ALGORITMOEN DISEINUA
G4: ZATITU ETA IRABAZI TEKNIKA

1) Problema jakin bat ebazteko, algoritmo sinple bat daukagu. Honek, *n* tamainako sarrerak ebazteko koadratikoa den denbora-ordena hartzen du (hau da, $t(n) \in O(n^2)$). Problema bera ebazten duen eta *zatitu eta irabazi* teknika jarraitzen duen beste algoritmo bat aurkitu da ordea. Estrategia hori jarraituz, *n lg n* eragiketa egiten da problema tamaina erdiko bi azpiproblematan banatzeko. Aldi berean, beste *n lg n* eragiketa egin behar da azpiproblemen emaitzak konbinatuz jatorrizko problemaren emaitza lortzeko. Zatitu eta irabazi algoritmo berria hasierako algoritmoa baino eraginkorragoa da?

2) Ondorengo algoritmoak osokoen bektore bat emanik, bertako osagai handiena eta txikiena mugatzen ditu. Kasu txarrenean algoritmoak osagaien arteko zenbat alderaketa egiten dituen kalkula bedi.

```
proz MaxMin (T[i..j], Max, Min)
-- i <= j
begin
  if T[i]<=T[j] then Max:= T[j]; Min:= T[i];
  else  Max:= T[i]; Min:= T[j]; end if;

  if i+1<= j-1
  then  MaxMin (T[i+1..j-1], Lag_Max, Lag_Min);
        if Max<Lag_Max then Max:= Lag_Max; end if;
        if Lag_Min<Min then Min:= Lag_Min; end if;
  end baldin;
end proz;
```

3) Ondorengo algoritmoa telefono-zerrenda batean abizenak bilatzeko balio du. (Noski, zerrenda alfabetikoki ordenaturik dago.)

```
fuction Bilatu (Zr: in Zerrenda; Abizena: in String;
                Salto: in Positive) return Integer is
-- Abizena beti Zr zerrendan dagoela suposatzen da.
-- Zr zerrendan Abizena osagaiaren indizea itzultzen du.
begin
   Indizea:= Zr'First+Salto-1;
   while Zr'Last>Indizea and then Abizena>Zr(Indizea) loop
     Indizea:= Indizea+Salto;
   end loop;
   if Zr'Last>Indizea then Muga:= Indizea;
   else Muga:= Zr'Last; end if;
   return (BilaketaLineala(Zr(Indizea-Salto+1..Muga), Abizena));
end Bilatu;
```

a) Kasu txarrenean zenbat alderaketa egiten dira Abizena eta Z erreko osagaien artean? Ezaguna da *BilaketaLineala(Zr(X..Y),Abizena)*-ri eginiko deiak $Y-X$ alderaketa egiten dituela.

b) Algoritmoaren exekuzio ordena aldatuko litzateke *BilaketaLinealari* deia egin ordez *BilaketaDikotomikoa(Zr(X..Y),Abizena)*-ri egingo bagenio (jakinez honek $1+\lfloor \log (Y-X) \rfloor$ alderaketa egiten dituela)?

4) *n* giltza desberdinez osatutako sekuentzia bat izanik, ondoren ematen den *Txertaketa* ordenazio algoritmoaren aldaera azter bedi:

∀i 2≤i≤n: S(1)≤S(2)≤ ... ≤S(i-1) sekuentzian S(i) zuzenean bere posizio erlatiboan txertatzen da. BilaketaDikotomikoa erabiltzen da S(i)-ren posizio zuzena S(1)≤S(2)≤ ... ≤S(i-1) sekuentzian mugatzeko.

Hau da, Txertaketa algoritmoak *S(i)* osagaia zein posiziotan txertatu behar duen mugatzeko ondorengo algoritmo hau erabiltzen du:

```
proz PosizioaDikotomikoki(Sek, Balioa, Posizioa)
-- Balioa  sekuentzian ez dago
   N:=Sek'length; H:=Sek'First;
   A:=Sek'Last; E:= (H+A) div 2;
begin

  if N=1
  then  if    Sek(H) < Balioa   then  Posizioa:= H+1;
        else Posizioa:=H; end if;
  else
     if Sek(E)<Balioa
     then PosizioaDikotomikoki(Sek(E+1..A), Balioa, Posizioa);
     else PosizioaDikotomikoki(Sek(H..(E-1)), Balioa, Posizioa);
     end if;
  end if;
end;
```

Adibideak:

S

2	5	8	18	20	30	Balioa	…	S(n)
1	2	3	4	5	6	i	…	n

PosizioaDikotomikoki(S(1..6), 1, P) → P = 1

PosizioaDikotomikoki(S(1..6), 3, P) → P = 2

PosizioaDikotomikoki(S(1..6), 9, P) → P = 4

Ondorengoa aztertzea eskatzen da:

a) Giltzen arteko alderaketak: Zein da kasu txarrena? Kasu txarren horren exekuzio denboraren ordena zein da?

b) Giltzen mugimenduak: Zein da kasu txarrena? Kasu txarrena horretan, zenbat giltzen mugimendu egiten dira guztira?

c) Zein da algoritmoaren ordena kasu txarrenean?

5) Anderrek Z zenbaki osoko positiboa pentsatu du. Z zenbakia zein den asmatzeko Anderri hari buruz itaun diezaiokezu bakarrik >, < eta = erlazioak erabiliz. Idatzi galdeketa sistematizatuko duen algoritmo bat. Idatzitako soluzioak Anderri o(Z) galdera egin behar dizkio. Frogatu hala dela.

6) Bilaketa Dikotomikoaren estrategia erabiliz, emaniko N zenbaki arrunta ondorengo hiru zenbaki arrunten biderkatze den ala ez mugatuko duen algoritmoa idatzi, soluzioaren denbora-ordena o(N) dela ziurtatuz. Soluzioaren denbora-ordena kalkulatu.

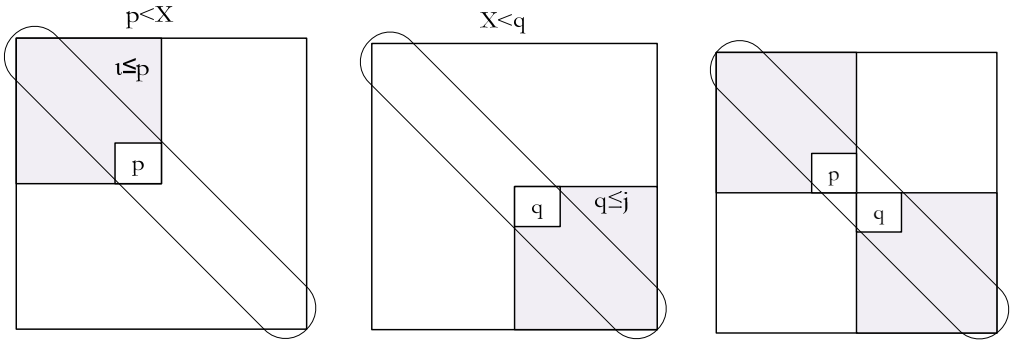
7) Demagun *V(1..N)* taulako zenbakiak txikitik handira ordenaturiko daudela bertako zenbait zenbaki negatiboak izan litezkeelarik. $i \in \{1,\ldots,n\}$ izanik, V(i)=i gertatzen den ala ez eraginkorki erabakiko duen algoritmoa egizu, dagoenean *i* itzuliaz. Soluzioa *o(n)* izan behar du (lineala baino hobea). Baldintza hori betetzen duen posiziorik existituko ez balitz, algoritmoak 0 itzuli beharko du.

8) Izan bedi `M(1..n, 1..n)` errenkadetako eta zutabeetako osagaiak ordena gorakorra jarraituz ordenaturik dituen matrize bat (ezkerretik eskuinera eta goitik behera). X osagaia M matrize horretan dagoen edo ez mugatzeko honako algoritmoa jarraitzen da:

   • Bilaketa dikotomiko bidez X diagonalean bilatzen da.

   • Baldin X aurkitzen badugu, amaitzen dugu.

   • Bestela, (*p* eta *q* aurkitzen ditugu non $p<X<q$) X egon ez daitekeen bi matrize zatiak baztertzen ditugu eta errekurtsiboki prozesu bera gelditzen diren beste bi zatitan aplikatzen da. Zein da algoritmoaren ordena kasu txarrenean?



9) $a_1, a_2, ..., a_n$ osokoen sekuentzia taula batean metaturik dago. Gainera, taulako edozein *ai* posizio jarraietan metaturik dauden sekuentziako osagaien balioak gehienez unitate batean desberdinak direla ezaguna da. Sekuentzian osagai jakin bat, *X*, ote dagoen eraginkorki mugatuko duen algoritmoa idaztea eskatzen da.

10) Osokoen bektore bat emanik, bertako osagai handiena eta txikiena zeintzuk diren erabakitzen duen algoritmo bat ebatzi. Bistakoa da *2n-2* alderaketa egiten dituen soluzioa. Hobea bilatu; hots, lineala bai, baina kopuru hori baino osagai arteko alderaketa gutxiago egiten dituena, adibidez ($3/2n \pm a$) egiten dituena.

11) *MergeSort* algoritmoetan zenbat osagaien mugimendu egiten da kasu txarrenean? Eta zenbat osagai arteko alderaketa? *BubleSort*, *SelectionSort* eta *InsertionSort* algoritmoen emaitzak eta honenak alderatu eta ondorioak atera. Osagaien arteko alderaketa kopuruak zenbatzea soilik, algoritmo hauek egiten duten lanaren adierazgarri ona da? Algoritmo guztietan?

12) *Mergesorte*-k memoria espazio estra lineala behar duela eta, gastu hori ekiditeko Mergesort-en ondorengo bertsio berria asmatu dugu: Bertsio berriak behar duen denbora ordena kalkula bedi.

```
procedure MSTaulaGainean (V: in out Tautla; I,J: in Indizea) is
  K: Integer:= (I*J) div 2;
begin
  if TxikiaDa?(I,J)  then  SortSinplea(V,I,J);
  else
     MSTaulaGainean(V,I,K);
     MSTaulaGainean(V,K+1,J);
     MSTaulaGainean(V,I,K,J);
  end if;
end MSTaulaGainean
```

Bertako *MSTaulaGainean(V, I, X, J)* prozedurak V-ren *V(I ..J)* azpitaula ordenatzen du *V* en bata bestearen segidan dauden *V(I..X)* eta *V(X+1..J)* segmentu ordenatuak ematen zaizkionean parametro bezala.

```
procedure MSTaulaGainean (T: in out Taula;  I,X,J: in Indizea) is
  Ezk: Indizea:=I; Esk: Indizea:=X+1; Lag: TElem;
begin
  while Ezk \= Esk loop
     if T(Ezk)>T(Esk)
     then  Lag:= T(Ezk);  T(Ezk):= T(Esk);
           Ordenatuta_utziz_Lag_T(Esk..J)n_txertatu
     end if;
     Ezk:= Ezk+1;
  end loop;
end MSTaulaGainean;
```

13) Osokoen taula bat emanik, bertako *K* osagai txikiena bilatuko duen algoritmoa idatzi eta ordena azter ezazu; hots:

m *Bek-eko K. osagai txikiena da* $\Leftrightarrow$ $\mathcal{N}i\{1 \le i \le n \wedge Bek(i) < m\} < K$
$\mathcal{N}i\{1 \le i \le n \wedge Bek(i) \le m\} \ge K$

K=m/2 denean, ebazten den problemari medianaren hautaketa deritzo. Honen adibide bi:

81	18	17	1	50	36	-8	

-8	1	17	18	50	36	81	90

14) (a) Ezaguna den bilaketa dikotomikoa errekurtsiboki kalkulatzen du ondorengo algoritmoak: (X bektorean ez badago, orduan 0 indizea itzuliko du)

```
proz    BilaketaDiko(V(i..j), X, Ind)
  if  i< j  then
     k:= (i+j) div 2;
     if  V(k)=X  then Ind:=k;
     else
       if  V(k)<X  then BilaketaDiko(V(k+1..j),X,Ind);
       else  BilaketaDiko(V(i..k-1), X, Ind);
       end if;
     end if;
  else
     if  i=j and then V(i)=X  then Ind:= i;
     else Ind:= 0;
     end if;
  end if;
end proz;
```

Algoritmoaren bi inplementazio desberdin kontsidera bitez, batean parametroen pasatzea erreferentzia bidez egiten dena eta bestean aldiz balio (edo kopia) bidez. Bi inplementazioen denbora-eraginkortasunean diferentziarik badago?

(b) Ohiko *MergeSort*-en azaldutako bi parametroen pasatze aukeren artean denbora-eraginkortasun diferentziarik badago?

15) 15 osoko dituen taula bat asma ezazu non taula hori *QuickSort*-i emanez gero *QuickSort*-en kasu onena den. Irudi bidez froga ezazu zure taula kasu onena dela.

16) Demagun bateraketa bidezko (*Merge*) ordenazio algoritmo bat erabili nahi dugula, baina bektorea hiru zatitan banatuz, errekurtsioz zatiak ordenatuz, eta ondoren hauek bateratuz.

a) `MergeSort3` algoritmoa idatzi eta aztertu.

b) Orain algoritmoa idatzi gabe, orokor ezazu aurreko prozesuaren analisia (kostua) 3 zatitan banatu ordez, jatorrizko bektorea K zatitan banatuko lukeena, eta hauek ordenatu ostean, K azpibektore ordenatuak bateratuko lituzkeena.

c) Aurreko analisitik zer ondorioztatzen duzu?

d) Eraman dezagun bektorearen zatiketaren ideia hori limiteraino, ondoren bateraketa bidez ordenatzeko. Horretarako `MergeSortApurrak` algoritmoa asmatzen dugu, non jatorrizko bektorea 2 osagai dituzten (n/2) zatitan banatzen duena, eta zati bakoitza ordenatu ostean, n/2 zatitxoak bateratzen dituena. Algoritmoa idatzi gabe, haren kostu funtzioa idatzi eta denbora ordena kalkulatu.

17) Aurreko ariketa kontuan izanik, `MergeSort`$\sqrt{n}$ algoritmoa asmatu dugu oraingoan: jatorrizko bektorea $\sqrt{n}$ tamaina duten $\sqrt{n}$ zatitan banatzen duena, eta $\sqrt{n}$ zatiak errekurtsioz ordenatu ostean bateraketa bidez bektore osoaren ordenazioa lortzen duena.

Ondorengo errekurtsioa gara ezazu:

T(n)= $\sqrt{n}$ * T($\sqrt{n}$) + n baldin n>1 T(1) = 1

Ebatzi duzun errekurtsio ekuazio, `MergeSort`$\sqrt{n}$ algoritmoari dagokio? Arrazoitu erantzuna.

18) *Mergesort* algoritmoaren *TA(n)*, *TM(n)*, *MEE(n)* eta ordena zehatzak (bai denborarako bai eta espazio estrarako) honakoak dira:

```
TA(n)= Kasu txarrenean alderaketa kopurua = (n-1) + 2T(n/2)
TM(n)= K. txarrenean mugimenduen kopurua = 2n + 2T(n/2)
```

Algoritmoak batez ere, eragiketa horiek egiten dituenez, algoritmoak egiten duen lana bi horien batuketa da: $T(n) = TA(n)+TM(n) \in \Theta(max(TA(n), TM(n)) = \Theta(n \lg n)$

$MEE(n) = n+a + MEE(n/2) \in \Theta(n)$

Orain, bertsio *MergeSort*-en aldaera berriak idaztea eskatzen da. *Merge* metodo berriak behar izanez gero hauek "idatzi". Ahalik eta onenak izan behar dute metodoek.

a) *Mergesort3*: tamaina berdineko *3* zatitan banatzen du sarrerako taula

b) *Mergesort4*: tamaina berdineko *4* zatitan banatzen du sarrerako taula

c) *Mergesort*$\sqrt{n}$: tamaina berdineko $\sqrt{n}$ zatitan banatzen du sarrerako taula

d) *Mergesort(n/2)*: tamaina berdineko *(n/2)* zatitan banatzen du sarrerako taula

e) *MergesortK-Heap*: tamaina berdineko K zatitan banatzen du sarrerako taula eta Meta egitura nonbait erabiltzen da. Bestera: Izan bitez K taula ordenatuak, guztira n osagai dituztenak. K osagaien arteko txikiena minimoen metak erabiliz muga dezakegu. Taula horien *merge* bidezko bateraketa metak erabiliz aztertu.

19) $1 \leq m \leq n$ izanik, n zenbaki osoko dituen bektore bateko m zenbaki txikienak mugatu nahi ditugu (adibidez, bektoreko aurrealdean kokatuz). Ez da beharrezkoa m osagaiak ordenaturik ematea.

   a) QuickSort algoritmoan inspiratuz, eta konkretuki, honen Banaketa prozedura erabiliz, problema hori ebazteko algoritmoa idatzi. Guztiz garrantzitsua da soberazko ordenazioak ez egitea.

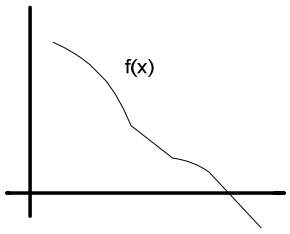
   b) Zure algoritmoaren exekuzio-ordena kasu txarrenean kalkula ezazu.

   c) Demagun zenbakizko bektore baten mediana kalkulatzeko Mediana algoritmo lineala dugula. Demagun prozedura hori lehenengo ataleko algoritmoan banaketa-puntua (pibotea) aukeratzeko erabiltzen dugula. Era horretan, bektorean pibotea baino txiki- agoak eta handiagoak diren osagaien kopuruak berdinak dira. Algoritmoa idatzi gabe, azter ezazu honen denbora ordena kasu txarrenean.

20) Argirik ez dagoen gela batean bi kaxoi ditugu: batean n torloju daude eta bestean aurrekoen n azkoinak. Problema torlojuen eta azkoinen doikuntza egitea da. Erlazioa biyektiboa da; hots, torloju bakoitzari azkoin bakarra dagokio eta alderantziz. Ilunpean gaudenez, aukera bakarra torloju eta azkoin parea hartzea eta estutuz probatzea da ea doitzen diren, azkoina handiegia den ala txikiegia den. Ataza burutzeko diseina ezazu algoritmo eraginkorra (bataz besteko kasuan $o(n^2)$ multzokoa den soluzioa).

21) $f(x)$ funtzio monotono beherakorra izanik, izan bedi n osokoa $f(n) \geq 0$ betetzen duen zenbaki osoko handiena. n existitzen duela asumituz, hura kalkulatzen duen algoritmoa honako hau da:

```
I:=0;
while F(I)>=0 loop
   I:= I + 1;
end loop;
N = I-1;
```



Halere, soluzio horren ordena O(n) da. Portaera asintotiko hori (n-ren funtziopean) hobetzen duen algoritmoa idaztea eskatzen da.

22) Osokoen taula bat emanik, batura maximoa ematen duen segmentua mugatu nahi da. Balioa eta segmentua kalkulatuko dituen metodoa idatzi eta aztertu. Adibidea: *baturaMax=29* litzateke honako sekuentziarentzat

22	37	8	12	1	-10	18	-25	20	-3

23) Osokoen $A=[a_1,a_2,\ldots,a_n]$ bektorea emanik, A-n dauden osoko berdinen segmentu luzeena mugatzeko algoritmo bat diseina ezazu *zatitu eta irabazi* teknika erabiliz. Algoritmoa optimoa dela uste duzu? Zure erantzuna arrazoitu.

24) Ondorengo algoritmoak sarrerako hitza palindromoa den mugatzen du:

```
function Pal (H:    Hitza; i,j: Indizea) return boolean is
begin
  if i≥j then return true;
  elsif H(i)≠H(j) then return false;
  else return Pal(H, i+1, j-1);
  end if;
end;
```

Aztertu P al(H it, 1, n) deiak behar duen denbora eta denbora-ordena kasu txarrenean eta batez besteko kasuan. Suposatu sarrera guztiak ekiprobableak direla eta kateen alfabetoa {a, b} dela.

25) Izan bedi $T(1..N)$ zenbakien bektore bat. $Kard(i|T(i)=X) > n/2$ propietatea betetzen duen X osagaia mugatuko duen algoritmoa idaztea eskatzen da; hau da, n/2 aldiz edota maizago agertzen den X osagaia T-n. Halere, T-ko N osagaien artean propietate hori betetzen duen X osagairik ez egotea gerta liteke.

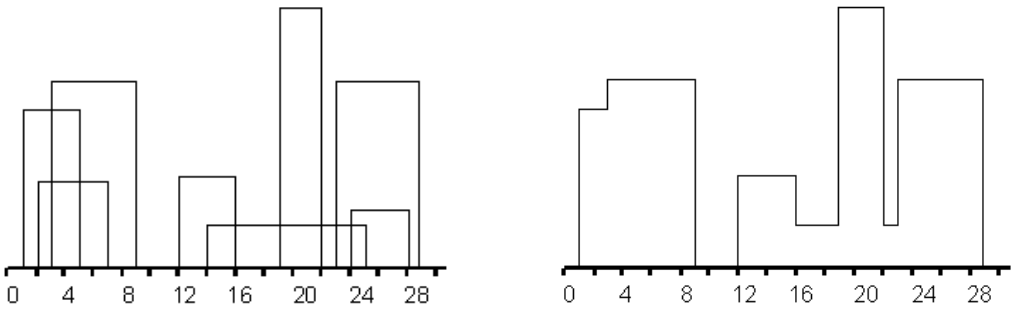
26) $A(a-1..b+1)$ taulako minimo lokala $A(k-1)\geq A(k)\leq A(k+1)$ baldintzak betetzen dituen $A(k)$ osagaia da. $a\leq b$, $A(a-1)\geq A(a)$ eta $A(b)\leq A(b+1)$ betetzen direla suposatuko dugu; baldintza hauek behintzat minimo lokal bat existitzen duela ziurtatzen dute. $A(a-1..b+1)$ tarteko minimo lokal bat mugatuko duen algoritmo bat idaztea eskatzen da.Soluzioa nabariak kasu txarrenean O(n) denbora hartzen duenez, zure soluzioa hori baino hobea izan beharko du. Horrela dela arrazoitu.

27) Ordenaturik dagoen V(1..n) bektorea emanik eta Z eta S bi balio (non $Z\leq S$), Z eta S balioen arteko balioz osaturiko V-ko azpibektorearen bi indizeak mugatzeko algoritmo bat egizu bilaketa dikotomikoaren ideia erabiliz. Soluzioak kasu txarrenean O(lg n) ordena duela frogatu. Adibidez, sarrera V=(3, 3, 3, 5, 5, 5, 7, 8, 8, 10, 12, 12, 12, 12, 12); Z=5 eta S=8 denean, irteerak (4,9) izan behar du.

28) N partaideko liga bidezko txapelketa bat antolatu behar dugu. Lehiakide bakoitzak eguneko partidu bakarra jokatuko du (salbuespen bakarra egongo da n bakoitia denean, egun bateko atsedena izango du). Lehien egutegi bat prestatu behar dugu, non denak denen kontra jokatu behar duten eta (**n-1**) egunetan amaitu beharko duena (egun bat

gehiago *n* bakoitia bada). *N* partaideak emanik eta zatitu eta irabazi teknika erabiliz, lehien egutegi bat ekoitziko duen algoritmoa egizu.

29) *Skyline* (poligonoen batura). Etxe-orratzak **(x,h,y)** hirukote bidez adieraz litezke; hau da, urrutitik begiratuta abszisa-ardatzarekiko **h** etxearen altuera litzateke eta **x** eta **y** balioak etxe-orratzaren hasiera eta amaiera puntuak lirateke ardatza horretan. Demagun $\mathbf{(x_i,h_i,y_i)}$ hirukoteen sekuentziak ($1 \le i \le n$) hiri bateko **n** etxe-orratz jasotzen dituela. Suposa dezakezu **n** etxe-orratzen sekuentzia ordenatuta dagoela etxe-orratzen hasierarekiko (x_i balioekiko). Helburua hiriko etxe-orratzek marrazten duten zeruertz-lerroa kalkulatzen duen algoritmoa egitea da; hots, ezkutuan gelditzen diren marrak ezabatuz zeruko perfila bi dimentsiotan kalkulatzen duen algoritmoa. Zeruertz-lerroaren adierazpideak (z_1, k_1, z_2, k_2,…, z_{m-1}, k_{m-1}, z_m) itxura izan dezake. Honek adieraziko luke, zeruertz-lerroa z_1 abszisan hasten dela k_1 altueraraino igoz, hau z_2 abszisaraino mantentzen dela, eta puntu horretan altuera k_2 izatera aldatzen dela, eta horrela behin eta berriro, azkeneko eraikina z_m abszisan amaituz. Adibidea: Adibidea, ((1,**11**,5), (2,**6**,7), (3,**13**,9), (12,**7**,16), (14,**3**,25), (19,**18**,22), (23,**13**,29), (24,**4**,28)) etxe-orratzei dagokien zeruertz-lerroa litzateke: (1,**11**,3,**13**,9,**0**,12,**7**,16,**3**,19,**18**,22,**3**,23,**13**,29,**0**), eta grafikoki:

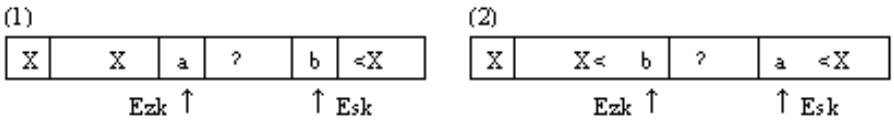


30) Problema hau *QuickSort*-ko Banatu prozeduraren aukera berri bat da.

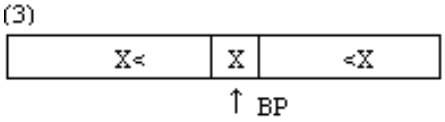
```
procedure QuickSort (Bek: in Sekuentzia) is
  BP, H,B: SekIndizea;
begin
  H:=Bek'First; B:=Bek'Last;
  if  H<B  then     Banatu2 (H,B,BP);
                    QuickSort(Bek(H, BP-1));
                    QuickSort(Bek(BP+1, B));
  end if;
end QuickSort;
```

Prozesuaren bertsio berria: *X=Bek(H). X*en balioaren funtzioan Bek sekuentziako balioak bi azpisekuentziatan banatzeko modu egoki bat, *Bek(H, B)* behin bakarrik korritzea da, baina sekuentziaren bi ataletatik hasita. Bi indize-aldagai behar dira, Ezk eta Esk. Indize hauek *H* eta *B+1* balioekin hasieratzen dira hurrenez hurren. Ezk inkrementatu egin behar da *Bek(Ezk)>X* gertatu arte eta, aldiz, Esk dekrementatu

Bek(Esk)≤*X* gertatu arte (1 irudia ikusi). Orduan *Bek(Ezk)* eta *Bek(Esk)* elkarraldatzen dira (2 irudia). Prozesuak *Ezk*<*Esk* baliozkoa den bitartean jarraitzen du.



Azkenean, Bek(H) eta Bek(Esk) permutatzen dira X bere posizio zuzen erlatiboan kokatzeko, hots, BP posizioan. (3 irudia)



31) Prozesuaren kodea ondorengo prozeduran jaso da:

```
procedure Banatu2 (Hasi,Buka: in SekIndizea;BP: outSekIndizea) is
  Ezk: SekIndizea:= Hasi;  Esk: SekIndizea:= Buka+1;
  X: SekBalioMota:= S(Hasi);

begin
  loop
     Ezk:=Ezk+1;
     exit when S(Ezk)>X or Ezk≥Buka;
  end loop;
  loop
     Esk:= Esk -1;
     exit when S(Esk)<=X;
  end loop;

  while  Ezk<Esk loop
     BekTaualanElkarraldaketa(Ezk, Esk);
     loop
        Ezk:=Ezk+1;
        exit when S(Ezk)>X
     end loop;
     loop
        Esk:= Esk-1;
        exit when S(Esk)≤X;
     end loop;
  end loop;
  BekTaualanElkarraldaketa(Hasi, Esk); BP:=Esk;
end Banatu2;
```

Galderak: n osagai dituen sekuentzia bat emanik:

a) *Banatu2* algoritmoan eta kasurik txarrenean, *Bek*-eko osagaien artean zenbat elkarraldatze egiten dira?

b) *QuickSort* algoritmoan eta kasurik txarrenean, *Bek*-eko osagaien arteko elkarraldatze kopuruaren ordena zein da?

c) *Bek*-ko osagaien arteko zenbat alderaketa egiten ditu kasu tzarrenean *Banatu2*-k? Hots, *Banatu2(BektF irst, BektLast, BanaketaP)* deiak *Bek*-eko osagaien artean egiten dituen alderaketa kopurua kasu txarrenean.

d) *QuickSort* algoritmoan eta kasurik txarrenean, *Bek*-eko osagaien arteko alderaketa kopuruaren ordena zein da?

e) *QuickSort*-ek sarrerako sekuentzia banatzeko goran azaldu den bertsioa erabiliko balu, bere ordena kasu txarrenean aldatuko litzateke? Eta batez besteko kasuan? Gehienez 10 lerrotan erantzuna arrazoi bedi.

32) Bektore bateko mediana, bektorea ordenatuta balego $\lfloor n/2 \rfloor$ posizioan legokeen balioa da. Honela, (2,4,6,5,3,1) bektoreko mediana 3 da.

a) Emaniko bektorearen mediana kalkulatuko duen algoritmoa idatz ezazu. Algoritmoaren konplexutasuna aztertu eta kalkulatu.

b) Orain, zatitu eta irabazi teknika bidez eta BANAKETA prozedura erabiliz *mediana*-ren problema ebazten duen metodoa garatu eta bere denbora-kostua kalkula ezazu kasu onenean eta txarrenean.

c) Demagun Θ(n) kostua duten bi prozedura ditugula dagoeneko definituak: MEDIANA (V(1..n),M), ematen zaion bektorearen mediana kalkulatzen duena M-en eta ANTOLATU(V(1..n),P) V bektoreko osagaien permutazio bat itzultzen duena V-en honako propietateak betetzen dituenak:

i) Baldin $V(i)<P$ eta $V(j)=P$ orduan $i<j$

ii) Baldin $V(i)=P$ eta $V(k)>P$ orduan $j<k$

d) MEDIANA eta ANTOLATU prozedurak erabiliz *QuickSort*(V(1..n))-en beste bertsio bat egizu kasu txarrenean o(n2) dena. Soluzioak aipatutako ordena betetzen duela frogatu behar duzu.

33) Aire kontrolerako sistema batek $\boldsymbol{n}$ espazio-ontzi ditu zirkulazioan bere ekintza-eremu barruan eta espazio-ontzi bakoitzaren $\boldsymbol{(x,y)}$ koordenatuak ezagunak izanik. Informazio hau guztia pantailaratu egiten da giza kontrolatzaile batek gertatzen denaren ikuspegi orokorra izan dezan. Dena dela, elkarren artean oso hurbil dauden hegazkinenei arreta berezia eman nahi diegu, elkar talka egiteko arrisku handiagoa baitute. Horretarako sistemak gorriz eta ñir-nir eginez elkarren artean hurbilen dagoen hegazkin parea adieraz zezan nahiko genuke. Egizu algoritmo bat kalkula dezan hegazkin pare hori, soluzioaren denbora-ordena kalkulatuz.