Adatbázis és szoftverfejlesztés elmélet

Témakör 8.

Összefoglalás

Programozási tételek

1. Egy sorozathoz egy érték hozzárendelése

Az összegzés tétele

Adott egy számsorozat. Számoljuk és írassuk ki az elemek összegét. A sorozatot most és a továbbiakban is az N elemű A[N] vektorban (tömbben) tároljuk.

Az algoritmus:

Eljárás Összegzés

s:=0

Ciklus i:=1-**től** N-**ig**

s:=s+A[i] //az s-be tegyük be s korábbi értékének és az A[i]-nek az összegét

Ciklus vége

Ki: s

Eljárás vége

Végig olvassuk az összes számot és összeadjuk őket az s nevű változóba. Vigyázni kell, hogy az összegzés megkezdése előtt az s-nek az értéke 0 legyen, hiszen nem tudhatjuk, hogy korábban milyen értéke volt. Ellenkező esetben rossz eredményt kaphatunk!

Az eldöntés tétele

Adott egy N elemű sorozat és egy, a sorozat elemein értelmezett T tulajdonság (pl. a kettővel való oszthatóság, vagy a számjegyek összege prímszám, stb.). Az algoritmus eredménye: annak eldöntése, hogy van-e a sorozatban legalább egy T tulajdonsággal rendelkező elem.

Az algoritmus:

Eljárás Eldönt

i:=1

Ciklus amíg i<=N **És** A[i] nem T tulajdonságú

i:=i+1

Ciklus vége

VAN:=i<=N // Van értéke igaz, ha i<=N igaz, egyébként hamis

Eljárás vége

Vesszük sorban az elemeket (ezért nő i mindig eggyel), és ezt addig csináljuk, amíg nem találunk megfelelő tulajdonságú elemet vagy már nincs több elem (éppen ez van leírva a ciklusfeltételben)

A kiválasztás tétele

Adott egy N elemű sorozat és egy, a sorozat elemein értelmezett T tulajdonság, valamint azt is tudjuk, hogy a sorozatban van legalább egy T tulajdonságú elem. A feladat ezen elem sorszámának meghatározása.

Az algoritmus:

Eljárás Kiválasztás

i:=1

Ciklus amíg A[i] nem T tulajdonságú //nincs szükség az i<=N vizsgálatra, mert biztosan van ilyen elem

i:=i+1

Ciklus vége

Sorsz:=i // a T tulajdonságú elem sorszáma kerül a Sorsz nevű változóba

Eljárás vége

Nincs szükség arra, hogy megvizsgáljuk: van-e még elem. Ha megtaláltuk a T tulajdonságú elemet, akkor a ciklus megáll, és már csak az i értékét kell lekérdeznünk. Az eredmény a Sorsz-ban lesz.

A megszámlálás tétele

Rendelkezésre áll egy N elemű sorozat, és egy, a sorozat elemein értelmezett T tulajdonság. Most a T tulajdonsággal rendelkező elemek megszámlálása és kiírása a feladat.

Az algoritmus:

Eljárás Megszámlálás

s:=0 // nem szabad elfelejteni

Ciklus i:=1-**től** N-ig

Ha A[i] T tulajdonságú **akkor** s:=s+1 // s értékét növeljük eggyel

Ciklus vége

Ki: s

Eljárás vége

A szélsőérték-kiválasztás tételei

a) A maximum-kiválasztás tétele

Ebben az esetben egy N elemű sorozat legnagyobb elemét kell megtalálni.

Az algoritmus:

Eljárás Maximum

Max:=A[1]

Ciklus j:=2-**től** N-**ig**

Ha Max<A[j] **akkor** Max:=A[j]

Ciklus vége

Ki: Max

Eljárás vége

Vesszük a sorozat első elemét. Ezek után ehhez az elemhez hasonlítjuk a többi elemet. Ha azonban találunk ettől az elemtől nagyobb számot, akkor innentől kezdve már a megtalált nagyobb számhoz hasonlítjuk a többi elemet. Vagyis amikor találtunk az eddig megvizsgáltak közül kiválasztott legnagyobbtól (ez van a Max változóban) nagyobbat (A[j]), akkor azonnal berakjuk a Max-ba, és innentől kezdve ez a legnagyobb.

b) A minimum-kiválasztás tétele

Ebben az esetben egy N elemű sorozat legkisebb elemét kell megtalálni.

Az algoritmus:

Eljárás Minimum

i:=1

Ciklus j:=2-**től** N-**ig**

Ha A[i]>A[j] **akkor** i:=j

Ciklus vége

Ki: A[i]

Eljárás vége

A megoldás majdnem azonos az előzővel, de most a legkisebb elem indexét tároljuk és nem magát az elemet. Ez azért lehet előnyös, mert ha például a sorozat elemei nagy számok, vagy valós számok, akkor ezek tárolása több helyet foglal a memóriában mint az index tárolása, ami mindig egész. Természetesen, a Ha utasítás feltételében a relációjel megfordul (>), hiszen most a legkisebb elemet keressük.

A keresési tételek

a) ***A lineáris keresés tétele***

Rendelkezésre áll egy N elemű sorozat, és egy, a sorozat elemein értelmezett T tulajdonság. Olyan algoritmust kell írni, amely eldönti, hogy van-e T tulajdonságú elem a sorozatban, s ha van, akkor megadja a sorszámát (ennyivel több mint az eldöntés tétele).

Az algoritmus:

Eljárás Lin_Keresés

i:=1

Ciklus amíg i<=N **És** A[i] nem T tulajdonságú

i:=i+1

Ciklus vége

Van:=i<=N

Ha van **akkor** Ki: i

Eljárás vége // csak egy elemet keres

b) ***A logaritmikus keresés tétele***

Rendelkezésre áll egy N elemű növekvő sorrendbe rendezett (!!!!!) sorozat és egy keresett elem (X). Olyan algoritmust kell írni, amely eldönti, hogy szerepel-e a keresett elem a sorozatban, s ha igen, akkor megadja a sorszámot.

Most kihasználjuk, hogy a sorozat rendezett. Ez alapján bármely elemről el tudjuk dönteni, hogy a keresett elem előtte vagy utána van-e, esetleg megtaláltuk. Az eljárás lényegének megértéséhez tudni kell, hogy az E és az F változóknak kiemelt szerepük van: mindig annak a részintervallumnak az alsó és felső végpontjai, amelyben a keresett elem benne van.

Az algoritmus:

Eljárás Log_Keresés

E:=1: F:=N

Ciklus // K-ban lesz az intervallum közepe

K:=(E+F) DIV 2 // A DIV továbbra is egész osztás

Ha A[K]<X **akkor** E:=K+1

Ha A[K]>X **akkor** F:=K-1

Amíg E>F **vagy** A[K]=X // Ha összeért a két részintervallum, vagy megtaláltam, akkor vége

Van:=E<=F

Ha van **akkor** Sorsz:=K

Eljárás vége

Pld. Legyen a sorozat 1, 2, 6, 8, 12, 22, 27 és a keresett elem a 22.

Mivel E=1 és F=7 ezért K=4, ezért az első lépésben a (1+7) DIV 2, azaz a 4. elemét, a 8-at hasonlítjuk a keresetthez, vagyis a 22-höz. Mivel a 8<22 ezért A egyenlő lesz öttel (K+1), így most már csak az 5. és a 7. elem között keressük a 22-t, vagyis egy hasonlítással „átvizsgáltuk" a sorozat felét.

2. Egy sorozathoz egy sorozat hozzárendelése

A kiválogatás tétele

Egy N elemű sorozat összes T tulajdonsággal rendelkező elemét kell meghatározni. Gyűjtsük a kiválogatott elemek sorszámait a B vektorban!

Az algoritmus:

Eljárás Kiválogat

j:=0 // Nem szabad elfelejteni

Ciklus i:=1-**től** N-**ig**

Ha a[i] T tulajdonságú **akkor**

j:=j+1

B[j]:= i // a sorszámot tároljuk

Elágazás vége

Ciklus vége

Eljárás vége

Itt is azért tároljuk az indexeket, mert így kevesebb helyet foglal a B[] vektor a memóriából. Ha ki is kell íratni az A[] vektor T tulajdonságú elemeit, akkor az a következőképpen történhet:

Ciklus i:=1-**től** j-**ig** // azért megy j-ig a ciklus, mert b-nek j db eleme van

Ki: A[B[i]]

Ciklus vége

Az A[B[i]] az a vektor B[i]-edik elemét jelenti. Ha pl. i=1 és a B[i] (a B vektor i-edik eleme) egyenlő 5, akkor A[5]-öt, azaz a vektor 5. eleme kerül kiírásra.

3. Több sorozathoz egy sorozat hozzárendelése

Rendezés minimum-kiválasztással

A módszer lényege:

A felesleges cserék kiküszöbölése érdekében két segédváltozó bevezetésére van szükség. Az ÉRTÉK nevű változó tartalmazza az adott menetben addig megtalált legkisebb elemet, az index pedig annak vektorbeli sorszámát, indexét. Az A vektor elemeit mindig az ÉRTÉK változó tartalmával hasonlítjuk össze. Ha ÉRTÉK-nél kisebb elemet találunk, azt betesszük az ÉRTÉK nevű változóba és az index-ben megjegyezzük a szóban forgó elem indexét. A menet végére az ÉRTÉK a vektor soron következő legkisebb elemét tartalmazza, az index pedig azt a sorszámot, ahol ezt az elemet találtuk. Csak a menet utolsó lépésében van szükségünk cserére, amikor az ÉRTÉK-ben lévő legkisebb elemet a helyére tesszük.

Az algoritmus:

Eljárás MinKivRend

Ciklus cikl:=1-**től** N-**ig**

index:=cikl; ÉRTÉK:=A[cikl]

Ciklus j:= cikl+1-**től** N-**ig**

Ha ÉRTÉK>A[j] **akkor** ÉRTÉK:=A[j]; index:=j

Ciklus vége

A[index]:=A[cikl]; A[cikl]:=ÉRTÉK

Ciklus vége

Eljárás vége

Rendezés maximum-kiválasztással

Az algoritmus:

Eljárás MaxKivRend

r:=n

Ciklus amíg r>1

i:=1; m:=A[1]; k:=1

Ciklus amíg i<r

Ha A[i]>m **akkor**

M:=A[i]; k:=i

Elágazás vége

i:=i+1

Ciklus vége

A[k]:=A[r]; A[r]:=m

r:=r-1

Ciklus vége

Eljárás vége

Buborékos rendezés

A módszer lényege

Végigmenve az adatsort tartalmazó vektoron minden szomszédos elempárt növekvő sorrendbe rakunk, azaz meghagyjuk helyükön, vagy megcseréljük őket aszerint, hogy jó sorrendben voltak vagy sem. Egy ilyen menet végén a legnagyobb elem a vektor végére kerül. Ugyanezt a páronkénti cserét végrehajtjuk a még rendezetlen A[1..(n-1)] részvektoron. Ezzel a második legnagyobb elem az A[n-1] elem helyére kerül. Addig folytatjuk az egyre rövidebb rendezetlen sor párcseréit, míg minden elem a helyére kerül. Egy menetben a maradék sor legnagyobb eleme, mint egy buborék halad végig az adatsoron. Innen származik a módszer neve.

Az algoritmus:

Eljárás Buborék

r:=n; csere:=**hamis**

Ciklus amíg r>1 **és Nem** csere

csere:=**igaz**

Ciklus i:=1-**től** r-1-**ig**

Ha A[i]>A[i+1] **akkor** //itt a páronkénti összehasonlítás

Csere(A[i], A[i+1])

csere:=**hamis**

Elágazás vége

i:=i+1

Ciklus vége

r:=r-1

Ciklus vége

Eljárás vége

A buborékrendezési algoritmust a következőképpen javíthatjuk: A módszer azon az észrevételen alapul, hogy egy menetnek az eredményeként a maximális elem a sor végére kerül. Eszerint a menet közbeni adatcseréket megtakaríthatjuk, ha a maximális elemet és ennek helyét megállapítjuk és csak a menet végén egyetlen cserével tesszük a helyére. Ezt a módszert maximum-kiválasztásos rendezésnek nevezzük.

A metszetképzés tétele

Rendelkezésünkre áll egy N és egy M elemű halmaz az A[] és B[] vektorokban. Készítsük el a két halmaz metszetét a C[] vektorba. (Két halmaz metszetébe azok az elemek tartoznak, amelyek mindkettőben szerepelnek.)

Az algoritmus:

Eljárás Metszet

szamol:=0

Ciklus i:=1-**től** N-**ig** // vesszük A[] elemeit

j:=1

Ciklus amíg j<=M **és** A[i]<>B[j] //amíg nincs az A[i]-vel egyező, vagy van még elem

j:=j+1 // vesszük B[] elemeit

Ciklus vége

Ha j<=M **akkor** // ha volt egyező eleme

szamol:=szamol+1; C[szamol]:=A[i] //betesszük C[] vektorba

Elágazás vége

Ciklus vége

Eljárás vége

Vesszük az egyik halmaz (A) elemeit sorban, először az elsőt, majd a másodikat, stb. (ezért indul az első ciklus). Majd a másik halmaz (B) elemeit olvassuk, de csak addig, amíg van a halmaznak meg nem vizsgált elme vagy nem találtunk az A-belivel egyezőt. Ellenkező esetben befejezzük az olvasást, azaz vége a ciklusnak. Ha még nem olvastuk végig a B halmazt (j<=M) az csak azért lehet, mert találtunk közös elemet. Ezt az elemet rakjuk C[] vektorba.

Az egyesítés (unióképzés) tétele

Rendelkezésre áll egy N és egy M elemű halmaz, az A[] és a B[] vektorban ábrázolva. Készítsük el a két halmaz egyesítését a C vektorba! Két halmaz egyesítésében azok az elemek tartoznak, amelyek legalább az egyikben szerepelnek.

Az algoritmus:

Eljárás Unió

Ciklus i:=1-**től** N-**ig**

C[i]:=A[i] // először az A[] elemei átkerülnek a C[]-be

Ciklus vége

szamol:=N

Ciklus j:=1-**től** M-**ig**

i:=1

Ciklus amíg i<=N **és** A[i]<>B[j] // B[]-beli elemhez keresünk A[]-ból

i:=i+1

Ciklus vége

Ha i>N **akkor** //ha nincs közös elem

szamol:=szamol+1

C[szamol]:=B[j] //akkor berakjuk az elemet C[]-be

Elágazás vége

Ciklus vége

Eljárás vége

Ha ki is szeretnénk íratni C[]-t, akkor ahhoz célszerű ciklust szervezni:

Ciklus cikl:=1**–től** szamol-**ig** //szamol eleme van C[]-nek

Ki: C[cikl]

Ciklus vége

Az algoritmus csak abban tér el a metszetképzéstől, hogy először az A[] vektor tartalmát bemásoljuk C[]-be (így annak már N eleme van) majd B[] minden egyes eleméhez (ezért megy a ciklus M-ig) keresünk A[]-beli egyezőt. Ha nem találunk (azaz i>M), akkor rakjuk C[]-be B[j]-t.

Az összefuttatás tétele

Rendelkezésre állnak két rendezett sorozat elemei. Állítsunk elő egy sorozatot úgy, hogy az eredeti sorozatok minden elem szerepeljen benne, és ez a sorozat is rendezett legyen! A feladat tulajdonképpen az uniótétel speciális esete: uniót kell előállítani úgy, hogy a rendezettség megmaradjon. A két sorozat: A[N], B[M]

Az eredmény: C[N+M]. A megoldásban a +α jelöli a számítógépen ábrázolható legnagyobb értéket. Ezt alkalmazzuk végjelként a megoldás egyszerűsítése érdekében.

Az algoritmus:

Eljárás Összefuttat

i:=1: j:=1: k:=0

A[N+1]:= +α: B[M+1]:= +α

Ciklus amíg i<N+1 **vagy** j<M+1

K:=k+1

Elágazás

A[i]<B[j] esetén C[k]:=A[i]: i:=i+1

A[i]>B[j] esetén C[k]:=B[j]: j:=j+1

A[i]=B[j] esetén C[k]:=A[i]: i:=i+1: j:=j+1

Elágazás vége

Ciklus vége

Eljárás vége

Mivel elképzelhető, hogy egyik vektor elemei elfogynak, így csak a másikból kell olvasni az elemeket és rakni át C[]-be. Ezért rakjuk a vektorok utolsó értékes eleme után a lehető legnagyobb értéket, hiszen így a másik vektorban minden elem kisebb lesz ettől a nagy értéktől, vagyis már csak az értékes elemeket tartalmazó vektort olvassuk.