ALGORITMOEN DISEINUA
G1: ALGORITMOEN ANALISIA

1) Balantza bat eta txanpon-multzoa ditugu. Haien arteko bat faltsua da. Pisatze kopuruak mugatu kasu onenean, txarrenean eta batez bestekoan:

   a) 10 txanpon baditugu eta faltsua arinagoa bada besteak baino, zenbat pisatze behar ditugu txanpon faltsua mugatzeko?

   b) *n* txanpon baditugu eta faltsua arinagoa bada besteak baino, zenbat pisatze behar ditugu txanpon faltsua mugatzeko?

   c) *n* txanpon baditugu eta faltsua arinagoa ala astunagoa den ez badakigu, zenbat pisatze behar ditugu txanpon faltsua mugatzeko?

   d) Zein metodo erabiliko genuke 24 txanponen artean faltsua 3 pisatze eginez mugatzeko, jakinez honek gehiago pisatzen duela?

2) Suposa dezagun *Alg1* algoritmoak *Konp1* konputagailuan $10^{-4}2^n$ segundo behar dituela n tamaina duen instantzia bat kalkulatzeko.

   a) zenbat denbora beharko du n=10 bada? n=20 bada? Eta n=30 bada?

   b) urtebetez egikaritzen utziko bagenu, gehienez zer tamainako instantzia egikari lezake?

   c) Demagun 100 aldiz azkarragoa den *Konp2* konputagailua erosteko aukera dugula. Zenbat denbora beharko luke *Alg1* algoritmoak *Konp2*an? Urtebetez egikaritzen utziko bagenu, gehienez zer tamainako instantzia ebatziko luke?

3) Algoritmo batek 1000 osagai ordenatzeko segundo 1 behar du zure etxeko konputagailuan. Kalkula ezazu zenbat denbora beharko duen 10.000 osagai ordenatzeko.

   a) algoritmoa n^2 denboraren proportzionala bada.

   b) algoritmoa $n \times lg\ n$ denboraren proportzionala bada.

4)

   a) $t(n)=n^3$ denbora kostua duen algoritmo batekin N tamaina duten problemak ebatz ditzakegu ordu batean. Makina 1000 aldiz azkarragoa balitz, algoritmo bera ordu betez egikaritzen utziez gero, gehienez zein tamainako soluzioak ebatziko genituzke?

b) $t(n)=2^n$ denbora kostua duen algoritmo batekin N tamaina duten problemak ebatz ditzakegu ordu batean. Makina 1000 aldiz azkarragoa balitz, algoritmo bera ordu betez egikaritzen utziez gero, gehienez zein tamainako soluzioak ebatziko genituzke?

5) *n* osagai dituen taula batean x osagaia bilatu egin nahi dugu. Alderaketa kopuruak mugatu kasu onenean, txarrenean eta batez bestekoan:

a) *x* taulan dago eta hau ez dago ordenaturik.

b) *x* taulan ez egotea gerta liteke eta hura ez dago ordenaturik.

c) *x* taulan dago eta hau ordenatuta dago.

d) *x* taulan ez egotea gerta liteke eta hura ordenatuta dago.

e) *x* taulan dago, hau ordenatuta dago eta bertako osagaiak ezin dira sekuentzialki banan bana aztertu.

f) *x* taulan ez egotea gerta liteke, hura ordenatuta dago eta bertako osagaiak ezin dira sekuentzialki banan bana aztertu.

6) Demagun programa jakin bat exekutatzeko PUZko ordu bete daukagula eta ordu horretan makinak kudea dezakeen programaren sarrerarik handiena **n**=1000 000 dela. Kalkulu Zentroko denbora-berrasignatzea ondoren, PUZko 3 ordu esleitu dizkigute. Zein da makinak denbora horretan kudea dezakeen programa beraren sarrerako tamainarik handiena T(*n*) programaren konplexutasuna honakoa bada (k_i konstante batentzat)?

a) $k_1 \times n$.

b) $k_2 \times n^2$.

c) $k_3 \times 10^n$.

7) Eman dezagun gauero programa jakin bat exekutatzeko PUZko ordu bete daukagula eta ordu horretan makinak kudea dezakeen programaren sarrerarik handiena *n*=1.000.000 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 sarrerako tamainarik handiena T(*n*) programaren konplexutasuna honakoa bada (k_i konstante ba- tentzat)?

a) $k_1 \times n$.

b) $k_2 \times n^2$.

c) $k_3 \times 10^n$.

8) Jarraiko espresioak egiazkoak ala faltsuak diren arrazoituz adierazi:

a) $2^{n+1} \in O(2^n)$

b) $2^{2n} \in O(2^n)$

c) $n^3 + 2n^2 \in O(n^2)$

d) $3n \in O(2n)$

e) $(n+1)! \in O(n!)$

f) $\log_a n! \in \Theta(n \log_a n)$

9) Ondorengo funtzioak goranzko ordenan zerrenda itzazu. Ordena berdinekoak direnak nolabait hala direla azaldu:

$n^2 + \log n$	$\log (\log n)$	$n^{1+\varepsilon}$ eta $0 < \varepsilon < 1$
2^{n-1}	$\sqrt{n}$	$\ln n$
e^n	$\log_3 n$	$n \log n$
$(\log n)^2$	$\log n^2$	$\frac{n^2}{\log n}$

10) Ondorengo sententzia bakoitzeko froga ezazu egiazkoa den ala ez, bigarrenerako kontrako adibidea emanez.

a) $2^{3n+1} \in O(2^n)$

b) $5 \lg_{10} n \in \Theta(\lg_2 n)$

c) $O(\lg f(n)) = O(\lg g(n)) \Rightarrow O(f(n)) = O(g(n))$

11) Ordena gorakorra jarraituz ordena itzazu ondorengo konplexutasun borneak (hazkuntza tasak), berdinak direnak horrela direla adieraziz: $O(\log n)$, $O(n^n)$, $O(2n^2)$, $O(n!)$, $O(3\ 2^n)$, $O(n \log \log n)$, $O(20)$, $O(n \log n)$, $O(2 \lg n)$, $O(n\ (n+\log n))$, $O(1)$, $O(3^n)$, $O(n\ 2^n)$, $O(n+1)$, $O(n^{n+1})$. Begi-bistakoak ez diren kasuak laburki arrazoi itzazu

12) Ondorengo baieztapenak egiazkoak ala faltsuak diren arrazoi itzazu. Adibidez, "$n^2 \in O(n^3)$" eta "$n^2 \in \Theta(n^3)$" baieztapenei, lehenengoari egiazkoa dela erantzun behar zaio eta faltsua, aldiz, bigarrengoari.

a) $n/\lg n \in \Theta(n)$ eta $n/\lg n \in o(n)$

b) $\lg \lg n \in o(\lg n)$ eta $\lg \lg n \in \omega(\lg n)$

c) $4^{\lg^2 n} \in O(n^2)$ eta $4^{\lg^2 n} \in \Omega(n^2)$

d) $(\lg n)^2 \in o(\sqrt{n})$ eta $\sqrt{n} \in O((\lg n)^2)$

13) Ondorengo errekurtsio ekuazioaren ordena zehatza kalkula ezazu:

t(n)= 8 t(n/2)+ n if n>1;

t(0)= t(1)= 1

Ondorengo multzoren bateko partaidea bada, adieraz ezazu zenena den: $\Theta(n)$, $\Theta(n \lg n)$, $\Theta(n^2)$, $\Theta(n^2 \lg n)$, $\Theta(n^3)$, $\Theta(n^3 \lg n)$, $\Theta(n^4)$.

14) Ondorengo programa zatien T(*n*) eta ordena (ahal baduzu zehatza) kalkulatu.

(a)
```
for IND in LISTEN_FREKUENTZIA'RANGE loop
                        -- Lista 0-z hasieratu
   LISTEN_FREKUENTZIA(IND) := 0;
end loop;
```

(b)
```
IND := 1;              -- Handienaren posizioa aurkitu
for ZENB  in 2..LISTA'LAST loop
   if    LISTA(ZENB)>LISTA(IND)  then IND:= ZENB; end if;
end loop;
```

(c)
```
AURKITUA:= false; IND:= 0; --  Taula  batean  bilaketa  lineala
while IND<LISTA'LENGTH and not AURKITUA loop
IND:= IND +1;
if  ELEMENTUA=LISTA(IND) then AURKITUA := true; end if;
end loop;
```

15) Azter ezazu ondorengo algoritmoa:

```
function Foo (n: in Integer) return Integer is
  X, I:Integer;
begin
  if n<= 1
  then return 1;
  else for I in 1..n loop
         X:= 1;
         while X<n loop
           X:= X*2;
         end loop;
       end loop;
       return (Foo(n/2) + Foo(n/2));
end;
```

16) V(I) ≤ N (*1*≤*I*≤*N*) betetzen duen V(1..N) taula eta X zero ez den digitua emanda, ondoan ematen den algoritmoaren konplexutasuna kalkulatu. Algoritmo honek K aldagaian V taularen elementu positibo guztien X digituaren agerpen kopurua bueltatzen du.

```
for I in V'RANGE loop
   if V(I)>0
   then B:= V(I); K:= 0;
        while B> 0 loop
              if (B rem 10=X) then K:= K + 1; end if;
              B:= B/10;
        end loop;
        Idatzi(V(I),"-an: ", K," aldiz ");
   end if;
end loop;
```

17) BubbleSort, SelectionSort eta InsertionSort algoritmoetan zenbat osagaien mugimendu egiten da kasu txarrenean? Eta zenbat osagai arteko alderaketa? Emaitzak alderatu eta ondorioak atera.

18) BubbleSort eta SelectionSort algoritmoak (eman dizkizuegun metodoak) ez dira sarrerarekiko sentikorrak. InsertionSorta, ordea, bai. Honek, batez bestean egiten dituen alderaketa kopurua kalkula ezazu? Ordena (goi-ordena edo zehatza) hobetzen du batez besteak?

19) Ondorengo programa errekurtsiboen T(*n*)-a eta ordena (O edo Θ) kalkula bedi

```
function FAKTORIALA (N: in INTEGER) return INTEGER is
begin
     if  N=0 then return 1;
     else return     N * FAKTORIALA(N-1); end if;
end FAKTORIALA;


procedure HANOI (N: in INTEGER; I,J:  in Datu123 ) is
-- N diskoen kopurua adierazten du.
begin
     if  N> 0
     then      HANOI (N-1, I, 6-I-J);
               Idatzi(I,"  -> ",J);
               HANOI (N-1, 6-I-J,J);
     end if;
end HANOI ;


function FIBONACCI (N: in INTEGER) return INTEGER is
begin
     if  N=1   then return N;
     else return     FIBONACCI(N-1) + FIBONACCI(N-2); end if;
end FIBONACCI;
```

20) Waste(n) prozedura deiak idazten dituen lerro kopurua jaso beza T(n)-k. T(**n**)-ren ordena zehatza kalkula ezazu

```
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;
```

21) 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 *Pal(Hit, 1, n)* deiak behar duen denbora eta denbora-ordena kasu txarrenean.

22) 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)
```

23) `a` bektorea [0..n,0..n] osokoen bektorea izanik, ondorengo eskema azter ezazu. Identifika itzazu sarrera onena (azkarrena) eta txarrena (motelena).

```
I = 1;  J = 1;  kont=0
while I+J ≤ n+1
   if  (a[I-1,J-1] > a[I,J]) { kont++;  I++; }
   else J++
   endif
endwhile
```

24) *n* sarreraren tamaina 3-ren anizkoitza dela suposatuz eta konbinatu funtzioaren ordena konstantea, ondorengo algoritmoaren exekuzio denbora eta ordena kalkula itzazu:

```
programa (a:datuak; n: tamaina): balioa
baldin   n≤3   orduan   zuzenean(a,n)
bestela   m1= n/3
```

```
            m2= n/32
            s1 = programa (a,m1)
            s2 = programa (a,m2)
            itzuli(konbinatu (s1,s2))
endbaldin
```

25) `tratatu` algoritmoa azter ezazu. Sarrera onena (azkarrena) eta txarrena (motelena) zeintzuk diren identifika itzazu

```
kopiatu(a:array[1..n] of integer; i,j:1,…,n)
for k=i+1 to j
    a[k]=a[i]
endfor
end kopiatu

tratatu (a:array[1..n] of integer)
for i=1 to n
   j=i+1
   while  j≤n  and  a[i]≤a[j]
      j++
   endwhile
   kopiatu(a,i,j-1)
endfor
end tratatu
```

26) Honako errekurtsio ekuazioa legokiokeen algoritmo bat idatz ezazu:

$t(n)= 2\ t(n/2) + n \lg n$
$t(0)= t(1)= O(1)$

27) T(n)-ak emanik haien ordenak kalkula bitez:

(a) $T(n)= 2T(n-1)+n+2^n$ $n\geq 1$ $\Theta(n2^n)$
$T(0)= 0$

(b) $T(n)= 2T(n-1)+(n+5)+3^n$ $n\geq 2$ $\Theta(3^n)$
$T(1)= 1$
$T(0)= 0$

(c) $T(n)= 2T(n-1)+3^n$ $n\geq 2$ $\Theta(3^n)$
$T(1)= 1$
$T(0)= 0$

(d) $T(n)= 3T(n-1)+(n+5)+3^n$ $n\geq 1$ $\Theta(n3^n)$
$T(0)= 0$

(c) $T(n)= 3T(n-1)+3^n$ $n\geq 1$ $\Theta(n3^n)$
$T(0)= 0$

28) Jakina da $O(n) \subseteq O(n\lg n) \subseteq O(n\sqrt{n}) \subseteq O(n^2) \subseteq O(n^3)$. Klasifika ezazu O(t(n)) partekotasun kate horretan jakinik: (Lagungarria gerta dakizuke honakoa: $\log_3 4 = 1.26184$)

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

29) Errekurtsio ekuazioen denbora-ordenak kalkulatu. Hauek ahalik eta sinpleenak eta borne hurbilenak izan behar dute:

a) $t(n) = \begin{cases} t\left(\frac{n}{2}\right) + \mathrm{b} & n > 2 \\ \mathrm{a} & n \leq 2 \end{cases}$ $\Theta(\lg n)$

b) $t(n) = \begin{cases} t\left(\frac{n}{2}\right) + n & n > 2 \\ \mathrm{a} & n \leq 2 \end{cases}$ $\Theta(n)$

c) $t(n) = \begin{cases} 2t\left(\frac{n}{2}\right) + 2n & n > 2 \\ \mathrm{a} & n \leq 2 \end{cases}$ $\Theta(n \lg n)$

d) $t(n) = \begin{cases} t\left(\frac{n}{2}\right) + \lg n & n > 2 \\ \mathrm{a} & n \leq 2 \end{cases}$ $\Theta((\lg n)^2)$

e) $t(n) = \begin{cases} 4t\left(\frac{n}{2}\right) + n & n > 2 \\ \mathrm{a} & n \leq 2 \end{cases}$ $\Theta(n^2)$

f) $t(n) = \begin{cases} 4t\left(\frac{n}{3}\right) + 3n - 5 & n > 3 \\ 1 & n \leq 3 \end{cases}$ $\Theta(n^{1.26})$

g) $t(n) = \begin{cases} 2t\left(\frac{n}{4}\right) + \lg n & n > 4 \\ 1 & n \leq 4 \end{cases}$ $\Theta(\sqrt{n})$

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

i) $t(n) = \begin{cases} 8t\left(\frac{n}{2}\right) + n & n > 2 \\ 1 & n \leq 2 \end{cases}$ $\Theta(n^3)$

j) $t(n) = \begin{cases} 2t\left(\frac{n}{4}\right) + \sqrt{n} & n > 4 \\ 1 & n \leq 4 \end{cases}$ $\Theta(\sqrt{n} \lg n)$

k) $t(n) = \begin{cases} 2t\left(\frac{n}{2}\right) + \sqrt{n} & n > 2 \\ 1 & n \leq 2 \end{cases}$ $\Theta(n)$

30) 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, 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 Adan testa bakarrik J≠M+1 baliozkoa denean kalkulatzen dela.

31) `Mokokatzen` izeneko algoritmoak V[a...b] bektoreko hainbat balioen batura kalkulatzen du. Algoritmoaren exekuzio ordena kalkula ezazu:

```
funtzioa Mokokatzen (V,a,b) return zenbakia
X ← ⌊(a+b)⌋/2
if (b-a+1)≤3 then return V(X)
else Y ← ⌊(2a+b)⌋/3
     Z ← ⌈(a+2b)⌉ /3
     return V(X) + Mokokatzen (V,a,Y) + Mokokatzen (V,Z,b)
```

Idazkera asintotikoek definitutako o(n), $O(\sqrt{n})$, $\Omega(\sqrt{n})$, $\Omega(2^n)$ edo Ω(n lg n) multzoetako zein multzokoa da lortu duzun kostu funtzioa?

32) Izan bedi P(*n*) n mailako polinomioa: $P(x) = a_nX^n+a_{n-1}X^{n-1}+ \ldots +a_1X^1+a_0$ Bere koefizienteak *Koef* izeneko taula batean metatu dira, ondorengo moduan: *Koef*(i) = a_i X aldagaiaren balioa emanik, polinomioaren balioa kalkulatuko duen programa definitu nahi dugu:

```
Koef: array (0..N) of Zenbakia := HASIERATU_K;
X: Zenbakia:= HASIERATU_X;
function POLIEBAL (M: 0..N) return Zenbakia is
begin
     if M=0 then return Koef(0);
     else      Ber:= 1;
               for Ind in 1..M loop Ber:=Ber * X; end loop;
               return (Koef(M) * Ber)+ POLIEBAL (M-1);
     end if;
end POLIEBAL;
```

a) Jaso beza T(n)-k POLI EBAL(n) deiak egiten dituen eragiketa aritmetiko (+,−,∗,/) kopurua. Kalkula ezazu T(n) exekuzio-denbora.

b) Aurreko ataleko kopurua errez hobe daiteke. Honela, eta ** berreketaren eragile estandarra erabili gabe, emaitza berdina ekoiztuko duen eta T(n) hobea duen algoritmoa idatzi eta haren exekuzio-denbora ekuazio berria txikiagoa dela froga ezazu.

33) Ondorengo funtzio errekurtsiboak X arruntaren M. berredura kalkulatzen du:

```
X: Oinarria := HASIERATU_X;
function BER (X,M: Arrunta) return Arrunta is
begin
     if M=0 then return 1;
     elsif M=1 then return X;
     elsif M mod 2 = 0
        then return BER(X, M/2) * BER(X,M/2);
        else return X* BER(X,M/2) * BER(X,M/2); end if;
end BER;
```

a) Algoritmoak egikaritzen duen biderketa kopuru konkretua kalkulatu, bai eta ordena zehatza ere.

b) Aurreko ataleko algoritmoak kalkulu berdinak errepikatzen dituela ikusiz, balio bera kalkulatzen duen algoritmo errekurtsibo berria idaztea eskatzen da, baina berreketa (**) eragiketa estandarra erabili gabe. Bertsio berriak biderketa kopuru gutxiago egin beharko du. Algoritmo berriaren ordena hobea izan behar du, eta hala dela frogatu.