Algoritma: Bir problemin çözülmesi için izlenen yol, yöntem

BBG 1 50

İkili → Binary

tabanlar

10 (decimal) → 0 – 9

16 (Hex) → 0 – 9 A – F (10 → A, 15 → F)

2 (Binary) ← 0 – 1

8 (Octal) → 0 – 7

Onluk → İkilik

$(45)_{10} = (\ ?\ )_2$

$(45)_{10} = (101101)_2$ → bit, bit

En yüksek anlamlı

32 → 1 $\quad 45 - 32 = 13$

16 → 0

8 → 1 $\quad 13 - 8 = 5$

4 → 1 $\quad 5 - 4 = 1$

2 → 0

1 → 1

En düşük anlamlı

İkilik → Onluk

$(101101)_2 = (?)_{10}$ → 00101101 → 1 byte

$(101101)_2 = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 + 0*2^4 + 1*2^5 = 1+4+8+32 = (45)_{10}$

Adreslenebilir en küçük bilgi 8 bit → 1 byte

bit → 0 veya 1 değeri alabilen en küçük bilgi.

Adresleme 8'in katı olmak zorunda. 8 bit → 16 bit → 32 bit → 64 bit

$(11111111)_2$

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 → 255

EYA — EDA

7	6	5	4	3	2	1	0

−128 → 127

İşaret biti (sign bit)

0: +

1: −

7	6	5	4	3	2	1	0
0	0	1	0	1	1	0	1

+ boşluk kalmadı

128

$\boxed{10000000}$

01111111 → 1'e tamleyen (1's complement)

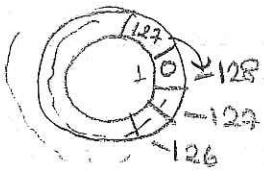
1

$\boxed{10000000}$ → 2'ye tamleyen (2's complement)

İşaret bitti

$\Downarrow$

-128



$$\begin{array}{r} 10000000 \\ 01111111 \\ + \quad\quad 1 \\ \hline 10000000 \end{array} \Rightarrow -128$$

$\underline{\underline{-22}}$

$+22$ 00010110

11101001 1'e tamleyen

$+ \quad\quad 1$

$\boxed{1}$1101010 2'ye tamleyen

İşaret bitti

$$\begin{array}{r} 11101010 \\ 00010101 \\ + \quad\quad 1 \\ \hline (00010110)_2 \rightarrow (22)_{10} \end{array}$$

$\boxed{-22}$

← nk 1'e kadar aynı sonra tamleyen

$\Leftarrow$

(00010110)

$\Downarrow$

11101010

$$\begin{array}{r} 10000000 \\ 10000000 \end{array}$$

$(01111111)_2 = (127)_{10}$

$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$

10000001

↓

İşaret bitti

Floating Point Binary

$2.5 * 10^{-3}$

mantissa, exponent, sign

$-3.154 * 10^5$

$3 * 10^0 \quad 1 * 10^{-1} + 5 * 10^{-2} + 4 * 10^{-3}$

$(\overset{\text{mantissa}}{\boxed{11.1011}})_2 * 2^3 = (3.6875 * 8)_{10} = (29.5)_{10}$

3 $\quad 1 * \frac{1}{2} + 0 * \frac{1}{4} + 1 * \frac{1}{8} + 1 * \frac{1}{16}$

$$\frac{8+2+1}{16} = \frac{11}{16}$$

İkili	Onluk Gösterim		Onluk Değer
.1	1/2	→	.5
.01	1/4	→	.25
.001	1/8	→	.125
.0001	1/16	→	.0625

.101
↓
.5+.125

Onluk → İkilik Dönüşümü

$(2.625)_{10} = (__)_2$ $\qquad$ $(2.625)_{10} = (10.101)_2$

↓

$0.625 * 2 = 1.25 \quad 1$

$0.25 * 2 = 0.5 \quad 0$

$0.5 * 2 = \underline{1.0} \quad 1$

$(0.40625)_{10} = (\ldots)_2$

$0.40625 * 2 = 0.8125 \rightarrow 0$

$0.8125 * 2 = 1.625 \rightarrow 1$

$0.625 * 2 = 1.25 \rightarrow 1$

$0.25 * 2 = 0.5 \rightarrow 0$

$0.5 * 2 = 1.0 \rightarrow 1$

$(0.01101)_2$

$(-4.75)_{10} = (__)_2$

$0.75 * 2 = 1.5 \rightarrow 1$

$0.5 * 2 = 1 \rightarrow 1$

$(-100.11)_2$

$(1.7)_{10} = (__)_2$

$0.7 * 2 = 1.4 \rightarrow 1$

$0.4 * 2 = 0.8 \rightarrow 0$ $\qquad$ $(1.1\overline{0110})_2$

$0.8 * 2 = 1.6 \rightarrow 1$

$0.6 * 2 = 1.2 \rightarrow 1$

$0.2 * 2 = 0.4 \rightarrow 0$

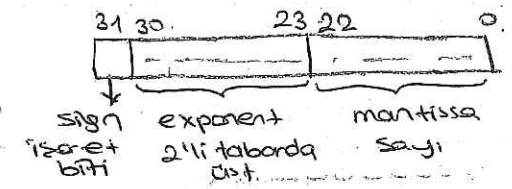
$0.4 * 2 = \overline{0.8}$ Devir

IEEE 754 standard

single precision: 32 bit (short real)

double precision: 64 bit (long real)

32 bit :



Ondalık düzendeki bir sayının İkilik düzende (yaklaşık olarak) gösterimi

1) Sayı İkili düzene çevrilir.

2) Noktanın solunda tek bit kalacak şekilde (tek bit ve bu bitin değeri 1 olacak şekilde) sayı normalize edilir.

3) Noktadan sonrası mantissa bölümüne yazılır. Üste (exponente) offset eklenerek elde edilen sayı İkili düzene çevrilir ve exponent bölümüne yazılır.

belli sınıta ilişkili olan

İşaret bitti yazılır.

32 bit kesirli sayılarda offset değeri 127'dir.

$(3.14)_{10} = (---)_2$ (11.0010001111010111...)

11.

0.14 * 2 = 0.28 0
0.28 * 2 = 0.56 0
0.56 * 2 = 1.12 1
0.12 * 2 = 0,24 0
0,24 * 2 = 0,48 0
0.48 * 2 = 0,96 0
0.96 * 2 = 1,92 1
0.92 * 2 = 1,84 1
0.84 * 2 = 1,68 1
0.68 * 2 = 1,36 1
0.36 * 2 = 0,72 0
0.72 * 2 = 1,44 1
0.44 * 2 = 0,88 0
0.88 * 2 = 1,76 1
0.76 * 2 = 1,52 1
0.52 * 2 = 1,04 1
0.04

36 / 2 = 18 2
32 / 2 = 184
84 / 2 = 176
68 / 2 = 136 1
86 / 2 = 176 1
76 / 2 = 152

11.0010001111010111

1	8	23
sign bit	exponent	mantissa

kaise: 127

1) $-0.00101 = -1.01 * 2^{-3}$

exponent $= 127 - 3 = 124$

23 bit

1	01111100	01000000000...0
sign	exponent	mantissa

64
32
16
8
4

floating pointte
eşitlik
sağlamayabilir
(duyarlılıkla ilgili)

$1.01.010 \rightarrow 1.01010 * 2^2$

$127+2=129$

$(1\ 10000010\ 11110110000000000...0)_2 = (?)_{10}$

$-1.1111011 * 2^3$

$1.\underbrace{111101}$

$128+2=130$

$(1111,1011)_2 = 15,6875$

$\frac{1}{2} + \frac{1}{8} + \frac{1}{16}$

(8) (2)

$\frac{11}{16}$

floating pointte

$(1e100 - 1e100) + 1.0 \cong 1.0$

$(1e100 + \underbrace{1.0}_{\text{sayısının 16 bir.}}) - 1e100 \cong 0.0$

$(x+y)+z \neq x+(y+z)$

$(x*y)*z \neq x*(y*z)$

0.1 kiu bir zaman tam gösterilemez

$0.1 * 2 = 0.2$

$0.2 * 2 = 0.4$

$0.1 \rightarrow 0.4 * 2 = 0.8$

$0.8 * 2 = 1.6$

$0.6 * 2 = 1.2$

$\rightarrow 0.2$

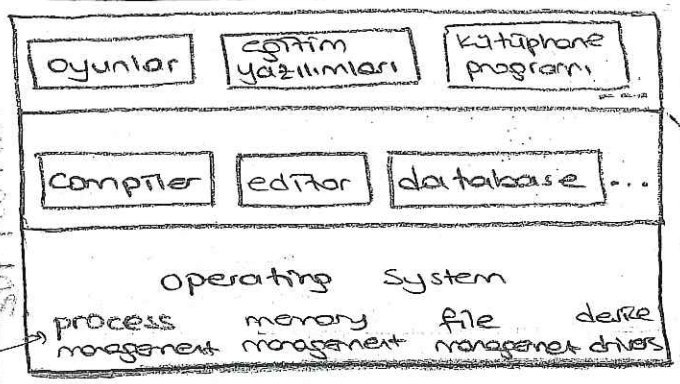
0-9

A-F

$23F \rightarrow 2 * 16^2 + 3 * 16^1 + 15 * 16^0$

Bilgisayar Sistemi

2 parçadan oluşur. software-hardware



compiler yazdığınız programların bilg.de çalışabilir hale getirmesini sağlar.

database: belli bir hiyerarşiye göre bilgilerin saklanabilmesi, gerektiğinde ortaya çıkıp kullanılabilmesini sağlar. (sistem yazılımı)

son kullanıcının paket prog. kullanımını sağlayan

İşletim sist.: programların donanım ile birlikte kullanılabilmesini sağlayan

process management: sıra belli edilmede, getiriş yapılarak d'da farklı fakat sanki aynı anda yapılıyormuş gibi görüntü yaratır.

(mouse-beğlam, etc) aslında aynı anda değil. çünkü durumu daha

memory management: sistemde iş yaptığınız bir belleğ var.

yapılan İşler orada tutulur. kullanımı kontrol eder.

yerini boş, orada tut. kullanmağı unutulan fiziksel belleğin bölgelerini boşaltıyor, arka arkaya 1 mb yokken 1mb'lık aynı aynı yer toplanmamıştır olduğunda yer yaratıp.

eskiden böyle değildi sanal virtual memory.

1mb'lık yer alır bile 10 mb'lık yer varmış gibi gösterir.

belleğin düzgün bir şekilde saklanabilmesi

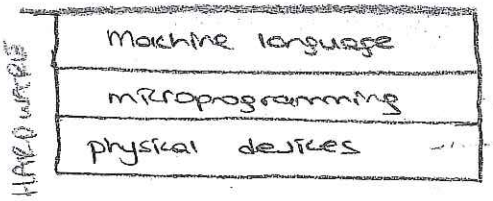
file management: dosyaların yapılara göre uygun şekilde saklanabilmesi ve getirilebilmesi. directory - ağaç yapısı

device drivers: hardwareler printer, mouse, video kart

bunların bilg. donanımıyla bağlantısını ve kullanılabilmesini sağlayan programlar.

printer kullanılabilmesi, bilgiden bilgi gönderme, haberleşebilmesi

donanım - yazılım operating system

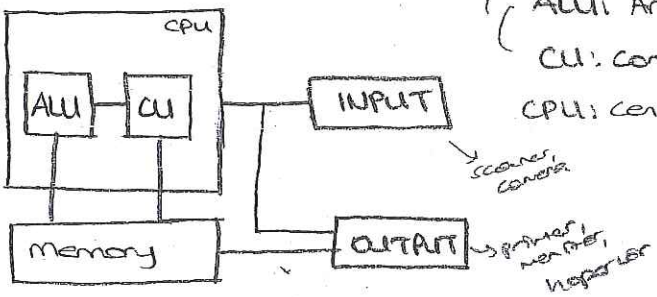


machine language: makine diline çevrilmeli. O ve 1'lerden oluşan komutlar. makine dili doğrudan makineyi (CPU) birim ingilizce anlatabilen komutlar (mesela doğrudan bilgisayarın çalışması)

microprogramming: CPU'yu ilgilendiren mikro komutlarla olur. bu komutların yaptırılmasını sağlayan. (sistemin) registera ulaşılması, 2'ye tümleyen alınması mikrokodlarla mikroprogram.

physical devices: program donanımı oluşturan parçalar.

von-Neumann Mimarisi



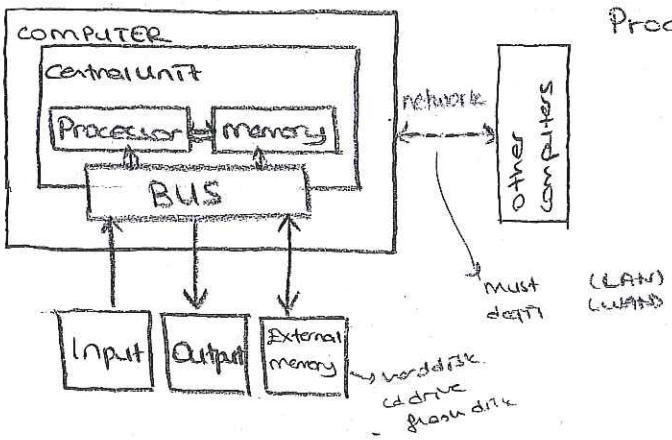
bütün hesaplamalar (kullanılan fonksiyonlar sonuçlar bilgi elde edilmesi)

(matematiksel ve mantıksal işlemler, bilgi karşılaştırmaları)

ALU: Arithmetic-Logic UNIT

CU: Control UNIT (bilgi akışını kontrol ve idare)

CPU: Central Processing UNIT



Processor: CPU ⇒ CU + ALU

Data Bus - Address Bus

BUS: cihazlar, birimler arasında bilgileri taşımayı sağlar, haberleşebilmeye sağlar.

Bilgi Saklama

2 tür bellek

Volatile : main memory **RAM** Random Access memory

non-Volatile : external (auxiliary)

RAM, sequential

volatile (uçucu)

bilgin kapandığı anda diğer bilgileri sağlayan geçici bellek (RAM)

RAMin üzerindeki gibi. (1-4 GB) sunucuda (10 GB)

bilg. açık olduğu sürece geçerli. olan bilgiler

non-volatile

bilg. kapansa bile pitoyer bilgilerin tutulduğu (external derler)

CD-DVD, memory stick, teyp, floppy disket

erişim 2 türlü

volatile'da **RAM** Random Access memory
dağınık erişilebilir.
CD, DVD

non-volatile RAM, sequential
↳ small erişim
teyp, kaset
erişim tüm sırmak gerekir
önce sırayla istenen istenilene erişmek zor

Saklanabilir bilgi

10^{-12} : pico
10^{-9} : nano
10^{-6} : micro
10^{-3} : milli
10^{3} : kilo
10^{6} : mega
10^{9} : giga
10^{12} : tera

en küçük birim byte (8 bit)

1024 byte

ALGORİTMA Bilgisayarda kesinleşen sonuçlar çıkarmak amaçlı geliştirilen yöntem
↑
El-Harezmi

Bilg'de tasarım, problem çözme gibi

1st generation → makine dili

2nd generation → Assembly diller → Low Level Prog. Lang.

makinenin anlamına bağlı olan diller LBDlle makinaya özgü Assembly diller

konutlar daha anlaşılır şekilde

mov ax,bx add bx,5	assembler → makine dili

assembler aracılığıyla makinada çalışabilir hale gelir

makine dilinde yazılmışsan bir şekilde ing.'in anlaşılır şekle getirilmesi

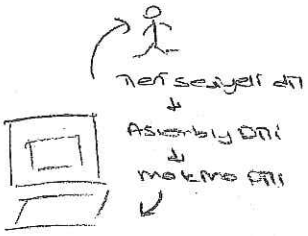
3rd generation → High Level Programming Languages

Pascal, C, Fortran

↓ middle-level

→ sitem prog. yazılımı kullanılmaz yazılımlı

READ A



Donanımlar geliştikçe, yapılması gereken işler çoğaldıkça, ihtiyaçlar arttıkça yeni diller geliştiriliyor.

4th generation

SQL
query language

bakkallık, kasiyerlik, muhasebe, öğrenci işleri otomasyonu gibi yazılımlar istendiğinde dilleri kullanmak daha iyi.

SELECT MUSTERI FROM MUSTERILISTE WHERE

MUSTERINO = 3157

5th generation → Prolog, Lisp gibi (yapay zeka dilleri)
→ Görsel Diller (Visual C, C++, Visual Basic)

Doğal diller, konuşma dili gibi ama Prolog, Lisp gibi yapay zeka amaçlı kullanılan diller (doğal dil işlemek, insan dili gibi konuşma gibi olan diller) yapay zeka problemlerini çözmek. ex: uzmandil (mantıksal sorular, konuşma dili vs.).

Görsel diller. Visual C, C#, Visual Basic

yapılan ekranda görünüyor. (görsel bir bilgi var)

ekran tasarımı var. mouse-enter) boşluk doldurma ile

1'de fatura bilgi bir ordde, grafik bir tasarım lambası ve sayfaları

Object Oriented Programming Lang.

C++
→ Java
C#

nesneye dayalı prog. dili = yapılması gereken işler obje hesabına ve objeler kullanılıyor.

Compiler = Ter sevye dili, makine diline çevirmek için
(Derleyici)

Hedef 1) Yazdığınız programın hata analizini yapmak

1) İmla hataları (Syntax Analysis)

for i:=1 to 10 do
↓
i:=1

i:=5 (j) mutlaka fırlar.

2) Semantik analiz (Semantic Analysis)
mantıksal hataları var mı bunlara bakmak

~~a+b:=c;~~

c:=a+b;

← program code'a çevrilir, execute hale gelir.

2) Ara işlemler yapıyor. Çalıştırılır kod haline programınızı çevirir.

1. Akış Diyagramı + Analiz
(Flow Chart)

2. Yarı kodlama
(Pseudo code)

1. Giriş: İki sayı
2. İşlem: verilen iki sayı arasında toplama işlemi
3. Çıkış: İki sayının toplamı

yazdığım değişkene atanan sabit değer

→ variable
değişken

değişik değerler tutmak çalışabilmemize

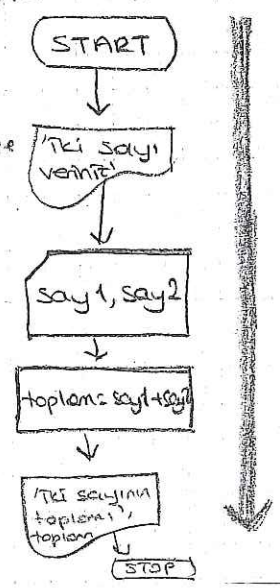
1) harfle başlamalı
ilk harfe sayı ve underscore olabilir.

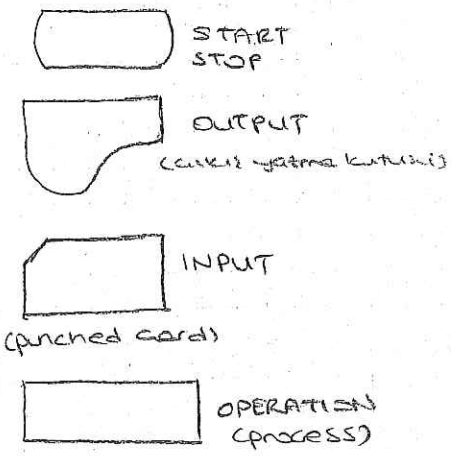
2) anlamlı isim

desAd
kelimeler

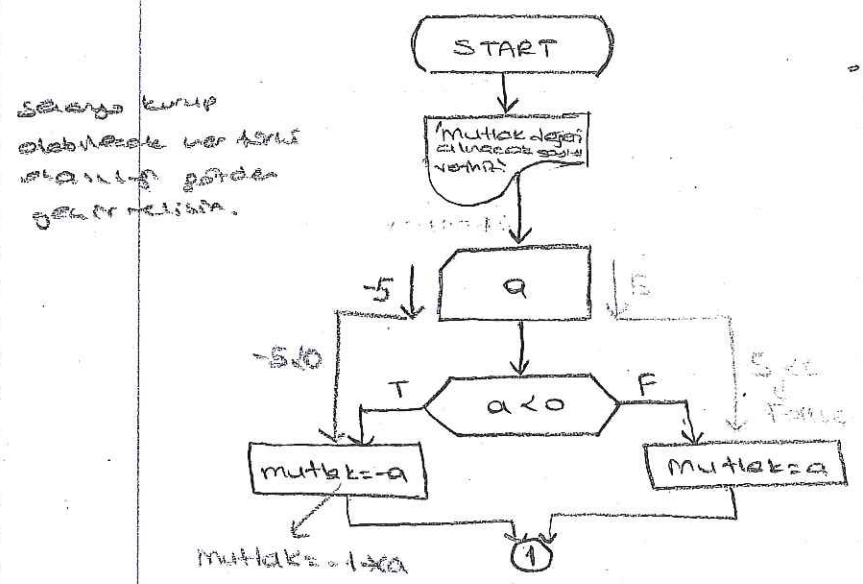
3) Türkçe karakter kullanma

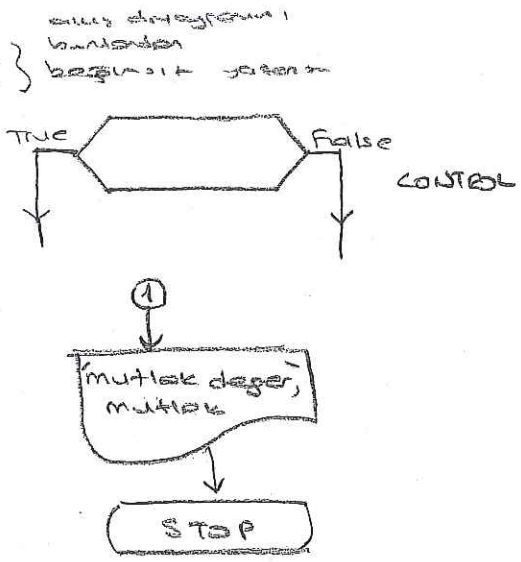
say1
say2
toplam





1. Giriş: Sayı
2. İşlem: Sayı 0'dan küçükse pozitif yapılmalı
3. Çıkış: Sayının mutlak değeri





Analiz

a=5
5<0 → F → mutlak=5

a=-5
-5<0 → T → mutlak=5

a=0
0<0 → F → mutlak=0

ÖDEV 1) İki sayı veriliyor, küçük olan sayının değerini ekrana yazdıran algoritmayı tasarlayınız.

2) a, b, c katsayıları verilen 2. dereceden $ax^2+bx+c=0$ gibi bir denklemin kökleri gerçel olduğu biliniyor. Bu kökleri bulan algoritmayı tasarlayınız. x1, x2 nedir.

LAB

Tüm sertifika ilk 2 hafta eğitim
sonraki 4 hafta sınav Linux Sure

thinclient
↓
sunucu bağlanıyor
hardisk yok

terminal → command line dosya görmek, değişiklik yapmak
command prompt

>ls = terminali açtığımızda home/11106101
>cp
>mv
>pwd
>cd
>mkdir
>touch
>rm
>rmdir
>chmod
>chown
>man
>clear

>ls dizindeki dosya ve klasörler dizinler
→ adlar sadece klasör

gizli dosyaları → >ls -la
gösterir. gizlilik hepsi

. noktayla başlayan klasörler dizinler

>ls -l
Bir klasörün veya dosyanın ayrıntılı sistemini görürüm

iki şekilde >ls -lai

drwxr-xr-x 2 11106601 user 72 2021-07-11 11:30

d → directory
owner → read write execute
group → kullanıcım ait olduğu dizinler
others → diğer kullanıcı

klasöre verilen link sayısı | kullanıcı adı | byte cinsi boyut | değiştirme tarihi | klasör adı

>cp copy komutu 1. parametreyi 2. parametreye kopyalar

dosya, klasör dizini

home
bbglab1 → klasör
bbglab2

>cp bbbglab1/deneme.txt bbglab2/

>mv

parametreler aynı şekilde
"cut" gibi
Tek yerde dosya kalmaz

>pwd

print working directory
(bulunduğum directory adını veriyor)
/home/11106601

>cd

change directory

>cd bbglab1 → klasör ismine geçiriyor

>pwd

bilgi alınır /home/11106601/bbglab1

>cd ..

geri dönüş

>pwd

home/tulin

>mkdir bbglab3

directory oluşturma

>cd bbglab3

>pwd

home/11106601/neel3

>mkdir neel
>cd neel
~/neel>cd ..

yeni dosya yaratma

>touch test.txt

>ls

>rm test.txt parametre olarak dosya adı
(silerim ve geri dönülmez)

>rmdir bbglab3
(directory siler eğer içi boşsa)
eğer dolu ise uyarı verir boşalt diye.

>chmod

change mode

bir dosya veya klasörün rwx haklarını değiştirebilir.

>chmod
>ls -l

-rw-r--r--. yapmalıyım

6 6 6

dosya olduğunu d yok, - var.

text olduğunu - ile execute hakkımız yok değil ama

>chmod 666 test.txt

110|110|110

2'lik düzen
6 6 6

>ls -l test.txt
-chmod 644

>chown <yeni sahip > dosya adı>

Change owner

dosyanın sahibini değiştirir

root

kullanıcıdan, en tepeyi olan, herşeyi yapar/qualistim

parametre almak, argüman kullanılan

>man ls

↓

manual

kullanım/kullanılan

ne yapar, parametreler yazılır

"q" ile çıkış

>clear

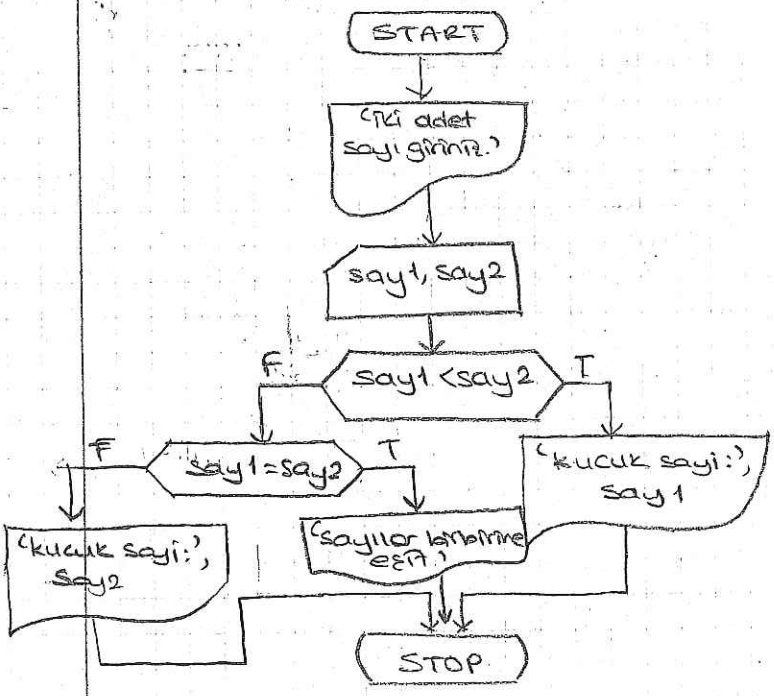
www.ce.yildiz.edu.tr

documents

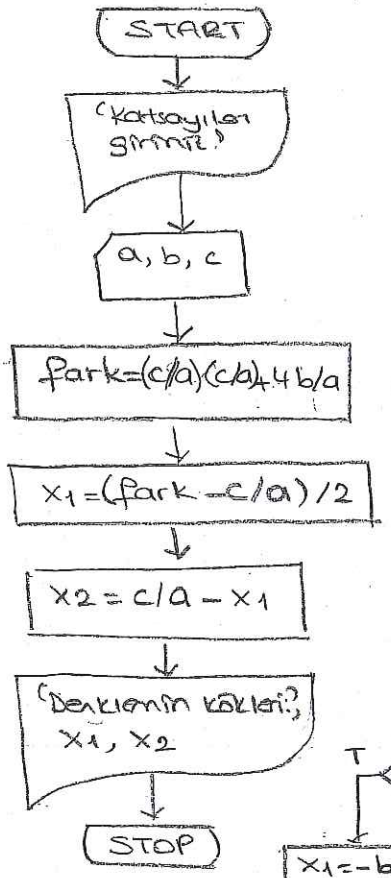
sw lab related

ödev

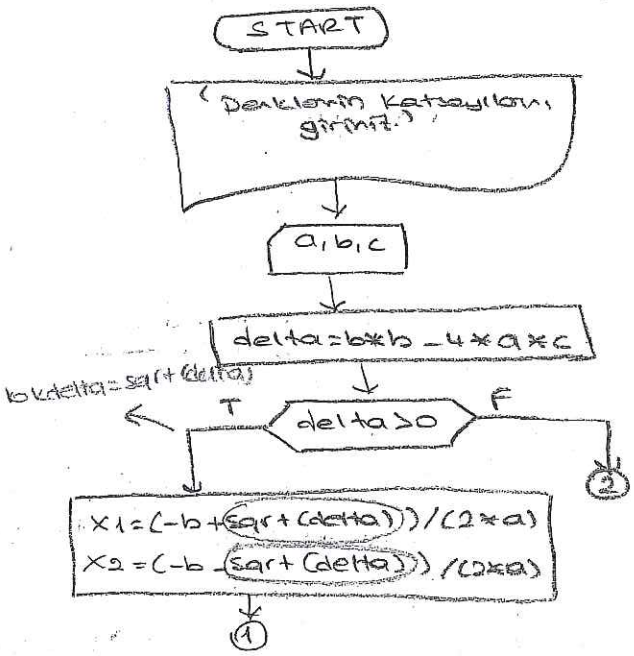
4) Iki sayı girilir, küçük olan sayının değerini ekrana yazdıran prog.

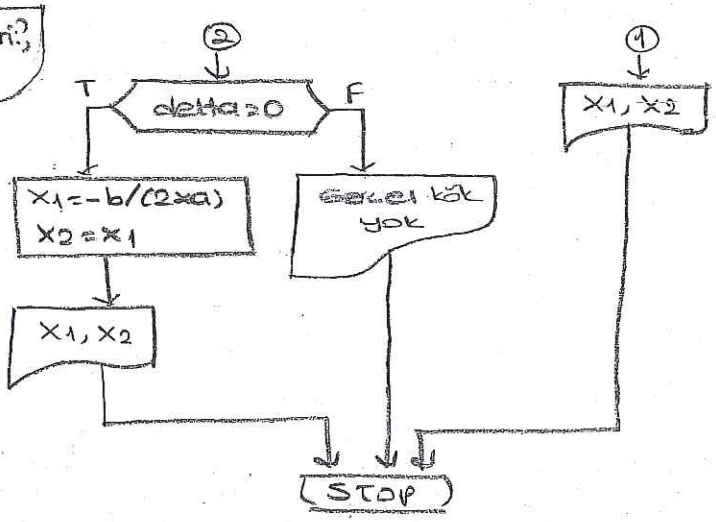


2.) a,b,c katsayıları verilen 2. dereceden $ax^2+bx+c=0$ gibi bir denklemin köklerinin gerçel olduğu bilinmiyor. Bu kökleri veren algoritma?



Pascal'da sqrt(delta) square root





En yüksek olasılıklı duruma göre akış diyagramını çiz.

Neyin olmasını bekliyorsak TRUE kolunu onu yaz.

1 Anlaşılır
2 Kısa

İşlem Öncelikleri

a ← b*b+c

1

2

a ← b+b*c

1

2

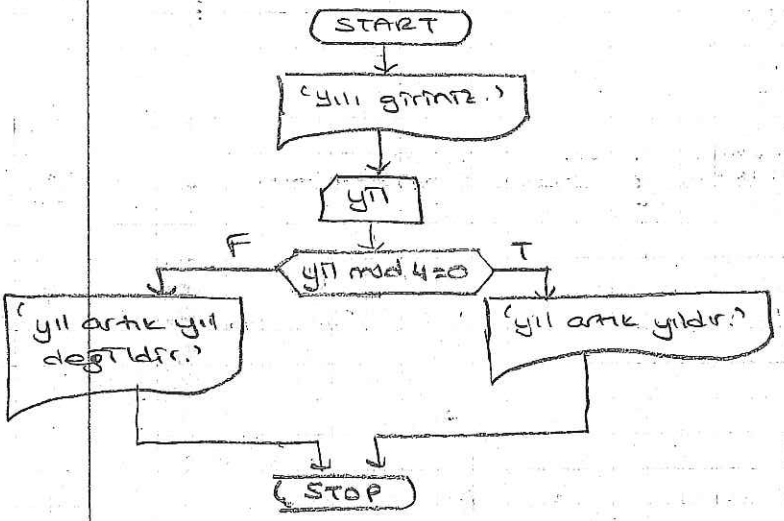
x1=(-b+sqrt(delta))/(2*a)

( ) olmazsa 2'ye bak, a ne sorar.

1. ( )
2. * /
3. + -

+
-
*
/
mod

Artık yıl 4'e tam bölünen yıl olduğuna göre bir yılın artık yıl olup olmadığını bularak yazdıran algoritmayı tasarlayınız

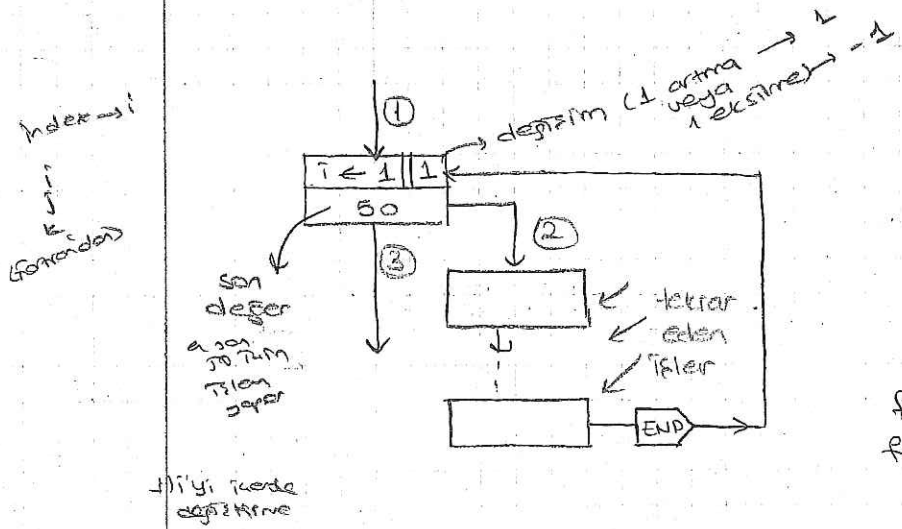


Hınak tarafa tüm kullanılma okler
↓
değişkende kullanma
↓
nesnede kullan

1. Giriş
2. İşlem
3. Çıkış

Döngü Kullanımı
(LOOP)

1. Adım sayısı belli olan
2. Adım sayısı belli olmayan

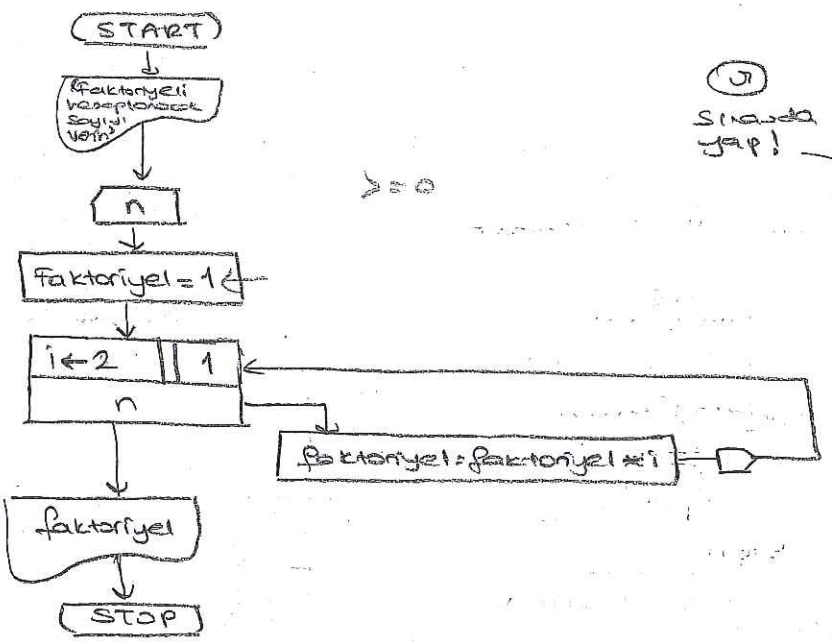


Parçalda ilerler
ilerlar artar

→ bağlantı kutusu yön gösterir.

for i:=1 to 50 do
for i:=50 downto 1 do

Verilen bir sayının faktöriyel karşılığını hesaplayan algoritmayı tasarla



N
↓
4 → 4! = 1*2*3*4
7 → 7! = 1*2*3*4*5*6*7

Sınıfta yap!

Analiz

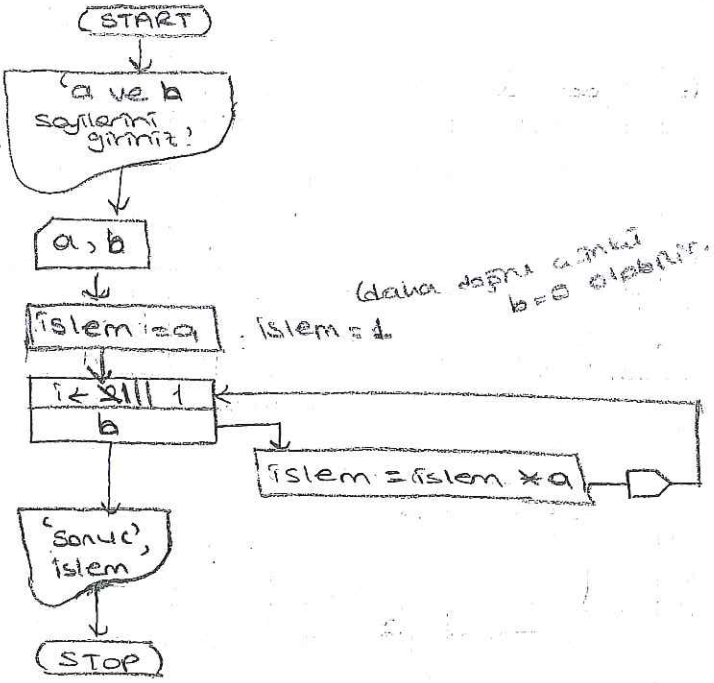
n	faktöriyel	i
3	1	2
	2	3
	6	4
1	1	2

İşlem i'yi 4 döneceğini yazar

Bir a sayısının n. dereceden kuvveti a sayısının (n−1) defa kendisiyle çarpılmasıdır. Buna göre verilen a ve n sayıları için a sayısının n. dereceden kuvvetini hesaplayan algoritmayı tasarlayınız.

a^n

$a^4 \rightarrow a * a * a * a$



a=4
b=3
İslem=4

i
2
3
4

İslem
a*İslem = 4*4 = 16
a*İslem = 4*16 = 64

(daha doğru a ancak b=0 olabilir.

ara birde değişen değerler tek

kullanıdan birde bütün değerleri bir dizide, sağ tarafta kullanıp sol tarafa atanabilir

İslem = temp
temp = a

İslem = a*a
a = a*a
(kutlama karıma gitsin)
2*2 = 4
4*4 = 16

NS Pascal

platform bağımsız: herkes aynı desteklenir.
(İşletim sistemleri farklı olsa da)

Pascal platform bağımsız.

girdi yapıldığını gösterir (nesnelerle ilgili sonuçlara gerek yok)

çıktı yapıldığını gösterir.

program kuvvet (input,output);

bir isim bıraktığını gösterir. (isim gibi)

parantez derleyicide karıştırılır.

var ← var

varsayılan kullanılmıyor dağınık olan değişkenler tanımlanır.

islem, a, b, i : integer;

begin

writeln ('a üzeri b işlemi için a ve b sayılarını veriniz');

writeline kullanılmı.

readln (a,b);

readline 3 eter 5 eter

islem := 1;

for i := 1 to b do

begin

islem := islem * a

end;

writeln ('Islem sonucu =', islem);

end.

islem: integer
ch: integer
da olur.

program kodunun bütün bir bloktur. (ana program bloğu)
blok başı, sonu begin end ile ayrılır.

blok tek bir satırsa begin-end'le gerek yok

end'lerden önce son yapılan iş sonra ; konmalı (sadece en sonuncu)

değişken tipleri
karakter → '?', 'a'
kesirli sayı → 3.7 2.8
tamsayı → 32719 -99
tipleri uygun kullan.

CP tamsayı çünkü 4 ter 1 er artıyor.

(a=b)

a=b ← a:=b;

for i := 1 to b do

İlk değer — son değer

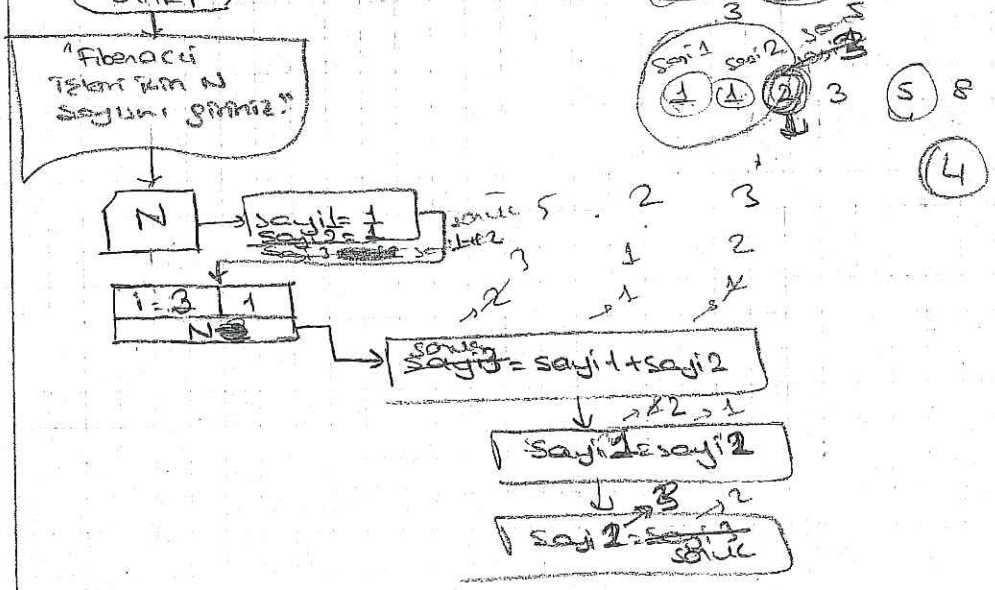
bu değişim 1 arttırır 1 ekliyor.

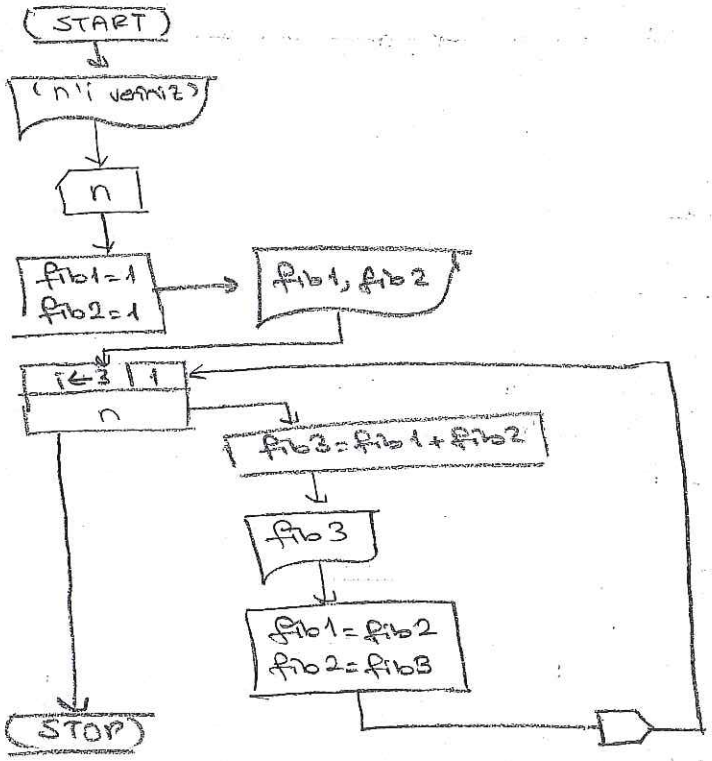
for i:=10 downto 3 do

end'den önce noktalı virgül yok.

döngü içinde ilk veya son değer değiştirme.
if (a=b)
i=8 olmalı nesne

İlk iki değeri 1 olan ve bundan sonraki her değer kendisinden önce gelen son 2 değerin toplamına eşit olan sayılara Fibonacci sayıları denir. İlk N Fibonacci sayısını hesaplayan algoritmayı tasarlayalım.





2 1 'den de başlayabilir

1 1 2 3 5

fib1 fib2 fib3

fib1 fib2

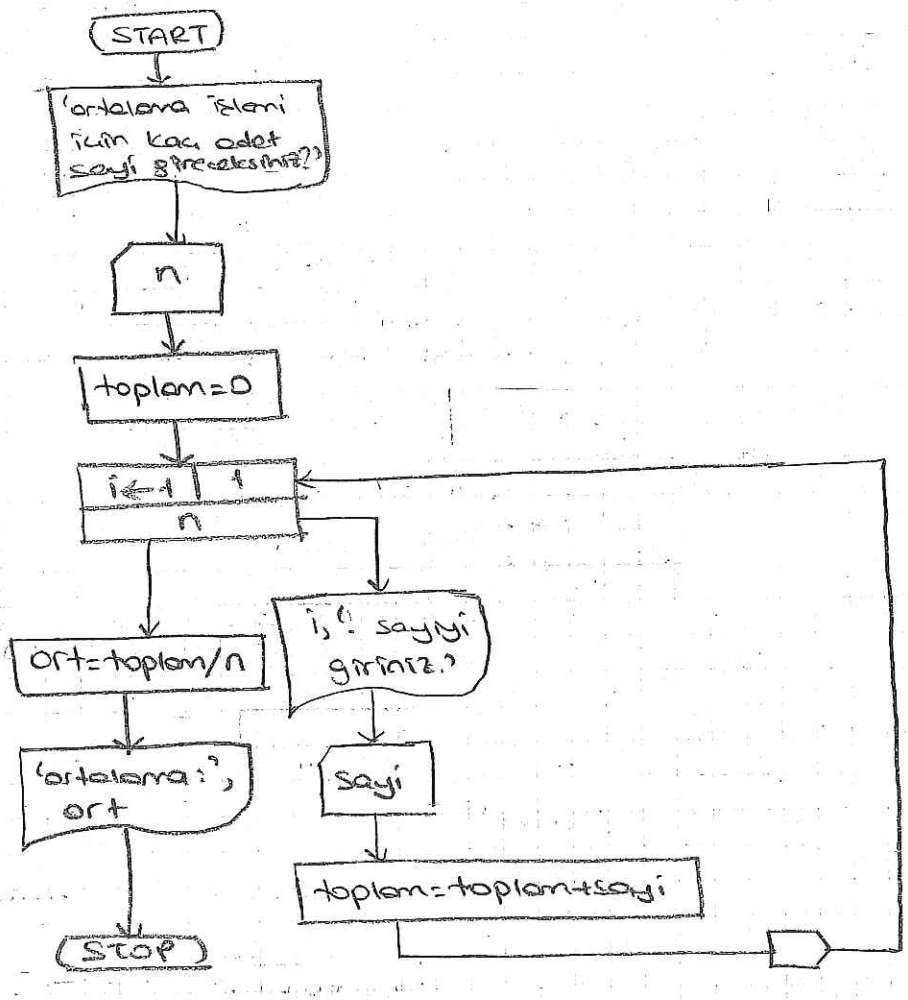
N=7

i	fib1	fib2	fib3	Term
3	1	1	2	2
	1	2		
4	2	3	3	3
5	3	5	5	5
6	5	8	8	

```
Program fibonacci (input,output);
var
   n, i, fib1, fib2, fib3 : integer;
begin
   writeln ('n sayısını veriniz.');
   readln (n);
   fib1:=1;
   fib2:=1;
   writeln (fib1, fib2);   altta yazarsan
   for i:=3 to n do
   begin
   fib3:=fib1+fib2;
   writeln (fib3);
   fib1:=fib2;
   fib2:=fib3
   end
   end.
```

HW

Dışarıdan girilen n adet sayıyı toplayarak ortalama sını yazdıran algoritmayı tasarlayıp programını yazınız




```
Program    ortalama (input, output);
var
   toplam, n, i, sayi : integer;
   ort : float;
begin
  writeln ('ortalama işlemi için kaç adet sayı girecekseniz?');
  readln (n);
  toplam := 0;
  for  i := 1  to  n  do
   begin
     writeln (i, '. sayıyı giriniz.');
     readln (sayi);
     toplam := toplam + sayi
   end;
  ort := toplam / n;
  writeln ('ortalama:', ort)
  end
```

n	i	sayi	toplam	ort
4	1	2	2	
	2	4	6	
	3	6	12	
	4	8	20	
	5			20/4=5

Dizi → Array

Aynı amaç için kullanılan, ortak değişkenli bilgi.

```
var
    arr : array [1..10] of integer;
```

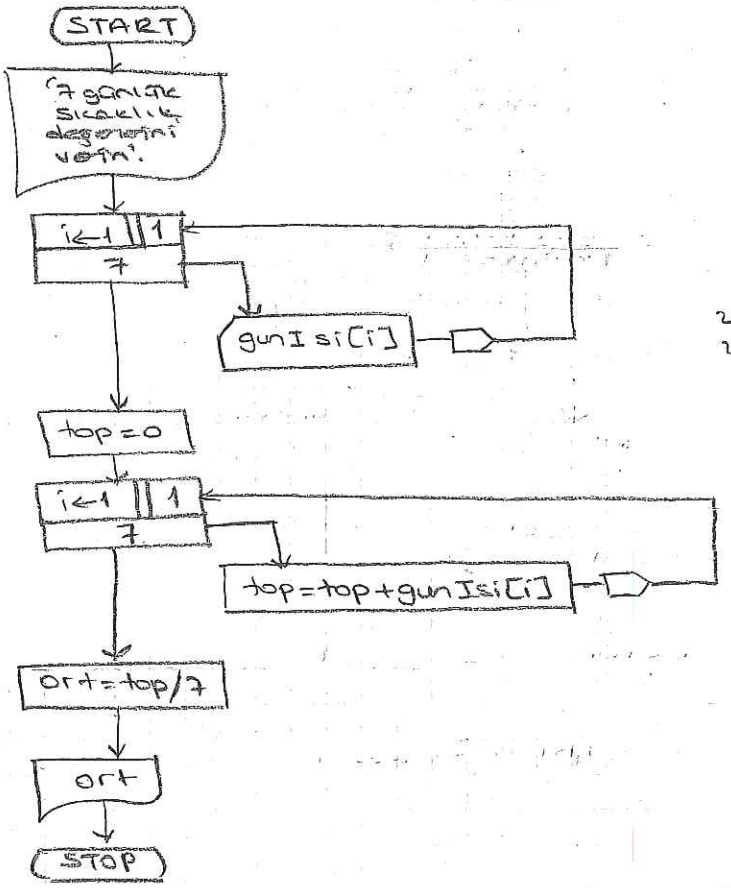
indis — başlangıç — bitiş indisleri — index

	1	2	3			10
arr	9	8		--	--	

Bir bölgedeki haftalık sıcaklık değerleri veriliyor. Bu bölgedeki o haftanın ortalama sıcaklık değerini hesaplayan algoritmayı tasarlayalım

arr[1]
arr[5]

i	top
1	0+13→13
2	13+15→28
3	28+14→42
7	top6+14=top7
8	



giriş blok
↓
işlem blok
↓
çıkış blok

27 kasım
28 Aralık

```
program ortIsi (input, output);
var
  i, top, ort : integer;
  gunIsi : array [1..7] of integer;
begin
  for i:=1 to 7 do
    begin
      writeln (i,'. günün ısı degerini veriniz.');
      readln (gunIsi [i])
    end;
  top:=0;
  for i:=1 to 7 do
    top:=top+gunIsi[i];
```

GİRİŞ

İŞLEM

girilecek sayı belirlediğine mantıklı değil.

0 i'nin 2'den 0'dan da başlatabilirim.

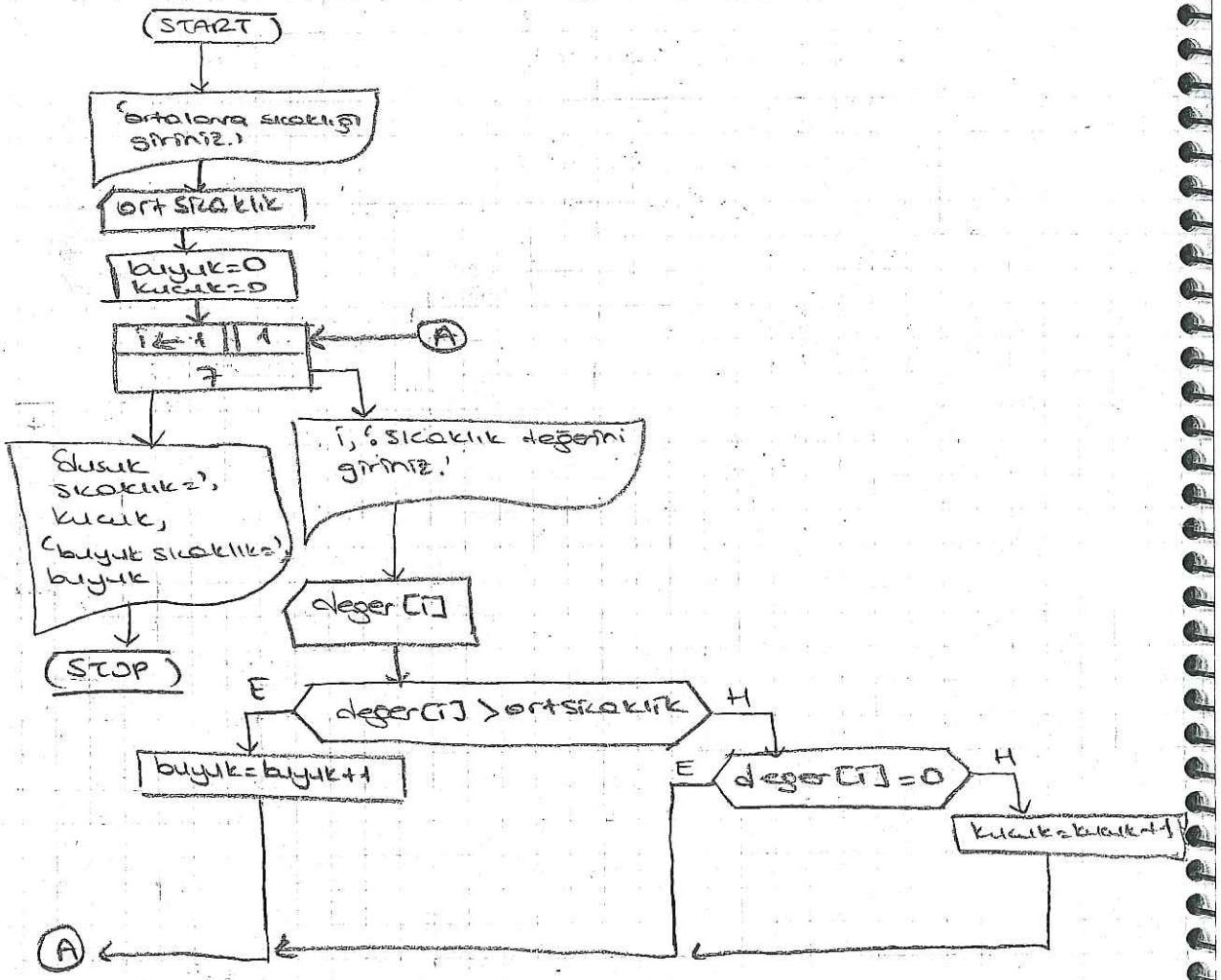
```
ort := top div 7;
writeln('7 günlük ısı degerleri için
ortalama: ', ort)
end.
```

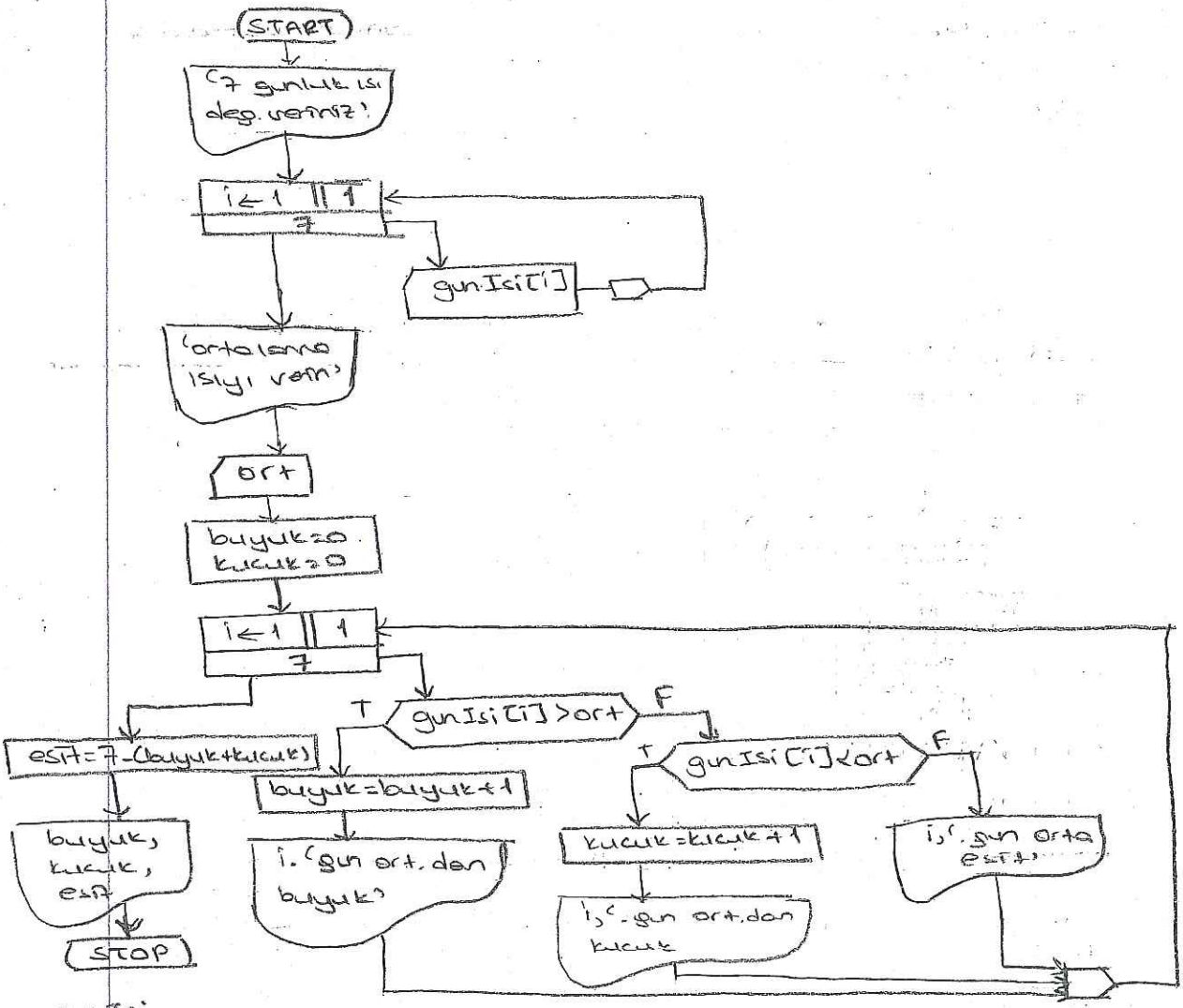
tam sayı bölme: div
kesirli bölme: /

1'de farklı dizileri array'e al, döngüde ekleme

/* — —— */

Bir bölgede 7 günlük sıcaklık dağılımı ve ortalama sıcaklık degeri veriliyor. Kaç günün ortalama sıcaklıktan büyük, kaç günün küçük olduğunu hesaplayan algoritmayı tasarlayınız





gunIsi

1	2	3	4	5	6	7
20	21	22	18	15	23	17

gunIsi[1]

ort: 19//

i	gunIsi[i]	buyuk	kucuk
1	20	~~0~~ 1	0
2	21	2	
3	22	3	
4	18		1
5	15		2
6	23	4	
7	17		3

esit=7-(4+3)=0

buyuk→4

kucuk→3

esit→0

> →Buyuk

>=→Buyuk esit

< →kucuk

<=→kucuk esit

= →esit

if a=b

Bir sınıftaki 20 öğrencinin numarası veriliyor. Ayrıca bir X sayısı veriliyor. X sayısı bu öğrencilerden birinin numarası olduğuna göre, kaçıncı öğrencinin numarası olduğunu bulan algoritmayı tasarlayınız.

9 7 13 4 ← (13)

n elemanlı bir dizinin elemanları pozitif (0 dahil) ve negatif sayılardan oluşmaktadır. Bu sayıların mutlak değerini hesaplayarak yeni bir diziye yerleştiren algoritmayı tasarlayınız.

→ a: [-9 | 4 | 0 | -5]

→ mut a: [9 | 4 | 0 | 5]

LAB

Cihan

20 MB

/home/11106601/

⇒ du -sm .

↓ disk usage — parametreler

dosyalarım ne kadar yer tutuyor.

rm -f dosya adı kalıcı sil

top

İşlemci kullanım oranları (process) hangi process'e ait

ID'si

ne kadar İşlemci harcanmış

hangi program sıfırdan çalıştırılmış

grafik arayüzde kullanım yüksek programlar

kill/k -9/k process_id bırakılan programı durdurmak için

ps/k -ef Ø 1006000

sistemdeki bütün processleri gösterir.

o anlıktaki gösterir.

anlık değişim yerleşerek gösterir

sistemde bana ait olan processleri görmek için

ps/k -ef | grep/k 11106601

↓ alt sır → [> , <]

man man

↓ ma kullanımla ilgili data

manual pages

man du

man printf

↓ C fonksiyonu

space ne ↓

q ne çıkış

↓ quit

free pascal compiler

pascal → fpc test.pas -o test.exe

input file — output file

./test.exe → calıştırma kısmı

C / C++

Java

prolog

lisp

→ gcc -Wall -O2 test.c -o test.exe

GNU C compiler

Warning All

en ufak uyarıya kadar kaleci oylar

optimizasyonu programın ne kadar optimize edileceği (2. seviye optimizasyon)

compile run

uzantılar hangi dilde ona göre var

.pas

.c

dizayn

pas ne

bazı fonksiyonlar

değişken fonksiyonlar

kod aktarır.

file transfer protocol

*ftp → dosya aktarmak, göndermek

WEB BROWSER, URL:

ftp://cihan@tweety.ce.yildiz.edu.tr

user name

MM06601

account adı, hesap şifresi — dosya gönderme, alma, kopyalama ona çalıştırmasını.

ne yapmak istediğe bağlı

SSH

yazdığım dosyayı derlemek, denemek çalıştırmak için bunu kullan.

Secure Shell

Linux'ta bağlanacaksam — ssh cihan@tweety.ce.yildiz.edu.tr

windows

Third party yazılım

web sayfası — Secure CRT — Open SSH

ücretli web sayfa + kullanım ücreti

Open terminal'den terminale bağlan

vi (text editor)

sonite

dosya yaratır, kaydet, çalıştır.

rm -f test.pas

silmek ne zaman

eğer müsaade edersanız

kepini silmek için root

rm -Rf *

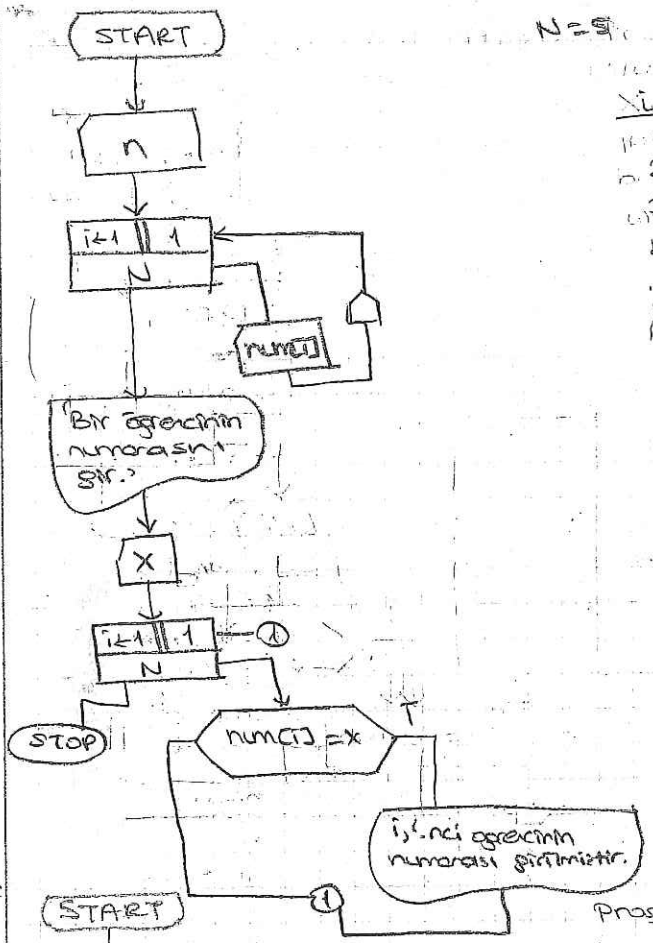
rm -f *

bulundığım klasördekiler

Ex

1-988

adım sayısı belli olan - for -



N=5

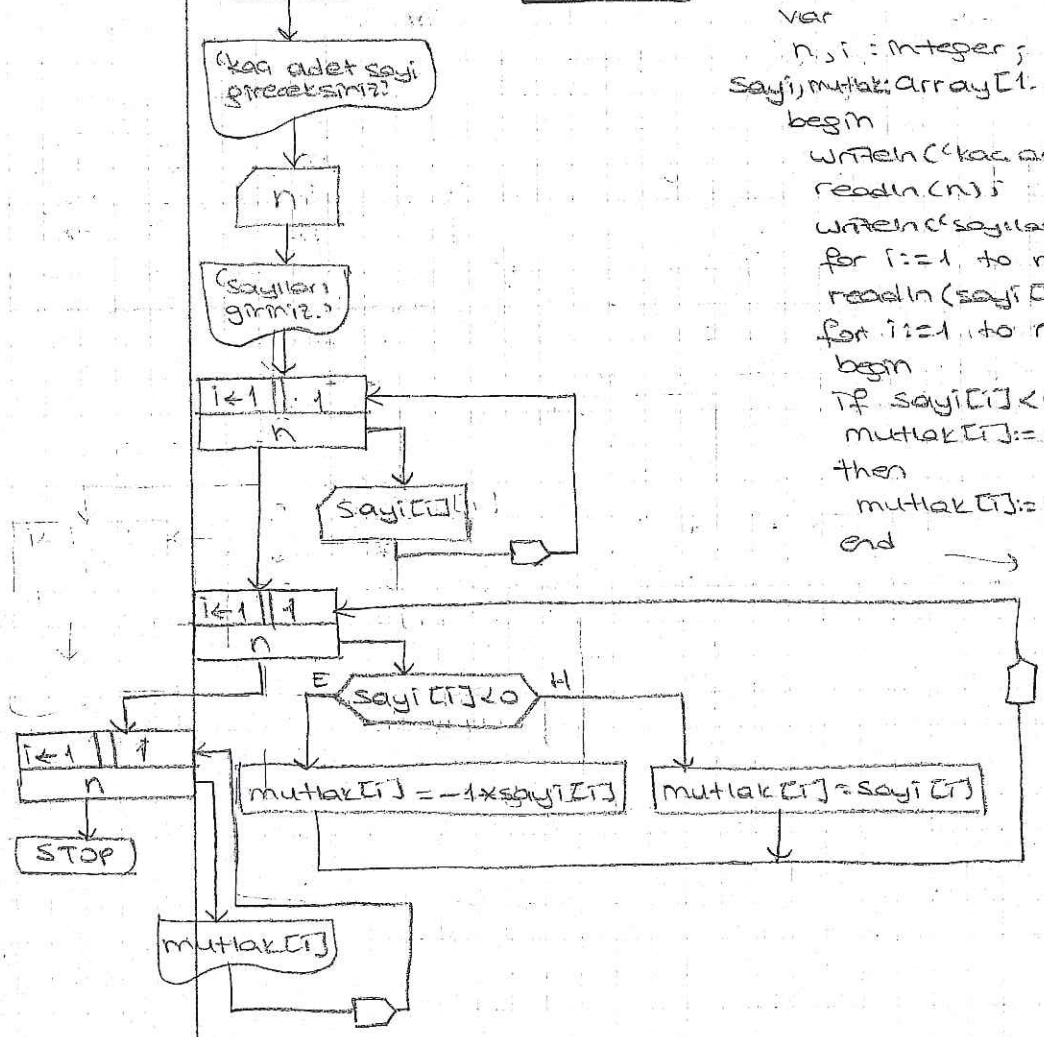
1	2	3	4	5
3	9	17	5	22

num x: 17

i	num[i]	num[i] ? x
1	3	3 ? 17 E
2	9	9 ? 17 E
3	17	17 ? 17 E
4	5	5 ? 17 E

for i:=1 to n do

Ex



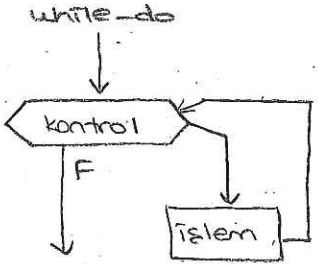

```
Program mutlak (input,output)
var
  n,i : integer;
sayi,mutlak: array[1..10] of integer;
  begin
   writeln('kaç adet sayı girecaksiniz');
   readln(n);
   writeln('sayıları giriniz');
   for i:=1 to n do
   readln(sayi[i]);
   for i:=1 to n do
    begin
    if sayi[i]<0 then
     mutlak[i]:=-1*sayi[i];
    then
     mutlak[i]:=sayi[i]
    end
```

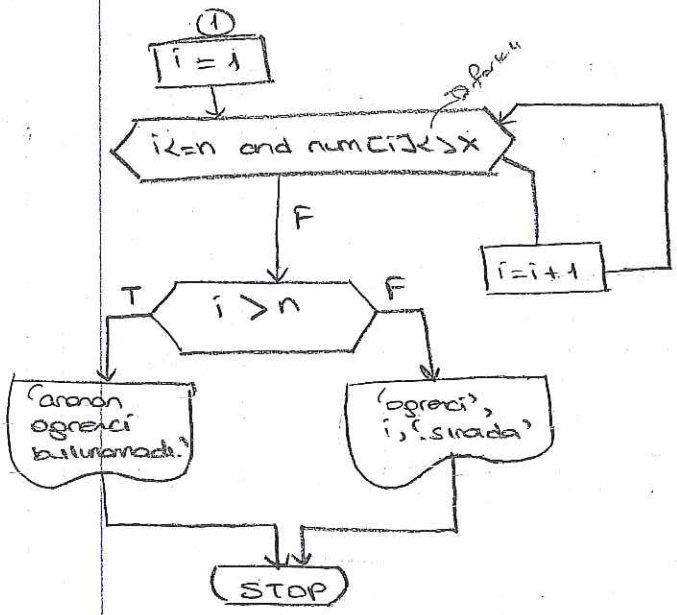
```
for i:=1 to n do
writeln(mutlak[i])
end.
```

Adım sayısı belli olmayan döngüler

while-do : olduğu sürece

repeat-until : olana kadar





num[i]<>x and i<=n yazarsak

6 tam num[6] ya bakar

Şartlardan birisi bozulduysa

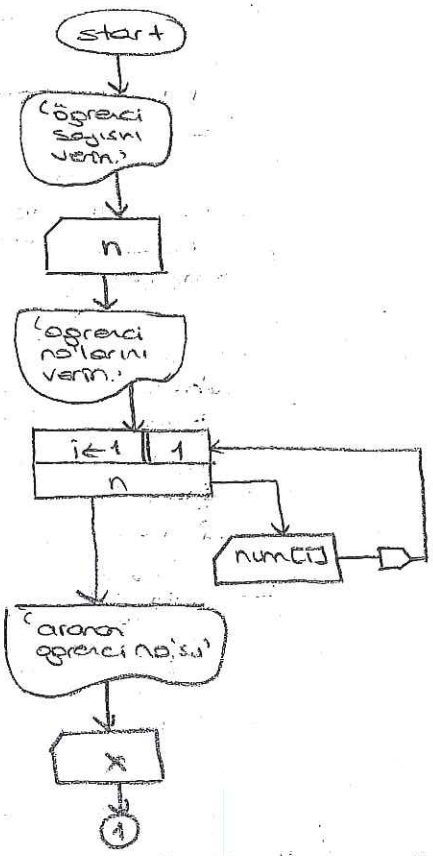
Ilk şarta bakılmalı

i<=n ise num[i]'ye bakılır

i>6 ise num[6]'ya bakmaya çalışmaz

1 bulunanı

2 bulunmama ne olacağını kontrol et.



GENEL ANALİZ

i	num[i]	num[i] = x
1	3	3<>17 T
2	9	9<>17 T
3	17	17<>17 F

uç noktalar

1 öğrenci

öğrenci yoksa ne yapar.

UÇ NOKTA ANALİZİ

x: 3

1	3

3<>3 F

x: 19

1	3
2	9
3	17
4	5
5	22
6	→ aranan öğrenci bulunamadı

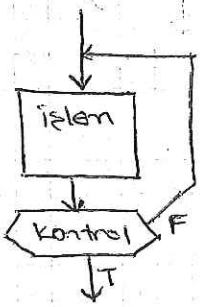
```
program  prog1 (input,output);
var
  i,n,x : integer;
  num : array[1..200] of integer;
begin
  writeln('ogrenci sayisini verin.');
  readln(n);
  writeln('ogrenci no'larini verin.');
  for i:=1 to n do
    readln(num[i]);
  writeln('aranan ogrenci no'su');
  readln(x);
  i:=1;
  while i<=n and num[i]<>x do
    begin
      i:=i+1
    end;
  if i>n
    then
      begin
        writeln('aranan ogrenci bulunamadi')
      end
    else
      begin
        writeln('ogrenci',i,'.sirada')
      end
  end.
```

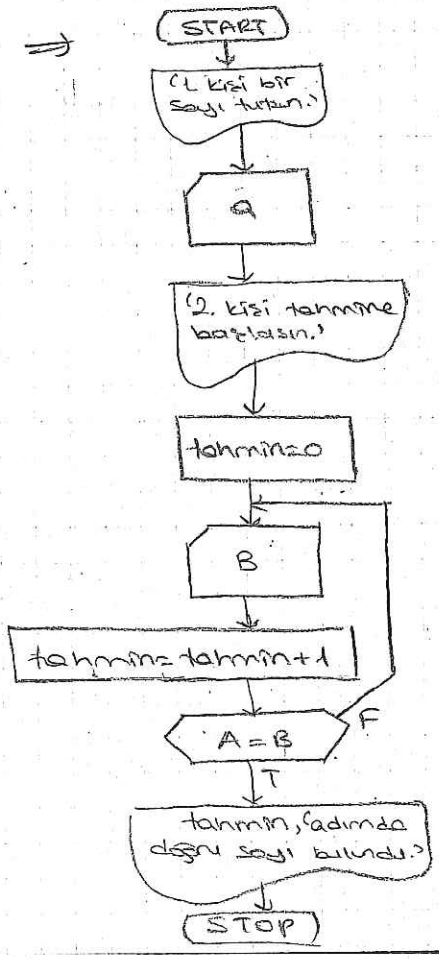
while ... do
(begin)
(end)

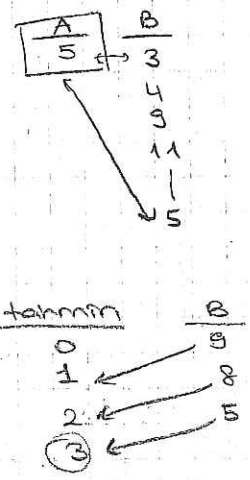
* elle'den sonraki ; hata verir.
; kaynak

Bir sayı tutmaca oyunda A kişisinin tuttuğu sayıyı B kişisi bulmaya çalışıyor. B kişisinin sayıyı kaç tahminde bulduğunu hesaplayan algoritmayı tasarlayınız.

repeat
  işlem
until kontrol;

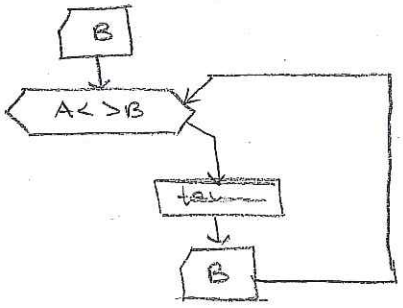






3 adımda doğru sayı bulundu

while ⇒



* bir tek
* repeat-until'in begin-end'i yok

```
program say(tahmin input,output)
var
 A, B, tahmin : int ;
begin
 writeln ('1. kisi bir sayi tutsun.') ;
 readln (A) ;
 tahmin := 0 ;
 repeat
   writeln ('B nin yeni tahmini :') ;
   readln(B);
   tahmin := tahmin+1
 until A=B ;
 writeln (tahmin, 'adımda doğru sayı bulundu')
 end.
```

D Ş

Kesin algoritmadan bahsetmek kolay olur.

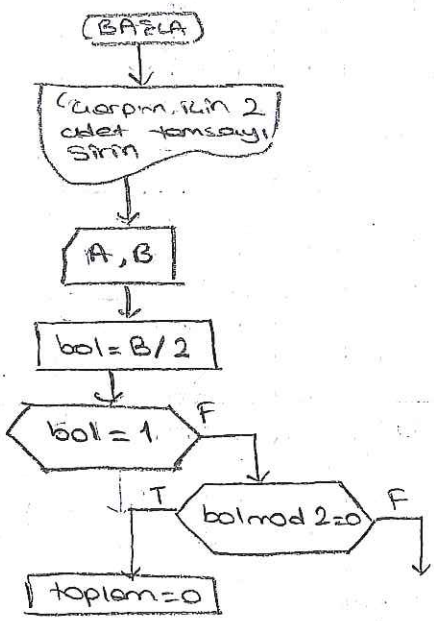
Iki tamsayıyı çarpmanın bir yolu sayılardan birini 2 ile çarparken diğerini 2'ye bölmektir. Bölünen tarafta sayı tek sayıya çarpılan tarafta sayılar toplanarak işlem bölünen sayı 1 olana kadar tekrarlanır. Toplam değer çarpım sonucunu verir. Bu yöntemle iki sayıyı çarpan algoritmayı tasarlayınız.

çarp	13	26	52	104
böl	12	6	3	1

toplam : ∅
52
+ 104
13 x 12 = 156

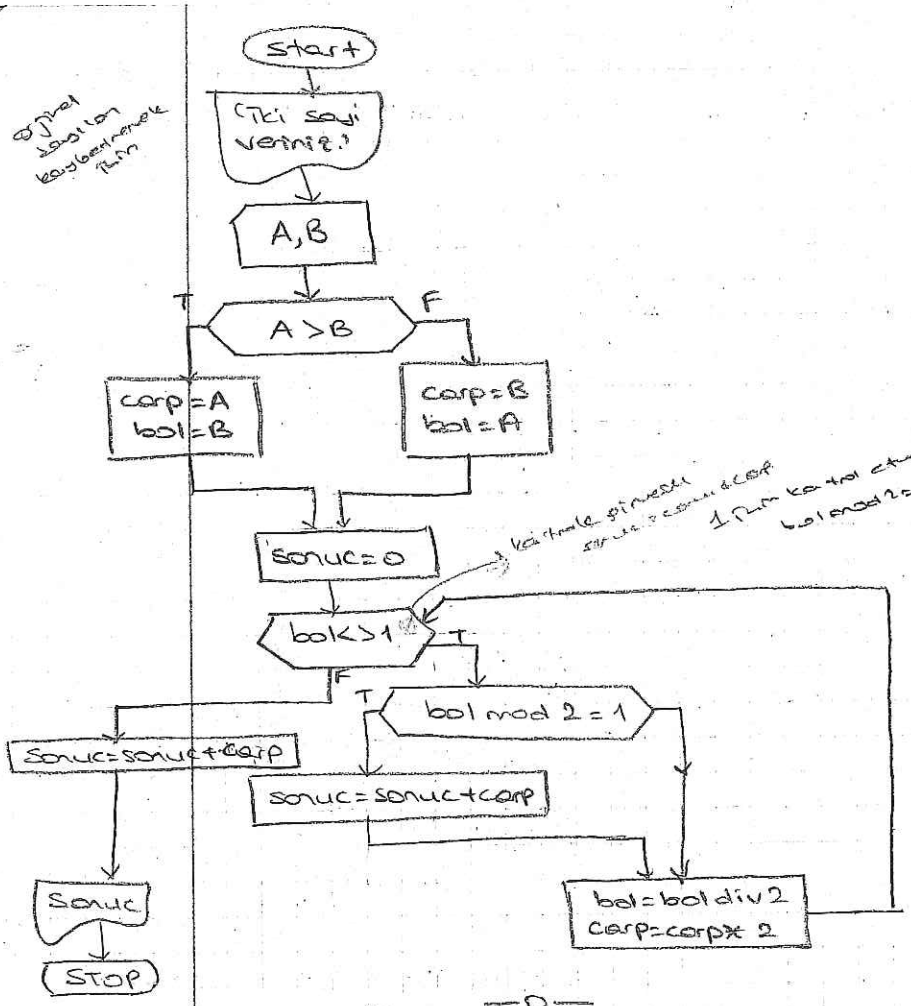
çarp	13	26	52	104
böl	9	4	2	1

toplam: ∅
13
104
117



bölmesini çarp.

bölümünün tek olduğu yerlerde en son toplayoruz.



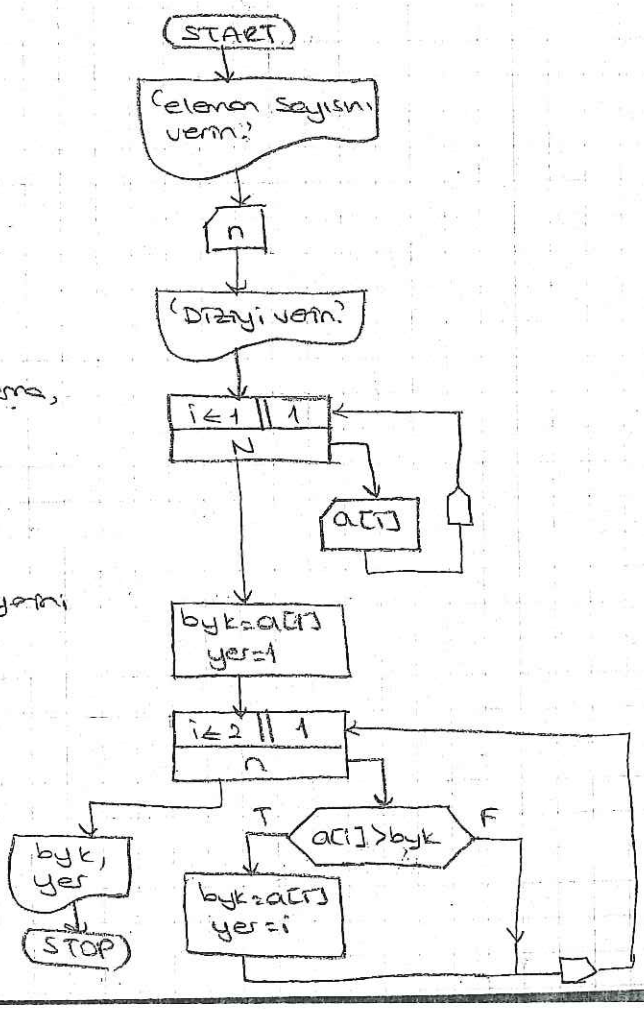
carp	bol	sonuc
13	9	Ø 13
26	4	
52	2	
104	1	117

—o—

Bir Dizinin En Büyük Elemanın Bulunması

	1	2	3	4	5
A	3	1	4	7	5

i	a[i]	byk	yer
2	1	3	1
3	4	3	1
4	7	4	3
5	5	7	4



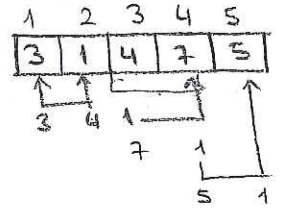
ama, pascalda onu sonsuz bırakma, mutlaka dizinin sonunda olabilecek en büyük sayı verilir. Her zaman bilinen değer olmalı.

Bir dizinin 2. büyük elemanın yerini bulan algıyı tasarlayınız.

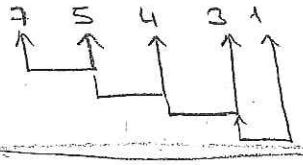
	1	2	3	4	5
A	3	1	4	7	5

Bubble Sort

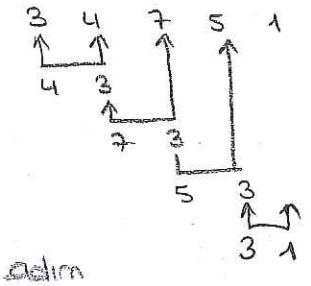
Arka arkaya 2 elemanı birbirleriyle karşılaştırıyorlar



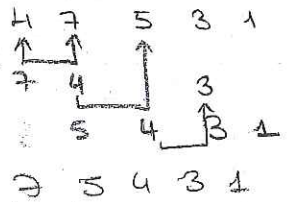
Arantar nokta hiçbir değişiklik olmayana kadar karşılaştırma yapar.



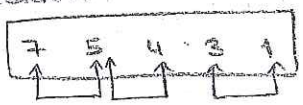
2.adım



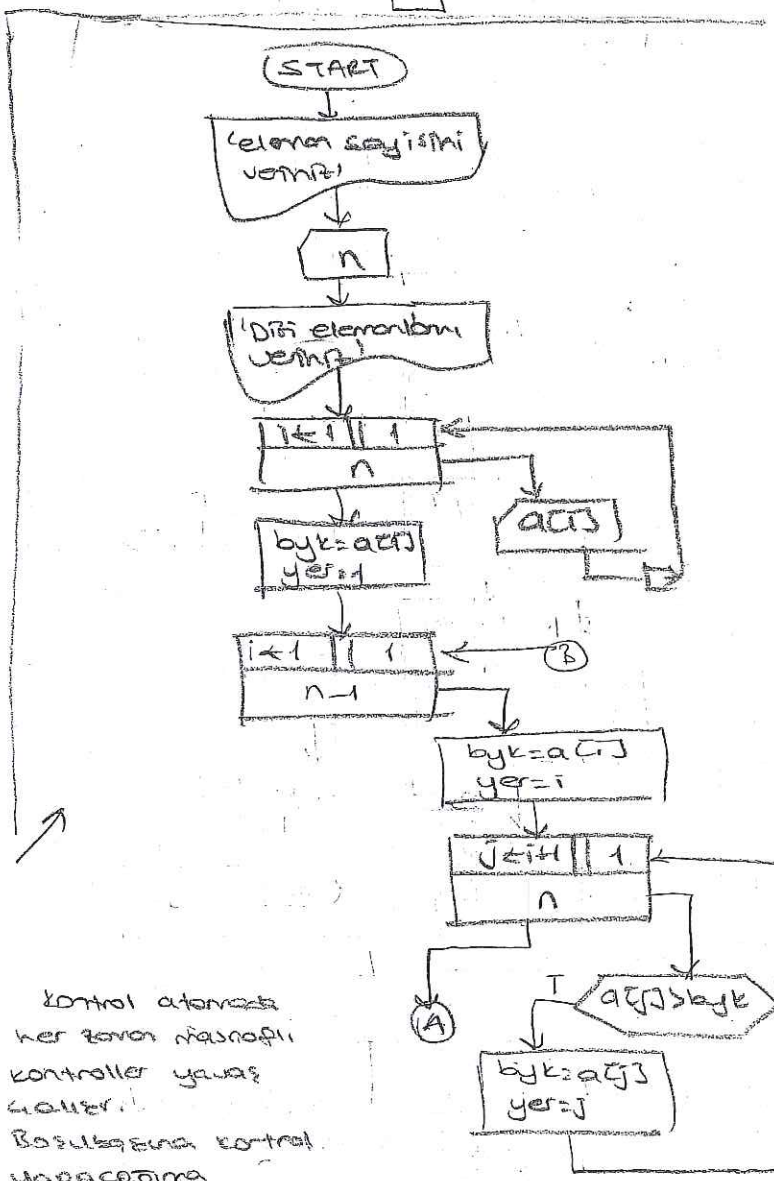
3.adım



4.adım



yer değişmeyene kadar devam eder

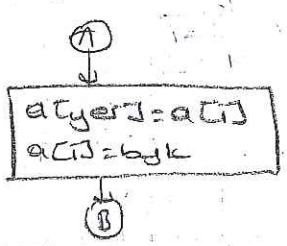


Selection Sort

A | 3 | 1 | 4 | 7 | 5 |

	3	1	4	7	5
1.adımın sonu	7	1	4	3	5
2.adım	5	4	3	1	
3.adım	4				
4.adım	3	1			

Kontrol ederken her zaman nasıl kontroller yapılır. Boşluğuna kontrol yapacağına sayıyı kendiliğinden değiştirecek



n-1 defa en büyük eleman bulma

1) denekki tekrar eden iş n-1 defa

2) tekrar eden iş ne?

Dizinin en b. elemanı bulunur ve onunla yer değiştirilir.

Her adımda baktığım bölgenin en büyüğünü bularak baktığım bölgenin en sağındaki değere yer değiştiriyorum.

Merge Sort

1	2	3	4	5	6	
13	12	9	7	5	2	A

1	2	3	4	
18	11	3	1	B

Sıralı varmış bir diziyi sıralıyoruz

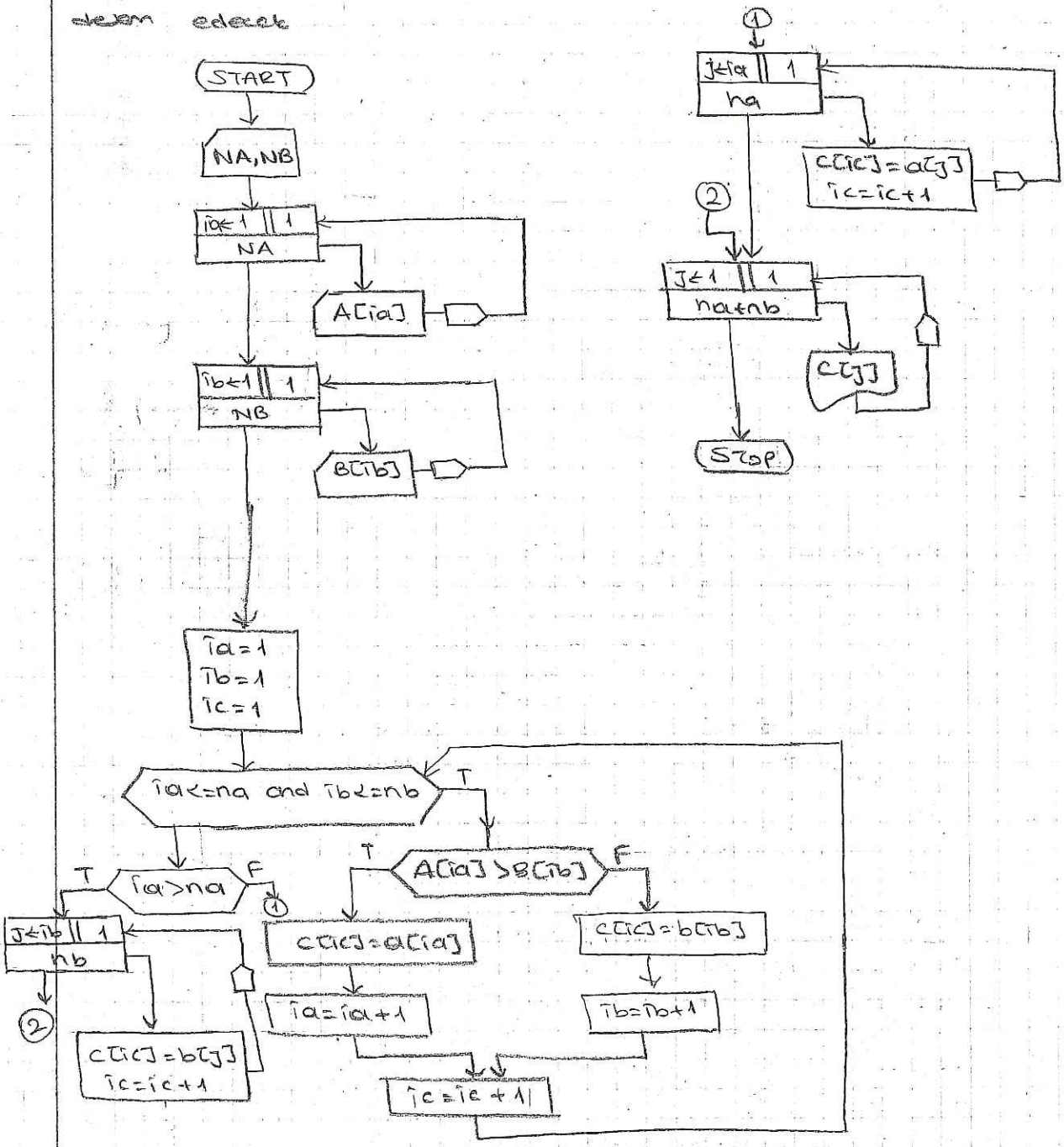
1	2	3	4	5	6	7	8	9	10
18	13	12	11	9	7	5	4	3	1

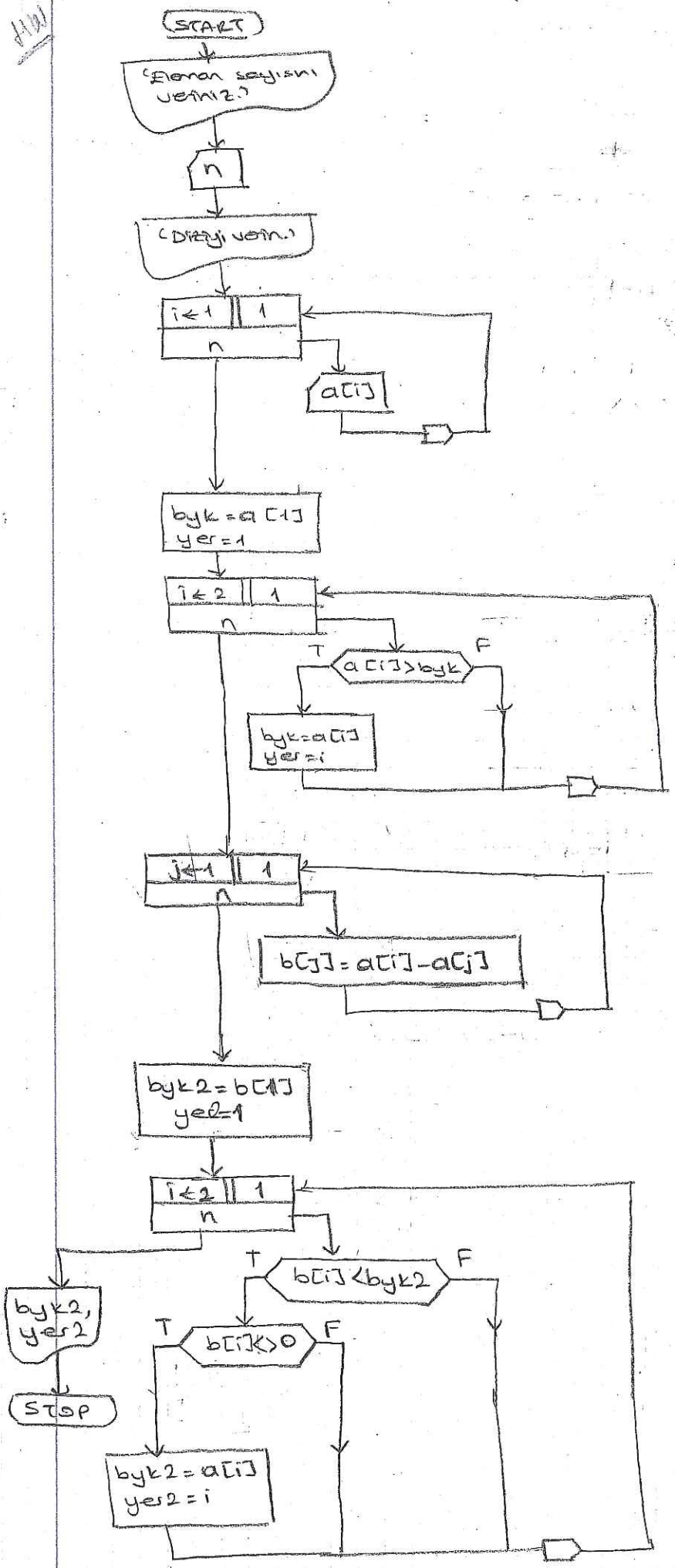
* Sayılar eşit olduğunda rastgele yazılır.

Genellikle T(n) göz (A,n) minimum

(-1) İyi yöntem değil

Dizilerde herhangi birî bitene kadar dönüm emrediyim. Bundan dolayı adım sayısı belli olmayan bir iştir. diğer çok birisi bitene kadar kontrol dönüm edecek





J M
2 5
7 8
6 2
M J
7 6

Binary Search – İkili Arama

1	2	3	4	5	6	7
2	7	8	10	12	17	19

X = 17

① 17 > 10 Bağıllar sıralıysa ortadaki elemandan karşılaştırmaya göre avantajı kullanırım

17 > 10 F 17 = 17 I

1	2	3	4	5	6	7	8	9	10	11	12
2	7	8	10	12	17	19	20	23	25	31	34

orta

x = 12

② 12 = 8 F
12 > 8 I

③ 10 = 12 F
12 > 10 I

④ 12 = 12 I

① 12 = 17 F
12 > 17 F

$\boxed{\log_2 N}$ en fazla adım sayısı

diye bulabiliyorsunuz

K → B

Baş = 1
Son = N → orta = (Baş + Son) div 2

X > A[orta] → $\boxed{\text{Baş} \to \text{orta} + 1}$
→ Son → Son

X < A[orta] → Baş → Baş
→ $\boxed{\text{Son} \to \text{orta} - 1}$

1	2	3	4	5	6	7	8	9	10	11	12
2	7	8	10	12	17	19	20	23	25	31	34

x = 20

Baş	Son	orta	X > A[orta]
1	12	6	20 > 17 I
7	12	9	20 > 23 F
7	8	7	20 > 19 I
8	8	8	20

X = 11

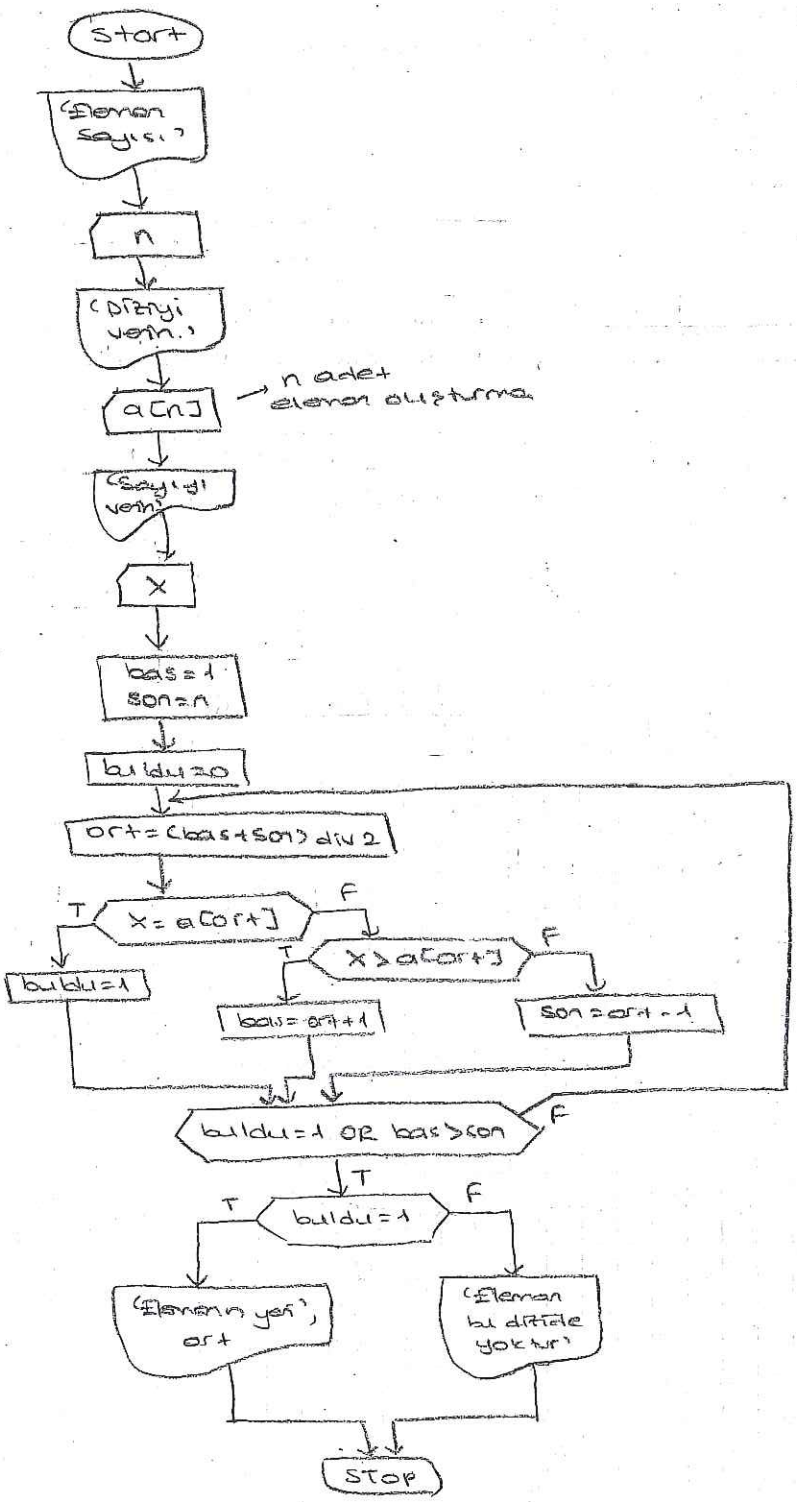
Baş	Son	orta	X > A[orta]
1	12	6	11 > 17 F
1	5	3	11 > 8 I
4	5	4	11 > 10 I
5	5	5	11 > 12 F
5	4		

① $\boxed{\text{Baş} <= \text{Son}}$

Baş son geçerse bulamadı demek

K → B

B → K'da tam tersi



Her döngü her yanlış kolon değişmektir.

buldu=0 sonu değiştirmesin

Önce i'deki sonra kontrol
↓
repeat-until

Önce kontrol sonra i'deki
↓
while

---

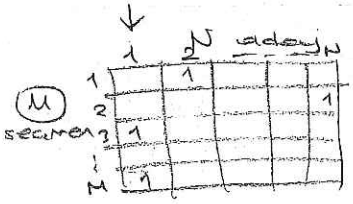
i=1

byk	yer	j	a[j] > byk	
3	1			a[yer] ← a[i]
3	1	2	1 > 3 F	a[4] ← a[i]
4	3	3	4 > 3 T	a[i] ← byk
7	4	4	7 > 4 T	
$\boxed{7}$	$\boxed{4}$	5	5 > 7 F	

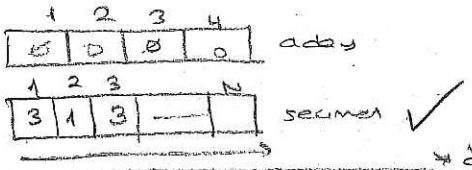
i=2

byk	yer	j	a[j] > byk	
1	2	3	4 > 1 T	a[5] ← a[2]
4	3	4	3 > 4 F	a[2] ← byk
5	5	5	5 > 4 T	

Bir maraşlarda müntaz seçimi tüm N adet müntaz adayı vardır. M adet seçmen oy kullanıyor. Hangi adayın en çok oyu aldığını bulan algoritmayı tasarlayınız.

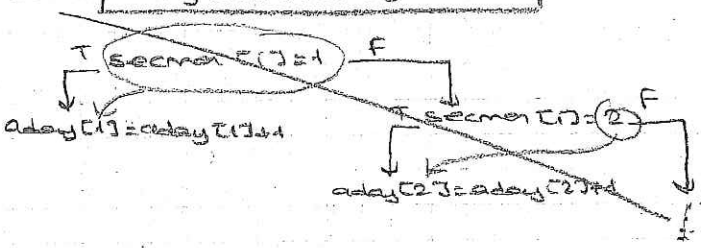


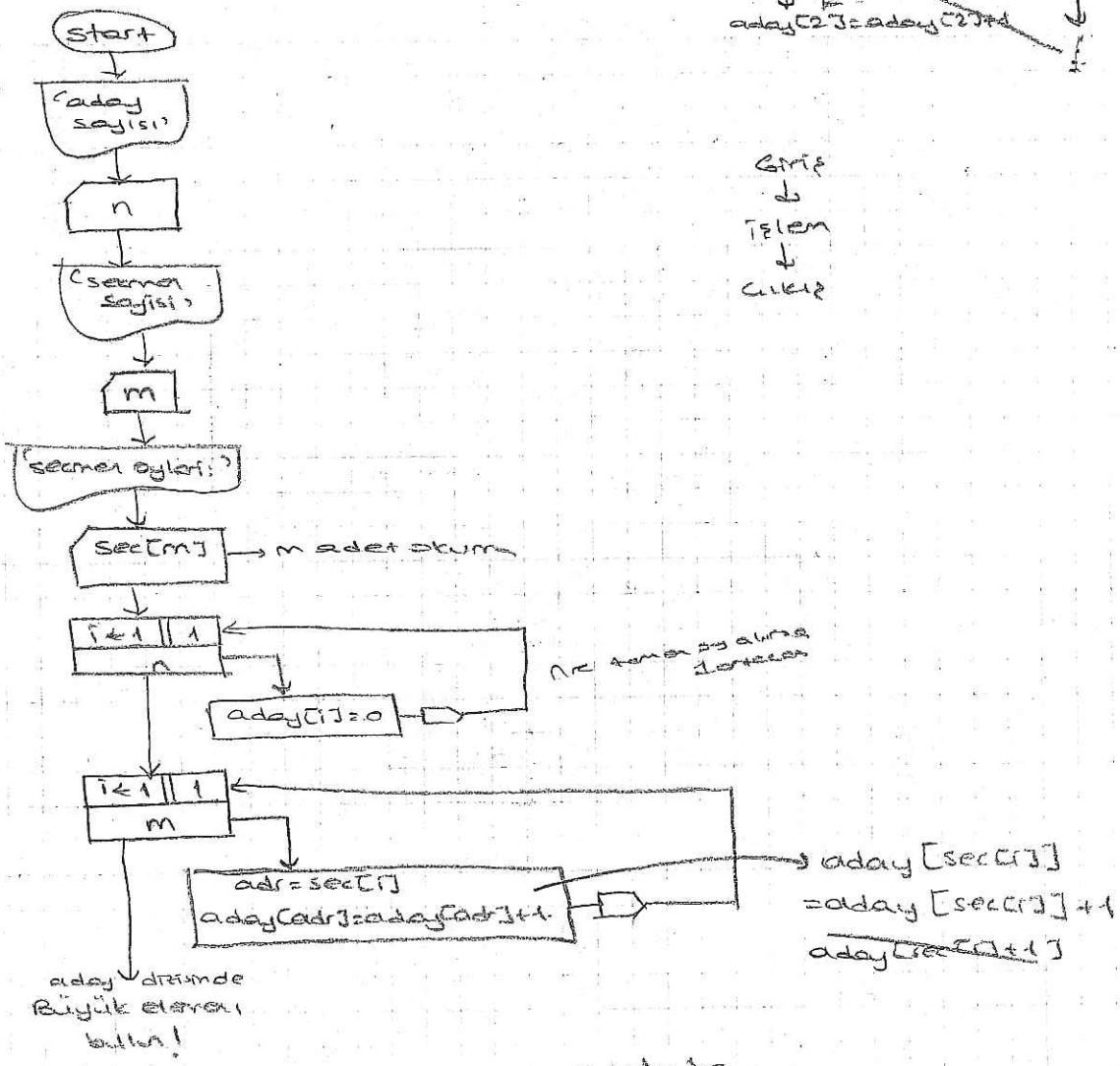
1 ve 3. aracılığıyla oy kullanmış.



adr = seçmen[i]
aday[adr] = aday[adr]+1

→ dizide bilgileri tutacağım



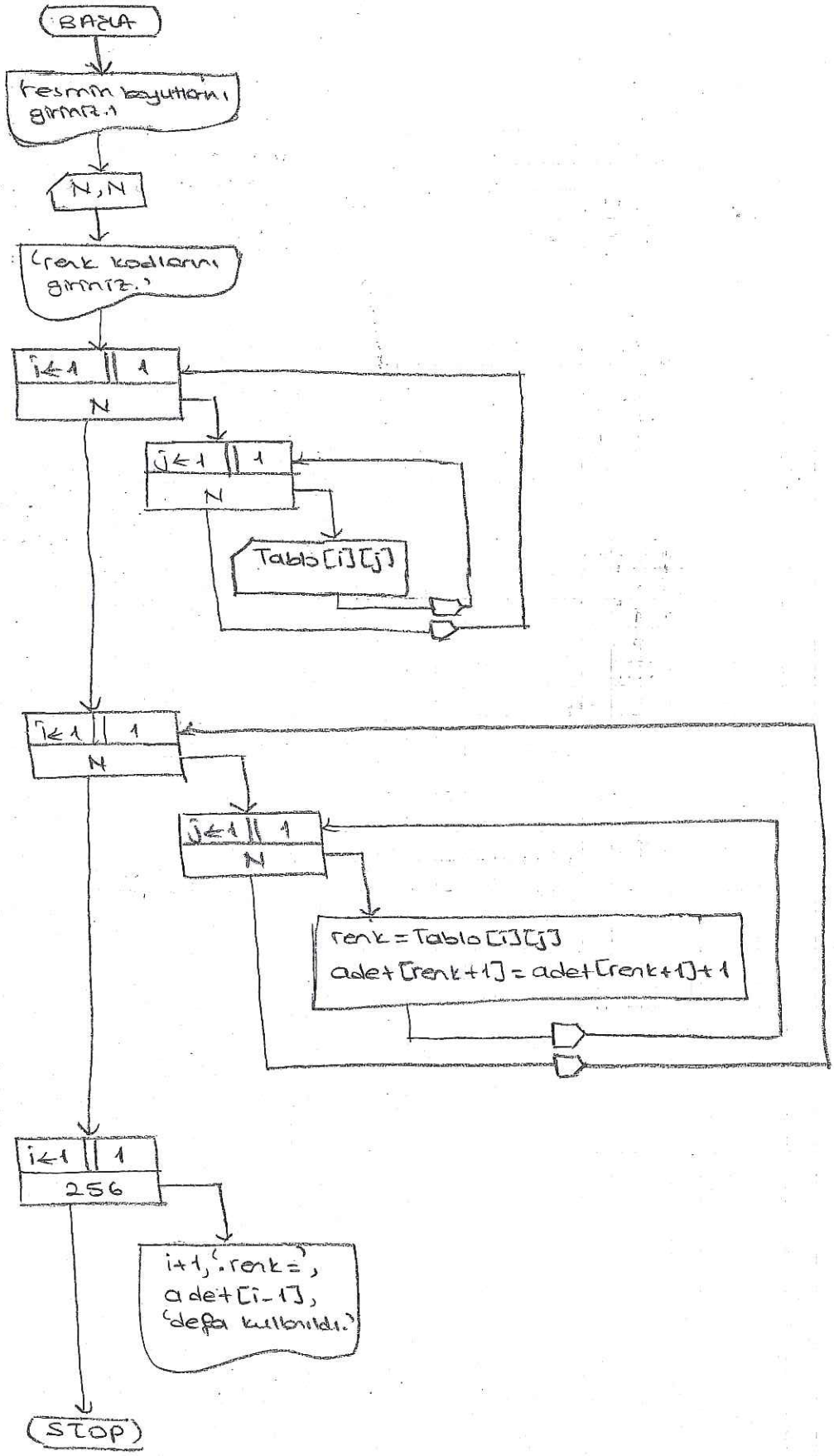


Giriş
↓
İşlem
↓
Çıkış

DEMİR AYDOĞ

HW

NxN lik bir resim 0-255'ye 256 renkten oluşmaktadır. Bu resimde her rengin kaç defa kullanıldığını hesaplayınız.



Linear Sort with Counting

Dizide değişiklik, bilgide değişiklik

demeli sıralamak gerekir, yoksa

ona dizisi bozmadan yedek iste tersinde

Çok zorlu karmaşıkca yapmaktan olmuş ama bu ötesi bir durum.

1	2	3	4	5	6	7	
3	7	1	2	4	8	9	A

Kuyu

1	1	1	1	1	1	1	index
3	5		2	4	6	7	

index dizisinde elemanların sıralı olacağı diziyi bacak bacak el olduğu durumlarda dana sonraki işlemler için kullanılır.

2, 7 kişiye
3, 7. kişi

1	1	1	1	1	1	1
2	2	1	2			
3	3					
	4					
	5					

3 - 8 2 - 7
8 - 3 7 - 2) geriye doğru bakmamalıyım

Sırasıyla kullanıldığı müşteri sıraları.

dizi geçiş yoğu

3	7	1	2	14	8	9

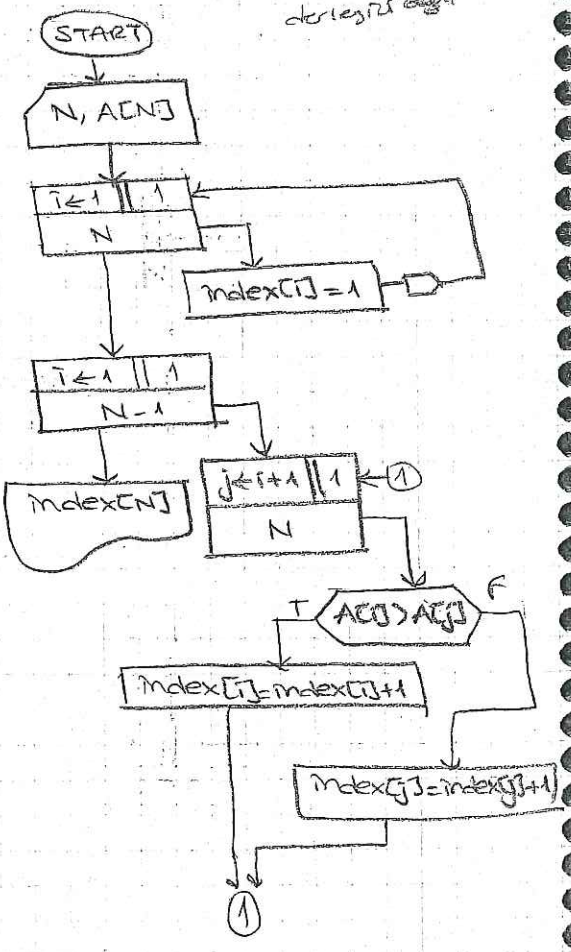
A

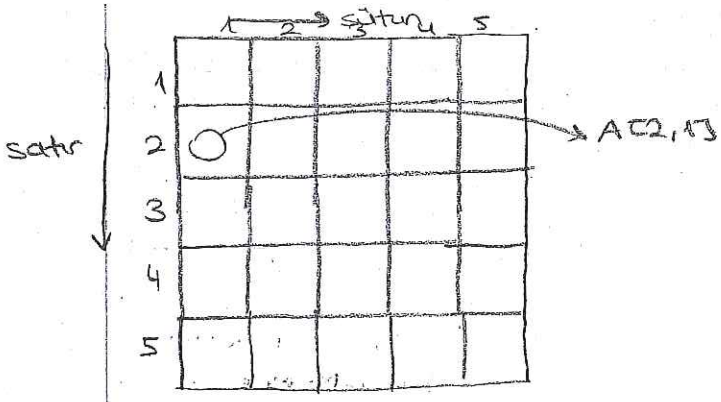
1	1	1	1	1	1	1	index
2	2	1	2	2	2	2	
3	3			3	3	3	
	4			4	4	4	
				5	5	5	
				6		6	
				7			

her elemanı kendisinden sonra gelenlere koşulastır. n-1 sefer

i=2

$C[\underbrace{index[i]}_{4}] = A[\underbrace{i}_{7}]$





A: array[1..20, 1..10] of integer;

satır sütun

A[M,N]

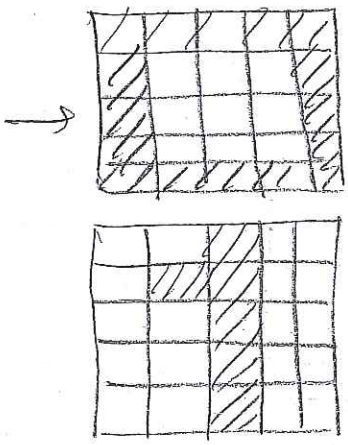
5 4

satır sütun

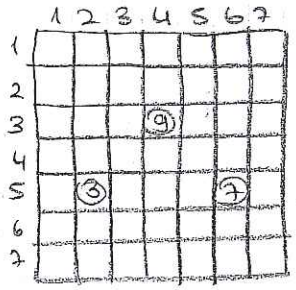
10 x 5 x 5

sayılar
10 tane 5x5
3 boyutlu bilgi

sayi: array[1..10, 1..5, 1..5] of integer;



Sparse matrix yoğunluğu fazlası 0 olan matrislere denir. Yerden kazanmak için sparse matrislerin 0(sıfır)dan farklı elemanları başka bir yerde saklanarak kullanılabilir. Bunun için gerekli yapıyı oluşturan algoritmayı tasarlayalım.



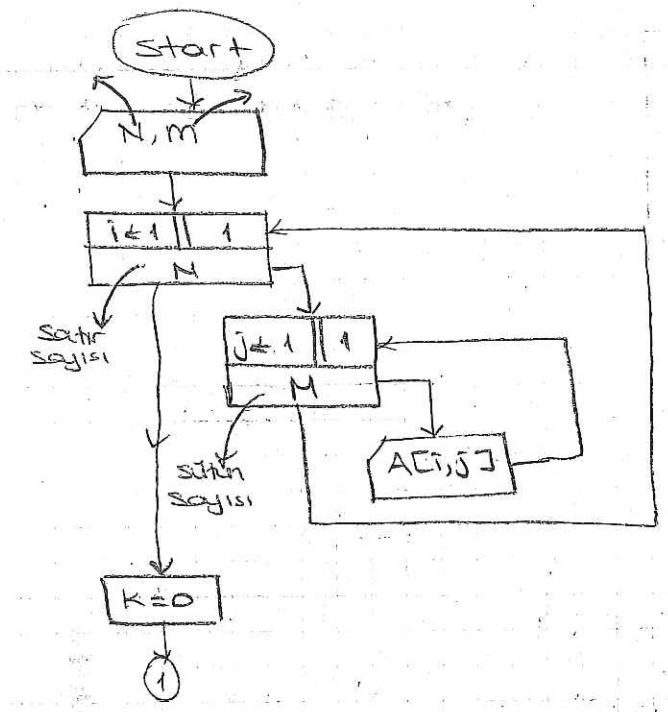
yeni birer beşin bir sıranı
(yeni sıranı)

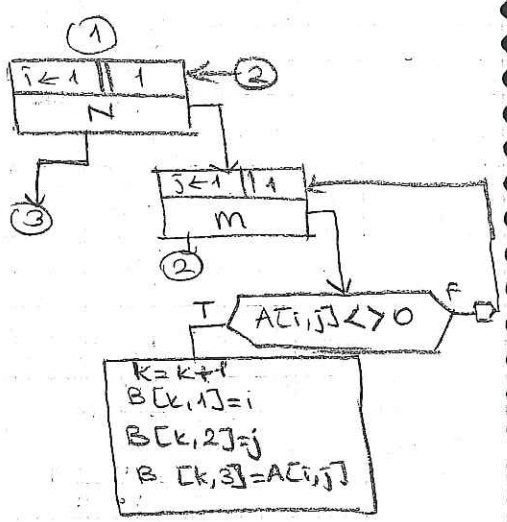
satır, sütun, değer

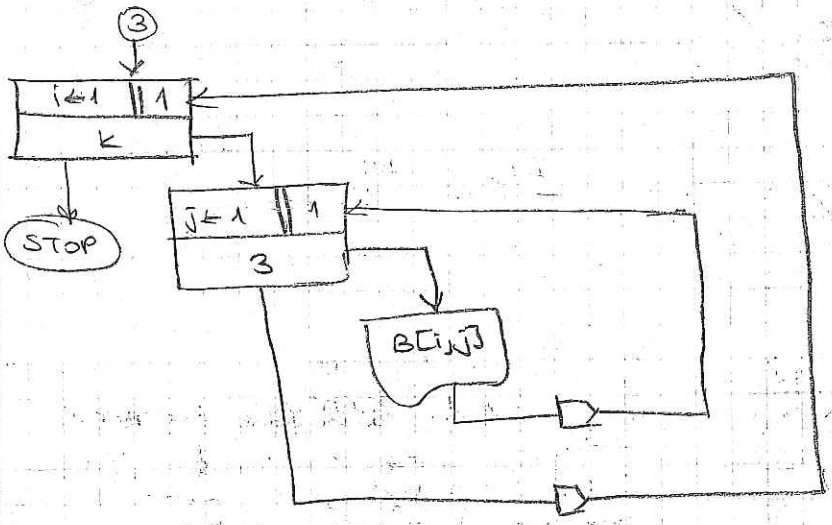
25 * 3 → 75

X

A: array[1..10, 1..3] of integer







A

	1	2	3	4	5
1					
2	4				
3					9
4					

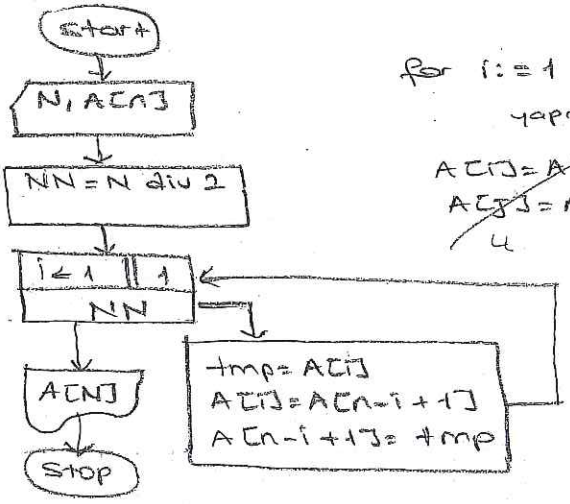
B

	1 satır	2 sütun	3 değer
→ k=1	2	1	4
k=2	3	5	9

A[3,5] ≠ 0

```
var
b: array [1..40, 1..30] of integer;
a: array [1..100, 1..100] of integer;
a[i,j]:=9;
```

Bir dizinin elemanlarını 1. ve n. sırı, 2 ve n-1. sırı, 3 ve n-2 şeklinde elemanların yeri değiştirelim.



for i := 1 to [N div 2]

yapma

~~A[i] = A[j]~~

~~A[j] = A[i]~~

4 4

NN=2

i	N-i+1	tmp	A[i]	A[N-i+1]
1	5	9	4	9
2	4	2	1	2

	1	2	3	4	5
1	3	7	0	0	0
2	4	0	0	0	0
3	1	5	9	0	0
4	2	1	8	4	0
5	3	0	0	0	0

Bir anket sonucunda n sorunun en fazla n şıkkı vardır. Buna göre alt şıklı sorular için o sorunun bulunduğu satırda şıkların bittiği yerden sonra boşluklar 0(sıfır) ile doldurulmuştur. Yerden kazanmak için bu matrisin 0(sıfır)dan farklı elemanlarını bir diziye aşağıdaki yapıda yerleştiren algoritmayı tasarlayınız.

1-satır 1. poz
2.satır 7. poz
3 satır 4. satır
4. poz

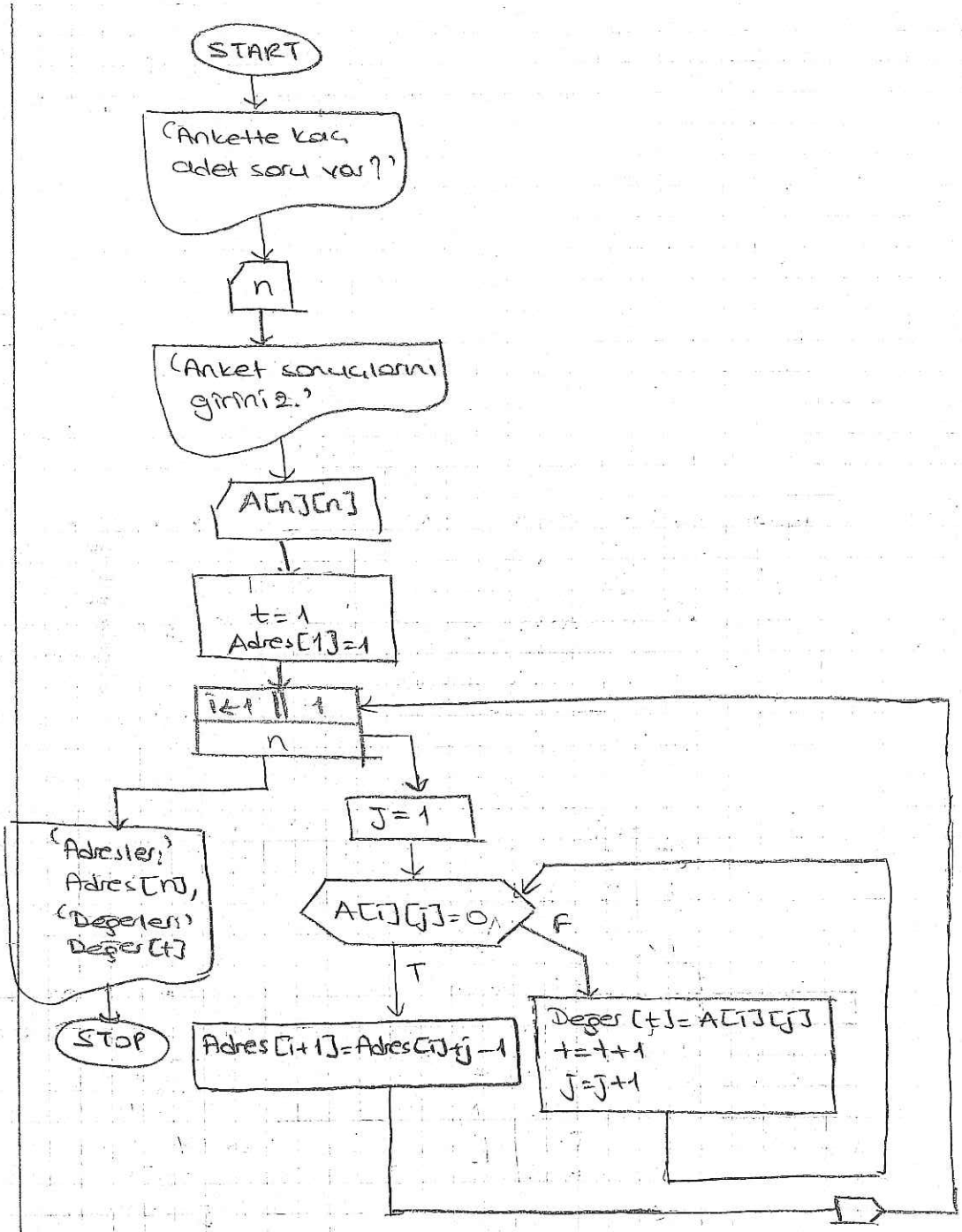
	1	2	3	4	5
Adres	1	3	4	7	11

Değer	3	7	4	1	5	9	2	1 - - -

1.satır 2.satır 3.

her satır nerede bitiyor hesaplıyorum

Uygun yapıyı oluşturduktan sonra A matrisinin i'yej. elemanının değerini A[i][j] yer oluşturduğunuz yapıyı kullanarak bulunuz.



ANALİZ

n	i	j	t	Adres[i+1]	Deger[t]
5	1	1	1	1	
		2	2		3
		3	3	3	7
	2	1			4
		2	4	4	
	3	1			
		2	5		1
		3	6		5
		4	7		9
	4	1			2
		2	8		

Sieve (Kalbur) Yöntemi ile Asal Sayı Bulma

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...	N
asal	1	1	1	~~1~~	1	~~1~~	1	~~1~~	1	1	1	~~1~~	1	~~1~~	1	...	1
2→				0		0		0				0		0			
3→						~~0~~			0			0			0		
5→										0					0		
7→														0			

Elemeye yöntemi, yedek diziye aktarma gerek yok.

karesine gelene kadar zaten katlanma önceki tıklamalar ile ugrarım:

3 → 2, 3 → 3

5 → 2 3 4 5

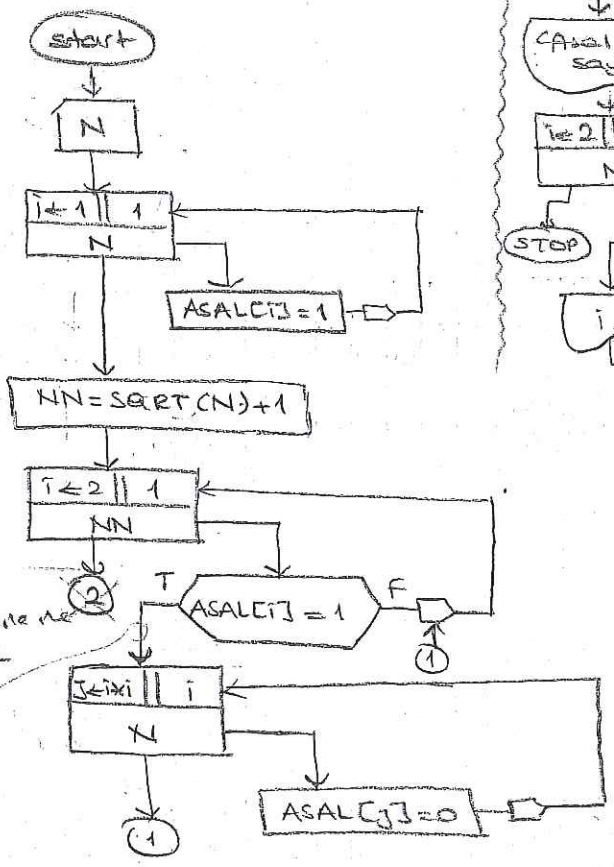
7 → 2 3 4 5 6 7

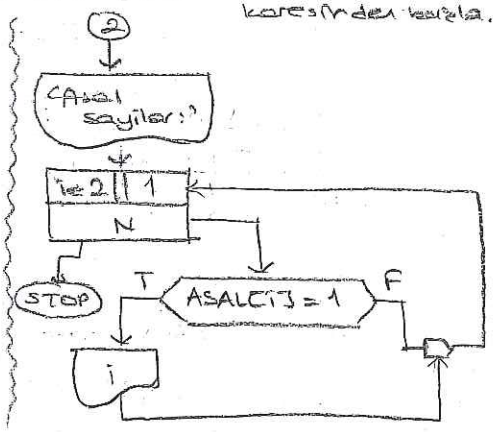
$\sqrt{N}$ 'e kadar gitmek yok. 7 → 49 yok. 11 → 121 yok. eleyebiliyorsun. karesinden kontrol.

$\sqrt{N}+1$

karesinden önce kaç kalan olacak

while





döngüdeki while ne (1'ler olduğunda

Asal olan sayı fonksiyonu

karesinden sonrakileri ele

DÖNEM ÖDEVİ

Market kasası simülasyonu.
(kuyruk)

[K1] [K2] [K3] [K4]

Bir markette 4 tane kasa var.
Başlangıçta hiçbir kasada müşteri yok.
Bir random sayı üreteci kullanılarak her adımda önce kasalara doğru gelen müşteri olup olmadığı belirleniyor. Eğer müşteri varsa yine random olarak kuyruktaki işlem ne kadar süreceği belirleniyor.
(sepettteki eşya da random belirleniyor)
Her müşteri en kısa kuyruğa yerleştiriliyor ve aynı zaman dilimlerde kuyruğun en önündeki müşterinin kuyruktaki zamanı 1 azaltılarak süresi dolan müşteri kuyruktan çıkartılıyor, onun yerine arkasındaki müşteri alıyor. Bu simülasyonu yapan sistemi tasarlayınız.

Sistem zamanı

Döngüde her adımda → random $X_0 = random(10)$; 0 → N-1, 1 → N-1

1 ile 9 arası sayı üretir.

ona kendine geldiğinde işlem ne kadar sürecek

Seçenekler
1. Önce müşteri var mı yok mu yoksa sadece kuyruğun en önündekiler
2. İşlem ne kadar sürecek [5]

Sadece en önündekiler azalır

en kısalarda iki durumu

K1	K2	K3	K4
0 ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~	1 ~~2~~ ~~3~~	~~5~~	4
müşteri var mı yok mu	0	4	

1 azalır yerleştirince
(her kuyruktaki işlem 1 azaldı)

Müşteri eklenince azalıyor
beklerken azalıyor.

Toplam iş yapanın gözü ile

ekrana göre → her adımın sonunda (pointerlar ile yapılır)
nasıl şekilde
kuyruğa göre

ekrana basmak → önceki adım müşteri varsa yerleştirildi, yoksa yok

t anında →

	K1	K2	K3	K4
	4	3	5	9
∞	9	5	4	1
Süre olarak en kısa		1		

e sıfır ile yerleştir.

random'u ikilik olarak yıl olan

0 - 1000 mod 2 yapsan $X_1 = random(1000) \bmod 2$;

çıkan bir 1, bir 0 (kuyruk tanımı ile çalıştı)

SO - 4

müşteri geldiğinde de sıra 1 azalacak (anlaşılsın)

yeteri kadar alan al.

ve kuyruğa yet ki ayrılan kuyruğa gire, sıranı gelsin kişisi alınırı

queue simülasyonu

stack

FIFO

first in first out

LIFO

last in first out

~~3~~	5	9	4	

~~3~~ 9 4

baş, son

baş ↓ 9 4

son

1 2 3 4 5 6

~~3~~	~~4~~	5	7	8	1

Circular queue

(dairesel kuyruk)

kapalı bir daire gibi

effective bir çözüm değil.

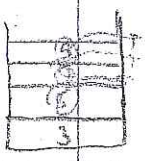
4 kuyruktan çıktı

baş = 3

son = 1

modülü başa dönüşümü sağlıyor

ekranda şöyle ama göstereceğim



X := A + B * C - D

stack kullanılır

kuyruk	baş	son
3	1	1
3,4	1	2
3,4,5	1	3
3'ün işi bitti		
4,5	2	3
4,5,8	2	5
4,5,8,1	2	6
4,5,8,1,9	2	1

(bunu yapanak 1 yeri serbest bırakıyorum)

baş 2 de 1 serbest

5, 10 ve 10 dikkat

* Her kuyruk en fazla n kişi almaktadır. Bundan kuyruk dolu ise o kuyruk kısa bile olsa müşteri alınmamaktadır.

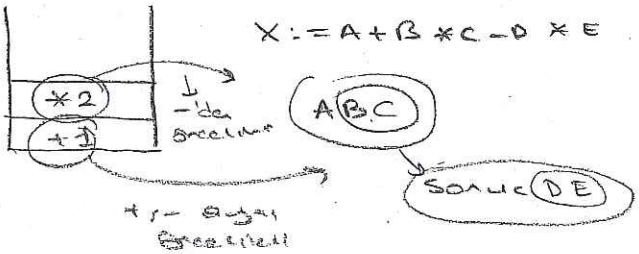
(Alınmayan 2. kasaya?)

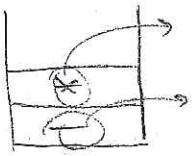
1 br zamanda

kasaya al. paralel

paralel ise

o kısa ya ekle, en önceleri 1 ayrılır





öncelikşöle sıralı geçiş kadar benzetiyor.

	1	2	3	4
N				

Teslim Tarihi final günü (15 Ocak) pzt.

teslim sa günü

ÖDEV RAPORU şöyle %20

yaptığım rapor anlatımı şöyle

1 sayfa → ana problemin tanımı
(varsa) şekilde)
yaptığın işin tanımı

→ arkasına blok diyagramınızı 1 sayfa
bitirmiş yapılan iş.
(panel akış diyagramı)
(kuyruğu nasıl çözdün,
ne kullandın nasıl bulunur)
ondeki kuyruğu 1 dakikan
yol geçen ve kuyruğa ekledim
↑
nasıl bulunda
nerede ekledin.

panel akış diyagramına
şöyle yaptığımı iki
gösterdim.
Ötel (detaylı) akış diyagramı
detaylı verdim
sana da
paralel ekle code
Comment satırıyla
değişken satırında
//
değişkenlerin yerlerini
açıklamaların yazı.
(deklare edilen)

Arkasına her (sayfa) yaptığımız iş anlatıyorum.

bitirmeden --

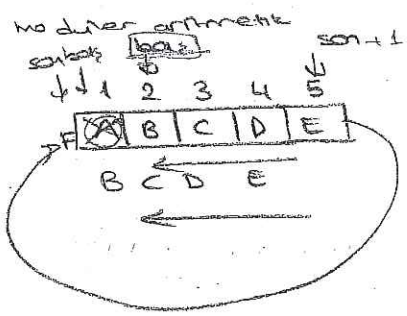
kertim başlıyacağım iş.

(programa bakıldığında program anlaşılır olsun)

(Kuyruğu kuyruğa yapısında ulaşmaçlar 75 satır)

Yönlü, çıkılı yapılı ilerlerler.

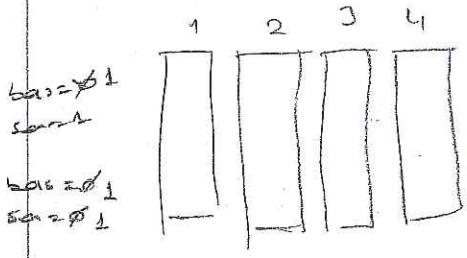
modüler aritmetik



random(5)
0-4

modüler aritmetik

Son = son + 1 5 ile 6 olmasın.
mod 5



bas, son dizisi

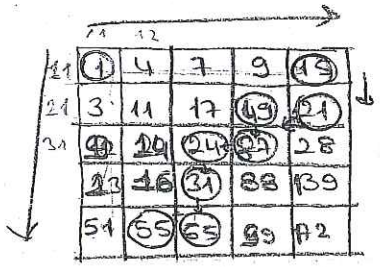
Pop 1 x

4 değil 40 kişi olsa?

if yavaş çalışır.
and → 2 if

4 kişi tam dolu, boş kontrolü

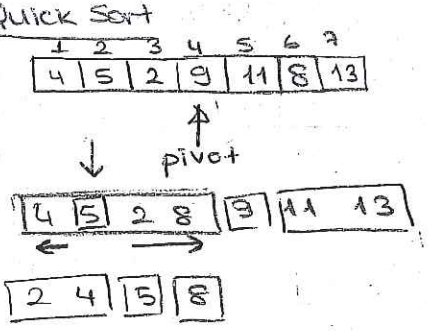
Bir matriste her satır ve sütun kendi içinde sıralıdır. Dizisinde verilen bir elemanın yerini en uygun şekilde bulan algoritmayı tasarlayınız.



x = 55

Ya baştan ya sondan bakarsa

Quick Sort



Genel Adım

pivotun solundakiler daha ortanın pivot, solda sağ
sağındakiler " " " "

rekürsif
recursive

1) 7 el.
2) 4 el.
3) 2 el.

herbirinin daha parçasının bası ve sonu değişiyor.

n elemanlı bir dizinin $\frac{n}{2}$. elemanını pivot kabul ederek pivottan küçük elemanları dizinin baş tarafına, büyük elemanları da sağ tarafına yerleştiren algoritmayı tasarlayınız. (Elemanların sıralı yerleştirilmesi zorunlu değil)

1	2	3	4	5	6	7	8	9	10	11	12	13
11	5	9	13	8	17	10	2	4	13	16	7	14

10 8

1	2	3	4	5	6	7	8	9	10	11	12	13
11	~~15~~	9	~~13~~	~~8~~	17	10	2	4	13	16	~~7~~	14
~~10~~	9	~~15~~	7	15		11					13	
7		8	10									

15<10

adr	i	
2	2	
3	3	→ 9 ↔ 15
4	4	
	5	8 ↔ 15
	6	
	7	
	8	
	9	
	10	
	11	
5	12	7 ↔ 13
	13	

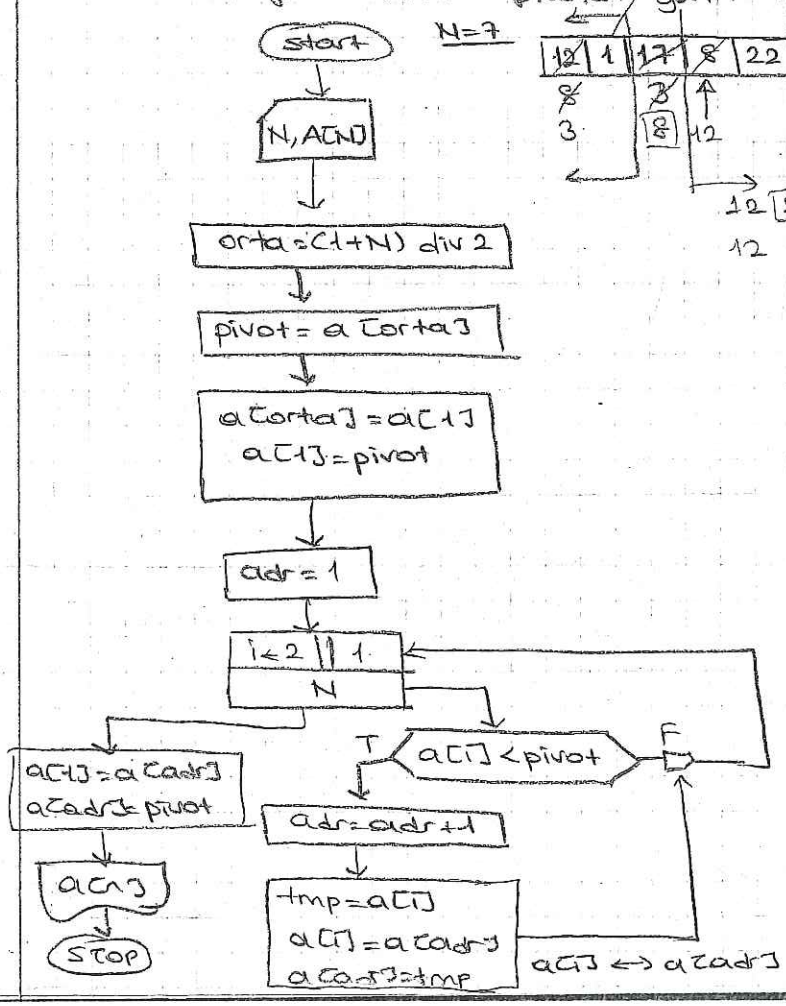
Kontrol daha pahalı bir işlem.

en son bulduğum kıyaslıkla pivotun yerini değiştiriyorum.

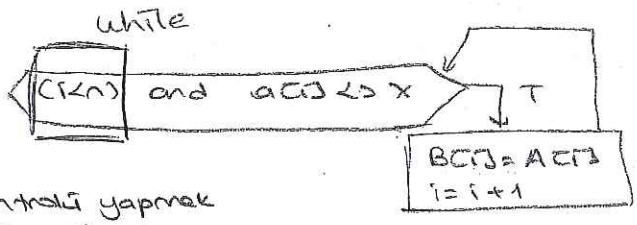
N=7

~~12~~	1	~~17~~	~~8~~	22	~~3~~	9
~~8~~		~~3~~			17	
3		$\boxed{8}$	12			
				12	$\boxed{22}$	17 9
				12	9	17 $\boxed{22}$

adr	i	değişim
1	2	
2		1 ↔ 1
	3	
	4	
	5	
	6	
3		17 ↔ 3
	7	



(a) #



kontrolü yapmak zorundasın.