UPV/EHU—Informatika Ingeniaritza
DEA-II 2 deialdia (2011-vi-28) - E taldea

Deiturak	1	2b	2j	3	4	

1. Taula batek n zenbaki natural desberdin ditu. T-ko m balio txikienak inprimatuko dituen algoritmo bat idaztea eskatzen zaizu. Ezaguna da m<<<n dela. Nola egingo zenuke modurik eraginkorrenean?

   - `MergeSort` edo `QuickSort` bidez ordenatu eta ondoren m lehenengoak inprimatu
   - `AukeratuTxikienak(T,1,n,m)` deia bitartez, non `AukeratuTxikienak` honako kodea duen:

```
proz AukeratuTxikienak (T,i,j,m)
   for K in 1..m loop
      print(K-Hautaketa (T(i,j,k));
   end loop
```

   - `Beste metodoren bat erabiliz.`

   Arrazoitu zure erantzuna ezagutzen diren kasu txarreneko eta bataz besteko denbora ordena ezagunak erabiliz.

   Goian eskatu zaizuna bukatu ostean, $m \in \Theta\left(\sqrt{n}\right)$ eta $m \in \Theta\left(\lg n\right)$ kasu partikularrek zerbait aldatuko lukete?

2. Grafo zuzendu bat emanik eta programazio dinamikoaren teknika erabiliz, egizu algoritmo bat datu egitura bat osa dezan, non egitura honek X eta Y (edozeintzuk) erpinen artean bidea dagoen ala ez denbora konstantean erantzuteko balioko duen.

3. X ikastordu irakasteko irakasleak kontratu behar ditu ikastegi batek. Ikastegiak N klaseetako kontratuak egin ditzake, non kontratu bakoitzak k_i ogerleko kostua duen eta gehienez h_i ikastordu estaltzeko balioko duen hurrenez hurren ($1 \le i \le n$). Ikastegiak gehienez M ogerleko inberti lezake kontratazio hauetan.
   - Backtrack teknika erabiliz,problemak soluzioa duen erabaki ezazu, eta hala balitz, zeintzuk kontratazio egin behar diren. Lehenik, bilaketa zuhaitza irudikatzen has zaitez.
   - Orain, teknika jalea bidez problema ebazteko prozedura hautesle bat defini ezazu. Zure definizioa zuzena den ala ez arrazoi ezazu.

4. Laboratze prozesu konplexu baten analisian, egin beharreko zenbait ataza at_i ($1 \le i \le n$) identifikatu dira bai eta zenbait ataza pareen arteko aurrekotasun erlazioa ere (adibidez, at_{17} ataza at_5 atazaren aurretik burutu behar da). Egizu algoritmo bat, aipaturiko informazio jasota (komeni zaizun moduan kodeturik), atazak guztiak sekuentzian burutzeko baliagarria den ordenazio bat sortuko duena; hots aurrekotasun murriztapenak errespetatzen dituen atazen antolaketa bat. Proposatutako soluzioren denbora-analisia ere eman ezazu.

UPV/EHU—Informatika Ingeniaritza
DEA-II 2 deialdia (2011-vi-28) - E taldea
SOLUZIO BAT

1. Taula batek n zenbaki natural desberdin ditu. T-ko m balio txikienak inprimatuko dituen algoritmo bat idaztea eskatzen zaizu. Ezaguna da m<<<n dela. Nola egingo zenuke modurik eraginkorrenean?

   - `MergeSort` edo `QuickSort` bidez ordenatu eta ondoren m lehenengoak inprimatu
   - `AukeratuTxikienak(T,1,n,m)` deia bitartez, non `AukeratuTxikienak` honako kodea duen:

```
proz AukeratuTxikienak (T,i,j,m)
   for K in 1..m loop
      print(K-Hautaketa (T(i,j,k));
   end loop
```

   - Beste metodoren bat erabiliz.

   Arrazoitu zure erantzuna ezagutzen diren kasu txarreneko eta bataz besteko denbora ordena ezagunak erabiliz.

   Goian eskatu zaizuna bukatu ostean, $m \in \Theta(\sqrt{n})$ eta $m \in \Theta(\lg n)$ kasu partikularrek zerbait aldatuko lukete?

SOL:

(1) Mergesort-en ordena bai bataz bestean bai kasu txarrenean Θ(n lg n) da, eta Quicksort baino hobea, azken hau kuadratikoa bait da kasu txarrenean. Hortaz, bien artetik onena Mergesort da

(2) AukeratuTxikienak m aldiz K-hautaketari dei egiten dio eta ondoren inprimaketa bat egiten du. Denbora ordena aldetik, bere kostua $m*\Theta(T_{\texttt{K-Hautaketa}}(n))+\Theta(m)$ da. Badakigu, kasu txarrenean k-Hautaketaren denbora kostua koadratikoa dela eta bataz bestean lineala (osagai kopuruan). Hori horrela, bigarrengo prozedura sarreraren antolamenduarekiko sentikorra da eta:
   a. bataz bestean, $m*\Theta(n)+\Theta(m)$ izanik, eta baturaren erregela aplikatuz, $\Theta(m*n)$ da denbora ordena,
   b. kasu txarrenean, $m*\Theta(n^2)+\Theta(m)$ izanik, eta baturaren erregela aplikatuz, $\Theta(m*n^2)$ da denbora ordena

(3) bigarren prozedura erraz hobe liteke. Zehazki, aski litzateke ondorengoarekin:

```
K-Hautaketa (T(1,n,m))   -- Dei bakarra
print(T(1..m))           -- Θ(m) kostuko inprimaketa bat
```

   Honen kostua $\Theta(T_{\texttt{K-Hautaketa}}(n))+\Theta(m)$ da.
   a. bataz bestean, $\Theta(n)+\Theta(m)$ litzateke eta baturaren erregela bidez, sinplikatuz, $\Theta(n)$ litzateke denbora ordena,
   b. kasu txarrenean, $\Theta(n^2)+\Theta(m)$ izanik, sinplifikatuz, $\Theta(n^2)$ litzateke.

Hiru prozeduren arteko konparazioa: hirugarren prozedura bigarrena hobetua da, eta beti soluzio eraginkorragoa da. Hortaz, 2.a alde batetara uzten da. 1 eta 3 prozeduren artean, 1 beti da Θ(n lg n) eta 3.a bataz bestean Θ(n) denez, *bataz bestean 3 prozedura onena da*. Halere, aipatu behar dugu 3.a kasu txarrenean, okerragoa litzatekeela koadratikoa baita, baina kasu hauek bakanak lirateke.

Bestalde, $m \in \Theta(\sqrt{n})$ eta $m \in \Theta(\lg n)$ kasu partikularrek ez lukete erantzuna aldatuko izanik kalkulatuko ordenetan, (bi kasuak $m \in o(n)$ izaten jarraitzen dute) eta haietan ez dute kalkulatutako sinplifikazioetan eraginik eta emaitzetan ez baita m aldagai agertu ere egiten.

2. Grafo zuzendu bat emanik eta programazio dinamikoaren teknika erabiliz, egizu algoritmo bat datu egitura bat osa dezan, non egitura honek X eta Y (edozeintzuk) erpinen artean bidea dagoen ala ez denbora konstantean erantzuteko balioko duen.

SOL:

Floyd algoritmoak edozeintzuk erpin pareen arteko bide motzenen distantziak kalkulatzen ditu programazio dinamikoaren teknika erabiliz. Enuntziatu berriari soluzioa bilatzeko aski da Floyd-en ekuazioa ekuazio-boolear bilakatzea honako moduan:

DE(i,i,0) = True
DE(i,j,0) = (G(i,j)>1) i≠j
DE(i,j,k) = **or**{DE(i,j,k-1), DE(i,k,k-1) **and** DE(k,j,k-1)}

Algoritmoa

```
proc    BideaDagoDatuEgituran    (G: in array (1..N,1..N) of Integer;
                                 DE: out array (1..N,1..N) of Boolean) is
begin
   for Irten in 1..N loop
      for Iritsi in 1..N loop
         DE(Irten,Iritsi):= (G(Irten, Iritsi)>0);    -- True positiboa bada eta False bestela
      end loop;
      G(Irten,Irten):=True);    -- Diagonala True, bertatik bertara beti iristerik baitago
   end loop;

   for Irten in 1..N loop
      for Iritsi in 1..N loop
        for K in 1..N loop
           DE(Irten, Iritsi) = or { DE(Irten, Iritsi),
                                    DE(Irten,k) and DE(k, Iritsi)}
        end loop;
     end loop;
   end loop;
```

Analisia

Begi bistakoa da soluzioa kubikoa dela eta itzultzen duen DE matrizean, denbora konstantean, edozeintzuk erpin pareen artean biderik balego *True* jasoa duela eta *False* bestela.

3. X ikastordu irakasteko irakasleak kontratu behar ditu ikastegi batek. Ikastegiak N klaseetako kontratuak egin ditzake, non kontratu bakoitzak k_i ogerleko kostua duen eta gehienez h_i ikastordu estaltzeko balioko duen hurrenez hurren ($1 \le i \le n$). Ikastegiak gehienez M ogerleko inberti lezake kontratazio hauetan.
   - Backtrack teknika erabiliz, problemak soluzioa duen erabaki ezazu, eta hala balitz, zeintzuk kontratazio egin behar diren. Lehenik, bilaketa zuhaitza irudikatzen has zaitez.
   - Orain, teknika jalea bidez problema ebazteko prozedura hautesle bat defini ezazu. Zure definizioa zuzena den ala ez arrazoi ezazu.

SOL: dirua iristen ez bada behar diren orduak kontratatzeko problemak soluziorik ez du. Soluzio optimoak, zehazki X ordu kontratuak izango ditu edo zerbait gehiago, baina M ogerleko gastua gainditu gabe (eta dauden soluzioen artetik, noski, merkeena izango da).

Backtrack:

Zuhaitza: maila bakoitzak i kontratu mota bat aztertuko du. i mailanko adarketa (etiketak) 0=i kontratu motako kontrataziorik ez, 1=kontratazio bakarra i motakoa, 2… Zuhaitzak gehienez n maila izango ditu, hain zuzen kontratu mota adina maila.

SP noiz ez da hedatzen jarraitu behar? (Zuhaitzaren hostoak).

- SP kontratazioekin X orduak edo ordu gehixeago estaltzen direnean (soluzioa, optimoa den aztertu beharko da);
- X estali gabe dirurik gelditzen ez denean kontratatzen jarraitzeko (soluziorik ez du eman esplorazio konbinazio horrek);

Aldagaiak:

- SP=<z1, z2,… zj, …,zi> i mailan itxura izango du, non zj j motako egin beharko liratekeen kontratu kopurua jasotzen du.
- SPK = Soluzio partzialaren kostua=z1*k1+z2*k2+…+zi*Ki. Soluzioak existi dezan SPK≤M izan behar du
- SKD= soluzio partzialak estaltzen dituen orduak= z1*h1+z2*h2+…+zi*hi. (SPD≤X) edo ((SPD-hi)<X<SPD) beteko da.
- i aztertzen ari garen kontratu mota (eta SP-ren luzera)
- SOPT= orain arteko kontratazioen zerrendaketa optimoa.
- SOPTK= SOPT-ren kostua. Beti beteko da SOPTK≤M. Hasieran M balioz hasieratuko da.
- Soluziorik= True, baldin emandako kontratu motekin X ordu estali balitezke gehienez M gastatu. False, bestela.

kima1: nahiz nahi adina kontratazio egingarri izan, i mailan mug_i izango da proba egiteko muga kontratazio kopuruaren muga. mug_i –ren balio konkretua KO div K_i edo (KO div K_i)+1 izango da, zehazki gehienez kontratazeko gelditzen diren orduak KO orduan Ki-ren anizkoitza denean edo kontratazio bat gehiago egin behar denean anizkoitza ez denean hurrenez hurren.

kima2: Kontratazio gehiago egiteko dirurik ez dagoenean eta ez ditugu ordu guztiak estali, ez da jarraitu behar. Hau modu sinplean konputatzeko, bektore baten bidez egingo da, MinKostua(1..n), non honako moduan hasieratu beharko den: MinKostua(j)=$min_{j\le y\le n}\{k_y\}$, hots aztergai gelditzen diren kontratazio merkeenaren kostua denbora konstantean ematen du bektoreak.

Kodea:

```
MinKostua(n)=Kn;
for i in (n-1)..1 loop
   MinKostua(i):=min{ MinKostua(i+1), Ki};
end loop;
Soluziorik:=False;
SOPTK:=M;     -- Soluzioak existituko balitz, kasu txarrenean M kostua izango luke
KontratazioaBT01 (1,X,0);

proc  KontratazioaBT01(i, KO, SPK) is
begin
   if (KO>=X) then  if (SPK<SOPTK)
                    then SOPTK:=SPK; Kopiatu(SOPT,SP,i); Soluziorik:= True; end if;
   else if SPK+ MinKostua(i)>M then null;
   else
     if (KO mod Ki=0) then mugi=KO div Ki; else mugi=1+ KO div Ki; end if;
     for A in 0 .. mugi loop
       if (SPK + Ki * A <SOPTK)k
       then KontratazioaBT01(i+1, KO-A*hi, SPK+A*ki);
       end if;
     end loop;
   end if;
```

Jalea:

Oharra: Motxila 01 problemak ez dauka soluzio jalerik. Motxila zatiekin bai, zatiketari esker doitzen baita da soluzioa.

Oraingoan, klaseak ditugu (nahi adina), eta 01 kasuan gaude. Hau da, adibidez, i kontratu motarekin norbait kontratatzen badugu, ki ordaindu beharko du eta hi ordu emango ditu gehienez, nahiz errealitatean, akaso, ordu gutxiago emango dituen.

Zenbait prozedura hautesle denak okerrak, kontrako adibideak baitituzte:

(1) kostu merkeen duen kontratu motako irakasleak kontratatu. Kontrako adibidea, Mren balioa behar bezain handia izanik, X=4; K=<3,4>; H=<3,4>. Azalpena: Datu hauekin k1+k1 kostuaz t1+t1 ordu kontratatuko lirateke, hobea izani t2=4 ordu kontratatzea k2=4 ordainduz.

(2) Ordu gehien ematen dituen kontratu motako ikasleak kontratatu. Kontrako adibidea, Mren balioa behar bezain handia izanik, X=4; K=<10,5>; H=<10,4>. Azalpena: t1 kontratatuko lirateke k1 kostupean, hobea izanik t2 kontratazioa k2 kostupean.

(3) Ordu kopuru/kostu proportzio maximoa duen motako kontratu mota aukeratu. Kontrako adibidea, M-ren balioa behar bezain handia izanik, X=4; K=< 2.5, 4> H=<3,4>. Azalpena: k1+k1 kostuz (5 ogerlekoz) 6 ordu kontratatuko genituzke, hobe litzatekeenean berriz k2 prezioz (4 ogerlekoz) t2 ordu (4 o.) kontratatzea.

4. Laboratze prozesu konplexu baten analisian, egin beharreko zenbait ataza at_i ($1 \le i \le n$) identifikatu dira bai eta zenbait ataza pareen arteko aurrekotasun erlazioa ere (adibidez, at_{17} ataza at_5 atazaren aurretik burutu behar da). Egizu algoritmo bat, aipaturiko informazio jasota (komeni zaizun moduan kodeturik), atazak guztiak sekuentzian burutzeko baliagarria den ordenazio bat sortuko duena; hots aurrekotasun murriztapenak errespetatzen dituen atazen antolaketa bat. Proposatutako soluzioren denbora-analisia ere eman ezazu.

SOL:

Atazen erlazioak grafo zuzendu bat adierazten dute. Erpinak atazak dira eta arkuak atazen arteko aurrekotasuna. Grafoa aziklikoa dela suposatuko dugu (soluzioak existi dezan). Ez da beharrezkoa jatorri ataza ezagutzea (indigree 0 duen erpina/ataza).

Grafoen sakonerako korritzea erabiliko da soluzioa eraikitzeko. Zabalerako korritzeak ez du balio soluzioa kalkulatzeko, atze prozesaketa behar baita. <t1,t2,t3,t4> zerrendaketak eraikiko bagenu, t1 lehendabizi exekutatuko genukela adieraziko luke, eta azkenekoz t4 ataza. Bestalde, t2 t3 eta t4 aurretik exekutatu behar dela adieraziko du.

Zerrendaketa atzetik aurrera eraikiko da: soluzioan lehendabizi azkenekoz burutu behar d(ir)en atazak gehituko dira. Zerrendara ataza gehiago gehitzeko orduan, soluzio zerrendaren burutik (zerrendaren aurrekaldetik) gehituko dira atazak.

Prozesua: Ataza hostoa bada, soluzio zerrendara gehitzen zaio zuzenean. Atazak umeak baditu, lehendabizi umeak sakoneran korrituko dira (eta zerrendaratuak izango dira). Berriro gurasora iristen denean kontrola, gurasoa zerrendara gehituko da burutik, honek adieraziko baitu besteen aurretik egin behar dela. Exekuzio sekuentziak markatutako erpin batetara eramaten badu arku berri baten tratamenduaren ondorioz, jomuga ataza jada zerrendaketan barnean dago eta ez da berriro aztertu behar (zerrendan egoteak gurasoaren ondoren exekutatu behar dela esan nahi du, beraz arazorik ez du ematen gertaera horrek).

```
procedure  GrafoaKorritu (G: in Grafoa) is
   Aztertua: constant Boolean := True;      // n= # erpinak; a= # arkuak
   Bisitak: Taula(1..n):= (Others=>  not (Aztertua));
   Listing: Zerrenda:= ZerrendaHutsaEgin();
begin
   for  Erp  in  1..n  loop
         if  not MarkatuaDago(Bisitak, Erp)  then
               MarkaIpini(Bisitak, Erp, Aztertua);
               SK(G, Erpina);
         end if;
   end loop;
   ZerrendaInprimatu(Listing); -- n atazen inprimaketa,
                               -- aurrekotasuna errespetatzen duen zerrendaketa
end GrafoaKorritu;


procedure  SK  (G: in Grafoa; A: in Erpina) is
   AAuzokideenKopia: ErpinenL;  Lehena: Erpina;
begin
   AAuzokideen_Kopia:= Auzokideak(G, A);
   while  not( HutsaDaL? (AAuzokideenKopia))  loop
      Lehena:= LehenengoaL(AAuzokideenKopia);
      Hondarra(A_AuzokideenKopia);
      if not Markatua_Dago(Bisitak, Lehena)  then
         Marka_Ipini(Bisitak, Lehena, Aztertua);
         SK(G, Lehena);
      end if;
   end loop;
   GehituBurutik(Listing,A);
end SK;
```