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

- Prelucrări asupra tablourilor uni şi bi-dimensionale.
- Execuţia pas cu pas şi identificarea erorilor.

Problema 1 Se consideră un tablou $x[1..n]$ cu elemente de tip întreg şi se pune problema să se decidă dacă toate elementele tabloului au aceeaşi valoare sau nu. (a) Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentaţi printr-un contraexemplu. (b) Stabiliţi operaţia dominantă şi estimaţi timpul de execuţie 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
```

Rezolvare:

(a) *identic1*: Incorect - algoritmul verifică doar dacă x[1] şi x[2] sunt identice sau nu.

(b) *identic2*: Incorect - algoritmul doar dacă x[1] şi x[2] sunt diferite sau nu.

(c) *identic3*: Corect - operaţia dominantă va fi comparaţia, în cel mai favorabil caz $T(n) = 1$ iar în cel mai defavorabil caz $T(n) = n - 1$.

(d) *identic4*: Incorect - incrementarea se alfă în intreriorul structurii *if*, iar dacă $x[1]! = x[2]$ se intră în ciclu infinit.

(e) *identic5*: Incorect - algoritmul va accesa x[0].

(f) *identic6*: Corect - operaţia dominantă va fi comparaţia, în cel mai favorabil caz $T(n) = 1$ iar în cel mai defavorabil caz $T(n) = n - 1$.

Problema 2 Se consideră un tablou $x[1..n]$ cu valori numerice şi se pune problema determinării celui mai mic element din tablou. (a) Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentaţi printr-un contraexemplu. (b) Stabiliţi operaţia dominantă şi estimaţi timpul de execuţie 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
```

```
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
```

Rezolvare:

(a) *minim1*: Incorect - algoritmul returnează minimul dintre x[n-1] şi x[n].

(b) *minim2*: Incorect - algoritmul returnează minimul doar dacă tabloul conţine valori mai mici decat 0.

(c) *minim3*: Corect - operaţia dominantă va fi comparaţia, $T(n) = n - 1$.

(d) *minim4*: Incorect - algoritmul returnează maximul.

(e) *minim5*: Incorect - algoritmul returnează minimul dintre x[n-1] şi x[n].

(f) *minim6*: Corect - operaţia dominantă va fi comparaţia, $T(n) = n - 1$.

Problema 3 Se pune problema verificării proprietăţii de simetrie a unei matrici pătratice $a[1..n, 1..n]$ (matricea este simetrică dacă $a[i, j] = a[j, i]$ pentru fiecare i şi j din $\{1, \ldots, j\}$. (a) Care dintre următoarele variante de algoritmi este/sunt incorectă/incorecte? Argumentaţi printr-un contraexemplu. (b) Stabiliţi operaţia dominantă şi estimaţi timpul de execuţie 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
```

Rezolvare:

(a) *simetrie1*: Incorect - algoritmul va afisa un raspuns pentru fiecare comparaţie.

(b) *simetrie2*: Incorect - algoritmul va afisa un raspuns pentru fiecare comparaţie.

(c) *simetrie3*: Corect - operaţia dominantă va fi comparaţia, $T(n) = (n-1)+(n-2)+...+2+1 = (n^2-n)/2$.

(d) *simetrie4*: Incorect - algoritmul va afisa dacă ultima pereche comparată este identică.

(e) *simetrie5*: Incorect - algoritmul va returna dacă există o pereche $a[i, j] = a[j, i]$, $i! = j$.

(f) *simetrie6*: Corect - operaţia dominantă va fi comparaţia, $T(n) = n^2$.

Problema 4 Fie $A = \{a_1, \ldots, a_m\}$ şi $B = \{b_1, \ldots, b_n\}$ două mulţimi cu elemente întregi. Propuneţi variante de reprezentare a mulţimilor şi descrieţi algoritmi pentru:

(a) Verificarea apartenenţei unui element la o mulţime.

(b) Calculul reuniunii a două mulţimi ($R = A \cup B$ este mulţimea elementelor prezente în cel puţin una dintre cele două mulţimi).

(c) Calculul intersecţiei a două mulţimi ($C = A \cap B$ este mulţimea elementelor comune lui A şi B).

(d) Calculul diferenţei dintre două mulţimi ($D = A \backslash B$ este mulţimea elementelor din A care nu fac parte din B).

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

(a) În cazul în care mulţimea este reprezentată prin tabloul valorilor, verificarea apartenenţei 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 mulţimea este reprezentată prin tablou cu indicatori de prezenţă atunci verificarea apartenenţei lui e la mulţimea reprezentată prin $a[1..k]$ constă doar în a verifica că $a[e]$ este 1.

(b) In prima variantă de reprezentare se iniţializează tabloul ce va conţine reuniunea cu una dintre mulţimi, după care se vor adăuga elementele din a doua mulţime ce nu fac parte din prima.

```
reuniune (integer a[1..n], b[1..m])
integeri, 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 mulţimile sunt reprezentate prin tablouri cu indicatori de prezenţă, pentru construirea tabloului $r[1..k]$ corespunzător reuniunii este suficient ca acesta să se iniţializeze cu 0 şi să se plaseze 1 pe toate poziţiile i pentru care fie $a[i] = 1$ fie $b[i] = 1$.

(c) In prima variantă de reprezentare se iniţializează tabloul cu mulţimea vidă (numărul de elemente este 0), se parcurge una dintre mulţimi şi se analizează fiecare element dacă aparţine sau nu celeilalte mulţimi (în caz afirmativ elementul se adaugă la mulţimea intersecţie, altfel se ignoră).

```
intersectie (integer a[1..n], b[1..m])
integeri, 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 mulţimile sunt reprezentate prin tablouri cu indicatori de prezenţă, pentru construirea tabloului $r[1..k]$ corespunzător intersecţiei este suficient ca acesta să se iniţializeze cu 0 şi să se plaseze 1 pe toate poziţiile i pentru care atât $a[i] = 1$ cât şi $b[i] = 1$.

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

(a) transformă imaginea color într-o imagine pe nivele de gri folosind pentru fiecare pixel regula: gri=(max(roşu,verde,albastru)+min(roşu,verde,albastru))/2

(b) construieşte histograma (tabelul cu frecvenţele corespunzătoare valorilor pixelilor) asociată imaginii pe nivele de gri

(c) determină valoarea medie folosind histograma construita 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 conţine pixeli negri pe diagonala principală şi pe cea secundară şi pixeli albi în rest.

(f) Stabiliţi operaţia dominantă şi estimaţi timpul de execuţie a fiecăruia dintre algoritmii descrişi.

Indicaţie. Imaginea poate fi reprezentată prin trei matrici, R, V şi A, cu n linii şi n coloane.
(a) Matricea G poate fi construită prin parcurgerea în paralel a elementelor celor trei matrici şi calculul $G[i, j] = \lfloor(\max(R[i, j], V[i, j], A[i, j]) + \min(R[i, j], V[i, j], A[i, j]))/2\rfloor$ (care este şi operaţia dominantă). Ordin de complexitate: $\Theta(n^2)$
(b) Histograma imaginii este un tablou $H[0..255]$ caracterizat prin faptul că $H[k]$ este numărul de elemente din matricea G care au valoarea k. Tabloul H se iniţializează cu 0 după care se parcurg elementele $G[i, j]$ ale matricii G şi se actualizează tabloul H: $H[G[i, j]] = H[G[i, j]] + 1$ (care este operaţia dominantă). Ordin de complexitate: $\Theta(n^2)$.
(c) Nivelul mediu de gri se calculează prin $M = \sum_{k=0}^{255} k \cdot H[k]/(255 \cdot 254/2)$. Ordin de complexitate: $\Theta(1)$ (numărul de prelucrări depinde de numărul de nivele de gri, 256, şi este constantă în raport cu dimensiunea imaginii)
(d) Matricea AN corespunzătoare imaginii alb-negru se construieşte element cu element: $AN[i, j] = 0$ dacă $G[i, j] \leq M$ respectiv $AN[i, j] = 1$ dacă $G[i, j] > M$. Ordin de complexitate: $\Theta(n^2)$.
(e) Se parcurg elementele matricii iar dacă expresia logică: (i==j and AN[i,j]==0) or (i+j==n+1 and AN[i,j]==0) or (i!=j and i+j!=n+1 and AN[i,j]!=0) este falsă atunci algoritmul returnează False. Ordinul de complexitate este $\Theta(n^2)$.

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 şi 1 pentru alb). O linie orizontală în imagine este o succesiune de pixeli albi aflaţi pe aceeaşi linie a matricii încadraţi de pixeli negri.

   (a) Să se determine numărul de linii orizontale din imagine

   (b) Să se determine cea mai lungă linie orizontală din imagine

2. Se consideră o matrice pătratică cu m linii şi n coloane. Descrieţi câte un algoritm pentru:

   (a) afişarea elementelor din matrice linie după linie

   (b) afişarea elementelor din matrice coloană după coloană

   (c) afişarea elementelor din matrice diagonală după diagonală (pornind de la elementul de pe prima linie ultima coloană şi parcurgând fiecare diagonalaă paralelă cu diagonala principală pornind de sus în jos)

   (d) afişarea elementelor în spirală pornind de la elementul de pe prima linie şi prima coloană şi parcurgând matricea în sensul acelor de ceasornic

3. O reţea constituită din n calculatoare (C_1, C_2, ..., 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 reţeaua este organizată ca un inel). Presupunem că reţeua este reprezentată printr-o matrice R cu n linii şi n coloane astfel încât elementul de pe linia i şi coloana j este 1 dacă între calculatorul C_i şi calculatorul C_j există conexiune şi este 0 în caz contrar.

   (a) propuneţi un algoritm care verifică dacă matricea R corespunde unei reţele de tip stea;

   (b) propuneţi un algoritm care verifică dacă matricea R corespunde unei reţele de tip inel.

4. Fie $A = a_n X^n + a_{n-1} X^{n-1} + \ldots + a_1 X + a_0$ şi $B = b_m X^m + b_{m-1} X^{m-1} + \ldots + b_1 X + b_0$ două polinoame cu coeficienţi reali. Descrieţi algoritmi pentru:

   (a) Calculul valorii polinomului A pentru argumentul x.

   (b) Determinarea polinomului sumă ($S = A + B$).

   (c) Determinarea polinomului produs ($P = A * B$).