ALGORITMOEN DISEINUA

G2: DATU-EGITURA AURRERATUAK:
meta, partiketa eta grafoa

1) Izan bedi G grafo zuzendua. G auzokide listen bidezko adierazpidea erabiliz emana dator. Idatzi eta aztertu

   a) G-ren adierazpidea auzokidetza matrizera bihurtzen duen metodoa eta aztertu.

   b) G-ren adierazpidea inzidentzi matrizera bihurtzen duen metodoa eta aztertu.

2) Izan bedi G grafo zuzendua. G auzokidetza matrize bidezko adierazpidea erabiliz emana dator. Idatzi eta aztertu

   a) G-ren adierazpidea auzokideen listara bihurtzen duen metodoa eta aztertu.

   b) G-ren adierazpidea inzidentzi matrizera bihurtzen duen metodoa eta aztertu.

3) $G=(E,A)$ grafo zuzendu baten karratua $G^2=(E,B)$ da, non $(u, w)\in B$ arkuak existitzen duen baldin eta soilik baldin $v\in E$ erpin batentzat existitzen badute $(u, v) \in A$ eta $(v, w) \in A$ arkuek. Hots, G^2-k arku bat du u-tik w-ra, G-k zehazki u-tik w-raino 2 luzerako bidea duenean. Idatzi algoritmo eraginkorrak G^2 kalkula dezaten:

   a) G auzokide listen bidez adierazirik dagoenean, eta

   b) G auzokide matrize bidez adierazirik dagoenean.

a) G grafo zuzendua emanik eta bertako i erpina, *Indegree*(i) zenbakia kalkulatu nahi dugu; hots, i auzokide duten erpinen kopurua. Grafoen ohiko bi adierazpideetarako algoritmo bana idatzi *Indegree*(i)a kalkulatzeko, hauen denbora kostuak mugatu eta argudia ezazu bi adierazpideetatik zein den egokiena.

4) G=(E,A) grafoa auzkokideen zerrendak erabiliz definiturik dagoena, irudika ezazu:

V=[V(1),V(2),V(3),V(4),V(5),V(6),V(7),V(8)]	
V(1)=[4]	V(5)=[1,2,6]
V(2)=[1]	V(6)=[2,7]
V(3)=[8]	V(7)=[3]
V(4)=[2,3,8]	V(8)=[7]

   a) 1 erpinetik hasiz Sakoneran korritzerakoan sortzen den zuhaitza marraztu, dagoeneko bisitatutako erpinak puntuzko marraz lotuz. 1 erpinetik hasi den korritze

horretako erpinen eta arkuen zerrenda idatzi lehenengo bisita ordenan. (a,b) idazkera erabil ezazu a-tik b-ra doan arkua adierazteko)

b) 1 erpinetik hasiz zabaleran korritzerakoan sortzen den zuhaitza marraztu, dagoeneko bisitatutako erpinak puntuzko marraz lotuz. 1 erpinetik hasi den korritze horretako erpinen eta arkuen zerrenda idatzi lehenengo bisita ordenan.

5) <2, 16, 7, 4, 18, 30, 52, 46> osokoen sekuentzia emanik meta-eraiki: (a) *Azaleratu* erabiliz eta (b) *Hondoratu* erabiliz

6) <10,2,9,16,8,6,1,3,12> taula erabiliz meta bat (*Heap*) eraiki ezazu soilik hondoratu prozedura erabiliz. Metaren egoera marraztu behar duzu (bai zuhaitz bitarra bezala bai eta honek adierazten duen taula bezala) eraiketa-pauso garrantzitsu bakoitzaren ondoren.

7) *n* osagai desberdin dituen bektore baten adibide konkretua eman ezazu, non *M* eta gaineko honako agindu sekuentzia aplikatu ondoren meta desberdina lortuko dugun.

```
M:=MetakoHandiena(B(1..n));
ErroaEzabatu(B);
OsagaiaGehitu(B(1..n-1), M);
```

8) Demagun 12 osagai dituen *B(1..12)* bektorea dugula, non *B(i)=i* gertatzen den $i \leq 12$ *i*-ren balio guztientzat. Jarraiko agindu sekuentzia egikaritu ondoren bektorearen egoera marraz ezazu. Aginduak bata bestearen ondoren exekutatzen dira, eta lehengoa ezik, gainontzekoak aurreko aginduak itzuli duen bektore gainean egikarituko da.

```
MetaEraiki(B);
EguneratuMetaBalioz(B,12,10);
EguneratuMetaBalioz(B,1,6);
EguneratuMetaBalioz(B,5,8);
```

9) Ondorengo galderei erantzun:

a) *h* altuera duen meta batek, gutxienez zenbat osagai eduki ditzake? Eta gehienez?

b) *Min-Meta* oro txikienetik handienera ordenaturik dagoen bektorea da? Eta alderantziz?

c) *Max-Meta*-k eraikitzeko Hondoratu prozedura erabiliz egin liteke. Metodo honek zergatik `n/2..1` posizioak aipaturiko ordenan hondoratzen ditu eta ez `1..n/2` ordenan?

10) Metatik_Ezabatu(A,i) izeneko algoritmoa egizu, A max-metako i posizioko osagaia metatik kenduko duena. Algoritmoaren denbora ordena O(lg *n*) izan behar du, max-metak *n* osagai dituenean.

11) *k* zerrenda ordenatuak, *n* osagaiko zerrenda ordenatu bakarrean biltzen dituen algoritmoa egizu (hots, *k* zerrendetan dauden osagaien kopurua guztira *n* da). Algoritmoaren exekuzio denbora O(n lg *k*). (Iruzkina: min-meta egitura bateraketa-bilketa egiteko erabiltzea gomendatzen zaizu).

12) `k-meta` funtsean beterik dagoen *k*-adarretako zuhaitzak erabiliz osatutako meta bezala defini dezakegu:

   a) `k-meta` bektore batean eraginkorki nola biltegi litekeen deskriba ezazu, azalduz zeintzuk eragiketa egin behar diren edozein erpin baten gurasoa eta umeak lortzeko.

   b) `k-meta` -ko `Hondoratu` eta `Azaleratu` eragiketen kostuak aztertu.

   c) Metak maneiatzen dituzten gainontzeko eragiketetan (`Max-Meta-Eraiki(V)`, `MetariGehitu(V,B)`, `MetatikKendu(V,p)`, `EguneratuMeta(V,p,B`)) egin behar diren aldaketak zehaztu, `k-meta` berrira egokituak izan daitezen.

   Metak erabiltzen dituen ordenazio algoritmoa berriz ere azter ezazu, oraingoan `k-meta` darabilkien inplementazioa duen bertsioa duzula suposatuz.

13) Izan bitez `T1` eta `T2` bi meta, zehazki biek 2^n osagai dituztenak. Hauetan dauden osagaiak konbinatuko dituen prozedura bat aurki ezazu `T3` zuhaitza eraikitzeko, prozedurak egin ditzakeen osagaien arteko konparaketa ordena `o(n)` izan behar du. (Oharra: prozeduraren kostu ordena handiagoa izan badaiteke ere, metetako elementuen arteko eragiketa kopurua `o(n`)koa izan behar du).

14) Ondorengo ariketa `Heapsort` algoritmoaren bertsio berri bat aztertzen du. Hain zuzen, bertako metaren eraiketaren prozesua aldatzen da.

```
procedure HeapSort (S: in out Sek) is
begin
     MetaEraiki2(S'First, S'Last);
     for Heaparen_Luzera   in reverse  S'First+1 .. S'Last loop
         STaulanTrukea(S'First, Heaparen_Luzera);
         ErroaHondoratu(S'First, Heaparen_Luzera -1);
     end loop;
end HeapSort;
```

15) Meta datu-mota abstraktua abiapuntutzat harturik, taula bat emanik hura metan bilakatuko duen `MetaEraiki2` algoritmoa idaztea eskatzen da.

Baina, haren eraiketarako *AzkenaAzaleratu* eragiketaz bakarrik balia daiteke programatzailea. *AzkenaAzaleratu* prozeduraren zehaztapena ondorengoa da:

```
procedure AzkenaAzaleratu (Hasi,Buka: in SekIndizea);
Sarrera:  Hasi eta Buka-1 indizeek S taulan meta bat mugatzen
          dute.
Buka indizean dagoen balioa S-en azaleratu nahi den osagaia da.
Irteera:  Hasi eta Buka-1 indizeek S taulan meta bat mugatzen
          dute.
Efektua:  S(Hasi,Buka) maximoen metan, metaren propietatea
          berreskuratu arte S(Buka) balioa azaleratzen du.
```

a) *AzkenaAzaleratu* prozeduraren kodea honako hau bada:

```
procedure AzkenaAzaleratu (Hasi,Buka: in SekIndizea) is
  Ind_J: SekIndizea;Ind_K: SekIndizea:=Buka;
begin
  loop
    Ind_J := Ind_K;
    if  Hasi<Ind_J  and then  S(Ind_J/2)<S(Ind_K)
    then  Ind_K:= Ind_J/2;  end if;
    STaulanTrukea (Ind_J, Ind_K);
    exit when Ind_J = Ind_K;
  end loop;
end AzkenaAzaleratu;
```

Egindako lana neurtzeko eragiketa adierazgarri bezala S taulako osagaien arteko alderaketak hartzen baditugu (non S taulak hasieran n osagai dituen), `MetaEraiki2` kasu txarrenean zein ordena zehatzekoa da? `HeapSorten` zein bertsio da azkarragoa: aurreko atalekoa edo Hondoratu prozedura erabiliz lineala dena? Erantzuna arrazoitu bedi gehienez 5 lerro erabiliz.

16) (a) Taula meta bihurtzen duen ondorengo algoritmoa aztertu:

```
procedure MetaEraiki (V(1..m)) is
begin
  for J in reverse 1 .. m div 2 loop
    Hondoratu(V(1..m), J);    -- V(J) posizioko balioa hondoratu
  end loop;
end;
```

(b) Aurreko ataleko ordena O(*m*) lortu ondoren, non dago akatsa HeapSort aztertzerakoan honako arrazonamenduan?

• O(*n*) denboran *MetaEraiki(T(1..n))* algoritmoak $\left\lfloor \frac{n}{2} \right\rfloor$-ren aldiz egikaritzen du Hondoratu.

• Beste algoritmo honek:

```
procedure HeapSort (T(1..n)) is
begin
  MetaEraiki(T(1..n));
  for J in reverse 2 .. n loop
    Trukea(T(1),T(J));
    Hondoratu(T(1..J-1), 1);  -- Erroa hondoratu
  end loop;
end;
```

Meta eraikitzen du lehengo ataleko algoritmoa erabiliz. Ondoren, begiztak *n−1* aldiz egikaritzen du (*Trukea*+*Hondoratu*); edo ordena aldetik berdina dena, $2(\lfloor \frac{n}{2} \rfloor$x *Trukea*+*Hondoratu))*

Aurreko bi puntuetako hiru zatiek O(*n*) ordena dutenez, `HeapSort` algoritmoaren ordena O(*n*) da.

17) Demagun *n* osagai dituen taula bateko *K* giltza handienak kalkulatzeko honako algoritmoa ematen digutela:

```
function  KGiltzaHandienak  (V: in Taula; K: in Integer)
                                  return Taula is
  M: Taula; Giltzak: Taula(1..K);
begin
  MMetaEraikiVEmanik(V,M);
  for  J  in 1 .. K  loop
    Giltzak(J):= M(1);
    M(1):= M(N-J+1);           -- Azkena errora
    ErroaHondoratu(M(1..n-J));
  end loop;
end KGiltzaHandienak;
```

Algoritmoaren ordena lineala, O(*n*), izan dadin, zer ordena eduki behar du *K*-k (*n*-ren funtziopean)?

18) *V* taulak *n* zenbaki arrunt desberdin du. V-ko *m* osagai txikienak kalkulatu nahi ditugu. Bestalde, *m*<<<*n* betetzen dela badakigu. *m* zenbakien kalkulua modurik eraginkorrenean egin nahi dela eta, nola kalkulatuko zenituzke?

a) *V* ordena gorakorra jarraituz sailkatu eta lehenengo *m* osagaiak itzuliz?

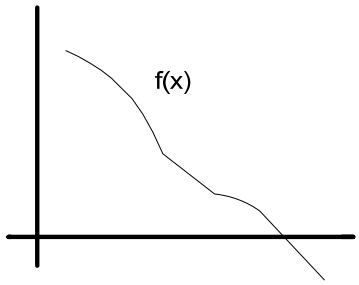
b) *KHautaketa(V,k)*ri m aldiz deituz (non k=1,2, ... eta m)?

c) Beste metodoren bat erabiliz? Zein?

Zure erantzuna argudia ezazu kasu txarrenetako eta batez besteko kasuetako analisi ezagunak alderatuz.

19) Ondorengo algoritmoa osatu n osagaien arteko K garren handiena kalkula dezan. n osagaiak banaka lortu behar dira *get-number(X : out integer)* prozedurari n dei eginaz. Kasu honetan erabilitako espazio estra minimizatu behar da. Hori dela eta, beti $\Omega(n)$ memoria- espazio estra behar duten soluzioak ez dira onartuko.

```
for  I  in  1 .. n  loop
   get-number (X)
   …
   end loop;
return KGarrena;
```



20) G=(ERP,ERT) grafo ez-zuzendu baten osagai konexuak kalkulatzeko partiketa egitura erabiltzen duen ondorengo algoritmoa ematen da:

```
proz  OsagaiKonexuak (G)
-- Partiketaren hasieraketa
hasi
   for  Da(a, ERP erpin bakoitzarentzat)  egin
      Multzoa_egin(v)
   end for;
   for  Da((x,y), ERT ertz bakoitzarentzat) egin
      if Bilatu(x) <> Bilatu(y)
      then Bateratu(Etiketa(x),Etiketa(y))
      end baldin;
   end for;
end proz;
```

Grafoak *K* osagai konexu baditu, zenbat Bilatu eragiketa egingo da? Eta zenbat *Bateratu* eragiketa? Emaitza $|ERT|$, $|ERP|$ eta *K*-ren funtziopean eman.