Data Structure

Dr. Rastgoo

فصل ۵:

لیست پیوندی

لیست پیوندی

لیست پیوندی، ساختمان داده ای پویا است که اشیاء با یک ترتیب خطی در آن قرار گرفته اند. بر خلاف آرایه ، که در آن ترتیب خطی توسط اندیسهای آرایه تعیین می شود، ترتیب در لیست پیوندی بوسیله یک اشاره گر در هر شیء تعیین می گردد. لیست پیوندی بر دو نوع یک طرفه و دو طرفه می باشد.

لیست پیوندی یک طرفه(یک سویه)

لیستی که در آن، هر عنصر فقط آدرس عنصر بعدی را نگهداری می کند. هر یک از گرههای این لیست پیونـدی از دو قسـمت داده و آدرس تشکیل شده که در زبان c به صورت زیر تعریف می شود:

```
typedef  struct  list-node  *list-pointer;
typedef  struct  list-node{
            int            data;
            list-pointer   next;
 };
```

در next، آدرس گره بعدی ذخیره می شود. (به جای next از link نیز استفاده می شود)

- درج و حذف در لیست پیوندی نسبت به آرایه ساده تر است. چون نیاز به جابجایی فیزیکی عناصر در حافظه نمی باشد.
- دسترسی به عناصر آرایه نسبت به لیست پیوندی سریع تر است، چون اندیس هر عنصر در آرایه مشخص است.
- امکان جستجوی دودویی در لیست پیوندی وجود ندارد.

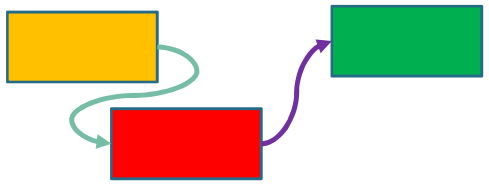
الگوریتم های کار بر روی لیست پیوندی یکطرفه

درج یک گره

تابع زیر یک گره با داده d بعد از یک گره با آدرس مشخص n درج می کند:

```
void   insert ( list-pointer  *first , list-pointer  n){
    list-pointer   t;
    t = malloc(sizeof(list-node));
    t – > data = d;
    if (*first) {
          tnext; – > next = n   – >
          n – >  next = t;
       }
   else{
          t  – > next = NULL:
          *first = t;
          }
}
```

اگر لیست تهی باشد، گره اضافه شده، تنها گره لیست خواهد بود.

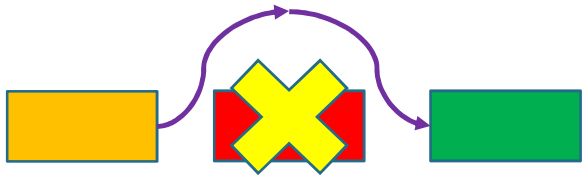


الگوریتم های کار بر روی لیست پیوندی یکطرفه

حذف یک گره

تابع delete **گره واقع در قبل از گره با آدرس مشخص** n **را حذف می کند.**(t : آدرس گره قبل از گره n)

```
void   delete( list-pointer  *first , n , t )
{
 if (t)
      t  - >  next = n - >next;
 else
     *first = (*first)  - >next;
 free(n);}
```



تذکر: اگر t برابر NULL باشد، آنگاه n اولین گره لیست می باشد.

الگوریتم های کار بر روی لیست پیوندی یکطرفه

چاپ داده های لیست

تابع زیر داده های یک لیست با آدرس شروع p را چاپ می کند.

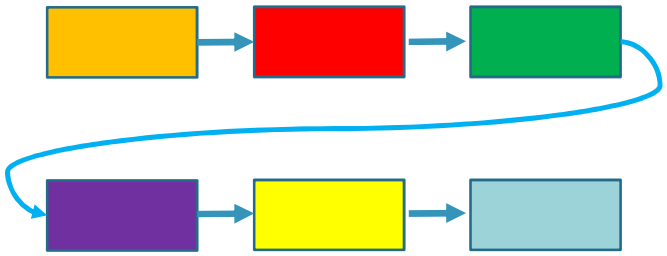
```
void  print-list (list-pointer   *p)
{
  for( ; p ;  p = p – >next )
       cout <<  p – >data;
}
```

الگوریتم های کار بر روی لیست پیوندی یکطرفه

اتصال دو لیست پیوندی به یکدیگر

تابع زیر لیست p2 را به انتهای لیست p1 متصل می کند:

```
list-pointer  concatenate (list-pointer  p1 , p2)
{
   list-pointer  t;
   if  (p1==NULL)  return  p2;
   else{
         if (p2 !=NULL)
         {
           for(t = p1 ; t – >next ; t = t – >next);
           t – >next = p2;
         }
    return  p1;
  }
}
```



اگر شرط p1=NULL برقرار باشد، یعنی لیست p1 موجود نیست و نتیجه همان p2 است و در صورت موجود بودن لیست p2 ، لیست p1 را توسط حلقه تا آخر پیمایش کرده و سپس p2 را به انتهای آن توسط دستور t->next=p2; ، اضافه می‌کنیم.

الگوریتم اتصال دو لیست xو y (x به انتهای y و یا y به انتهای x)، از مرتبه O(max(m,n)) می‌باشد، زیرا در بدترین حالت لیست بزرگتر باید تا انتها پیمایش شود. (mوn طول لیست ها می باشند)

حذف عناصر تکراری در یک لیست (با شبه کد CLRS)

الگوریتم زیر عناصر تکراری در یک لیست را طوری حذف می کند که ترتیب عناصر باقی مانده تغییر نکند. مثلا لیست < 1,2,3,2,1,3,4 > به < 1,2,3,4 > تبدیل می شود.

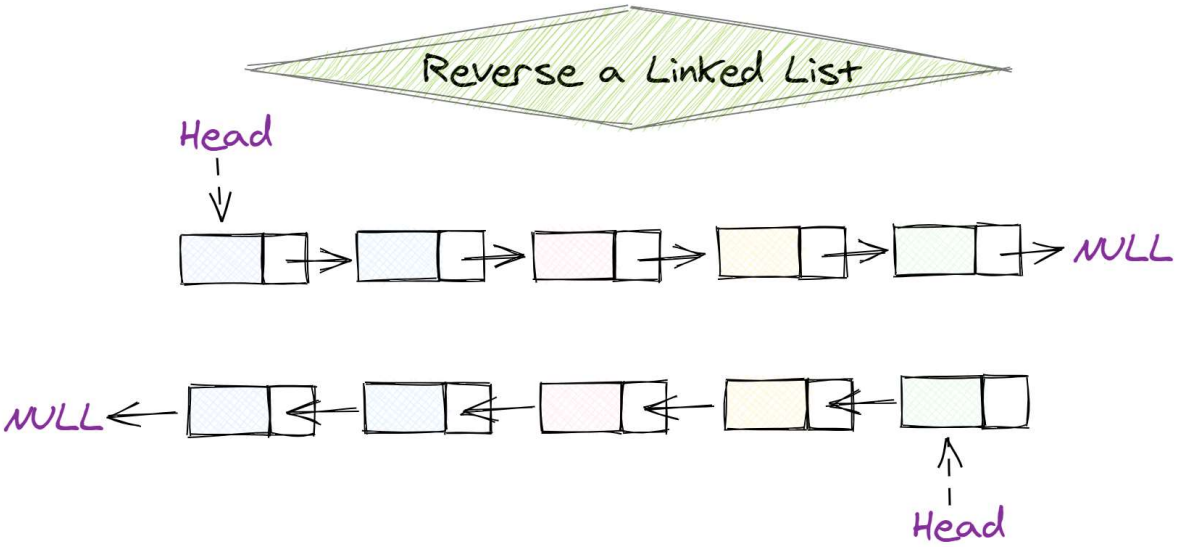
```
PurgeList(L)
1   p ← First(L)
2   while  p< > null
3      do  q ←p
4          while  next[q] < > null
5             do  if  element[p]=element[next[q]]
6                       then   delete-after(L,q)
7                       else   q← next[q]
8          p ← next[p]
```

تذکر: در الگوریتم بالا، تابع **delete-after(L,q)** ، عنصر بعد از **q** را در لیست **L** حذف می کند.

الگوریتم بالا از مرتبه $O(n^2)$ است. می توان لیست را مرتب کرد و سپس با $O(n)$ کـار اضـافی، عناصـر تکـراری را در لیست مرتب حذف کرد. اما لیست نهایی این الگوریتم که عناصرش مرتب هستند، با لیستی که الگوریتم فوق تولید مـی کند، متفاوت است.

وارون کردن یک لیست پیوندی

می توان یک لیست n عضوی را تنها با تغییر اشاره گرهای آن وارون کرد، به طوری که عنصر i ام لیست بعد از وارون سازی عنصر n-i+1 ام لیست جدید شود. این کار را به صورت بازگشتی و غیر بازگشتی می توان انجام داد.

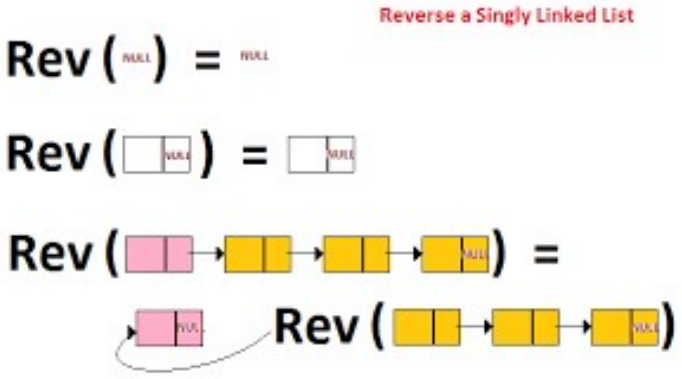


الف- روش غیر بازگشتی

در روش غیر بازگشتی از سه اشاره گر p ، q و r استفاده می کنیم،که به سه عنصر متوالی اشاره می کنند. در زمان اجرا p وq به اولین و دومین عنصر لیستی که تا این مرحله وارون شده اشاره می کنند و r به عنصر بعدی در لیست که هنوز وارون نشده است. ابتدا p تهی است و q به عنصر اول و r به عنصر دوم لیست اشاره می کنند. در هر مرحله، مولفه next عنصر r را به عنصر q تغییر داده و هر سه اشاره گر را به جلو می بریم. با تهی شدن r ، لیست معکوس شده است. این کار در زمان خطی انجام می شود.

ب- روش بازگشتی

رویه بازگشتی reverse(L,p) ، زیر لیست شامل عناصر p به بعد را در لیست L ،وراون شده بر می گرداند. برای وارون کل لیست، p باید برابر first[L] باشد. با فرض اینکه تعداد عناصر از p به بعد در لیست L برابر n باشد،الگوریتم ابتدا عنصر p را کتار می گذارد و بقیه لیست با n-1 عنصر که با q شروع می شود را به صورت بازگشتی وارون می کند و از همان عناصر یک لیست r ایجاد می کند. در نهایت،عنصر p را به انتهای لیست r وصل می کند.



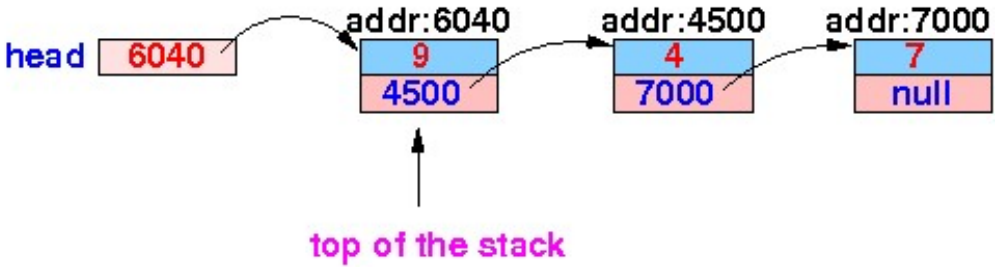
پیاده‌سازی پشته با لیست پیوندی

می توان پشته را با لیست پیوندی پیاده سازی کرد. درج و حذف هر عنصر فقط در ابتدای لیست انجام می شود و محـدودیتی برای تعداد عناصر پشته نداریم.

Representing a stack with a list:

Stack:

List:



اضافه کردن به پشته پیوندی

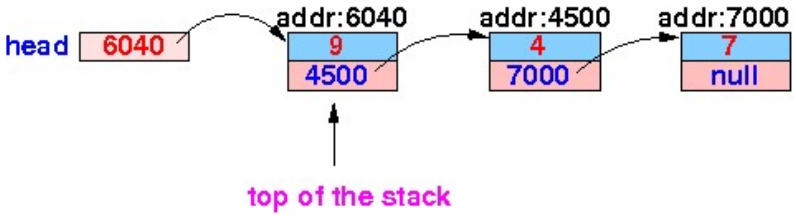
اضافه کردن یک گره به پشته پیوندی، مشابه اضافه کردن یک گره به اول لیست پیوندی است.آدرس شروع پشته پیوندی برابر top می باشد:

```
void  add(stack-pointer  *top , int   a){
   stack-pointer  p;
   p = malloc(sizeof(stack) );
   p – > item =a;
   p – > next = *top;
   *top = p;
}
```

Initial state:

Stack:

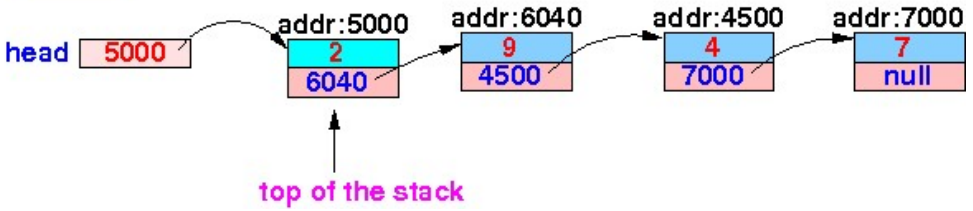
List:



After push(2):

Stack:

List:



حذف از پشته پیوندی

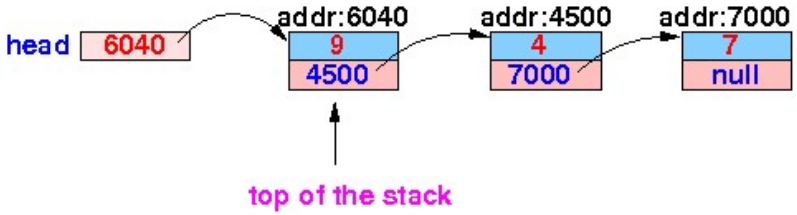
حذف یک گره از پشته پیوندی مشابه حذف اولین گره لیست پیوندی می باشد:

```
int  delete(stack-pointer   *top ){
    stack-pointer  t;
    int  a;
     t =*top;
    if  (t ==NULL)  exit ( );
    else{
            a = t  – >  item;
             *top = t  – >  next;
            free(t);
            return  a;
          }
}
```

Initial state:

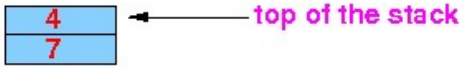
Stack:

List:

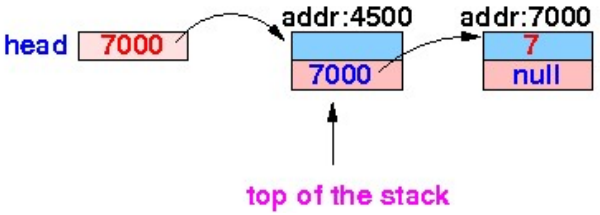


After pop():

Stack:



List:

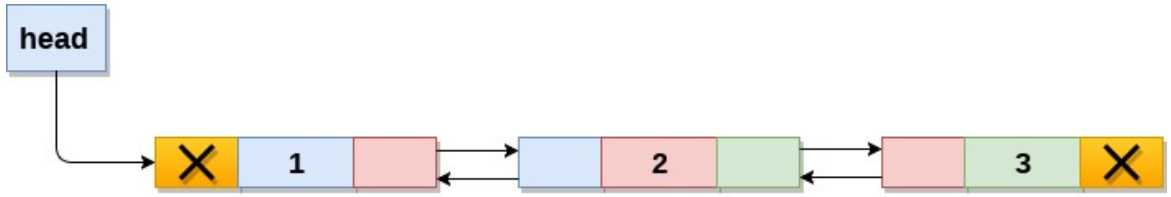


پیاده‌سازی صف با لیست پیوندی

برای پیاده سازی صف Q می توان از یک لیست پیوندی یک طرفه با دو اشاره گر front[Q] (اشاره گر به عنصر اول لیست) و rear[Q] (اشاره گر به عنصر آخر لیست) استفاده کرد. عمل EnQueue یک عنصر به انتهای لیست درج می کند. و عمل DeQueue عنصر اول لیست را حذف می کند و در اختیار قرار می دهد.

لیست پیوندی دوطرفه

در لیستهای پیوندی دو طرفه(دو گانه یا دو سویه)، هر عنصر علاوه بر مولفه های لیست یک طرفه، مولفه prev هم دارد که به عنصر قبلی اش در لیست اشاره می کند.



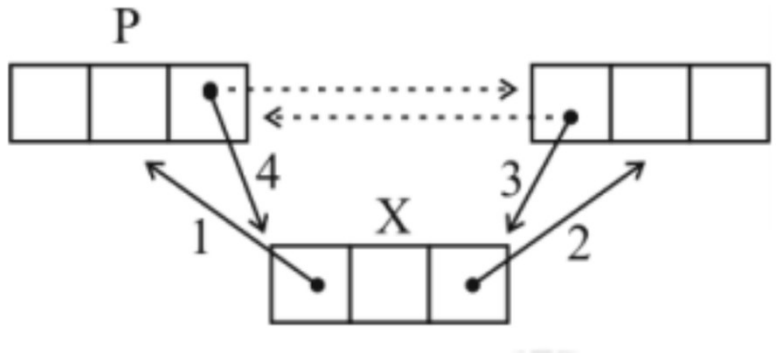
Doubly Linked List

به جای Prev از Llink و به جای next از Rlink نیز استفاده می شود.

اضافه کردن گره

اضافه کردن گره x به سمت راست گره با آدرس مشخص p

```
void   insert(node-pointer  p , node-pointer x){
   x -> prev = p;              (1)
   x -> next = p -> next;      (2)
   p -> next -> prev = x;      (3)
   p -> next = x;              (4)
}
```



می توان ۸ حالت مجاز برای ترتیب شماره ها در نظر گرفت:

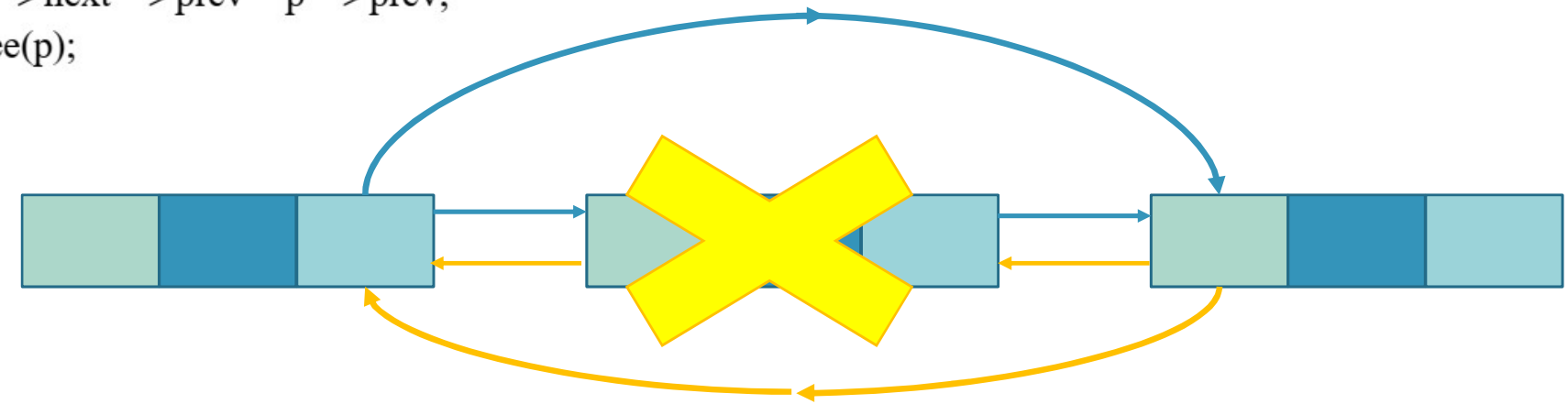
1234 , 2134 , 2314 , 2341 , 1324 , 3124 , 3214 , 3241

باید ابتدا خط ۲و۳ تنظیم شوند و سپس خط ۴.

حذف گره

تابع زیر گره با آدرس مشخص p را از لیست پیوندی دو طرفه حذف می کند. برای این کار کافی است اشاره گر next گره قبل از p ، به گره بعد از p اشاره کند و اشاره گر prev گره بعد از p ، به گره قبل از p اشاره کند.

```
delete(node-pointer  p)
{
   p->prev->next = p->next;
   p->next->prev = p->prev;
   free(p);
}
```



- برای حذف یک گره از یک لیست پیوندی دو طرفه به ۲ تغییر اشاره گر نیاز است.

- برای اضافه کردن یک گره به یک لیست پیوندی دو طرفه به ۴ تغییر اشاره گر نیاز است.

- از مزایای لیست پیوندی دوطرفه نسبت به یک طرفه، می توان ساده شدن عملیات زیر را نام برد:

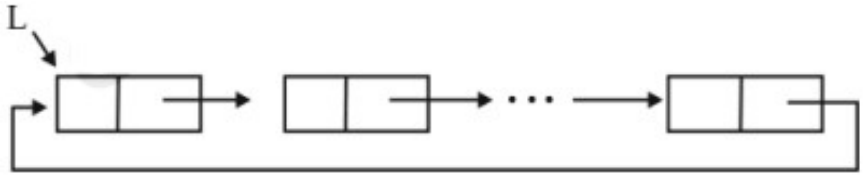
  ۱ـ حذف یک گره با معلوم بودن مکان آن گره.

  ۲ـ اضافه کردن یک گره قبل از گره‌ای با مکان معلوم.

لیست پیوندی حلقوی

لیست حلقوی(چرخشی)، نوعی لیست یک طرفه است که در آن اشاره‌گر آخرین گره، به جای مقدار دهی با NULL به ابتدای لیست اشاره می‌کند.

شکل زیر یک لیست حلقوی یک طرفه را نشان می دهد:

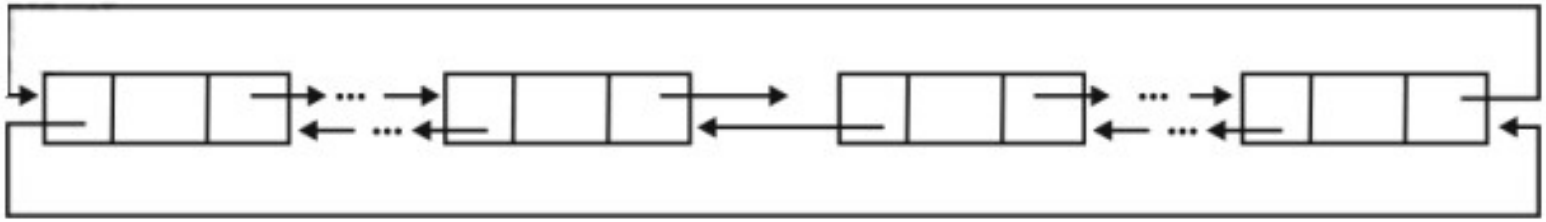


- مزیت لیست یکطرفه چرخشی نسبت به لیست یکطرفه غیرچرخشی در این است که به راحتی می توان بـه گـره قبلـی رفت.

- در لیست چرخشی با داشتن آدرس هر گره می‌توان به کلیه گره‌ها دسترسی داشت.

لیست دو طرفه چرخشی

نوعی لیست دو طرفه است که در آن اشاره‌گر راست آخرین گره به ابتدای لیست و اشاره گر چپ اولین گره به انتهای لیست اشاره می‌کند. به عبارتی next آخرین گره به گره اول و Prev اولین گره به گره آخر اشاره می‌کند. شکل زیر یک لیست چرخشی دو طرفه را نشان می دهد:



حذف گره x از یک لیست پیوندی دو طرفه چرخشی L با گره head:

```
if  (x==L)  Halt;
x->prev->next = x->next;
x->next->prev = x->prev;
free(x);
```

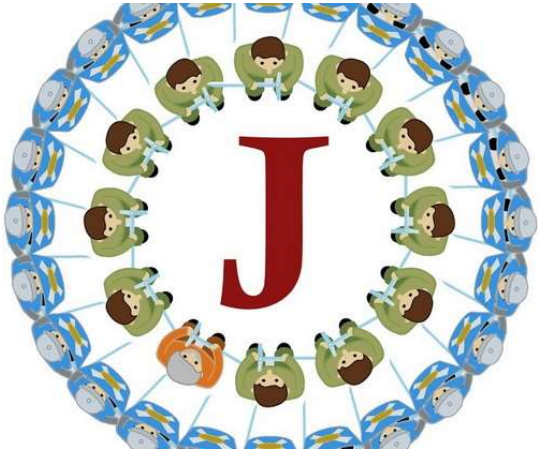
اضافه کردن گره p بعد از گره x در یک لیست پیوندی دو طرفه چرخشی:

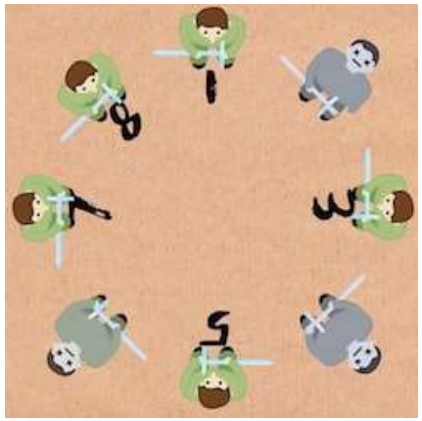
```
p->prev = x;
p->next = x->next;
x->next ->prev= p;
x->next = p;
```

تذکر: اگر آخرین خط یعنی x->next=p; را در ابتدا اجرا کنیم، آن گاه دیگر به گره های واقع در سمت راست گـره x در لیست اولیه، دسترسی نخواهیم داشت و به آنها زباله (Garbage) می گوییم.

مسئله ی ژوزفوس

ژوزفوس یکی از ۴۱ یهودی ای بود که به وسیله ی رومیان در یک غار محاصره شده بودند. به جای تسلیم، این گـروه تصـمیم گرفتند که همگی به این صورت خودکشی کنند: آن ها قرار گذاشتند تا با شروع از نفر اول و به صورت حلقوی، هر بار نفر دوم زنده ها خود را بکشد و نوبت به نفر زنده ی بعدی برسد، تا این که هیچکس باقی نماند. ولی ژوزفوس زرنگ تـر از آنهـا بـود و مکان نشستن آخرین فردی را که قاعدتا باید خود را به تنهایی می کشت محاسبه کرد و از ابتدا در آن مکـان نشسـت و جـان سالم در برد.





مسئله ی ژوزفوس در حالت کلی

در مسئله ژوزفوس اگر هر بار k امین نفر زنده خودکشی کند، خواهیم داشت:

$$f(n,k) = ((f(n-1,k) + k - 1) \mod n) + 1$$
$$f(1,k) = 1$$

که در حالت $k=2$ خواهیم داشت:

$$f(n) = ((f(n-1) + 1) \mod n) + 1$$

مثال

مقدار $f(4)$ را با فرض $k=2$ مشخص کنید.

$$f(4) = ((f(3)+1) \mod 4)+1 = 4 \mod 4+1 = 0+1 = 1$$
$$f(3) = ((f(2)+1) \mod 3)+1 = 2 \mod 3+1 = 2+1 = 3$$
$$f(2) = ((f(1)+1) \mod 2)+1 = 2 \mod 2+1 = 0+1 = 1$$

بررسی چند نمونه در حالت $K=2$:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f(n)	**1**	**1**	3	**1**	3	5	7	**1**	3	5	7	9	11	13	15	**1**