Prolog3

Listeler

- Liste: elemanlar dizisi
- Örnek liste : ann, tennis, tom, skiing
- Prolog ‘da ifadesi:

[ann, tennis, tom, skiing]

Listelerin gösterimi (1)

- Boş bir dizi

  [ ]

- Boş olmayan bir dizi
  - İki öğeden oluşur
    - İlk öğe : *head* liste'nin başı
    - İkinci öğe : *tail* listenin geri kalanı
    - Örneğimizdeki dizi için Head ve Tail :
      - Head :

        ann
      - Tail :

        [tennis, tom, skiing]

Listelerin gösterimi (2)

- **Head|Tail gösterimine örnekler**

Liste	**[Head|Tail] değerleri**
[a, b, c, d, e]	Head = [a] Tail = [b, c, d, e]
[book, table, pen]	Head = [book] Tail = [table, pen]
[a,b,[c,d]]	Head = [a] Tail = [b,[c,d]]
[clock]	Head = [clock] Tail = []
[]	No head no tail

Listelerin gösterimi (3)

- head herhangi bir prolog objesi olabilir.
- Tail liste olmak zorunda.
- head ve tail özel bir gösterimle liste yapısı haline getirilirler:

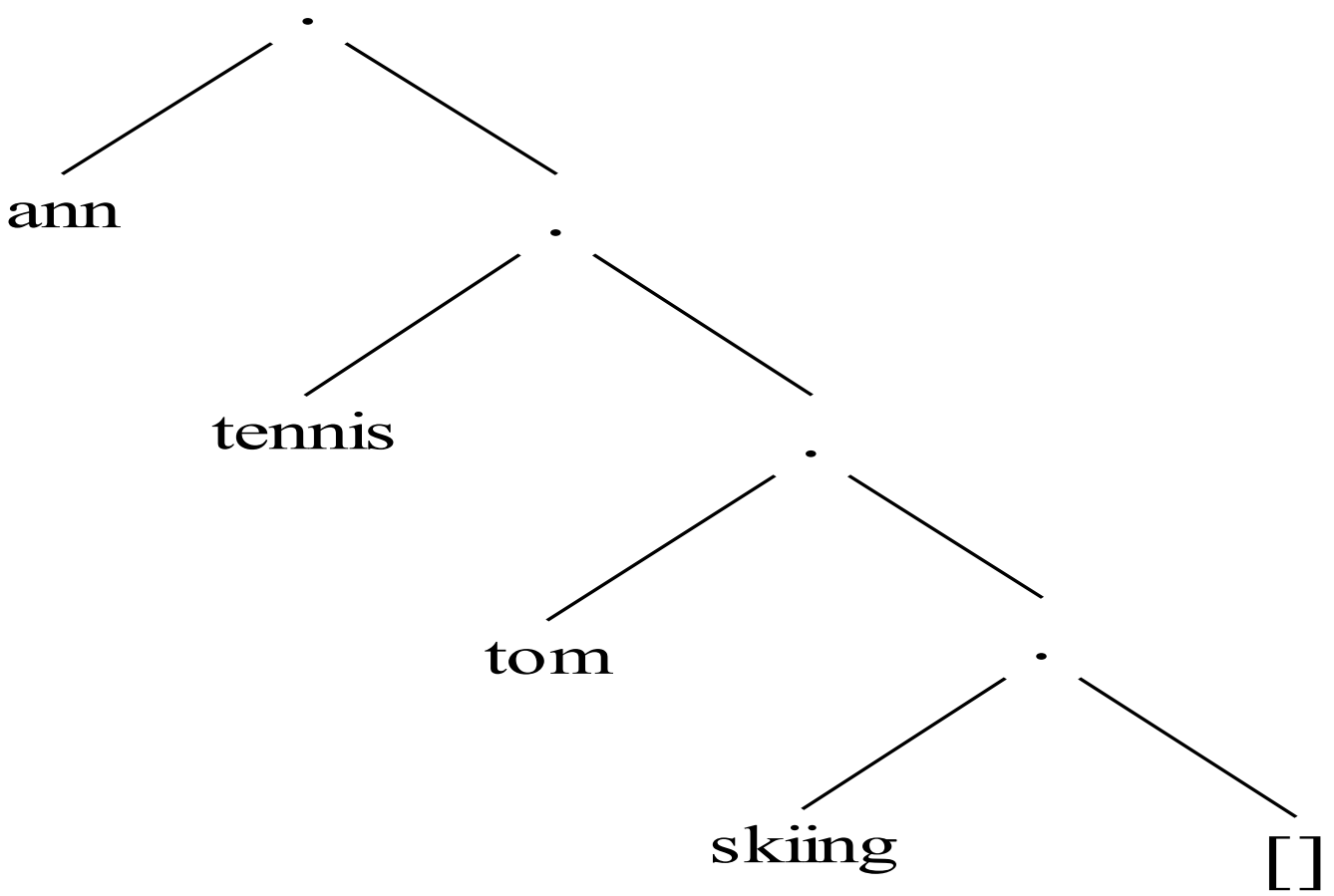
  .(Head, Tail)

- Yukarıdaki gösterimdeki Tail yine bir listedir.

Listelerin gösterimi (4)

– İlk örneğimiz aşağıdaki şekilde yazılabilir :

.(ann, .(tennis, .(tom, .(skiing, []))))



Listelerin gösterimi (5)

- Boş liste bütün listelerin sonunda vardır:

  [skiing] = .(skiing, [])

- Liste gösteriminde nokta ve parantezli ya da köşeli parantezli notasyon kullanılabilir.
- Arka planda listelerin işlenmesi ağaçlarla yapılır ancak programın çıkışında listeler köşeli parantezlerle gösterilir.

```
?- List1 = [a, b, c],
List2 = .(a, .(b, .(c, []))).
List1 = [a, b, c]
List2 = [a, b, c]

?- Hobbies1 = .(tennis, .(music, [])),
Hobbies2 = [skiing, food],
L = [ann, Hobbies1, tom, Hobbies2].
Hobbies1 = [tennis, music]
Hobbies2 = [skiing, food]
L = [ann, [tennis, music], tom, [skiing, food] ]
```

Listelerin gösterimi (6)

– Aynı dizinin farklı yazılışları

[a, b, c] = [a | [b, c]]
= [a, b | [c]]
= [a, b, c | []]

Listelerle İşlemler

```
/*bir listenin elemanlarını yazdırma */
print_list([]).
print_list([Head|Tail]):-
        write(Head),
        write(' '),
        print_list(Tail).
```

- ?- print_list([9,7,3]).
- 9 7 3
- Yes

- ?- print_list([9,7,[3,6,8]]).
- 9 7 [3, 6, 8]
- Yes

```
/*bir dizinin eleman sayısını bulmak*/
size([],0).
size([H|T],N) :-
    size(T,N1), N is N1+1.
```

- ?- size([34,6,4,3],H).
- H = 4 ;
- No
- ?- size([34,6,[4,6,[2,1],3],3],H).
- H = 4 ;
- No

/*bir dizinin eleman sayısını bulmak*/

- size([34,6,4,3],N) :- size ([6,4,3],N1), N is N1+1 N1=3 → size([34,6,4,3],4)
- size([6,4,3],N) :- size ([4,3],N1), N is N1+1 N1 N1= 2 → size([6,4,3],3).
- size([4,3],N) :- size ([3],N1), N is N1+1 N1= 1 →size([4,3],2).
- size([3],N) :- size ([],N1), N is N1+1 N1 = 0 → size([3],1).
- size ([],N).
- N= 0

```
/*bir dizinin eleman sayısını bulmak*/
size([],0).
size([H|T],N) :- size(T,N1), N is N1+1.
size([H|T],N) :- N is N1+1, size(T,N1). (?)
```

?- size([34,6,4,3],H).

- H = 4 ;
- No
- ?- size([34,6,[4,6,[2,1],3],3],H).
- H = 4 ;
- No

?- size([2,4,5],H).
ERROR: Arguments are not sufficiently instantiated

```
/*bir elemanın, listenin elemanı olup olmadığını bulma*/
member(Element,[Element| _ ] ).
member(Element,[ _ |Tail] ) :-
    member(Element,Tail).
```

- ?- member(4,[6,4,8]).
- Yes
- ?- member([5,6],[6,[5,6],8]).
- Yes
- ?- member(5,[6,[5,6],8]).
- No

- **/*bir elemanın, listenin elemanı olup olmadığını bulma*/**

- **member(3, [4,3,5,6,7]) → member(3, [3,5,6,7]) --> Kural biri sağladı**
- **member(3, [4,5,6,7]) → … member(3, [7]) → member(3, [])**

Bir liste, bir başka listenin altkümesi midir?

- sublist(X,L) doğrudur eğer X in tüm elemanları L'nin de elemanı ise.
  - member(X,[X|_]).
  - member(X,[_|R) :- member(X,R).

  - subset([],_).
  - subset([X|R],L) :- member(X,L), subset(R,L).

sum(Xs,N)
N, Xs listesindeki rakamların toplamı

- listetopla([X|[]],X).
- listetopla([H|T],R):-
  listetopla(T,G),
  R is G+H.

```
?- listetopla([10,2,4,4,7],G).
G = 27 ;
No
```

- listetopla([10,2,4,4,7],R) :- listetopla([2,4,4,7],G), R is G+10 G= 17 → listetopla([10,2,4,4,7],27)

- listetopla([2,4,4,7],R) :- listetopla([4,4,7],G) , R is G+ 2 G=15 → listetopla([2,4,4,7],17)

- listetopla([4,4,7],R) :- listetopla([4,7],G) , R is G+ 4 G= 11 → listetopla([4,4,7],15)

- listetopla([4,7],R) :- listetopla([7],G) , R is G+ 4 G=7 → listetopla([4,7],11)

- listetopla([7],G) G=7

sum1(Xs,Ys)

Xs is [x1,x2,...,xn] ve Ys, Xs in her elemanının bir fazlasını içerir. Ys(i)=Xs(i)+1.

- ?

- ?- Sum1([2,3,4],H)
- H= [3,4,5]

Büyükten Küçüğe Sıralı mı?

- s([_]).
- s([X|[Y|T]]):-s([Y|T]),X>Y.

- s([_]).
- s([X,Y|T]):-s([Y|T]),X>Y.

```
/*bir dizinin ilk elemanını silmek*/

removefirst([],[]).
removefirst([Head|Tail],Tail).

?- removefirst([8],H).
H = [] ;
No
?- removefirst([8,7,5],H).
H = [7, 5] ;
No
?- removefirst([[4,5],7,5],H).
H = [7, 5] ;
No
?- removefirst([],H).
H = [] ;
No
```

Listenin İlk N elemanını silmek

- trim(N,L,L1) doğrudur eğer L1, L'nin ilk N elemanı silinmiş hali ise.
- trim(0,[],[]).
- trim(0,[H|T],[H|T]).
- trim(N,[_|T],L) :- N > 0, M is N - 1, trim(M,T,L).
  - ?- trim(3,[1,4,5,6,7,8,9],U).
  - U = [6, 7, 8, 9] ;

- ?- trim(3,[1,4,5,6,7,8,9],L) →
- trim (2,[4,5,6,7,8,9],L) →
- trim (1,[5,6,7,8,9],L) →
- trim (0,[6,7,8,9],L)
- trim (0,[6,7,8,9],L) → trim (0,[6,7,8,9],L) = trim (0,[6,7,8,9], [6,7,8,9])

Listeden istenilen elemanı silmek

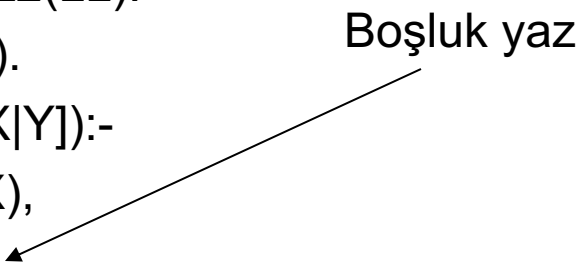
- **del(X,[X|Tail],Tail).**
- **del(X,[Y|Tail],[Y|Tail1]):-del(X,Tail,Tail1).**

- ?- del(a,[1,a,3,7,8],H).
- H = [1, 3, 7, 8] ;
- No
- ?- del(a,[1,a,3,a,a],H). ?
- H = [1, 3, a, a] ;
- H = [1, a, 3, a] ;
- H = [1, a, 3, a] ;
- No

Listeleri Yazdırmak

```
listeyaz([]).
listeyaz([X|Y]):-
    write(X),
    nl,
    listeyaz(Y).
?- listeyaz([2,4,5]).
2
4
5
Yes
```

```
listeyaz2([]).
listeyaz2([L|LL]):-
      satiryaz(L),
      nl,
      listeyaz2(LL).
satiryaz([]).
satiryaz([X|Y]):-
      write(X),
      tab(1),
      satiryaz(Y).
?- listeyaz2([[2,3,4,5],[4,5]]).
2 3 4 5
4 5
Yes
```



Çeviri

```
means(0,zero).
means(1,one).
means(2,two).
means(3,three).
means(4,four).
means(5,five).
means(6,six).
means(7,seven).
means(8,eight).
means(9,nine).

translate([],[]).
translate([H1|T1],[H2|T2]) :-
  means(H1,H2),
  translate(T1,T2).
```

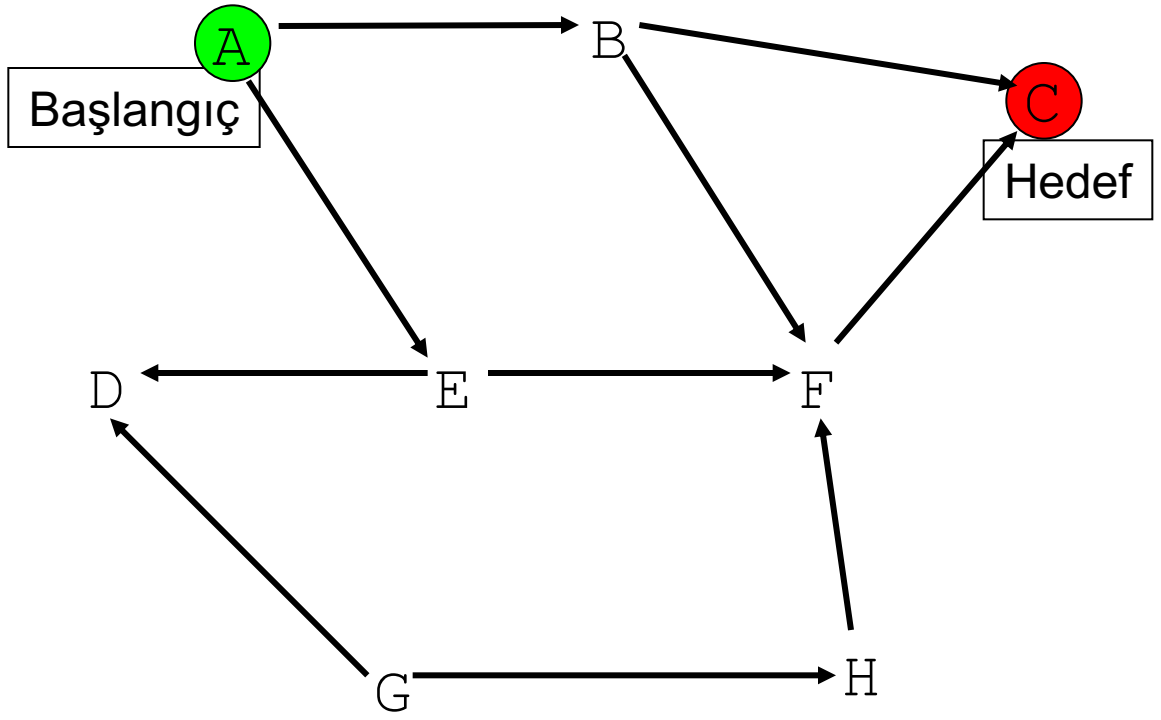
```
?- translate([1,2,3],H).
H = [one, two, three] ;

?- translate(H,[zero,one,nine]).
H = [0, 1, 9] ;
```

Yol Bulma

```
link(g,h).
link(g,d).
link(e,d).
link(h,f).
link(e,f).
link(a,e).
link(a,b).
link(b,f).
link(b,c).
link(f,c).
```

yollar




```
go(X,X,[X]).
go(X,Y,[X|T]):-
        link(X,Z),
        go(Z,Y,T).
```

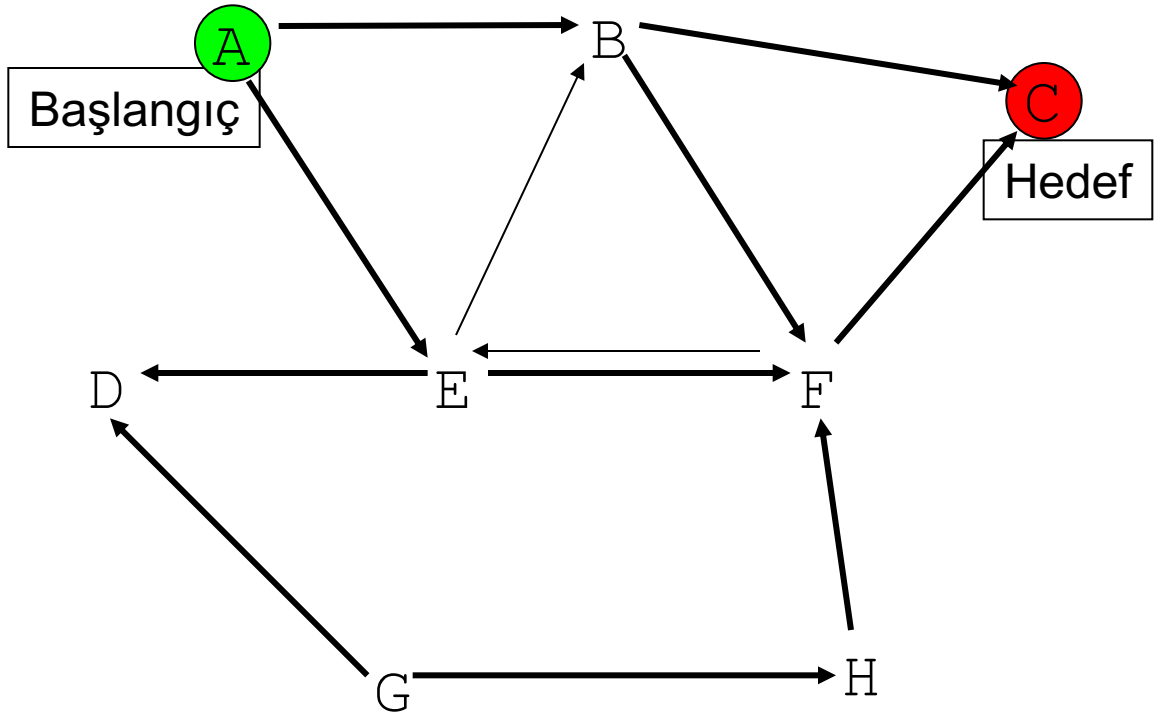
```
?- go(a,c,YOL).
YOL = [a, e, f, c] ;
YOL = [a, b, f, c] ;
YOL = [a, b, c] ;
No
```

Yol Bulma

```
link(g,h).
link(g,d).
link(e,d).
link(h,f).
link(e,f).
link(a,e).
link(a,b).
link(b,f).
link(b,c).
link(f,c).
link(f,e).
link(e,b).
```

yollar

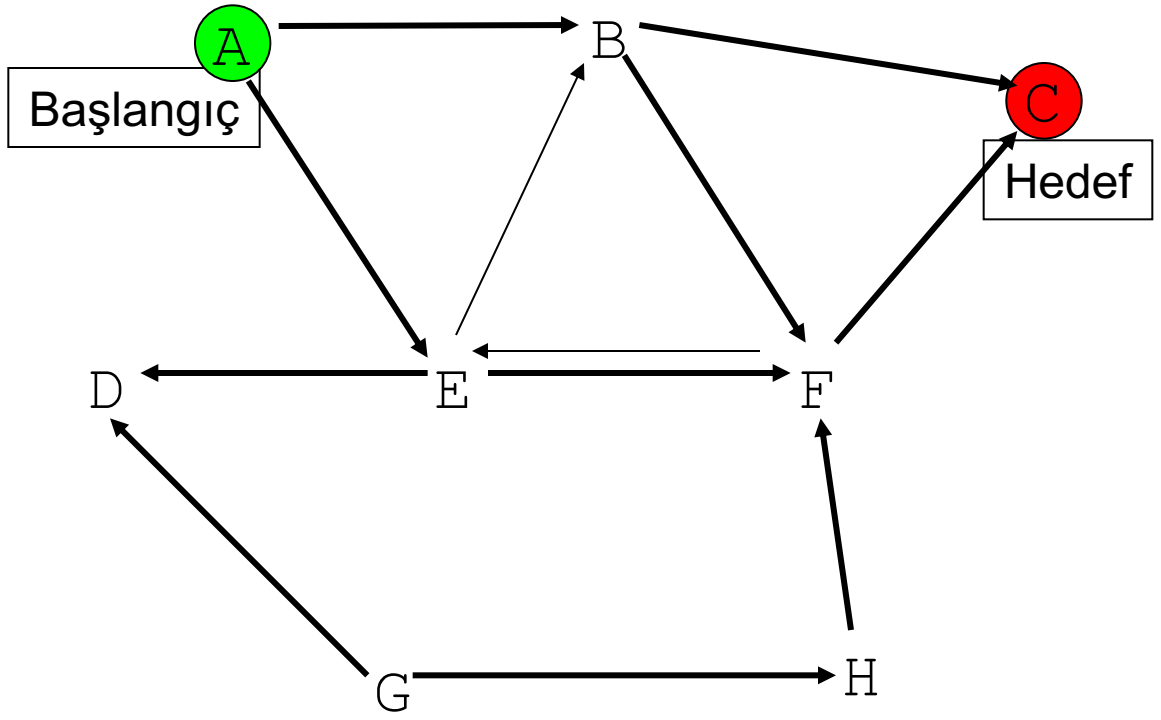
```
go(X,X,[X]).
go(X,Y,[X|T]):-
        link(X,Z),
        go(Z,Y,T).
```




```
?- go(a,c,G).
G = [a, e, f, c] ;
G = [a, e, f, e, f, c] ;
G = [a, e, f, e, f, e, f, c] ;
G = [a, e, f, e, f, e, f, e, f|...] ;
G = [a, e, f, e, f, e, f, e, f|...] ;
…
```

Yol Bulma

```
link(g,h).
link(g,d).
link(e,d).
link(h,f).
link(e,f).
link(a,e).
link(a,b).
link(b,f).
link(b,c).
link(f,c).
link(f,e).
link(e,b).
```



yollar

```
go(X,X,_,[X]).
go(X,Y,Visited,[X|T]):-
    link(X,Z),
    not(member(Z,Visited)),
    go(Z,Y,[Z|Visited],T).
```

```
?- go(a,c,[],G).
G = [a, e, f, c] ;
G = [a, e, b, f, c] ;
G = [a, e, b, c] ;
G = [a, b, f, c] ;
G = [a, b, c] ;
No
```

```
go(X,X,V,[X]):-write(V).
go(X,Y,Visited,[X|T]):-
    arc(X,Z),
    not(member(Z,Visited)),
    go(Z,Y,[Z|Visited],T).

arc([o,B,C, D,E,F], [B,o,C, D,E,F]).
arc([o,B,C, D,E,F], [D,B,C, o,E,F]).

arc([A,o,C, D,E,F], [o,A,C, D,E,F]).
arc([A,o,C, D,E,F], [A,E,C, D,o,F]).
arc([A,o,C, D,E,F], [A,C,o, D,E,F]).

arc([A,B,o, D,E,F], [A,o,B, D,E,F]).
arc([A,B,o, D,E,F], [A,B,F, D,E,o]).

arc([A,B,C, o,E,F], [o,B,C, A,E,F]).
arc([A,B,C, o,E,F], [A,B,C, E,o,F]).

arc([A,B,C, D,o,F], [A,B,C, o,D,F]).
arc([A,B,C, D,o,F], [A,o,C, D,B,F]).
arc([A,B,C, D,o,F], [A,B,C, D,F,o]).

arc([A,B,C, D,E,o], [A,B,o, D,E,C]).
arc([A,B,C, D,E,o], [A,B,C, D,o,E]).
```

1	o	3
4	5	6

6'lı Puzzle

```
?- go([1,o,3,4,5,6],[4,o,3,5,1,6],[],_).
[[4, o, 3, 5, 1, 6],
[o, 4, 3, 5, 1, 6],
[5, 4, 3, o, 1, 6],
[5, 4, 3, 1, o, 6],
[5, o, 3, 1, 4, 6],
[o, 5, 3, 1, 4, 6],
[1, 5, 3, o, 4, 6],
[1, 5, 3, 4, o, 6],
[1, o, 3, 4, 5, 6],
[o, 1, 3, 4, 5, 6]]
Yes
```

Değişken kullanmasaydık kaç geçiş yazmamız gerekirdi?

Kaynaklar

- PROLOG Programming for Artificial Intelligence, Bratko, I., 3rd Edition, Addison-Wesley, 2001