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

- Prelucrări asupra tablourilor uni și bi-dimensionale.
 - Execuția pas cu pas și identificarea erorilor.
-

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. Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentați printr-un contraexemplu.

```
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. Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentați printr-un contraexemplu.

```

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 pătratice $a[1..n, 1..n]$ (matricea este simetrică dacă $a[i, j] = a[j, i]$ pentru fiecare i și j din $\{1, \dots, j\}$). Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentați printr-un contraexemplu.

```

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 simet-
        rică")
    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 simet-
        rică")
    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).

Rezolvare. 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.

```

apartenenta(integer a[1..n], e)
integer i
boolean gasit
gasit = False
i = 1
while i ≤ n AND gasit == False do
    if a[i] == e then gasit == True
        else i = i + 1
    endif
endwhile
return gasit

```

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.

```

reuniune (integer a[1..n], b[1..m])
integer i, r[1..k], k for i = 1, n do
    r[i] = a[i]
endfor
k = n
for i = 1, m do
    if apartine(a[1..n], b[i]) == False then
        k = k + 1
        r[k] = b[i]
    endif
endfor
return r[1..k]

```

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).

```

intersecție (integer a[1..n], b[1..m])
integer i, c[1..k], k k = 0
for i = 1, m do
    if apartine(a[1..n], b[i]) = True then
        k = k + 1
        r[k] = b[i]
    endif
endfor
return r[1..k]

```

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:

- (a) transformă imaginea color într-o imagine pe nivele de gri folosind pentru fiecare pixel regula:
$$\text{gri} = (\max(\text{roșu}, \text{verde}, \text{albastru}) + \min(\text{roșu}, \text{verde}, \text{albastru})) / 2$$
- (b) construiește histograma (tabelul cu frecvențele corespunzătoare valorilor pixelilor) asociată imaginii pe nivele de gri
- (c) determină valoarea medie folosind histograma construită la punctul (b)
- (d) 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)
- (e) verifică dacă imaginea alb-negru construită la punctul anterior conține pixeli negri pe diagonala principală și pe cea secundară și pixeli albi în rest.