فصل سوم

سید ناصر رضوی

razavi@Comp.iust.ac.ir

۱۳۸۳

مقدمه

- نمایش لیست ها
- عملیات روی لیست ها
- تعریف عملگرها
- عملیات ریاضی

نمایش لیست ها

- تعریف لیست:

هر دنباله متناهی از تعدادی عنصر مانند $a_1, a_2, a_3, \ldots, a_n$ که ترتیب آنها مهم باشد. چنین لیستی در پرولوگ به صورت زیر نمایش داده می شود:

$[a_1, a_2, a_3, \ldots, a_n]$

مثال:

[ann, tennis, tom, skiing]

لیستها یک نوع ساختار هستند و بنابراین در پرولوگ نمایش داخلی آنها به صورت درختی می باشد.

نمایش لیست ها

- انواع لیست:
  - لیست تهی که با [] نمایش داده می شود.
  - لیست غیر تهی
- در مورد لیست غیر تهی می توان آن را متشکل از دو قسمت در نظر گرفت:
  - سرآیند: عنصر اول لیست
  - دنباله: بقیه عناصر لیست
- مثلا در لیست مثال قبل ann سرآیند لیست و دنباله خود یک لیست به صورت زیر است:

[tennis, tom, skiing]

به طور کلی سرآیند هر چیزی می تواند باشد (هر شیء پرولوگ) ، اما دنباله باید خود یک لیست باشد.
این دوقسمت را می توان با یک functor ویژه با هم ترکیب نمود:

.(Head, Tail)

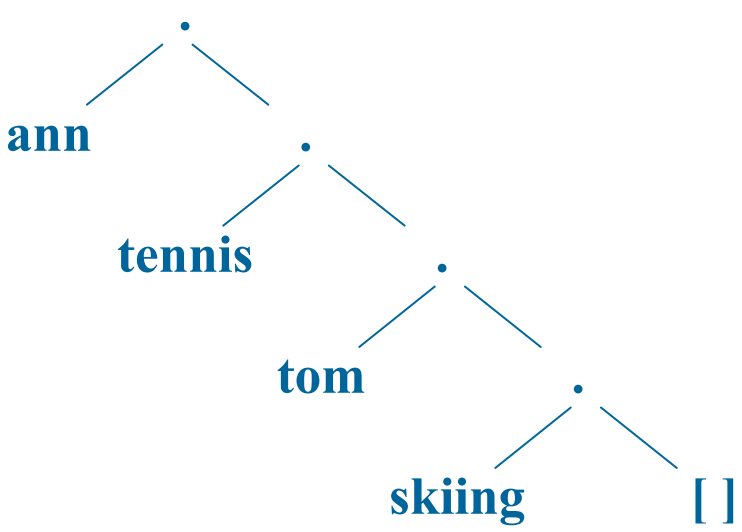
نمایش لیست ها

- چون دنباله یک لیست خود یک لیست می باشد، بنابراین می تواند تهی باشد و یا سرآیند و دنباله خود را داشته باشد.

[ann, tennis, tom, skiing] =

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

[skiing] = .(skiing, [])



نمایش لیست ها

- لیست های تودرتو

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

نمایش لیست ها

اگر لیست L به صورت زیر باشد:

L = [a, b, c]

می توانیم بنویسیم:

Tail = [b, c] , L=.(a, Tail)

روش دیگر:

L = [a| Tail]

به طور کلی:

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

نمایش لیست ها

- خلاصه:
- لیست یک ساختار داده ای است که یا تهی می باشد و یا از دو بخش **سرآیند** و **دنباله** تشکیل می شود.
- در پرولوگ با لیست ها به عنوان یک نوع خاص از درختهای دودویی رفتار می شود. در پرولوگ برای افزایش خوانایی روش دیگری نیز برای نمایش لیست ها وجود دارد:

[Item1 , Item2, …]

[Head| Tail]

[Item1 , Item2, …| Others]

عملیات روی لیست ها

- از لیستها می توان برای نمایش مجموعه ها استفاده نمود با این تفاوت که ترتیب عناصر در مجموعه ها مهم نیست.
- برخی عملیات روی لیست ها:
  - بررسی اینکه آیا یک شیء خاص در لیست وجود دارد.
  - اتصال دو لیست
  - حذف عناصر موجود در لیست
  - اضافه نمودن عناصر جدید به لیست

عملیات روی لیست ها

- پیاده سازی عمل عضویت:

member(X, L)

member(X, [X| Tail]).
member(X, [Head| Tail]) :-
member(X, Tail).

عملیات روی لیست ها

- مثال:

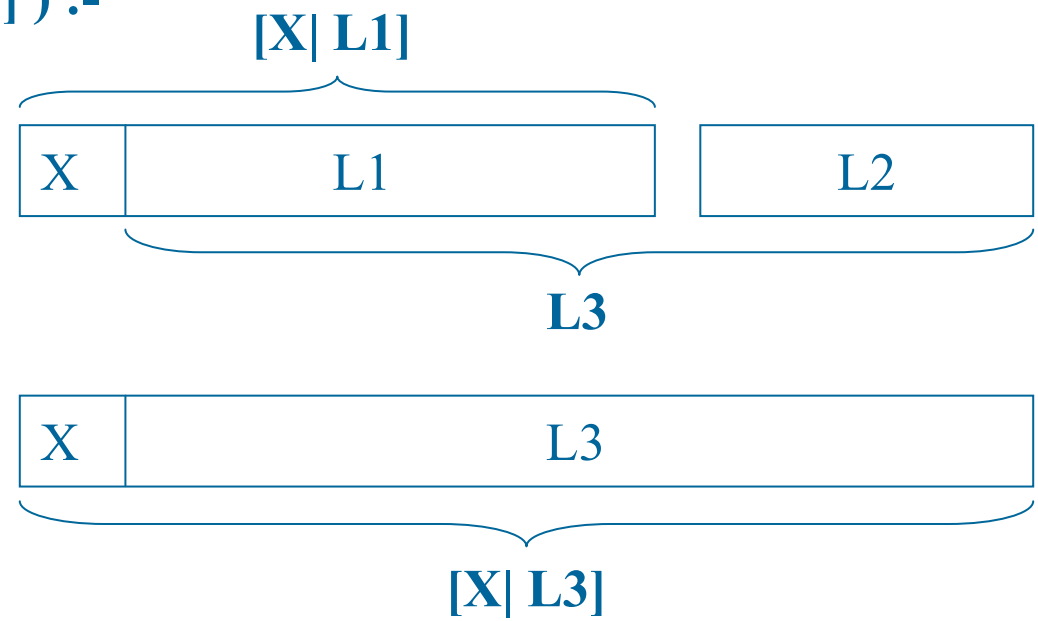
```
?- member( b, [a, b, c] ).
   yes
?- member( b, [a, [b, c] ] ).
   no
?- member( [b, c], [a, [b, c] ] ).
   yes
?- member( X, [a, b] ).
   X = a;
   X = b;
   no
```

عملیات روی لیست ها

- پیاده سازی عمل اتصال لیست ها

conc(L1, L2, L3)

```
conc( [], L, L).
conc( [X| L1], L2, [X| L3] ) :-
    conc( L1, L2, L3).
```



عملیات روی لیست ها

- مثال:

```
?- conc( [a, b, c],[1 ,2,3]  , L).
    L = [a, b, c, 1,1,2,3]
?- conc(L1, L2,[ a, b, c] ).
    L1 = []
    L2 = [a, b, c];
    L1 = [a]
    L2 = [b, c];
    L1 = [a, b]
    L2 = [c];
    L1 = [a, b, c]
    L2 = [];
    no
```

عملیات روی لیست ها

- مثال:

```
?- conc( Before, [may| After],
         [jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec] ).
   Before = [jan, feb, mar, apr]
   After = [jun, jul, aug, sep, oct, nov, dec]

?- conc( _, [M1, may, M2| _],
         [jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec] ).
   M1 = apr
   M2 = jun
```

```
member1( X, L) :-
   conc(L1, [X, L2], L).
```

عملیات روی لیست ها

- اضافه نمودن یک عنصر جدید به لیست

add(X, L, [X| L]).

مثال:

```
?- add( a, [b, c] , L).
   L = [a, b, c]
```

عملیات روی لیست ها

- حذف یک عنصر از لیست

del(X, L, L1)

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

عملیات روی لیست ها

```
?- del( a, [a, b, a, a], L).
    L = [b, a, a];
    L = [a, b, a];
    L = [a, b, a];
    no
```

از del می توان برای اضافه کردن عنصر جدید در مکان دلخواه استفاده نمود:

```
?- del( a, L, [1,2,3] ).
    L = [a, 1,2,3];
    L = [1, a, 2,3];
    L = [1,2, a,  3];
    L = [1,2,3, a  ];
    no
```

عملیات روی لیست ها

- حذف یک عنصر از لیست

به طور کلی می توان عمل درج X در مکانی از List را بصورت زیر انجام داد:

```
insert( X, List, BiggerList) :-
   del( X, BiggerList, L).
```

پیاده سازی عمل عضویت بوسیله del:

```
member2( X, L) :-
   del( X, L, _).
```

عملیات روی لیست ها

- پیاده سازی عمل زیرلیست:

```
sublist( S, L) :-
    con( L1, L2 , L),
    con( S, L3, L2).
```

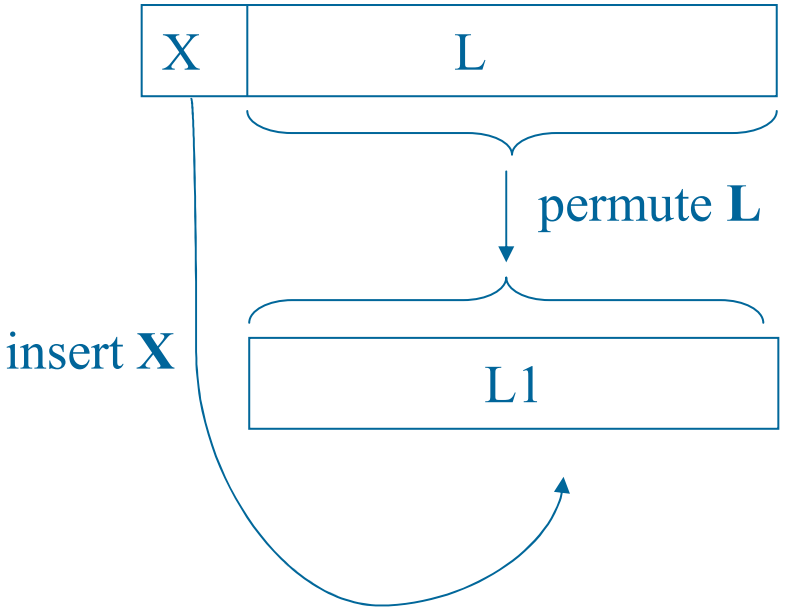
مثال:

```
?- sublist( [c, d, e], [a, b, c, d, e] ).
   yes
?- sublist( [c, e], [a, b, c, d, e] ).
   no
```

عملیات روی لیست ها

- جایگشت ها

```
permutation([], []).
permutation( [X| L], P) :-
    permutation( L, L1),
    insert( X, L1, P).
```



عملیات روی لیست ها

- مثال:

```
?- permutation( [r, b, g], P).
    P = [r, b, g];
    P = [r, g, b];
    P = [b, r, g];
    P = [b, g, r];
    P = [g, r, b];
    P = [g, b, r];
    no
```

عملیات روی لیست ها

- تمرین
- رابطه reverse(L1, L2) را برای معکوس کردن یک لیست تعریف کنید.
- رابطه shift(L1, L2) را برای شیفت چرخشی لیست L1 به اندازه یک واحد به سمت چپ تعریف کنید.
- رابطه palindrome(L) را به گونه ای تعریف کنید که مشخص کند آیا L با معکوس آن مساویست یا خیر.

عملیات ریاضی

عملگر	توضیح
+	جمع
–	تفریق
×	ضرب
/	تقسیم
××	توان
//	تقسیم صحیح
mod	باقیمانده

عملیات ریاضی

```
?- X = 1 + 2.
   X = 1 + 2

?- X is 1 + 2.
   X = 3
?- X is 5/2,
   Y is 5//2,
   Z is 5 mod 2.
   X = 2.5
   Y = 2
   Z = 1
```

برخی توابع ریاضی: sin(X), cos(X), atan(X), log(X), exp(x)

عملیات ریاضی

- عملگر های رابطه ای

عملگر	توضیح
X > Y	بزرگتر
X < Y	کوچکتر
X >= Y	بزرگتر یا مساوی
X =< Y	کوچکتر یا مساوی
X =:= Y	مساوی
X =/= Y	نامساوی

عملیات ریاضی

```
?- 1 + 2 =:= 2 + 1.
   yes
?- 1 + 2 = 2 + 1
   no
?- 1 + A = B + 2.
   A = 2
   B = 1
?- 277*37 >      10000
   yes
?- born( Name, Year),
   Year >= 1980,
   Year =< 1990.
```

عملیات ریاضی

- مثال : محاسبه بزرگترین مقسوم علیه مشترک X و Y:
  - اگر X = Y، آنگاه D = X.
  - اگر X < Y ، آنگاه D برابر است با ب.م.م. X و Y-X.
  - اگر X > Y، آنگاه مانند حالت دوم با تعویض نقش X و Y.

```
gcd( X, X, X).
gcd( X, Y, D) :-
    X < Y,
    Y1 is Y – X,
    gcd( X, Y1, D).
gcd( X, Y, D) :-
    Y < X,
    gcd( Y, X, D).
```

عملیات ریاضی

- مثال: محاسبه طول لیست

```
length( [], 0).
length( [ _| Tail], N) :-
    length( Tail, N1),
    N is N1 + 1.
```

```
?- length( [a, b, [c, d], e], N).
   N = 4
```

عملیات ریاضی

- مثال: محاسبه فاکتوریل N:

```
factorial( 0, F):  - F is 1.
factorial( N, F) :-
    N >0,
    N1 is N -1,
    factorial(N1, F1),
    F is N*  F1.
```

```
?- factorial( 5, F).
   F = 120
```

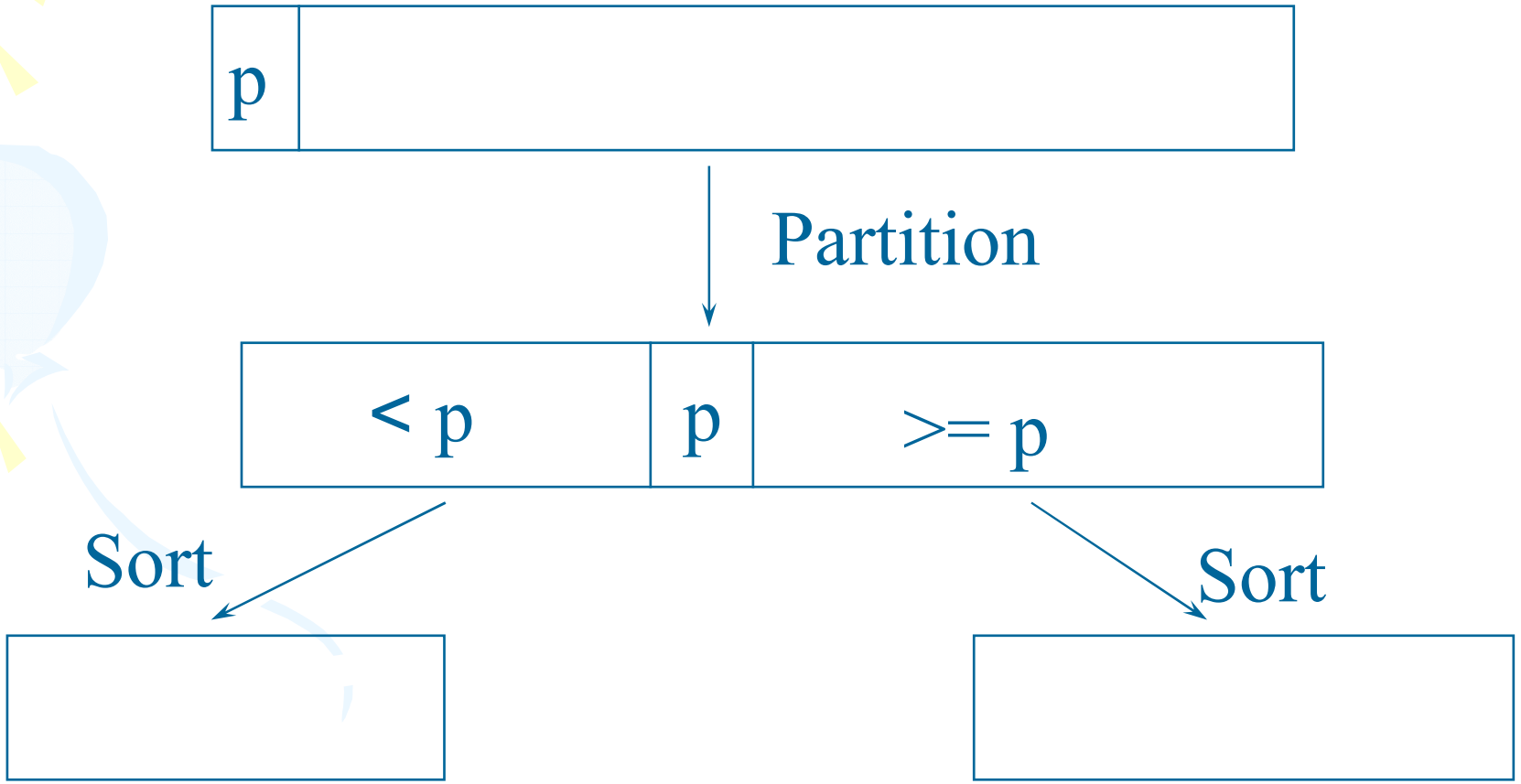
تمرین

۱- رابطه maxlist(List, Max) را به گونه ای تعریف کنید که Max برابر مقدار بزرگترین عدد List شود.

۲- رابطه ordered(List) را به گونه ای تعریف کنید که در صورت مرتب بودن اعداد لیست درست باشد.

۳- رابطه sumlist(List, Sum) را برای محاسبه مجموع عناصر لیست تعریف کنید.

مرتب سازی سریع (*Quick Sort*)



مرتب سازی سریع (*Quick Sort*)

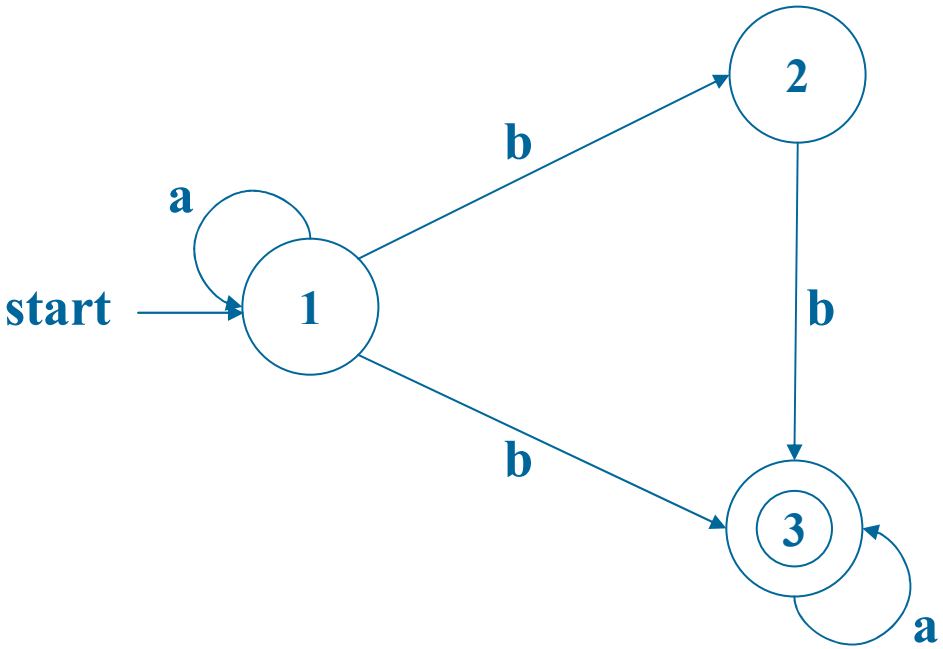
```
quickSort([ ], [ ]).
quickSort( [P | T], Y) :- partition( T, P, L, B),
   quickSort( L, SortLow), quickSort( B, SortBig),
   append( SortLow, [P | SortBig], Y).
partition([ ], P, [ ], [ ]).
partition( [H | T], P, [H | L], B) :-
   H < P, partition( T, P, L, B).
partition( [H | T], P, L, [H | B]) :-
   H >= P, partition( T, P, L, B)
```

شبیه سازی یک *NFA*

```
accept( W) :-
   start( S), path( S, W).
path( S, [ ]) :-
   final( S).
path( S, [H | T]) :-
   arc( S, H, N), path( N, T).
```

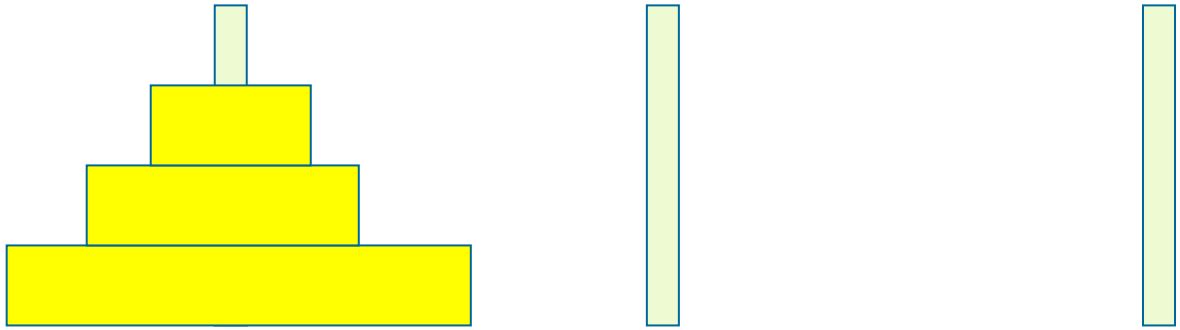
شبیه سازی یک NFA

```
start(1).
final(3).
arc(1, a, 1).
arc(1, b, 2).
arc(1, b, 3).
arc(2, b, 3).
arc(3, a, 3).
```



?- accept([a, b, a]).
yes

برجهای هانوی
(Towers of Hanoi)



- هدف: انتقال دیسکها، در هر لحظه یکی، از میله سمت چپ به میله سمت راست با کمک میله وسط می باشد.
- در هیچ مرحله ای نمی توان یک دیسک بزرگتر را روی یک دیسک کوچکتر قرار داد.

برجهای هانوی
(Towers of Hanoi)

```
hanoi(1, X, Y, Z) :-
   write(‘ Move top disk from ’), write( X),
   write(‘ to ’), write( Y), nl.
hanoi( N, X, Y, Z) :-
   N > 1, M is N-1, hanoi( M, X, Z, Y),
   hanoi(1, X, Y, Z), hanoi( M, Z, Y, X).
```

?- hanoi(3, left, right, middle).

برجهای هانوی
(Towers of Hanoi)

?- hanoi(3, left, right, middle).

Move top disk from left to right
Move top disk from left to middle
Move top disk from right to middle
Move top disk from left to right
Move top disk from middle to left
Move top disk from middle to right
Move top disk from left to right