ALGORITMIKA

AZTERKETETAKO ARIKETA ZERRENDA

1. $\forall n \geq n_0 f(n) \geq g(n)$ eta $f(n), g(n) \in \Theta(h(n))$ baldintzetan egonik, zer esan dezakezu $f(n) - g(n)$ren ordenari buruz?

2. Froga bedi: $\forall a, b(a, b > 1 \Rightarrow \lg_a n \in \Theta(\lg_b n))$. Baldin $a \neq b$, orduan egia da $2^{\lg_a n} \in \Theta(2^{\lg_b n})$?

3. f eta h funtzioak algoritmo batzuk aztertu ondoren lortu diren kostu funtzioak dira. Arrazoitu ondorengo baieztapenak egiazkoak ala faltsuak diren:

   (a) $O(f(n)) = O(h(n)) \Rightarrow O(\lg f(n)) = O(\lg h(n))$

   (b) $O(\lg f(n)) = O(\lg h(n)) \Rightarrow O(f(n)) = O(h(n))$

4. $Waste(n)$ prozedura deiak idazten dituen lerro kopurua jaso beza $T(n)$-k.

```
procedure  Waste ( x: in INTEGER) is
begin
     for I in 1..x loop
       for J in 1..I loop  PUT_LINE(I,J, x); end loop;
     end loop;
     if x>0 then
        for I in 1..4 loop Waste(x div 2); end loop;
   end if;
end Waste;
```

   $T(n)$-ren ordena zehatza kalkula ezazu.

5. $n \geq 1$ zenbaki arrunta triangeluarra dela diogu baldin eta 1ean hasi eta jarraiko zenbaki arruntaz osaturiko ordena gorakorreko sekuentzia ez-huts baten baturaren berdina bada. Honela, lehenengo bost zenbaki triangeluarrak 1, $3 = 1+2$, $6 = 1+2+3$, $10 = 1+2+3+4$ eta $15 = 1+2+3+4+5$ dira.

   (a) Idatzi ezazu programa bat non emaniko zenbaki osoko positiboa zenbaki triangeluarra den ala ez erabakiko duen. Programaren eraginkortasuna $O(n)$ren barnekoa izan behar du eta erabilitako memoria espazio estra konstantea.

   (b) Zure soluzioaren ordena aztertu.

6. Demagun programa jakin bat exekutatzeko gauero PUZko ordu bete daukagula eta ordu horretan makinak kudea dezakeen programaren sarrerarik handiena $n = 1000000$ dela. Kalkulu zentruko denbora esleipen berri baten ondoren, egunero PUZko 3 ordu esleitzen zaizkigu. Zein da makinak denbora horretan kudea dezakeen programa beraren sarreraren tamainarik handiena $T(n)$ programaren konplexutasuna honakoa bada (k_i konstante batentzat)?

   (a) $k_1 n$

(b) $k_2 n^2$

(c) $k_3 10^n$

7. Eman dezagun gauero programa jakin bat exekutatzeko PUZko ordu bete daukagula eta ordu horretan makinak kudea dezakeen programaren sarrerarik handiena $n = 1000000$ dela. Orain, gure nagusiak aurrekoa baino 100 aldiz azkarragoa den makina berri bat erosi du. Zein da makina berriak ordu bete batean kudea dezakeen programa beraren sarreraren tamainarik handiena $T(n)$ programaren konplexutasuna honakoa bada (k_i konstante batentzat)?

   (a) $k_1 n$

   (b) $k_2 n^2$

   (c) $k_3 10^n$

8. n osagai dituen bektore bateko balio maximoak eta minimoak kalkulatu, horretarako osagaien arteko $\frac{3}{2}n$ alderaketa baino gutxiago eginez. Frogatu soluzioak aipaturiko alderaketa kopuru hori baino alderaketa gutxiago egiten dituela.

9. Honako errekurtsio ekuazioa ebatz bedi. Zein da bere ordena?

$$t(n) = \begin{cases} 2 \cdot t(\frac{n}{4}) + \lg n & n > 1 \\ a & n = 1 \end{cases}$$

10. Ondorengo programen egikaritzapen-denbora ordena aztertu:

    (a)
    ```
    function dena (n:positibo)
    if n=1 then 1 else dena (n-1) + 2 * partziala(n-1)
    ```
    jakinik
    ```
    function partziala (m:positibo)
    if m=1 then 1 else 2 * partziala(m-1)
    ```

    (b)
    ```
    function dena (n,m:positibo)
    if n=1 then m else  m + dena (n-1, 2 * m)
    ```

11. Testu editore askok ondorengo algoritmoa erabiltzen du. String konkretu baten lehenengo agerraldia kalkulatzen du (hau da, $B(1..m)$ karaktere-zerrendaren lehenengo agerraldia $A(1..n)$ karaktere-zerrendan bilatzen du) B stringa A-n hasten den indizea itzuliaz, B barne dagoenean noski. $Muga = n - m + 1$ balioa B stringa A-n has litekeen eskuinaldeagoko indizea da.

```
procedure StringSearch (A,B: in String; Aurkitua: out boolean;
                        Hasiera: out Indizea) is
    N:= A'Length;   Topatua:= false;
    M:= B'Length;   Muga:= n-m+1;
    I, J : Indizea; Hasi:= A'First;
begin
   while not Topatua  and  (Hasi  Muga) loop
       I:= Hasi;
       J:= B'First;
       while J/= M+1 and then  (A(i)=B(j))  loop
           I:= I+1; J:=J+1;
       end loop;
```

```
        Topatua:= (J=M+1);
        if  not Topatua  then  Hasi:= Hasi+1;  end if;
    end loop;
    Aurkitua:= Topatua;
    Hasiera:= Hasi;
end StringSearch;
```

Kasu txarrenean zenbat aldiz egikaritzen da $A(i) = B(j)$ alderaketa? Zeintzuk sarrerak dira kasu txarrena ematen dutenak? Ohar zaitez *Ada*n testa bakarrik $J \neq M+1$ baliozkoa denean kalkulatzen dela.

12. $< e_1, e_2, ..., e_n >$ n osagaiez osatutako bektorea ordenatzeko $MergeSort$ algoritmoaren pareko estrategia bat erabiltzen da baina bektorea 2 osagai dituzten $\frac{n}{2}$ zatitan banatuz eta Merge kodea $\frac{n}{2}$ sekuentzietara orokortuz.

    (a) Metodoaren exekuzio denboraren funtzioa idatzi bertako batugai bakoitzaren jatorria arrazoituz.

    (b) exekuzio-ordena kalkulatu.

13. Ondorengo algoritmoak sarrerako hitza palindromoa den mugatzen du:

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

Aztertu $Pal(Hit, 1, n)$ deiak behar duen denbora eta denbora-ordena kasu txarrenean eta batez besteko kasuan. Suposatu sarrera guztiak ekiprobableak direla eta kateen alfabetoa $\{a, b\}$ dela.

14. 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?

15. 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
    baldin   T[i] <= T[j]   orduan Max := T[j]; Min:= T[i];
    bestela  Max := T[i]; Min:= T[j];
    end baldin;
    baldin   i+1<= j-1   orduan
        MaxMin (T[i+1..j-1], Lag_Max, Lag_Min);
```

```
      baldin Max < Lag_Max orduan  Max := Lag_Max;  end baldin;
      baldin Lag_Min < Min  orduan  Min:= Lag_Min;  end baldin;
   end baldin;
end proz;
```

16. n giltza desberdinez osatutako sekuentzia bat izanik, ondoren ematen den $Txertaketa$ ordenazio algoritmoaren aldaera azter bedi:

    $\forall i\ 2 \leq i \leq n : S(1) \leq S(2) \leq ... \leq S(i-1)$ sekuentzian $S(i)$ zuzenean bere posizio erlatiboan txertatzen da. $BilaketaDikotomikoa$ erabiltzen da $S(i)$ ren posizio zuzena $S(1) \leq S(2) \leq ... \leq S(i-1)$ sekuentzian mugatzeko.

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

```
Alg PosizioaDikotomikoki(Sek, Balioa, Posizioa)
-- Balioa sekuentzian ez dago
N:=Sek'length; H:=Sek'First; A:=Sek'Last; E:= (H+A) div 2
baldin  N=1  orduan
       baldin  Sek(H) < Balioa  orduan  Posizioa:= H+1;
          bestela Posizioa:=H;
bestela
       baldin  Sek(E) < Balioa
       orduan   PosizioaDikotomikoki(Sek(E+1..A), Balioa, Posizioa)
       bestela  PosizioaDikotomikoki(Sek(H..(E-1)), Balioa, Posizioa)
```

    Adibideak:

    S

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

    $PosizioaDikotomikoki(S(1..6), 1, P) \rightarrow P = 1$

    $PosizioaDikotomikoki(S(1..6), 3, P) \rightarrow P = 2$

    $PosizioaDikotomikoki(S(1..6), 9, P) \rightarrow P = 4$

    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?

17. $Mergesort$ek 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  MergeSortTaulaGainean  ( 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
        MergeSortTaulaGainean(V,I,K);
        MergeSortTaulaGainean(V,K+1,J);
        MergeTaulaGainean(V,I,K,J);
    end if;
end   MergeSortTaulaGainean;
```

Bertako $MergeTaulaGainean(V, I, X, J)$ prozedurak Vren $V(I..J)$ azpitaula ordenatzen du Ven bata bestearen segidan dauden $V(I..X)$ eta $V(X + 1..J)$ segmentu ordenatuak ematen zaizkionean parametro bezala.

```
procedure MergeTaulaGainean ( 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  MergeTaulaGainean;
```

18. 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 Kk (n-ren funtziopean)?

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

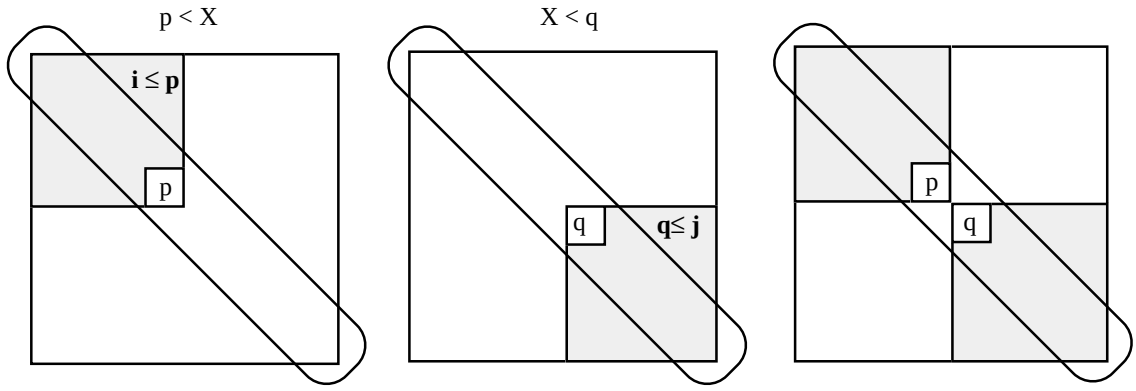
```
fuction Bilatu (Zr: in Zerrenda; Abizena: in String;
                          Salto: in Positive) return Integer is
    --  Abizena beti Zr zerrendan dagoela suposatzen da.
    --  Zr zerrendan Abizena osagaiaren indizea itzultzen du.
begin
    Indizea:= Zr'First+Salto-1;
    while  Zr'Last>Indizea  and then Abizena>Zr(Indizea) loop
        Indizea:= Indizea+Salto;
    end loop;
    if  Zr'Last>Indizea
    then  Hasta:= 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 *Zer*reko osagaien artean? Ezaguna da $BilaketaLineala(Zr(X..Y), Abizena)$ri eginiko deiak $Y - X$ alderaketa egiten dituela.

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

20. 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?

21. V arrayak n zenbaki arrunt desberdin du. Vko 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.

22. 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.

23. (a) Taula meta bihurtzen duen ondorengo algoritmoa aztertu:

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

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

- $Meta_Eraiki(T(1..n))$ algoritmoak $\lfloor \frac{n}{2} \rfloor$-ren aldiz egikaritzen du $Hondoratu$ $O(n)$ denboran.
- Beste algoritmo honek:

```
procedure HeapSort (T(1..n)) is
begin
    Meta_Eraiki(T(1..n))
      for I in reverse 2 .. n loop
            Trukea(T(1),T(I));
            Hondoratu(T(1..I-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, $2(\lfloor \frac{n}{2} \rfloor \cdot (Trukea + Hondoratu))$

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

24. Ondorengo ordenazio algoritmoa izanik:

```
procedure HerenkaSort (B: in out Bektorea; I,J: in Integer) is
      K: Integer:=  Lurra((J-I+1)/3);
begin
    if B(I) > B(J) then Trukatu(B(I),B(J)); end if;
    if J-I+1 <= 2  then return; end if;
    HerenkaSort (B, I, J-K);
    HerenkaSort (B, I+K, J);
    HerenkaSort (B, I, J-K);
end HerenkaSort;
```

- Bere exekuzio ordena aztertu.
- *InsertionSort* eta *HeapSort* algoritmoen exekuzio ordenekin alderatu.

25. Ondorengo agoritmoa osatu n osagaien arteko Kgarren handiena kalkula dezan. n osagaiak banaka lortu behar dira $get - number(X : outinteger)$ 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;
```

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.

(Zatitu eta irabazi teknika erabiltzea iradokitzen da.)

27. (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)
     baldin  i< j orduan
             k:= (i+j) div 2;
             baldin  V(k) = X   orduan  Ind:= k;
     bestela
             baldin V(k)<X orduan BilaketaDiko(V(k+1..j),X,Ind);
             bestela BilaketaDiko(V(i..k-1), X, Ind);
             end baldin;
     end baldin;
     bestela
             baldin  i=j  bai eta  V(i) = X  orduan Ind:= i;
             bestela Ind:= 0;
             end baldin;
     end baldin;
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-eragikortasunean diferentziarik badago?

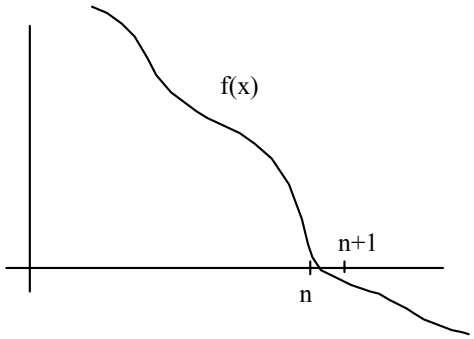
(b) Ohiko *MergeSort*en azaldutako bi parametroen pasatze aukeren artean denbora-eraginkortasun diferentziarik badago?

28. $f(x)$ funtzio monotono beherakorra izanik, izan bedi n osokoa $f(n) \geq 0$ betezen 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;
```

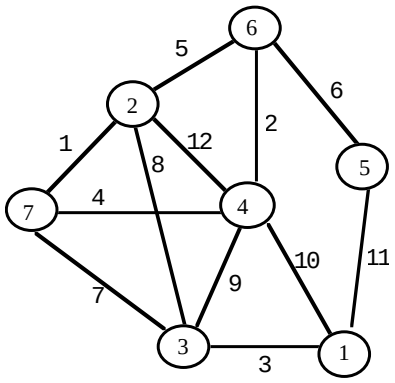


Halere, soluzio horren ordena $O(n)$ da. Portaera asintotiko hori ($n$ren funtziopean) hobetzen duen algoritmoa idaztea eskatzen da.

29. n konputagailuz osaturiko sare internazional bateko X konputagailu bakoitza dobloi bat ordainduz bere I konputagailu auzokidearekin komunika daiteke. Aldi berean, I beste batzuekin komunikatuta dagoenez, X beste konputagailuekin komunika daiteke. Noski, komunikatzen den konputagilu bakoitzari dagokion dobloia ordaindu beharko dio.

    $Prezioa(1..N)$ taula kalkulatuko duen algoritmoa idatz ezazu, non $Prezioa(J)$ gelazkak 1 konputagailua J konputagailuarekin konexioa eduki dezan, lehenengoak ordaindu beharko duen dobloi kopurua jasotzen duen.

30. Ondorengo grafoa emanik:



    (a) Idatzi, hautaketa ordenan, aurreko grafoan $Prim$ algoritmoak hautatuko lituzkeen eta onartuko lituzkeen ertzak.

    (b) Aurreko atalekoa baina $Kruskal$ algoritmoa erabiliz.

31. Zehazki pisu berdina duten bi ertz dituen grafo bat emanik:

    (a) Bi horietako lehenen hautatutako ertza edozein izanik, $Prim$ algoritmoak hedapen zuhaitz minimo bera eraikiko luke?

    (b) Aurreko atalekoa baina $Kruskal$ algoritmoa erabiliz.

32. $G = (ERP, ERT)$ grafo ez-zudendu baten osagai konexuak kalkulatzeko partiketa egitura erabiltzen duen ondorengo algoritmoa ematen da:

```
proz  OsagaiKonexuak(G)
    -- Partiketaren hasieraketa
    for  Dago(a, ERP erpin bakoitzarentzat egin MultzoaEgin(v)) end for;
```

```
    for  Dago((x,y), ERT ertz bakoitzarentzat) egin
        baldin ez(Bilatu(x) = Bilatu(y))
        orduan Bateratu(Etiketa(x),Etiketa(y))
            end baldin;
      end for;
end proz;
```

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

33. G grafo ez-zuzendua emanik $Kruskal$en algoritmoak Gren hedapen zuhaitz minimoa kalkulatzen du $Bilatu-Bateratu$ ($Union-Find$) datu-egitura erabiliz. Metodoak lehendabizi grafoaren ertzez osatutako multzoa haien pisu gorakorrean ordenatzen du, ondoren ertz bakoitza banan bana aztertzeko. $Kruskal$en ondorengo bariante berria aztertzea eskatzen da. Aldaera berriak *meta* ($Heap$) datu-egitura erabiliko du ertzak metatzeko, eta bertatik banan bana hartuko ditu ondoren iturri bertsioan bezala tratatzeko. Algoritmoa idaztea eta haren ordena kasu txarrenean kalkulatzea eskatzen da. Erantzun biak justifikatzea derrigorrezkoa da.

34. Demagun bidai-agentzia baten datubasean munduko aireportu guztien arteko hegaldiei buruzko informazioa gordetzen dela. Aire-lotura guztiak joan eta etortzekoak direla suposatuko dugu. Ondoko problema ebatzi nahi dugu: A eta B aireportuak emanda, zein da Atik Bra hegazkinez joateko geldialdi kopuru minimoa?

    (a) Grafoa korritzeko algoritmo ezagunetatik, zein uste duzu egokiena dela? Zergatik? Aukeratutako algoritmoari egin beharreko aldaketak azal itzazu.

    (b) Ezagutzen dituzun grafoei buruzko algoritmoen artean, problema hau ebazten duten haietatko bakoitzaren denbora-konplexutasuna esan ezazu

    (c) Aurreko bi ataletan aipatutakoen artean, zeinekin gelditu zinateke azkenean? Zergatik?

35. Demagun $Bib(G, v)$ algoritmoa dugula, non G ertz positiboak dituen grafo bat eta bertako v erpin bat emanik v-tik G-ko gainontzeko erpinetaraino dauden distantzia minimoak itzultzen dituen. Egia da $Bib(G, v) = Bib(Kruskal(G), v)$? Gogoratu $Kruskal(G)$ G-ko hedapen zuhaitz minimo bat dela.

36. $Dijkstra$ren algoritmoa (grafo zuzendu bat emanik, erpin batetik beste erpinetaraino doazen bide motzenen kostua kalkulatzen duena) ikusi ondoren, hura aldaraztea eskatzen da **gainera** ondorengoak kalkula eta itzuli ditzan exekuzio denboraren **ordena mantenduz**:

    (a) **Zenbat** bide minimo dauden erpin bakoitzeraino.

    (b) Bide minimoak (hots, bideak) zeintzuk diren.

    Erantzunak arrazoi itzazu.

    Arrastoa: $Dijkstra$-ren benetako algoritmoak bide motzenen kostua kalkulatzeaz at, gainera bide propioak kalkula ditzan (erpinen sekuentziak) aski da bide horiek metatuko dituen $BMD(1..n)$ taula eranstea. $BMD(x)$ gelazkak 1etik Xrainoko bideetan justu X erpinaren aurretik bisitatu behar diren erpinen zerrenda metatuko du. Adibidez, $BMD(x) = [x1, x2, x3]$ bada, 1etetik Xrainoko motzenetan $(x1, x)$, $(x2, x)$ eta $(x3, x)$ bisitatzen diren arkuak dira.

37. G grafo konexuko ertzeetako zein T azpimultzok G-ko erpin guztiak konektatzen dituen eta ertz haien pisuen batura minimoa ematen duen kalkulatzen badakigu. T multzoaren kalkuluak zentzua izaten jarraitzen du oraindik nahiz G-ko zenbait ertzek pisu negatiboa eduki. Hala ere, kasu honetan soluzioak ez du zergatik zuhaitza izan behar. *Kruskal*en algoritmoa egoki bedi pisu negatibodun ertzak dituzten grafoekin ere algoritmoak funtziona dezan.

38. Demagun grafo zuzendu azikliko bat dugula, non haren arku bakoitzak pisu ez-negatibo bat erlazionaturik duen. Helburua, abiapuntu batetik beste erpinetaraino doazen bideen arteko bide luzeenen distantziak zenbatekoa den kalkulatzea da. Zuzena litzateke ondorengo hautaketa jalea erabiltzea algoritmo jale batez problema ebazteko?

    Izan bedi S onartutako erpinen multzoa (hasieran $S = \{Abiapuntua\}$). Pauso bakoitzean bide *berezi* luzeena duen eta $e \notin S$ den erpina onartuko du hautaketa jaleak.

    (Bide bereziaren kontzeptua *Dijkstra*ren algoritmoan ikusitakoa da.)

39. Demagun G grafo azikliko arkuen pisuak 0 edo balio handiagoa dutela.

    *Dijkstra*ren algoritmoan, izan bedi w soluzio S multzotik at dagoen erpin bat S-ri v erpin berri bat erantsi ondoren. Ba al lezake abiapuntu erpinetik w-rainoko bide berezi motzen batek v-tik pasa ondoren S-ko beste erpin batetik pasatzea? Erantzuna matematikoki argudiatu behar da.

40. Z zuhaitza emanik bertako zein K mailak adabegi gehien dituen kalkulatuko duen algoritmoa idaztea eta bere ordena kalkulatzea eskatzen da. Horietako maila bat baino gehiago existituko balira, maila handiena itzuli beharko luke algoritmoak.

41. Ondorengo funtzio errekurtsiboak emanik

$$f(n,m) = \begin{cases} 0 & m = 0 \\ n & m = 1 \\ f(n, \lfloor \frac{m}{2} \rfloor) + f(n, \lceil \frac{m}{2} \rceil) & m > 1 \end{cases}$$

    (a) Bistakoa da $f(n,m)$ kalkulatzen duen programa errekurtsibo inozo batek kalkuluak errepikatuko dituela. Kalkulu berdinak ez errepikatzearren, funtzio hori kalkulatzen duen eta *programazio dinamikoa* teknika jarraitzen duten bi programa idatzi:
    - batak funtzio memoriadunak erabili behar ditu.
    - soluzio iteratiboa izan behar du.

    (b) Aurreko ataleko soluzioen konplexutasuna aztertu. Batuketa kopuruan diferentziarik aurkitzen duzu?

42. $f(1) = 1$ izanik, zein programazio teknika erabiliko zenuke $f(n) = n + \frac{2}{n} \sum_{i=1}^{n-1} f(i)$ kalkulatuko lukeen programa bat idazteko? Programa horren exekuzio ordena zein izango litzatekeen laburki arrazoitu.

43. Izan bedi $A : array(1..n) of\ integer$, non $n > 0$ den. Bestalde, *BatuketaDa* eta *Batura* ondoko bi funtzioak ematen dizkigute:

```
function BatuketaDa(A(1..i)) return boolean is
begin   -- i>0
    if i=1 and  A(1)/=0 then return false;
    elsif A(i)= Batura (A(1..i-1)) then return true;
    else return BatuketaDa(A(1..i-1));
    end if;
end BatuketaDa;
```

```
function Batura (A(1..k)) return integer is
    b:integer:=0;
begin
    for  j  in  1 .. k  loop b:=b+A(j); end loop;
    return b;
end Batura;
```

$BatuketaDa(A(1..n))$ funtzioaren deiak boolear bat itzultzen du: baldin $A(1..n)$ bektoreko osagairen baten balioa bektorean honen aurrekari guztien baturaren berdina bada orduan *true* itzuliko du, eta *false* bestela. Ebaluaketa honen eraginkortasuna aztertu. Beste teknika bat erabiliz, problema bera eraginkorkiago ebatzi. Proposatutako emaitzaren eraginkortasuna aztertu.

44. Erlazio bitar R baten istura transitiboa $\mathcal{R}$ kalkulatuko duen algoritmo idatzi eta aztertu. Erlazio bitarra R emanik, $x\mathcal{R}y \Leftrightarrow xRy \vee (x\mathcal{R}z \wedge z\mathcal{R}y)$

45. $[hi, bi)$ $1 \leq i \leq n$ ekintzak izanik, ekintza-hautaketa problema *programazio dinamikoa* teknika bidez ebaztea eskatzen da. Algoritmoa $Zenb_i$ $i = 1, 2, .., n$ zenbakiaren kalkulu *iteratiboan* oinarrituko da, non $Zenb_i$ zenbaki horrek $1, 2, ..., i$ ekintzen multzoko kardinalitate handieneko eta elkar bateragarri diren ekintzen kopurua metatuko duen. Ekintzak bukaera denbora balioekiko ordena gorakorra jarraituz ordenatuta daudela suposatu: $b_1 \leq b_2 \leq \ldots \leq b_n$

    Soluzio honen eta problema beraren teknika jale bidezko soluzioaren (hain zuzen, lehenen amaitzen den ekintza eta aldez aurretik onartutako ekintzekin teilakatzen ez den hautaketa funtzioa duen soluzio jalea) exekuzio-denbora eta exekuzio-ordena konpara bitez.

46. N objektuk $p_1, ..., p_n$ pisuak eta $b_1, ..., b_n$ balioak erlazionaturik dituzte hurrenez hurren. Bestalde, E edukiera duen motxila bat ere badugu. Helburua, motxilaren edukiera gaindituko ez duten objektuen azpimultzo bat bilatzea da haien balioen batura maximizatuz. (*Backtrack*ing eskema erabiltzen dituzten soluzioak ez ditugu onartuko.)

47. $b_1, b_2, ..., b_n$ n txanponen klaseak emanik eta jakinik txanpona klase bakoitzeko behar hainbat txapona eskuragarri ditugula, K kopuru jakin bat itzultzeko zenbat txanpon konbinazio **desberdin** dauden kalkulatuko duen algoritmo bat idaztea eskatzen da.

    Ad.: Txaponen klaseak honakoak balira: $b_1 = 25$, $b_2 = 50$, $b_3 = 75$ eta $b_4 = 100$

Kopurua	Konbinazio desberdinak
100	5 modu: $25 + 75$, $2 \cdot 50$, 100, $4 \cdot 25$, $50 + 2 \cdot 25$
116	0, ez dago konbinaziorik

    Iradokizuna: $25 + 75$ konbinazioa $75 + 25$ konbinazioaren berdina da, eta ondorioz, konbinazio bakarra kontsideratu behar da.

48. S-ko *azpisekuentzia* bat S-ko edozein posizioetatik edozein osagai kopuru kenduz lortzen da. Adib.: bbacabxxb sekuentziako azpisekuentzia bat acxb da. Bi sekuentzia emanik hauen azpisekuentzia *komun luzeena* mugatzen du ondorengo f funtzio errekurtsiboaren definizioak (kasuka):

    $f(R, \varepsilon) = \varepsilon$

    $f(\varepsilon, S) = \varepsilon$

    $f(a \cdot R, a \cdot S) = a \cdot f(R, S)$

$f(a \cdot R, b \cdot S) = f(R, b \cdot S)$ eta $f(a \cdot R, S)$ azpisekuentzien artetik luzeena $(a \neq b)$

Adib.: $f(belarri, berdela) = bela$, $f(kamamila, motxila) = mila$, $f(bai, ez) = \varepsilon$.

(Iradokizuna: Luzera luzeeneko azpisekuentzia bat baino gehiago existitzen badu, azpisekuentzia bakarra kalkulatzen du.)

$f(R, S)$ sekuentziaren *Luzera* kalkulatuko duen eta *programazio dinamikoa* teknika jarraituz diseinaturiko algoritmoa idaztea eskatzen da (memoriadun funtzioak erabili **gabe**). Zein da algoritmoaren exekuzio-denbora ordena? Eta erabilitako memoria espazio estraren ordena?

49. $G = (Erp, Ark)$ grafo zuzenduko arkuen pisuak negatiboak ez direla dakigu eta $|Erp| = n$. G grafoa L auzokidetza matrize bidez emanik dator. $i, j \in Erp$ multzoko erpin bikote bakoitzaren arteko distantzia minimoa kalkulatu nahi dugu honako formula erabiliz:

$D_{ij}{}^p$: i-tik j-rainoko distantzia gehienez p pauso eman ditzakegunean.

Oinarrizko balioak$D_{ij}{}^p = L_{ij}$dira eta i eta j guztientzat $D_{ij}{}^{n-1}$ balioak bilatu nahi ditugu, jakinez $D_{ij}{}^p$ errekurtsiboki honela defintzen dela:

Problema *programazio dinamikoa* teknika erabiliz ebaztea eskatzen da (memoriadun funtzioak erabili gabe), non goian azaldutako ekuazio errekurtsiboa jaso beharko duen; hots, ondorengo puntu guztiak eskatzen dira beraz:

(a) tarteko emaitza zein datu-egituratan metatuko den mugatzea: bektorea ala matrizea behar den, egituraren posizio bakoitzak zer adierazten edo metatzen duen, egituraren haiseraketa, nola betetzen joango den, soluzioa non dagoen.

(b) emandako definizio errekurtsiboa jarraituz (a)-n aukeratutako datu-egitura betetzen duela ikusi eta soluzioa itzultzen duen algoritmo iteratiboa eman.

(c) algoritmoaren denbora eta espazio estra ordenak kalkulatu: $O(T(n))$ eta $O(MEE(n))$.

50. X eta Y zenbaki positiboak izanik, demagun ondorengo programa:

```
function Nagusia (Znbk: positive) return real is
    Taula: array (integer range 1.. Znbk) of real:=
                                (1=>X; 2=>Y; others=>0);
    function LAG (Znbk: natural) return real is
    begin
        if Taula(Znbk-1)=0
        then Taula(Znbk-1):= LAG(Znbk-1);
        end if;
        if Taula(Znbk-2)=0
        then Taula(Znbk-2):= LAG(Znbk-2);
        end if;
        Taula(Znbk):= (Taula(Znbk-1)+Taula(Znbk-2)) / 2
    end LAG;
begin
    if Znbk<=2 then return Taula(Znbk);
    else return LAG(Znbk);
    end if;
end Nagusia;
```

(a) $Nagusia(n)$k kalkulatzen duen funtzioa definitu.

(b) Aurreko programaren diseinurako, zein teknika erabili da?

(c) $Nagusia(n)$ren egikaritzapen-denbora ordena kalkula ezazu.

(d) Sarrerako datuak $X = 1$ eta $Y = 1$ balira, aurreko programa erabiliko zenuke? Zergatik?

51. Bilaketa dikotomikoaren estrategia erabiliz, emaniko N zenbaki arrunta jarraiko hiru zenbaki arrunten biderkaketa den ala ez mugatuko duen algoritmoa idatzi, soluzioaren denbora-ordena $o(N)$ dela ziurtatuz. Soluzioaren denbora-ordena kalkulatu.

52. $Quicksort(T(1..n))$ algoritmoaren bertsio klasikoa kasu txarrenean $\Theta(n^2)$koa dela ezaguna dugu. Kasu txarrenean $\Theta(n)$koak diren ondoko bi algoritmoak erabiliz, $Quicksort(T(1..n))$ren beste bertsio bat idatz eta azter ezazu, baina soluzio berriak $o(n^2)$koa izan behar duelarik kasu txarrenean:

(a) $Mediana(T(1..n), M)$: $T(1..n)$ osokoen bektorearen mediana M-n kalkulatzen duena, eta

(b) $Banatu(T(1..n), P)$: $T(1..n)$ bektorearen elementuak permutatzen dituena, azkenean ondokoak

bete daitezen:

baldin $T(i) < P$ eta $T(j) = P$ orduan $i < j$

baldin $T(j) = P$ eta $T(k) > P$ orduan $j < k$

53. Arkuen pisuak positiboak dituen grafo zuzendu azikliko bat emanik, *Floyd*en algoritmoak erpin bikote bakoitzen arteko distantzia minimoak kalkulatzen ditu. *Floyd*en algoritmoa eguneratzea eskatzen da, egiten duenaz gain, erpin bikote bakoitzen arteko distantzia minimoa duten bide kopurua kalkula dezan.

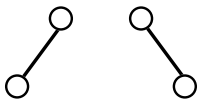
54. Imajina ezazu trenbide baten gainean eskuin-ezkerretara mugi daitekeen robota badaukagula. Trenbidea amaigabea da, eta nonbait haren ondoan pakete bat badago. Robotak paketea aurkitu nahi du, baina ez daki bere eskubian ala bere ezkerrean dagoen. Orduan, gero eta zabalagoak diren mugimendu pendularrak egiten ditu, paketea noizbait topatuko duelakoan. Robotaren portaera deskribatzeko *mugitu*(*tokia*, *norabidea*) prozedura eta *aurkitua*?(*tokia*) funtzio boolearra erabil ditzakegu. Lehenegoaren bidez pauso bateko mugimenduak egin daitezke esandako norabidean, eta robotaren tokia eguneratuz, eta bigarrenak paketea detektatzeko balio du. Ondoko algoritmoak robotaren bilaketa azaltzen du 0 posiziotik abiatuta:

```
noraino := 1    --hurrengo oszilazioaren zabaltasuna adierazten du
norabidea := eskubirantz;
loop
    tokia := 0
    while tokia<noraino loop
            mugitu (tokia, norabidea)
            if aurkitua?(tokia)
            then return (tokia, norabidea)
            end if
      end loop
      norabidea := kontrako(norabidea)
      for i in 1..noraino loop
```

```
            mugitu(tokia, norabidea)
        end loop
        noraino:= 2*noraino;
end loop;
```

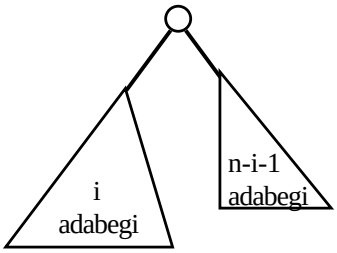
Paketea aurkitzeko *mugitu* eragiketa zenbat aldiz burutuko den analiza ezazu, paketearen posizioaren eta abiapuntuaren arteko distantziapean eman kalkulua.

55. n adabegi dituzten zenbat zuhaitz bitar desberdin osa daiteke? Izan bedi, $Z(n)$ kopuru hori. Badakigu $Z(0) = 1$ dela -zuhaitz hutsa-, $Z(1) = 1$ -erroa soilik duen zuhaitza-, eta $Z(2) = 2$ dela:



$Z(3)$ eta $Z(4)$ kalkulatzeko zuhaitzak marraz ditzakezu:

(a) $Z(n)$ kalkulatzen duen definizio errekurtsiboa idatzi ondorengo diagrama kontuan edukiz:



Orain errekurtsio-ekuazioa zuzena dela $n = 3, 4$ balioentzat ziurta dezakezu zure marrazkiekin.

(b) *Programazio dinamikoa* teknika erabiliz eta zuk aurreko atalean definitutako errekurtsioa jarraituz $Z(n)$ kalkulatzen duen algoritmoa idatzi. Soluzioaren konplexutasuna kalkulatu.

56. $A_1, A_2, ..., A_n$ matrize segida biderkatzean, biderkaketa eskalar kopurua minimizatzen duen algoritmoa gogora ezazu. Algoritmo horrek $Faktor(1..n, 1..n)$ taula eraikitzen du, non $Faktor(i, j) = k$ baldin i-garren eta j-garrenaren artean dauden matrizeen faktorketa optimoa $(A_i...A_k)(A_{k+1}...A_j)$ bada. $Faktor(1..n, 1..n)$taula emanik $A_1A_2...A_n$ biderkaduraren parentesiak jartzeko era optimoa inprimatzen duen algoritmoa idatz eta analiza ezazu.

57. Ondorengo errekurtsio-ekuazioaren denbora-ordena kalkulatu:

$$t(n) = \begin{cases} 1 & n \leq 4 \\ t(\frac{n}{4}) + \sqrt{n} + 1 & n > 4 \end{cases}$$

58. Jakina da $O(n) \in O(n \cdot \lg n) \in O(n \cdot \sqrt{n}) \in O(n^2) \in O(n^3)$. Klasifika ezazu $O(t(n))$ partekotasun kate horretan jakinik:

$$t(n) = \begin{cases} 1 & n \leq 1 \\ 4 \cdot t(\frac{n}{3}) + n & n > 1 \end{cases}$$

Lagungarria gerta dakizuke honakoa: $\log_3 4 = 1.26184$

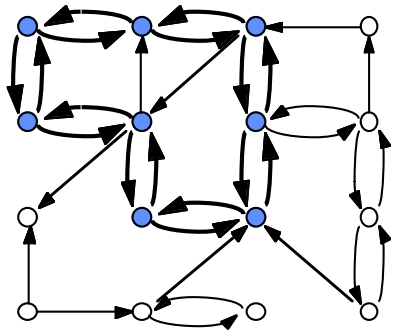
59. n osoko dituen bektore bat eta X osoko bat emanik, algoritmo bat eman mugatzeko existitzen duten ala ez Y eta Z bektoreko bi osoko non $Y + Z = X$ den. Algoritmoaren denbora-ordena $O(n \cdot \lg n)$ izan behar du. Zure algoritmoaren denbora-ordena aztertu eta eskatutakoa dela froga ezazu.

60. $G = (E, A)$ 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.

61. Grafo zuzendu bat emanik, **eraztuna** deritzogu hiru erpin edo gehiago ziklo moduan eta norantza bietan konektatuta dituen seriari. Adibidez, irudiko grafo zuzenduan:



    beltzez markaturik dago grafo honek duen eraztun bakarra. Sakonerako korritzean inspiratuz, diseina ezazu algoritmo eraginkor bat grafo zuzendu bat emanik honek gutxienez eraztun bat duen ala ez kalkulatuko duena. Zure soluzioaren denbora-ordena kalkula ezazu.

62. Gogoratu $Fib(0) = 0$, $Fib(1) = 1$ eta $Fib(n) = Fib(n-1) + Fib(n-2)$ $n \geq 2$ arruntentzat.

    (a) Demagun zenbakien tamaina kontuan izan gabe, bi zenbaki bat egin, ken eta biderka ditzakegula denbora-ordena konstantean,$O(1)$. $O(n)$ denbora-ordena duen eta $Fib(n)$ kalkulatuko duen algoritmo bat idatzi.

    (b) $O(\lg n)$ denbora-ordena duen eta $Fib(n)$ kalkulatuko duen beste algoritmo bat idatzi batuketak eta biderkaketak erabiliz. (Laguntza: Kontsidera ezazu ondorengo matrizea eta haren berredurak:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} fib_0 & 1 \\ fib_1 & 1 \end{pmatrix}$$

    (c) Orain, suposa ezazu β bit dituzten bi zenbaki batzeko $O(\beta)$ denbora-ordena behar dela eta biderkatzeko, aldiz, $O(\beta^2)$.

    Sarreraren tamainatzat zenbakien bit kopurua hartuz eta kontuan edukiz honi dagozkion eragiketa aritmetikoen kostua, lehenengo ataleko zure algoritmoa aztertu. Gogoratu: $Fib(n) = (\frac{1+\sqrt{5}}{2})^n$

63. K hasierako kopuru bat emanik eta honi $\times 2$, $\times 3$, $\times 5$ eragiketak behin eta berriro aplikatuz, zer balio X lor dezakegu M baliotik hurbilen dagoena baina hau gainditzen ez duena?

    Programazio dinamikoa teknika erabil ezazu Xren balioa mugatzeko eta proposatutako algoritmoaren denbora-ordena kalkulatu.

64. Izan bitez A eta B bi zenbaki osoko, m eta n digitu dituztelarik hurrenez hurren.

    (a) Arrazoitu A eta B arteko biderkaketa egiteko erabiltzen duzun ohiko biderkaketa algoritmoak $O(m{\cdot}n)$ *oinarrizko* biderkaketa eta $O(m{\cdot}n)$ oinarrizko batuketa burutzen dituela. (*oinarrizkoa* da bi digitu dezimalen arteko biderkaketa (edo batuketa)).

    (b) Egia esan, $A \times 10$, $\lfloor B/10 \rfloor$ eta $Bmod10$ eragiketak $O(1)$ ordenean egin daitezke (burutzeko aski da zifra dezimal bat gehitzea, kentzea edo aukeratzea hurrenez hurren). Beraz, ez dugu ez biderkaketarik ez eta batuketarik egin behar. Ondorengo biderkatze algoritmoarekin zenbat oinarrizko biderkaketa eta zenbat oinarrizko batuketa egiten diren aztertu:

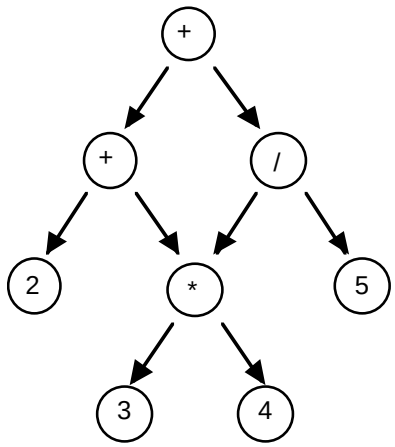
```
funtzioa Biderkaketa (A,B)
    baldin B=0 orduan itzuli 0
    bestela itzuli A*(B mod 10)+ Biderkaketa (Ax10, |B/10|)$
```

    non $+$ eta $*$ aurreko ataleko batuketa eta bideraketa adierazten duten. Ikus $*$, $\times 10$ eta Biderkaketa eragiketa desberdinak direla.

65. Demagun n klase txanpon dugula. i klaseko txanponak ($i = 1..n$ bakoitzarentzat) b_i balioa du eta zehazki klase horretako k_i txanpon kopuru dugu. Idatzi algoritmo bat M balioa lortzeko existitzen duten txanpon-konbinazio desberdinen kopurua kalkulatuko duena. Soluzioaren denbora-ordena kalkulatu.

    Adb.: Bi txanpon klase bagenu honako balio eta kopuruekin: $b_1 = 25$, $k_1 = 10$, $b_2 = 100$, $k_2 = 2$

Balioa	Txanpon-konbinazio desberdinak
250	3 modu: $2 \cdot 100 + 2 \cdot 25$, $1 \cdot 100 + 6 \cdot 25$, $10 \cdot 25$
300	2, modu: $2 * 100 + 4 * 25, 1 * 100 + 8 * 25$

    $25 + 100$ konbinazioa eta $100 + 25$a berdinak lirateke eta horregatik konbinazio bakarra kontsidera beharko zenuke.

66. Espresio aritmetikoa DAGen (grafo zuzendu aziklikoen) bidez adieraz daiteke. Adierazpen hau zuhaitzen bidezkoa baino trinkoagoa da, errepikatutako azpiespresioak behin bakarrik agertzen dira eta. Adibidez, irudiko DAGak $2 + 3 * 4 + (3 * 4)/5$ espresio aritmetikoa adierazten du. Eragile (edo barne adabegi) bakoitzak eragigai (edo arku) bi ditu zehazki. Ezaugarri hauek dituen DAGa ebaluatuko duen algoritmo lineal bat adabegien kopuruan idatzi. Bere denbora-ordena ere kalkula ezazu.