UPV/EHU—Informatika Ingeniaritza
DEA-II (2011-iv-18) - E taldea

3 P: Backtrack (7 puntu) + finaleko 1-4 ariketetako BAT (3 puntu)
3 P + Jalea: Backtrack (6 puntu) + Prim (1,5 puntu) + finaleko 1-4 ariketetako BAT (2,5 puntu)
AZTERKETA FINALA: 1-5 ariketak, bakoitzak 2 puntu

Deiturak	1	2	3	4	5	Prim	

1. Algoritmoaren denbora ekuazioa identifikatu eta denbora ordena kalkulatu

```
function Atalak(T, X, Y, M) return integer is
if Y-X+1≤3 then
   return T(X)+T(Y)+M
else
   zati← ⌊(Y-X+1)/3⌋
   for i in X..X+zati-1 loop M ← M+T(i) end loop
   M← Atalak(T, X, X+zati-1, M)
   for i in X+zati..X+(2*zati)-1 loop M ← M+T(i) end loop
   M← Atalak(T, X+zati, X+2*zati-1, M)
   for i in X+(2*zati).. Y loop M ← M+T(i) end loop
   M← Atalak(T, X+2*zati, Y, M)
   retun M
end
```

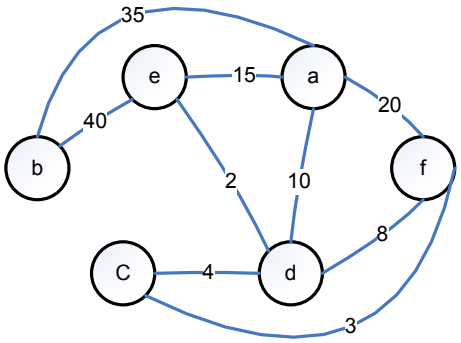
2. Demagun Europako eskualde bateko hiriburu guztien arteko tren bidezko konexioak ezagutzen ditugula. Eskualde horretako bi hiriburu emanik, lehenengotik bigarrengora joateko egin behar diren tren-aldaketa kopuru minimoa mugatuko duen algoritmo bat egin ezazu eta soluzioaren denbora ordena aztertu.

3. Enpresa batek K pertsona kontratatu nahi ditu G euroko gastua gainditu gabe. Enpresak n pertsonen txostenak ditu (n>K izanik) eta ezagutzen ditu i pertsona bakoitzeko ere haren s_i soldata eta p_i produkzio-etekina ($1 \leq i \leq n$).
   Enpresaren etekin maximoa zein den mugatzeko algoritmo bat idatz ezazu *programazio dinamikoaren* teknika erabiliz. Soluzioaren denbora ordena aztertu.

4. n zenbaki osoko desberdin ordenaturik dituen T taula bat emanik, T(i) = i gertatzen bada i indizea itzuliko duen algoritmo eraginkor bat idatz ezazu (lineala baino hobea). Baldintza hori betetzen duen posiziorik existituko ez balitz, algoritmoak 0 itzuli beharko du.

5. *Atazen banaketa agente artean.* Demagun N ataza eta N agente ditugula. *j* ataza ($1 \leq j \leq n$) *i* agenteari esleituko bagenio, honen egiteak K_{ij} kostua izango luke, kostuak K matrizean egonik. Backtrack teknika erabiliz, ataza bakoitza agente desberdin bati esleitu ataza guztiak burutu daitezen kostu global minimoz. Horretarako (a) esplorazio zuhaitza eraiki adarketa, kimak eta kasu nabariak identifikatuz; (b) erabilitako aldagaiak, parametrizazioa, hasieratzea eta lehenengo deia identifikatu; eta (c) algoritmoa idatz ezazu.

6. Primen algoritmoa aplika ezazu honako grafoaren hedapen zuhaitz minimoa kalkulatzeko. Zuhaitzaz gain, algoritmoaren begizta nagusiaren bira bakoitzaren ostean *Auzokide* eta *Pisu_Min* taulen egoerak erregistra itzazu ematen zaizkizun tauletan.



Auzokide

A	B	C	D	E	F

PisuMin

A	B	C	D	E

HZM ertzak


```
procedure  PRIM  (G: in Matrizea;  SErt: out  ErtzMultzoa) is
...
begin
      -- hasieraketa, ERP multzo fiktizioan 1 erpina dago soilik
      MultzoHutsaErt (SErt);
      for  K   in  G´First(1) +1..G'Last(1)  loop
            Auzokide(K) := 1;
            PisuMin := G(K,1);
      end loop;

      for Ind in  G´First(1)..G'Last(1)-1  loop
            Min := System.MAX_INT;
            for J in G´First(1)+1..G'Last(1) loop
               if 0≤ PisuMin(J)<Min then
                  Min:= PisuMin (J);
                  K:= J;
               end if;
            end loop;
            ErantsiErt(SErt,(K,Auzokide(K));
            PisuMin (K):= -1;
            for J in G´First(1)+1..G'Last(1) loop
              if  G(K,J)<PisuMin(J) then
                  PisuMin (J) :=  G(K,J);
                  Auzokide(J) := K;
              end if;
            end loop;
      end loop;
end PRIM;
```

```
procedure  GrafoaKorritu (G: in Grafoa) is
   Aztertua: constant Boolean := True;      // n= # erpinak; a= # arkuak
   Bisitak: Taula(1..n):= (Others=>  not (Aztertua));
begin
   for  Erp  in  1..n  loop
         if  not MarkatuaDago(Bisitak, Erp)  then
               MarkaIpini(Bisitak, Erp, Aztertua);
               SK(G, Erpina);   // ZK(G, Erp);
         end if;
   end loop;
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;
end SK;


procedure  ZK  (G: in Grafoa; X: in Erpina) is
   Ilara: IlaraDatuMota:=HutsaI;  YRenAuzokideak: ErpinenL;
   Y,Z: Erpina;
begin
   IlarariGehitu(Ilara, X);
   while not( HutsikDagoI?(Ilara))  loop
      Y:= Burua(Ilara);  Ilara:= IlaratikKendu(Ilara);
      YEnAuzokideak:= Auzokideak(Y).;
      while  not ( HutsikDagoL?(YRenAuzokideak))  loop
         Z:= LehenaL(YRenAuzokideak);  Hondarra(YRenAuzokideak);
         if  not MarkatuaDago?(Bisitak, Z)  then
               MarkaIpini(Bisitak, Z, Aztertua);
               IlarariGehitu(Ilara, Z);
         end if;
        end loop;
   end loop;
end ZK;
```

SOLUZIOAK

1) Algoritmoak 3 begizta lineal (n/3 aldiz) eta 3 dei errekurtsibo tamaina (n/3) egien ditu.

$T(n)= \Theta(n)+T(n/3)+ \Theta(n)+T(n/3)+ \Theta(n)+T(n/3)$

$= 3T(n/3) + n$

$T(3^k)-3T(3^{k-1})=3^k \qquad n=3^k$

$(x-3)(x-3)=0$

$T_k=a_1 3^k - a_2 3^k k; \quad T(3^k)=T(n)= a_1 n - a_2 n \lg_3 n \in \Theta(n\lg n)$

2- Bi erpinen (hiriburuen) arteko tren-aldaketa kopuru minimoa kalkulatzeko aukera desberdinak ditugu, grafoen gaineko korritzeak eraginkorrenak izanik. Sakonerako korritzea erabil liteke, atze prozesuan ume guztien arteko aukeren artean kostu minimoa ematen duena aukeratuz. Halere, zabalerako korritzea da zuzenena: abiapuntu erpinetik hasiz, aurreraka mailaz maila urruntzen joan topatu arte (jomuga) ala hedapena bukatu arte

o

```
procedure  GrafoaKorritu (G: in Grafoa; Irten,Iritsi: in erpina;
                          Bidea: out Boolean; Distantzia: in Integer) is
   Aztertua: constant Boolean := True;       // n= # erpinak; a= # arkuak
   Bisitak: Taula(1..n):= (Others=>  not (Aztertua));
begin
   Bidea:=false; Distantzia:=0;
   MarkaIpini(Bisitak, Irten, Aztertua);
   ZK(G, Irten, 0, Distantzia);
end GrafoaKorritu;


procedure  ZK  (G: in Grafoa; X: in Erpina; Luz: in Integer;
                Dis: out Integer ) is
   Ilara: IlaraDatuMota:=HutsaI;  YRenAuzokideak: ErpinenL;
   Y,Z: Erpina;
begin
   IlarariGehitu(Ilara, (X,Luz));
   while not( HutsikDagoI?(Ilara)) and not Bidea loop
      (Y,L):= Burua(Ilara);  Ilara:= IlaratikKendu(Ilara);
      YEnAuzokideak:= Auzokideak(Y);
      while  not ( HutsikDagoL?(YRenAuzokideak))  loop
         Z:= LehenaL(YRenAuzokideak);  Hondarra(YRenAuzokideak);
         if Z=Iritsi then Dist:=L+1; Bidea:=true;
         esle if  not MarkatuaDago?(Bisitak, Z)  then
               MarkaIpini(Bisitak, Z, Aztertua);
               IlarariGehitu(Ilara, (Z, L+1));
         end if;
        end loop;
   end loop;
end ZK;
```

3.- Programazio dinamikoa:

EMK(i,d,e)= Gehienez e euro gastatuz d pertsonen kontratazioak, 1..i pertsonen artean, produzi dezakeen etekin maximoa. //EtekinMaxiomokoKontratazioa

EMK(0,d,e) = 0

EMK(i,0,e) = 0

EMK(i,i,e) = 0 if s1+s2+…+si>e

= p1+p2+…+pi

EMK(i,d,e) = max{ EMK(i-1,d,e), pi+ EMK(i-1,d-1,e-si)} if si<=e

= EMK(i-1,d,e) if si>e

4.- bilaketa lineala baino hobe aizan behar duenez, bilaketa dikotomikoaren prozesua erabiliko dugu soluzioa, bektorea ordenatua baitago eta bilaketa esparrua ere.

```
function BilatuKointzidentzia (T(1..N)) return integer is
   Aurkitua:=false;
   H←1; B←N; P← ⌊ (B-H+1)/2⌋;
   While  B<H  and then  P ≠T(P) loop
      if P <T(P) then  B←P-1 else H←P+1 end if;
      P← ⌊ (B-H+1)/2⌋;
   end loop;
   if P ≠T(P) then return P else return 0; end if;
end;
```

Analisia: bilaketa N osagai dituen bektorean egiten da. Prozesuak bektor

5.- Backtrack

Iruzkinak

- Zuhaitzaren i mailan i ataza, libre dauden agentee artean esleituko da, guztiekin proba eginaz.
- Kasu nabariak: SP bektoreak n luzera duenean.

Aldagaiak:

Globalak izango dira:

- Ematen zaigun kostu matrizea: K(i,j), 1≤i,j≤N i ataza j agenteari esleitzearen kostua
- SOpt: soluzio optimoa
- SOptK: SOPT-ek ematen duen etekina orora

Parametro formalak

- SP: eraikitzen dugun soluzio partziala: $(X_1,...X_j,..X_{i-1})$, non $\forall j$ $(1 \le j < i \rightarrow X_j \in \{1,2,..,n\})$ eta $\forall j,k$ $(1 \le j,k < i \rightarrow X_j \neq X_k)$, j ataza X_j agenteari esleitzen zaio.
- SPK: SPko esleipenaren kostua; hots: BATUKARIA(j=1,i-1; K(j,SP(j)))
- Agente(1..N): agenteen egoera. Agente(j)=true, libre dago j agentea eta false, dagoeneko ataza bat esleitu zaio.

Haiseraketa eta 1 deia:

SOptK:= ∞;

AtazenEsleipenaBt (1, [], 0, Agente)

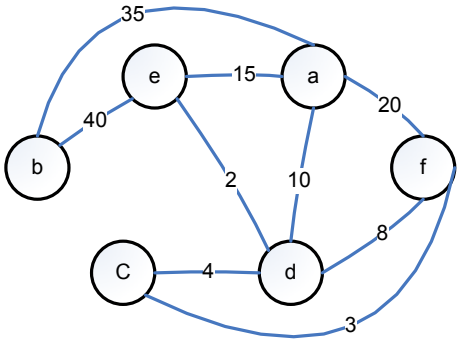
Kimak:

- Kima 1: agenteak dagoeneko atazen bat esleitua badu, ezin zaio besterik esleitu
- Kima 2: Daramagun esleipena eta aukera berriaren kostuak orain arteko soluzio optimoaren kostua gainditzen badu, ez jarraitu; hots, SPK+K(i,A)≥SOptK

Algoritmoa:

```
Procedure AtazenEsleipenaBt (i, SP,SPK, Agente)
begin
  if i=n+1 then     if SOptK>SPK then SOptK:= SPK; SOPT(1..n):=SP;
                    end if;
  else
      for A in 1..N loop
            if Agente(A)        -- Libre dago, ez da Kima 1 aplikatzen
                  and  SPK+K(i,A)<SOptK            -- ez da Kima2 aplikatzen
            then  Agente(A):= false;
                  SP(i):=A;
                  AtazenEsleipenaBt ((i+1), SP, SPK+ K(i,A), Agente)
                  Agente(A):= True;
      end if;
      end loop;
  end if;
end.
```

7. Primen algoritmoaren aplikazioa



Auzokide

--	a	a	a	a	a
a	b	c	d	e	f
--	a	d	a	d	d
a	b	c	d	e	f
--	a	d	a	d	d
a	b	c	d	e	f
--	a	d	a	d	c
a	b	c	d	e	f
--	a	d	a	d	c
a	b	c	d	e	f
--	a	d	a	d	c
a	b	c	d	e	f

PisuMin

--	35	∞	10	15	20
a	b	c	d	e	f
--	35	4	-1	2	8
a	b	c	d	e	f
	35	4	-1	-1	8
a	b	c	d	e	f
	35	-1	-1	-1	3
a	b	c	d	e	f
	35	-1	-1	-1	-1
a	b	c	d	e	f
	-1	-1	-1	-1	-1
a	b	c	d	e	f

HZM ertzak

(a,d)
(d,e)
(d,c)
(c,f)
(a,b)