

# Search algorithms

Sorted arrays

Unsorted Arrays

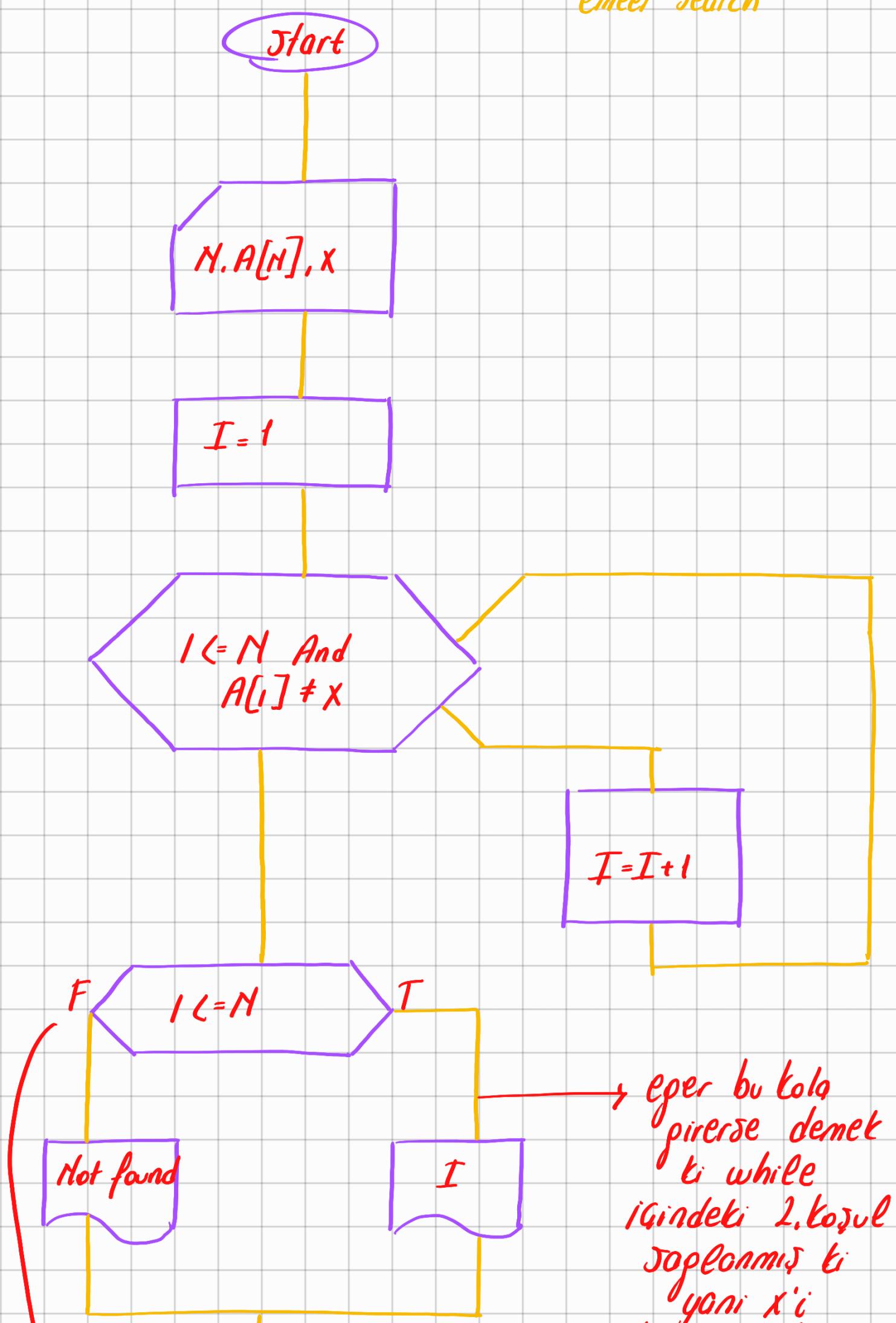
- Linear Search
- Binary Search
- Jump search
- Interpolation Search
- Exponential Search

5	1	2	3	6	4
---	---	---	---	---	---

$$N=6$$

$$\begin{array}{l} X=5 \rightarrow 1 \\ X=8 \rightarrow X \end{array}$$

linear search



bu ömür demet-  
tir

Stop

eğer bu kola girerse  
demek ki x'i bulamamış  
demektir

I      A[1]      x      N

1	5	3	6
2	1	3	6
3	2	3	6
4	3	3	

1	5	4	6
2	1	1	1
3	2	1	1
4	3	1	1
5	6	1	1
6	4	4	1

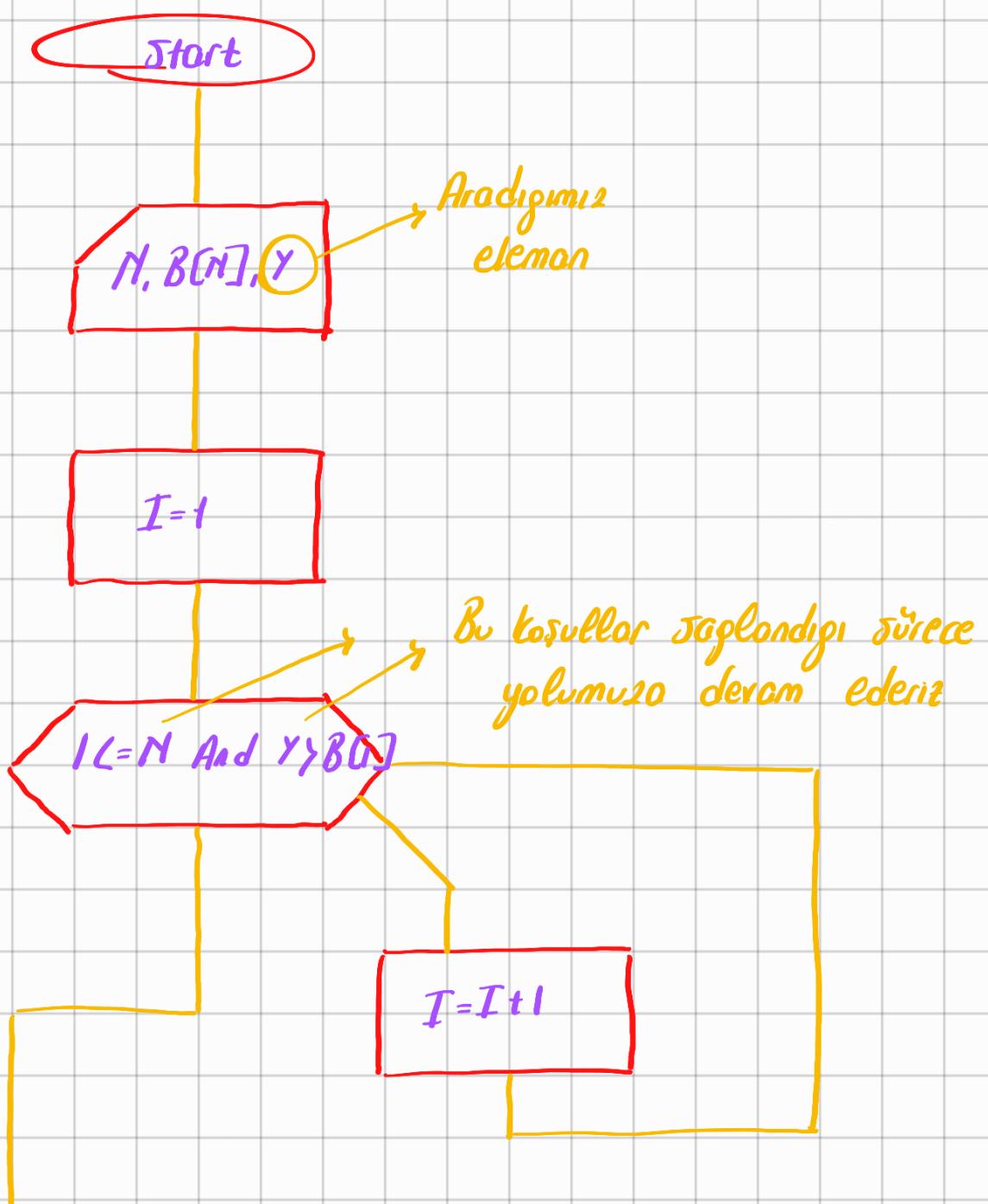
7	7
---	---

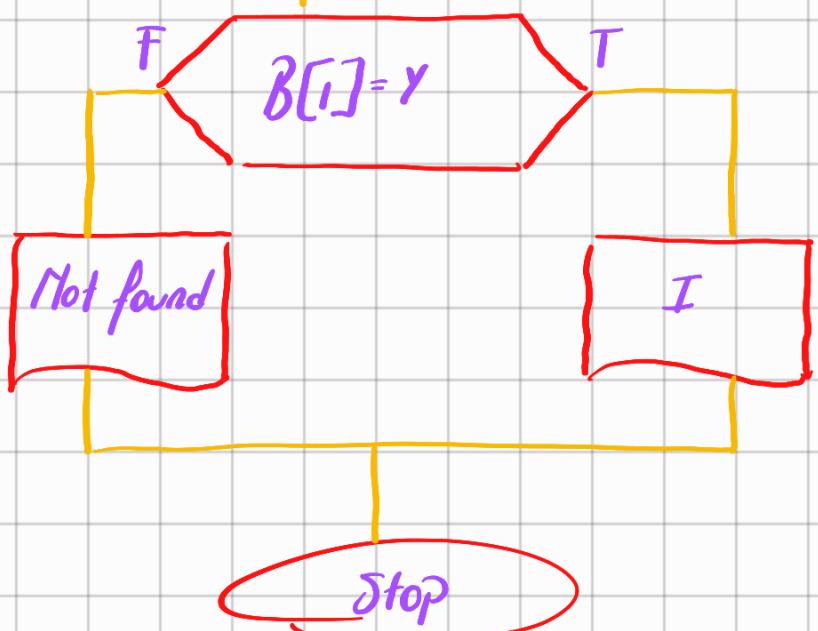
Not found durumu  
göntü dizi sınırları  
aşmış

5	1	2	3	6	4
---	---	---	---	---	---

Verilen bir elementin varsa yerini yoksa bulunmadığı bilgisini ekranı yorduran algoritmanın okus diagramını çiziniz (sırada bir düzide)

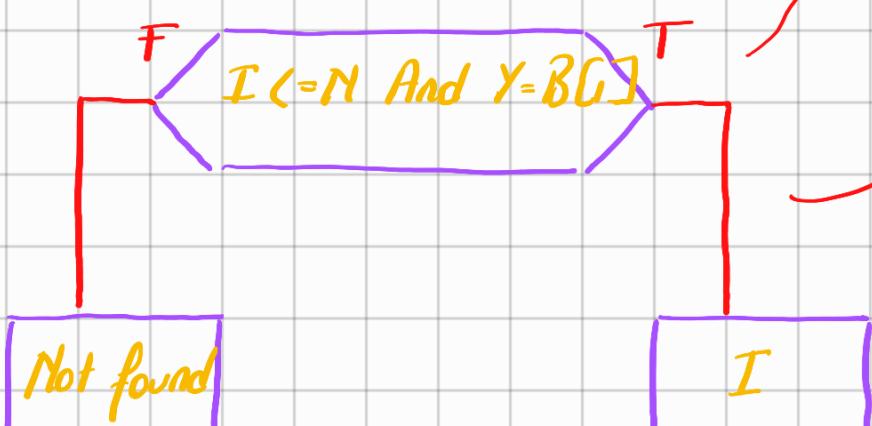
1	2	3	4	5	6	7
1	3	5	8	12	25	32





$1 \checkmark$  } Bu ikisi  
 $32 \checkmark$  } içiń galışır  
 $35 X$  → omo bu ifade içiń algoritma galışmaz  
 Bu algoritma elementin dizideki elementlərdən böyük oldugu durum içiń galışmaz

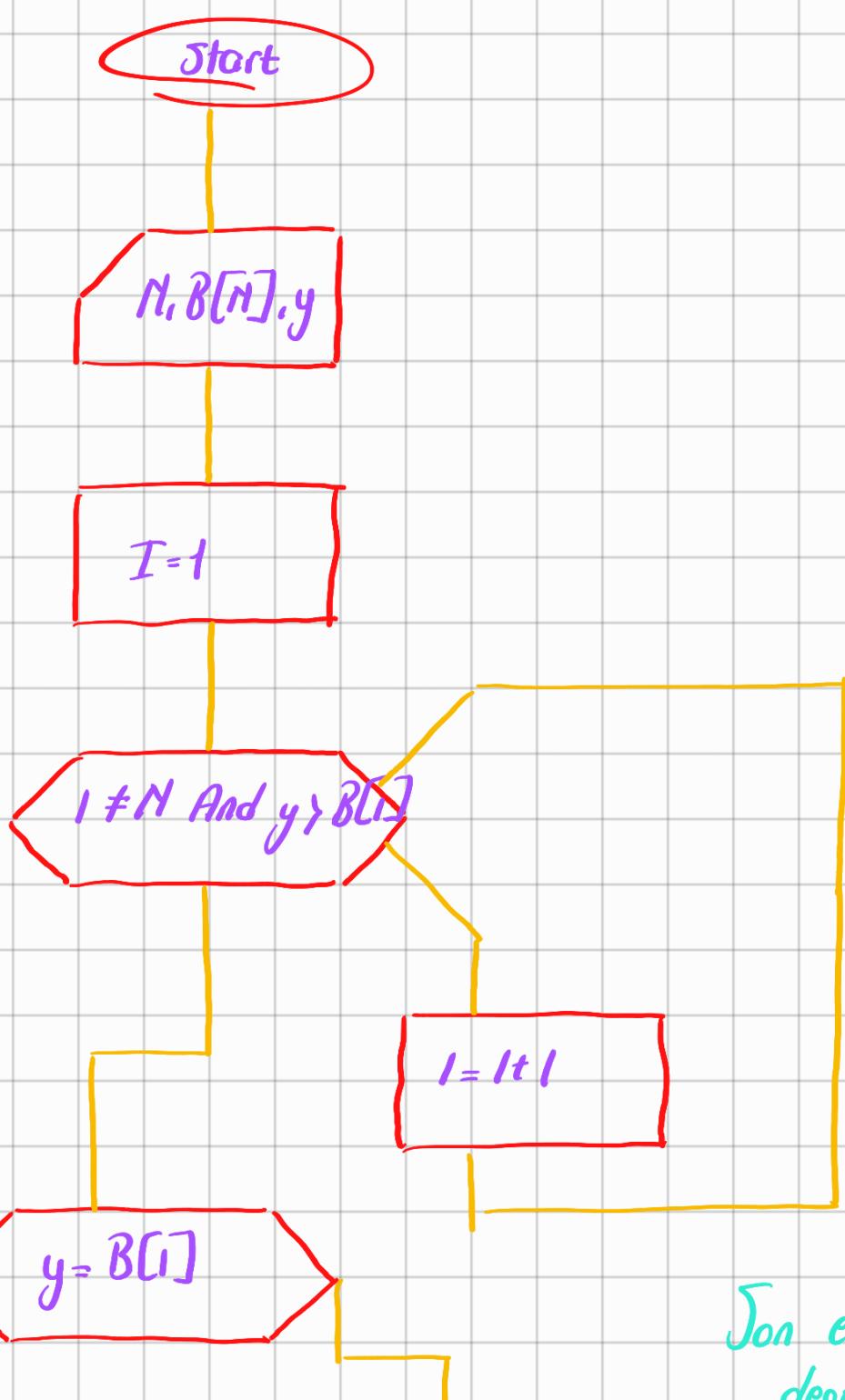
$B[8] = Y$   
 durumu yapıyor Gündə biziñ dizimiz + elementlə



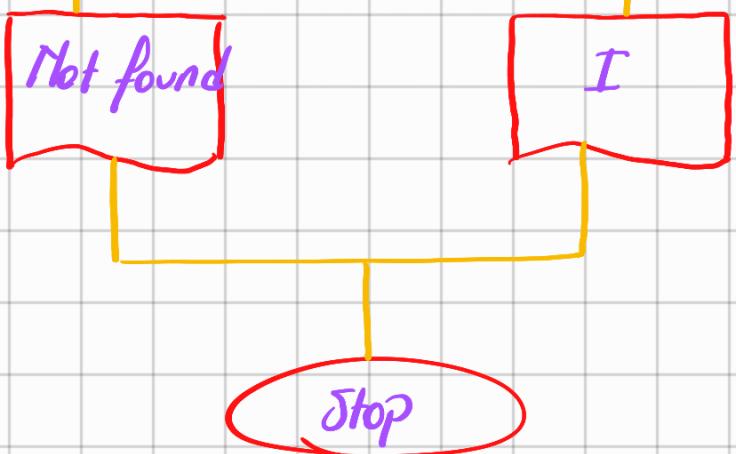
Böyle extra bir koşul ekleyerek galışmabiliriz



Peki bu if'in içine fazladan bir şart koymadan algoritmonu nasıl yaparız



Son elemeni while'de  
değer if'de kontrol



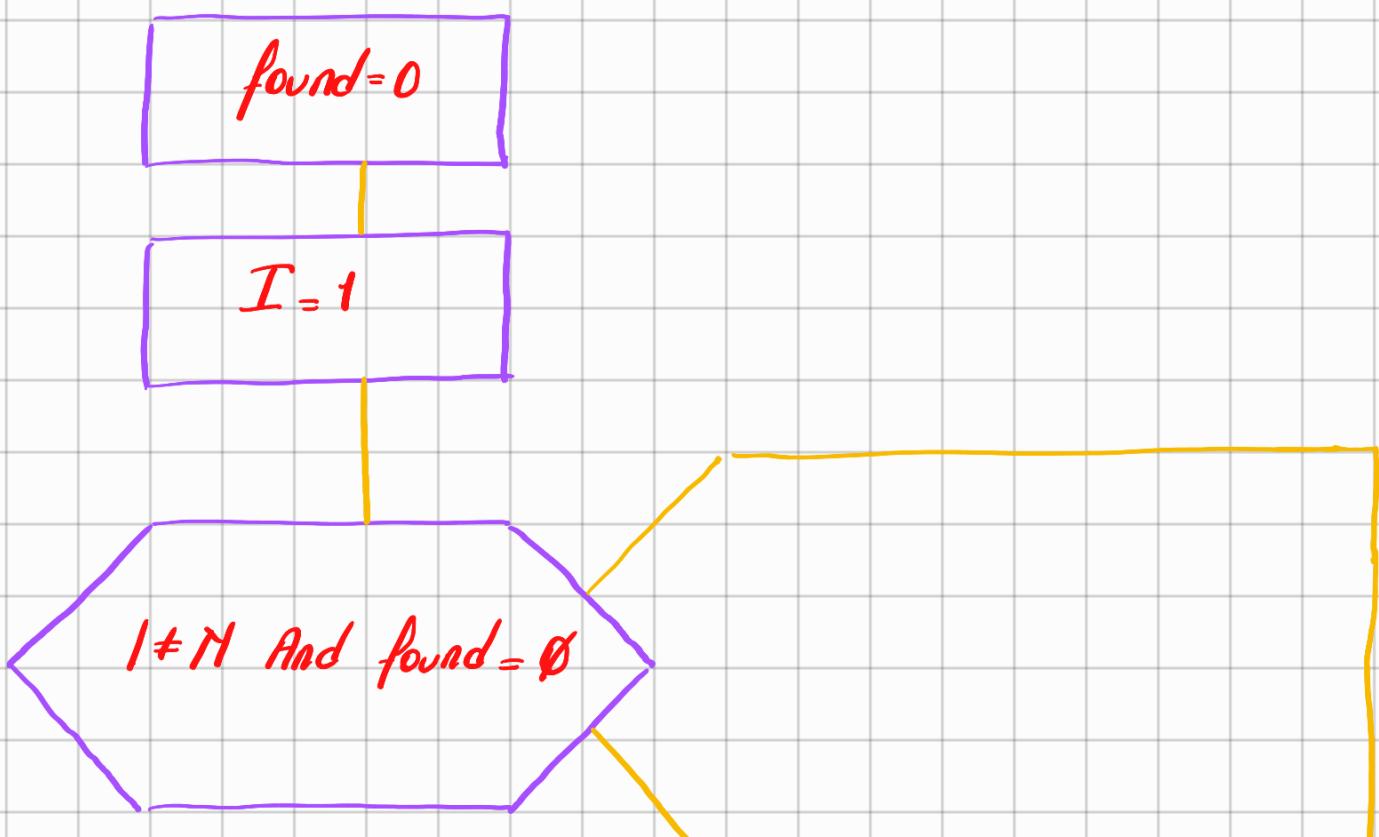
*... yine de ...  
ediyoruz Bu nedenle  
dizinin dizini  
akmuyoruz*

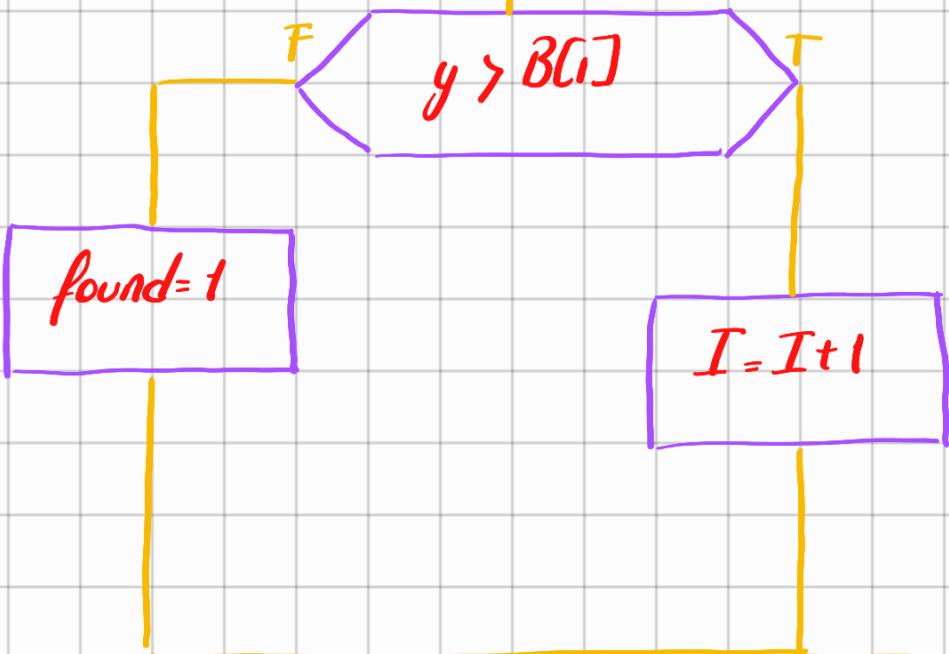
1 VV  
4 X V  
32 VV  
35 VV

*zahladan sort eklemeden  
de istedigimiz algoritma-  
yi gizmis olduk*

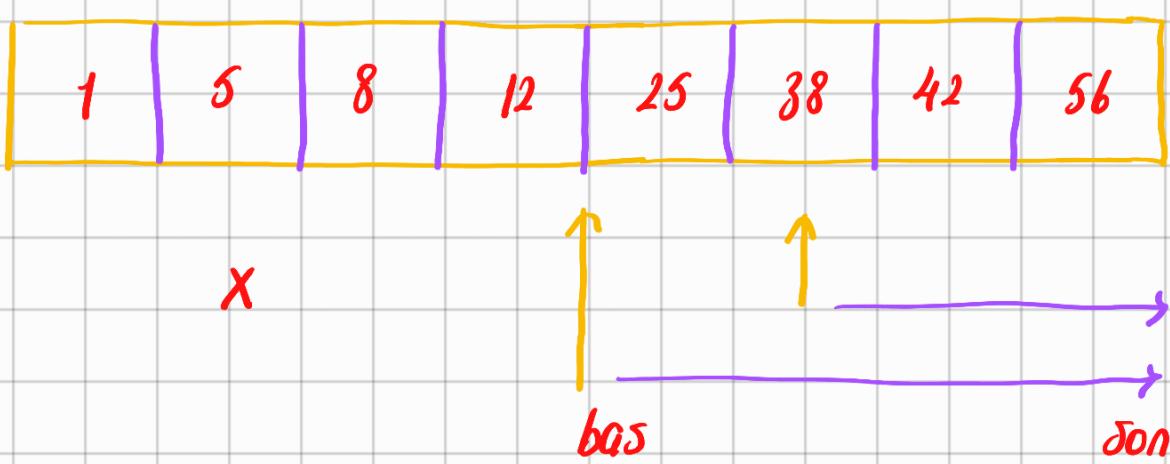
Bu algoritma farklı bir formatta  
da yopulabilirdi

Algoritmanın while durumunu böyle de  
yopabiliyor





Binary Search algoritması (elemlor burda sıralı olmalar gereklidir)  
 $(\log_2 N)$



Bu dizinin ortasından başlantz atiyorum aradığum sayı 42  
 8. kezde dizinin 1. yarısını elec ve 2. yarısını bittiğine göre

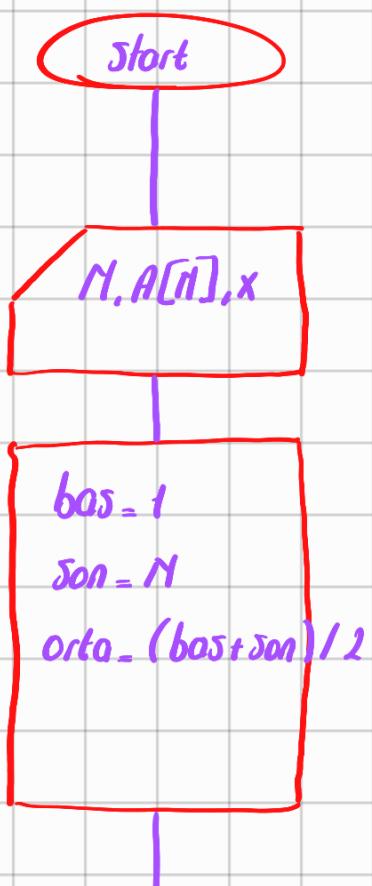
zumuruzun 1. yarısını da ve 2. yarısını da kuralım olsun  
38'e gelir ve ona bakarsın bakın bu sayıda da büyük olsa-  
man 38'inde şapka bakarsın

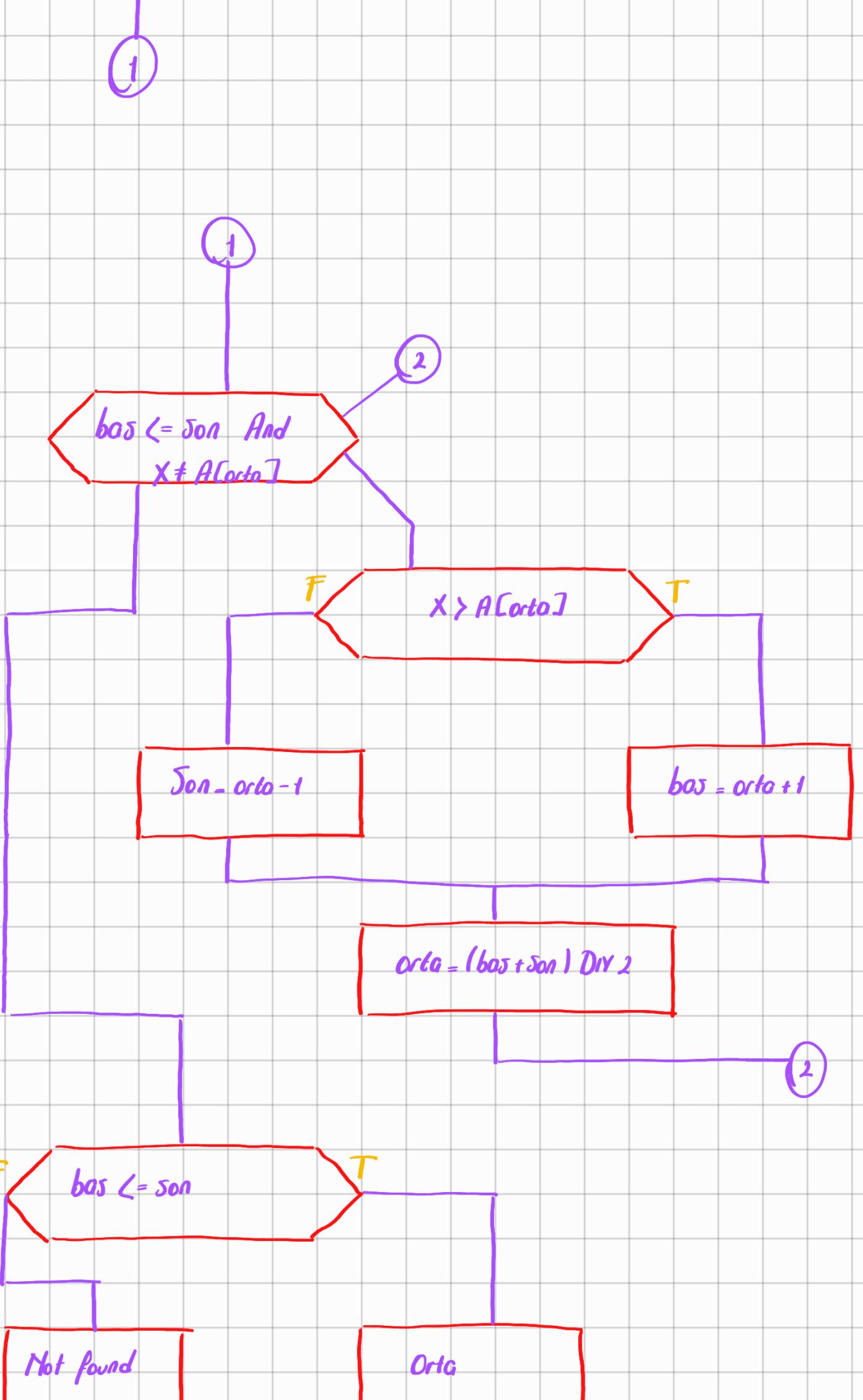
Linear search karmaşıklığı  
 $N$

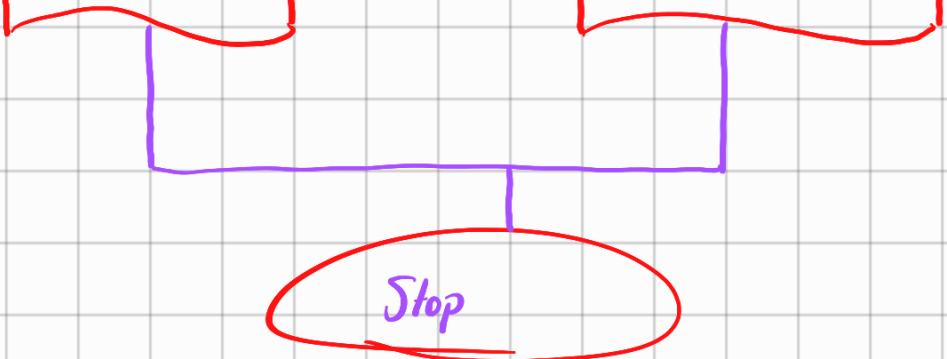
Binary search karmaşıklığı  
 $\log_2 N$

Yani Linear Search'de 100 adında yapabığın  
isi Binary Search'de 7 adında yapabilsin

## Algoritması



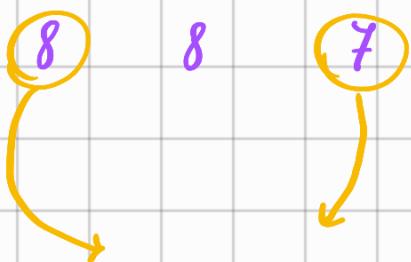




**Bas**   **Orta**   **Son**   **N**   **X**   **A[Orta]**

1	4	8	8	42	12
5	6	8			38
7	7	8		42	42

8        8        8                  43        56



Burda bas  $\leq$  son  
 koluna gider ve  
 konsul saqlanmadigi  
 icin false koluna  
 gider ve not found  
 diye gitir

time Complexity (worst case)

lineer search

Binary search

found	$O(N)$	$O(\log_2 N)$
Not found	$O(N+1)$	$O(\log_2 N + 1)$
	$N/100$	7

1000	10
------	----

Sıralı bir diziyse while kullanmak lazımdır

Sırasız bir diziyse for kullanmak daha mantıklı

Direct Address table

hash table

Sırasız bir dizide binary search uygulanmak için sıralama gereklidir

( $\rightarrow O(N \log N)$ )

( $\rightarrow$  Sonra bunu binary search uygulamak  $O(\log N)$ )

yani

$O(N \log N) + O(\log N)$

→ Büyüktür olan obrur  
 $O(N \log N)$

Sırasız bir dizide binary  
search uygulamının  
mooğyeti budur

işaretli tam sayı sisteminde

1 ve 0'larından oluşan 2'li  
sayı sisteminde sayıla-  
rin binary karşılık-  
ları verilmiştir

1	0	1	0	0	1	1	0
1	0	1	0	0	1		
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1

Bunların onluk sayı sisteme-  
mindeki karşılıklarını  
bir dizide yazdırın  
Algoritmanın okus  
diagramını çiziniz

		127	-120		1
--	--	-----	------	--	---

$$\begin{array}{r} 10001000 \\ 01110111 \\ \hline + \end{array}$$

$$01111000$$

64 82 16 8

-120

en soldaki bite bokoruz 1 iđe 256'dan aikorur  
ve ekisi isaretini koyoruz o iđe direkt  
hesaplaruz

Bunun kodunu yaz

---

Jump Search

1 5 8 12 96 172 251 255 3000  
↑      ↑      ↑

Belirli adimlarla atluyarak elemani ariyordunuz

Sonra arıyorum 96 gitti 12 ve 251 arasında bu iki yer  
arasında linear search yaporsun

Atlama aralıklarını dizinin uzunluguna göre belirlesin

$\sqrt{n}$  bize atlama sayımızı temsil ederse en optimum  
dizeyde elemanı ulaşma sonuna kadar oluyoruz

Bu algoritmonun karmaşıklığı  
 $O(\sqrt{n})$