   $i = i + 1$ 
  if  $x[i - 1]! = x[i]$  then
    return False
  end if
end while
return True
```

Problema 2 Se consideră un tablou $x[1..n]$ cu valori numerice și se pune problema determinării celui mai mic element din tablou. (a) Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentați printr-un contraexemplu. (b) Stabiliți operația dominantă și estimați timpul de execuție a fiecăruia dintre variantele corecte de algoritmi.

```
minim1( $x[1..n]$ )
for  $i = 1, n - 1$  do
  if  $x[i] < x[i + 1]$  then
     $min = x[i]$ 
  else
     $min = x[i + 1]$ 
  end if
end for return  $min$ 
```

```
minim2( $x[1..n]$ )
 $min = 0$ 
for  $i = 1, n$  do
  if  $min > x[i]$  then
     $min = x[i]$ 
  end if
end for
return  $min$ 
```

```
minim3( $x[1..n]$ )
 $min = x[1]$ 
for  $i = 2, n$  do
  if  $min > x[i]$  then
     $min = x[i]$ 
  end if
end for
return  $min$ 
```

```

minim4( $x[1..n]$ )
 $min = x[1]$ 
for  $i = 2, n$  do
    if  $min < x[i]$  then
         $min = x[i]$ 
    end if
end for
return  $min$ 

```

```

minim5( $x[1..n]$ )
for  $i = 1, n - 1$  do
    if  $x[i] > x[i + 1]$  then
         $min = x[i + 1]$ 
    else
         $min = x[i]$ 
    end if
end for return  $min$ 
return True

```

```

minim6( $x[1..n]$ )
 $min = x[1]$ 
 $i = 1$ 
while  $i < n$  do
     $i = i + 1$ 
    if  $min > x[i]$  then
         $min = x[i]$ 
    end if
end while
return  $min$ 

```

Problema 3 Se pune problema verificării proprietății de simetrie a unei matrici pătratică $a[1..n, 1..n]$ (matricea este simetrică dacă $a[i, j] = a[j, i]$ pentru fiecare i și j din $\{1, \dots, j\}$). (a) Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentați printr-un contraexemplu. (b) Stabiliți operația dominantă și estimați timpul de execuție a fiecăruia dintre variantele corecte de algoritmi.

```

simetrie1( $a[1..n, 1..n]$ )
for  $i = 1, n$  do
    for  $j = 1, n$  do
        if  $a[i, j] == a[j, i]$  then
            print("este simetrică")
        else
            print("nu este simetrică")
        end if
    end for
end for

```

```

simetrie2( $a[1..n, 1..n]$ )
for  $i = 1, n - 1$  do
    for  $j = i + 1, n$  do
        if  $a[i, j] == a[j, i]$  then
            print("este simetrică")
        else
            print("nu este simetrică")
        end if
    end for
end for

```

```

simetrie3( $a[1..n, 1..n]$ )
 $rez = True$ 
for  $i = 1, n - 1$  do
    for  $j = i + 1, n$  do
        if  $a[i, j] != a[j, i]$  then
             $rez = False$ 
        end if
    end for
end for
if  $rez == True$  then
    print ("este simetrică")
else
    print ("nu este simetrică")
end if

```

```

simetrie4( $a[1..n, 1..n]$ )
for  $i = 1, n$  do
    for  $j = 1, n$  do
        if  $a[i, j] == a[j, i]$  then
             $rez = True$ 
        else
             $rez = False$ 
        end if
    end for
end for
if  $rez == True$  then
    print ("este simetrică")
else
    print ("nu este simetrică")
end if

```

```

simetrie5( $a[1..n, 1..n]$ )
 $rez = False$ 
for  $i = 1, n - 1$  do
    for  $j = i + 1, n$  do
        if  $a[i, j] == a[j, i]$  then
             $rez = True$ 
        end if
    end for
end for
if  $rez == True$  then
    print ("este simetrică")
else
    print ("nu este simetrică")
end if

```

```

simetrie6( $a[1..n, 1..n]$ )
 $rez = True$ 
for  $i = 1, n$  do
    for  $j = 1, n$  do
        if  $a[i, j] != a[j, i]$  then
             $rez = False$ 
        end if
    end for
end for
if  $rez == True$  then
    print ("este simetrică")
else
    print ("nu este simetrică")
end if

```

Problema 4 Fie $A = \{a_1, \dots, a_m\}$ și $B = \{b_1, \dots, b_n\}$ două mulțimi cu elemente întregi. Propuneți variante de reprezentare a mulțimilor și descrieți algoritmi pentru:

- Verificarea apartenenței unui element la o mulțime.
- Calculul reuniunii a două mulțimi ($R = A \cup B$ este mulțimea elementelor prezente în cel puțin una dintre cele două mulțimi).
- Calculul intersecției a două mulțimi ($C = A \cap B$ este mulțimea elementelor comune lui A și B).
- Calculul diferenței dintre două mulțimi ($D = A \setminus B$ este mulțimea elementelor din A care nu fac parte din B).
- Stabiliți operația dominantă și estimați timpul de execuție a fiecăruia dintre algoritmi descriși.

Indicație. Mulțimile pot fi reprezentate fie prin tabloul elementelor lor distincte fie printr-un tablou cu indicatori de prezență (în cazul în care setul valorilor ce pot fi luate de elementele mulțimii este finit). În primul caz mulțimea A va fi reprezentată printr-un tablou $a[1..n]$ iar mulțimea B printr-un tablou $b[1..m]$. Elementele tablourilor sunt în corespondență cu elementele mulțimii ($a[i] = a_i, i = \overline{1, n}$). În al doilea caz fiecare mulțime va fi reprezentată printr-un tablou cu k elemente (k este numărul valorilor posibile pe care le pot lua elementele mulțimii). Presupunând că $S = \{s_1, \dots, s_k\}$ este această mulțime de valori, elementul de pe poziția i din tabloul $a[1..n]$ este 1 dacă valoarea s_i face parte din mulțime și este 0 în caz contrar.

(a) În cazul în care mulțimea este reprezentată prin tabloul valorilor, verificarea apartenenței este echivalentă cu problema căutării unei valori într-un tablou.

Dacă mulțimea este reprezentată prin tablou cu indicatori de prezență atunci verificarea apartenenței lui e la mulțimea reprezentată prin $a[1..k]$ constă doar în a verifica că $a[e]$ este 1.

(b) În prima variantă de reprezentare se inițializează tabloul ce va conține reuniunea cu una dintre mulțimi, după care se vor adăuga elementele din a doua mulțime ce nu fac parte din prima.

În cazul în care mulțimile sunt reprezentate prin tablouri cu indicatori de prezență, pentru construirea tabloului $r[1..k]$ corespunzător reuniunii este suficient ca acesta să se inițializeze cu 0 și să se plaseze 1 pe toate pozițiile i pentru care fie $a[i] = 1$ fie $b[i] = 1$.

(c) În prima variantă de reprezentare se inițializează tabloul cu mulțimea vidă (numărul de elemente este 0), se parcurge una dintre mulțimi și se analizează fiecare element dacă aparține sau nu celeilalte mulțimi (în caz afirmativ elementul se adaugă la mulțimea intersecție, altfel se ignoră).

În cazul în care mulțimile sunt reprezentate prin tablouri cu indicatori de prezență, pentru construirea tabloului $r[1..k]$ corespunzător intersecției este suficient ca acesta să se inițializeze cu 0 și să se plaseze 1 pe toate pozițiile i pentru care atât $a[i] = 1$ cât și $b[i] = 1$.

Problema 5 Se consideră o imagine color de dimensiune $n \times n$ pixeli. Stiind că fiecărui pixel îi corespund trei valori din mulțimea $\{0, 1, \dots, 255\}$ (câte una pentru fiecare dintre cele trei canale de culoare - roșu, verde și albastru) propuneți o structură de date pentru stocarea imaginii. Descrieți un algoritm care:

- transformă imaginea color într-o imagine pe nivele de gri folosind pentru fiecare pixel regula:

$$\text{gri} = (\max(\text{roșu}, \text{verde}, \text{albastru}) + \min(\text{roșu}, \text{verde}, \text{albastru})) / 2;$$
- construiește histograma (tabelul cu frecvențele corespunzătoare valorilor pixelilor) asociată imaginii pe nivele de gri;
- determină valoarea medie folosind histograma construită la punctul (b);
- transformă imaginea pe nivele de gri în imagine alb-negru (alb-1, negru-0) folosind valoarea determinată la punctul (c) ca valoare prag (dacă valoarea pixelului din imaginea pe nivele de gri este mai mică decât valoarea medie atunci valoarea pixelului în imaginea alb-negru este 0 altfel este 1);
- verifică dacă imaginea alb-negru construită la punctul anterior conține pixeli negri pe diagonala principală și pe cea secundară și pixeli albi în rest.
- Stabiliți operația dominantă și estimați timpul de execuție a fiecăruia dintre algoritmi descriși.

Probleme suplimentare

1. Se consideră o imagine alb-negru reprezentată printr-o matrice de dimensiune $m \times n$ cu elemente din $\{0, 1\}$ (0 pentru negru și 1 pentru alb). O linie orizontală în imagine este o succesiune de pixeli albi aflați pe aceeași linie a matricii încadrați de pixeli negri.
 - (a) Să se determine numărul de linii orizontale din imagine.
 - (b) Să se determine cea mai lungă linie orizontală din imagine.
 - (c) Stabiliți operația dominantă și estimați timpul de execuție a fiecăruia dintre algoritmi descriși.
2. Se consideră o matrice pătratică cu m linii și n coloane. Descrieți câte un algoritm pentru:
 - (a) afișarea elementelor din matrice linie după linie;
 - (b) afișarea elementelor din matrice coloană după coloană;
 - (c) afișarea elementelor din matrice diagonală după diagonală (pornind de la elementul de pe prima linie ultima coloană și parcurgând fiecare diagonală paralelă cu diagonală principală pornind de sus în jos);
 - (d) afișarea elementelor în spirală pornind de la elementul de pe prima linie și prima coloană și parcurgând matricea în sensul acelor de ceasornic.
3. O rețea constituită din n calculatoare (C_1, C_2, \dots, C_n) poate fi structurată după unul dintre modelele:
 - (i) *stea* (unul dintre calculatoare este conectat cu toate celelalte, iar fiecare dintre celelalte calculatoare este conectat doar cu acesta);
 - (ii) *inel* (fiecare calculator este conectat cu alte două calculatoare astfel încât rețeaua este organizată ca un inel).Presupunem că rețeaua este reprezentată printr-o matrice R cu n linii și n coloane astfel încât elementul de pe linia i și coloana j este 1 dacă între calculatorul C_i și calculatorul C_j există conexiune și este 0 în caz contrar.
 - (a) propuneți un algoritm care verifică dacă matricea R corespunde unei rețele de tip stea;
 - (b) propuneți un algoritm care verifică dacă matricea R corespunde unei rețele de tip inel.
4. Fie $A = a_n X^n + a_{n-1} X^{n-1} + \dots + a_1 X + a_0$ și $B = b_m X^m + b_{m-1} X^{m-1} + \dots + b_1 X + b_0$ două polinoame cu coeficienți reali. Presupunând că polinoamele sunt reprezentate prin tablourile coeficienților ($a[0..n]$ respectiv $b[0..m]$) descrieți algoritmi pentru:
 - (a) Calculul valorii polinomului A pentru argumentul x .
 - (b) Determinarea polinomului sumă ($S = A + B$).
 - (c) Determinarea polinomului produs ($P = A * B$).
 - (d) Pentru fiecare dintre prelucrări stabiliți dimensiunea problemei, selectați operația dominantă și determinați numărul de execuții ale operației dominante.