MONDRAGON UNIBERTSITATEA
GOI ESKOLA POLITEKNIKOA
ESCUELA POLITÉCNICA SUPERIOR

Ordenatze Algoritmoak

Informatika Saila

20/02/2012

Ordenatze metodoen helburua datuak ordenatzea da, normalean arrayak edo fitxeroak direlarik. Metodo desberdinak daude ordenatzeko, eta konparazioan eta elkartrukean oinarritzen dira. Ondoren, bakoitzaren azterketa egingo da.

Aurkibidea

Ordenatze Algoritmoak

1 Metodo Arrunta (selección directa)

Metodo Arruntaren funtzionamentua: zerrendako a_i (i=1 .. n-1, izanik) elementua hartu eta zerrendan atzetik datorren eta txikiagoa den a_j (j=i+1 .. n, izanik) elementua topatu ondoren, elkarrekin trukatzea.

i (1..n-1) aldatzea nahi dugun elementua izanik eta j (i+1..n) i baino txikiagoa den eta bilatzen ari garen elementu hori izanik hona hemen adibide bat:

390	**205**	**182**	45	45	45	45	45	45	45	45
205	390	390	390	**390**	**205**	182	182	182	182	182
182	**182**	205	205	**205**	390	390	**390**	205	205	205
45	45	**45**	182	182	**182**	205	205	390	**390**	235
235	235	235	235	235	235	235	**235**	235	**235**	390
i:1	i:1	i:1	i:1	i:2	i:2	i:2	i:3	i:3	i:4	o.k.
j:2	j:3	j:4	j:5	j:3	j:4	j:5	j:4	j:5	j:5	

Inplementazioa :

void arrunta (int zerrenda [], int kopura);

Metodo honen bidez, a_i elementua a_j elementuarekin ordeztuko dugu, txikiagoa den bat topatzen dugun bakoitzean.

Algoritmo hau hobetzeko aukera bat, txikiagoa den elementuaren posizioaren erreferentzia gorde eta behin zerrenda osoa pasatu ondoren, benetan txikiena dena topatua izango genukenez, orduan soilik aldaketa egitea litzateke. Hona hemen adibidea:

390	45	45	45	45
205	**205**	182	182	182
182	**182**	205	205	205
45	390	390	**390**	235
235	235	235	**235**	390
i: 1	i:2	i:3	i:4	O.K.
j: 2	j:3	j:4	j:5	
el_menor =4	el_menor =3	el_menor =3	el_menor =5	

Inplementazioa :

void arrunta_hobeto (int zerrenda [],int kopurua);

2 Burbuilaren Metodoa (Burbuja – bubleSort)

i = 1-etik i=(n-1) arte, baldin eta a_i a_{i+1} baina handiago bada, balioak elkarrekin trukatzen dira eta zerrendan errekorritzen jarraitzen da. Ordenatzea, trukerik egin gabeko egoerara iristean bukatzen da.

Algoritmoaren buelta bakoitzean, elementu txikiena gora igotzen delako (burbuilak uretan egiten duten moduan) izena du honela metodo honek.

390	205	205	205	**205**	182	182	182	**182**	45	45	45	45	45	45	45
205	**390**	182	182	**182**	**205**	45	45	**45**	182	182	182	182	182	182	182
182	**182**	**390**	45	45	**45**	205	205	205	205	205	205	205	205	205	205
45	45	**45**	**390**	235	235	235	235	235	235	235	235	235	235	235	235
235	235	235	**235**	390	390	390	390	390	390	390	390	390	390	390	390
i:1	i:2	i:3	i:4	i:1	i:2	i:3	i:4	i:1	i:2	i:3	i:4	i:1	i:2	i:3	i:4
C : F	C : T	C : T	C : T	C : F	C : T	C : T	C : T	C : F	C : T	C : T	C : F	C : F	C : F	C : F	OK

Inplementazioa: **void burbuila (int zerrenda [], int kopurua);**

Goian ikusitakoa hobetu asmoz, algoritmoaren iterazio bakoitzean, elementurik pisutsuenak (handienak), zerrendaren beheraino nola erortzen diren ikusi dezakegu. Honela, lehenengo bueltan elementurik pisutsuena zerrendako azken posiziora joaten da, bigarren bueltan, bigarren elementurik pisutsuena azkenbigarren posizioan,…Beraz, lehenengo iterazioan, 1-etik n-1 arteko elementuak konprobatu beharko ditugu, bigarrengo iterazioan, 1-etik n-2 artekoak eta modu honetan jarraitu beharko dugu, buelta bakoitzean txekeatu beharreko elementuen kopurua txikituaz:

390	205	205	205	**205**	182	182	**182**	45	45	45
205	**390**	182	182	**182**	**205**	45	**45**	182	182	182
182	**182**	**390**	45	45	**45**	205	205	205	205	205
45	45	**45**	**390**	235	235	235	235	235	235	235
235	235	235	**235**	390	390	390	390	390	390	390
i:1	i:2	i:3	i:4	i:1	i:2	i:3	i:1	i:2	i:1	OK
k=5	k=5	k=5	k=5	k=4	k=4	k=4	k=3	k=3	k=2	
C : F	C : T	C : T	C : T	C : F	C : T	C : T	C : F	C : T	C : F	

Inplementazioa: **void burbuila_hobetua (int zerrenda [], int kopurua);**

3 Sartze bidezko Ordenazioa (Inserción directa)

Algoritmo honen ideia nagusia, a_k elementua iada organizaturik dagoen azpizerrenda batean sartzen datza.

k (n-1)-etik 1 arte ondorengoa errepikatu
    gorde = a_k (** zerrenda [n+1]-ean gordeko dugu**)
    i = k+1 –etik n arte ondorengoa egin
        baldin eta a_i < gorde , orduan
            $a_{i-1} = a_i$
            a_i = gorde

k: zerrendan ezarri beharreko elementua markatzen du
j : ordenaturik/organizaturik dagoen zerrendaren hasiera markatzen du
gorde : zerrendan sartu beharreko elementua gordeko duen aldagaia,
    Zerrendako n+1 posizioa izango da.

390	390	390	390	45
205	205	205	45	182
182	**182**	45	182	205
45	45	182	205	235
235	235	235	235	390
45	182	205	390	o.k.
k:4	k:3	k:2	k:1	
j:5	j:4	j: 3	j:2	

Algoritmo honek sartzen du zenbaki bat ordenatutako zerrenada baten tokatzen den tokian

```
k = 2-tik n-arte errepikatu
        i = k-1
                // balioa joango da tokatzen den tokira
                // trukaketa bitartez
        ((i>=1) eta (bek[i]>bek[i+1])) den bitartean
                // trukatu i-posizioan dagoena eta hurrengoa
                Aux = bek[i+1]
                Bek[i+1] = bek[i]
                Bek[i] = aux
```

Inplementazioa: **void sartzez_ordenatzea (int zerrenda [], int kopurua);**

4 ShellSort metodoa

Konparaketa eta elkarbanaketan oinarritzen da.
Insercion directa-ren algoritmoaren zabalpen bat bezala hartu, beraz, Shell-arekin sartu aurretik kontutan hartzea komeni da.
Ordenatu behar den lista bat daukagu... adibidez:

74, 14, 21, 44, 38, 97, 11, 78, 65, 88, 30

Inserción directa-ko algoritmoan banaka joaten ginen, bigarrenetik(14) azkenera arte elementu bakoitza ezkerreruntz eramaten.

Shell Insercion directa-ri buruz array-aren ordenazioa bat egitea proposatzen digu, baina array-a zatitzen, k elementuz zatitutako sub-arraya sortuz (zatiketa honi salto edo gap deitzen diogu) … k=n/2tik hasi behar da, n array-ko elementu kopurua izanik, eta zatiketa osoa bezala hartuz …gero, k txikiagotzen joango gara zati bi eginez, k=1 izaera iritsi arte

Goazen horretara... Gure adibidean, n=11 (11 elementu). Beraz k=n/2=11/2=5

Lehenengo elementua hartzen dugu (74) 5 leku kontatzen ditugu eta beste elementu bat hartzen dugu (97) 5 kontatzen dugu berriz eta beste elemenntu bat hartzen dugu (30) eta bukatzen dugu zeren array-a bukatu da.

74, 14, 21, 44, 38, 97, 11, 78, 65, 88, 30

Sub-arrayaren elementuak beraien artean bakarrik ordenatzen dira.

30, 14, 21, 44, 38, 74, 11, 78, 65, 88, 97

k=5 saltoko sub-array bat osatuko dugu... bigarren elementutik asiz (14) eta 5 kontatuz (11 ere hartzen dugu) eta listo, array-a bukatzen baita.

74 zenbakiko posiziora iritsi arte egiten dugu hau.
Array-a 5-ordenatuta dagoela esaten dugu orain.

Algoritmoarekin jarraitzeko, k txikiagotzen joango gara pixkanaka 2rekin zatituz eta k sub-arraya ordenatuz (oroitu k sub-array aterako zaizkigula). K=1era iristen garenean bukatuko dugu.

5 Bilatze Algoritmoa (Bilaketa Dikotomikoa) (busqueda binária o dicótomica)

Arazoa: zerrendako elementu baten posizioa bilatzea.

Zerrenda sekuentzialki pasatu beharrean, algoritmo honek, zerrenda bi zatitan banatzen du. Hauek, aldi berean beste bitan egiten dira eta zatiketak egiten jarraituko dugu bilatzen ari garen elementua aurkitu arte.

Algoritmoa:

```
if (balioa < zerrenda  [erdia] )
    bilatzen ari garen elementuaren eremua:  azpi limitea .. erdia
else
  if (balioa > zerrenda [erdia])
        bilatzen ari garen elementuaren eremua:   erdia .. goi limitea
    else
        elementua topatu dugu
```

Inplementazioa : **int bilaketa (int zerrenda [], int kopurua, int elementua);**

6 Ordenatze Algoritmoi buruzko Ondorioak

- Algoritmo arrunta, beti pausu kopuru berdina erabiltzen du. Berdin da zerrendaren hasierako egoera zein den. Algoritmo honen bertsio hobetua erabiliaz berriz:

(n-1) + (n-2) + ... + 1 = 1/2 n(n-1) = Orden (n^2)

- Burbuilaren algoritmoan, zerrendaren hasierako egoeraren arabera pauso kopuru gehiago edo gutxiago erabiliko ditugu:
  - Zerrenda ordenaturik baldin badago: n-1 = Orden (n)
  - Zerrenda guztiz desordenaturik baldin badago (alderantzizko ordena):

    (n-1) + (n-2) + ... + (n-$K_{ordenados}$)

    $K_{ordenados = n-1}$

    1/2 n (n-1) = Orden (n^2)

  Beraz, batazbestekoa hobetu egiten da. Baina barneko operazioetan, hiru mugimendu behar dira. Beraz, algoritmo hau nahiko txarra da.

- Sartze metodoa, zerrendaren hasierako egoeraren araberakoa da. Kanpoko buklea, beti exekutatuko da, baina barrukoa, ordenaturik baldin badago ez da exekutatuko, beraz Orden(n). Kasurik txarrenean 1+2+ ... + n-1 = 1/2 n (n-1) = Orden (n^2)

  Barne buklea, metodo arrunta baino konplexuagoa da. Baina ia ordenaturik dauden zerrendetan, barne buklea oso gutxi exekutatzen da. Beraz, konplexutasuna konpentsatu egiten da. Metodo hau asko erabiltzen da *quicksort*-ekin.