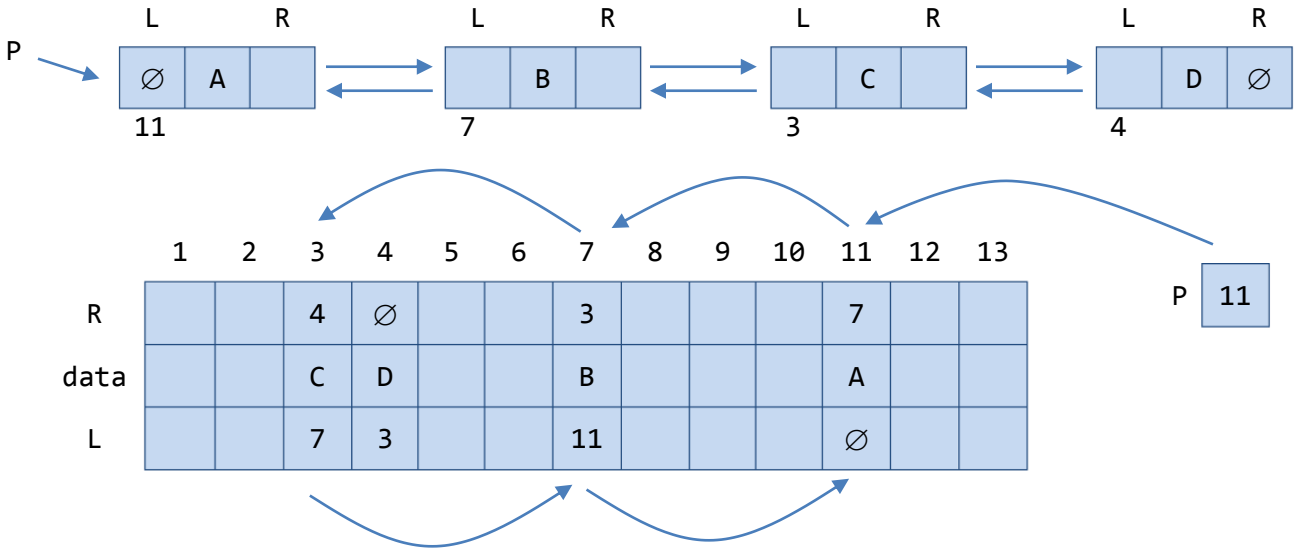
پیاده سازی لیست پیوندی دوطرفه به کمک آرایه

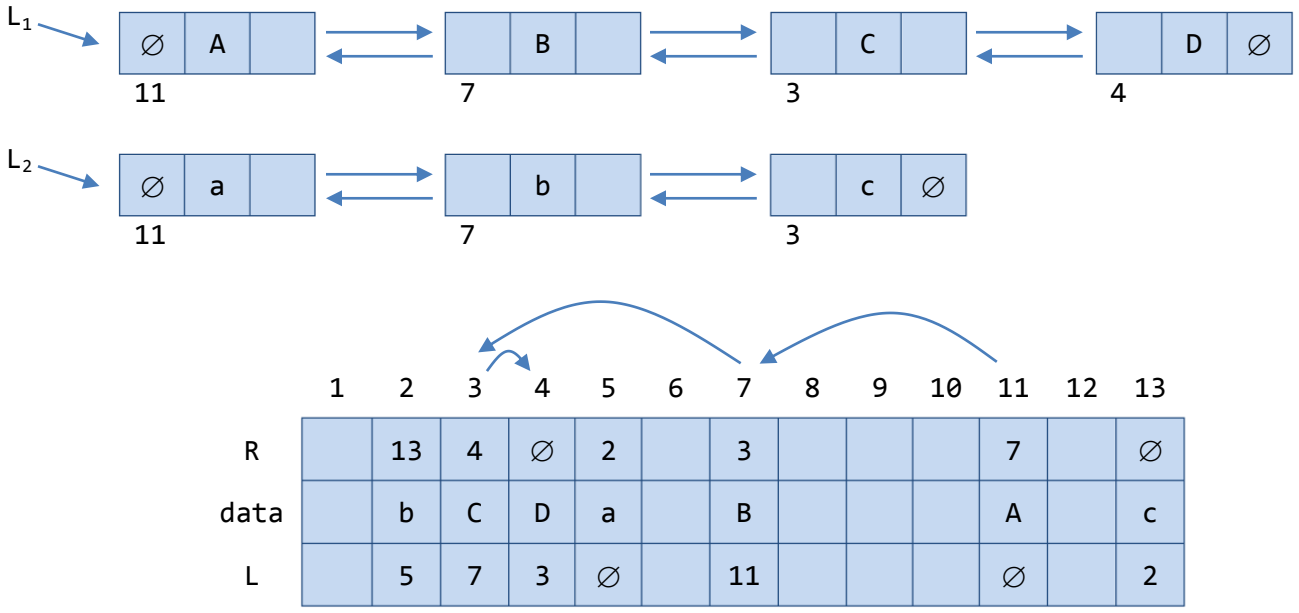
۱. آرایه های دوبعدی

۲. آرایه های یک بعدی

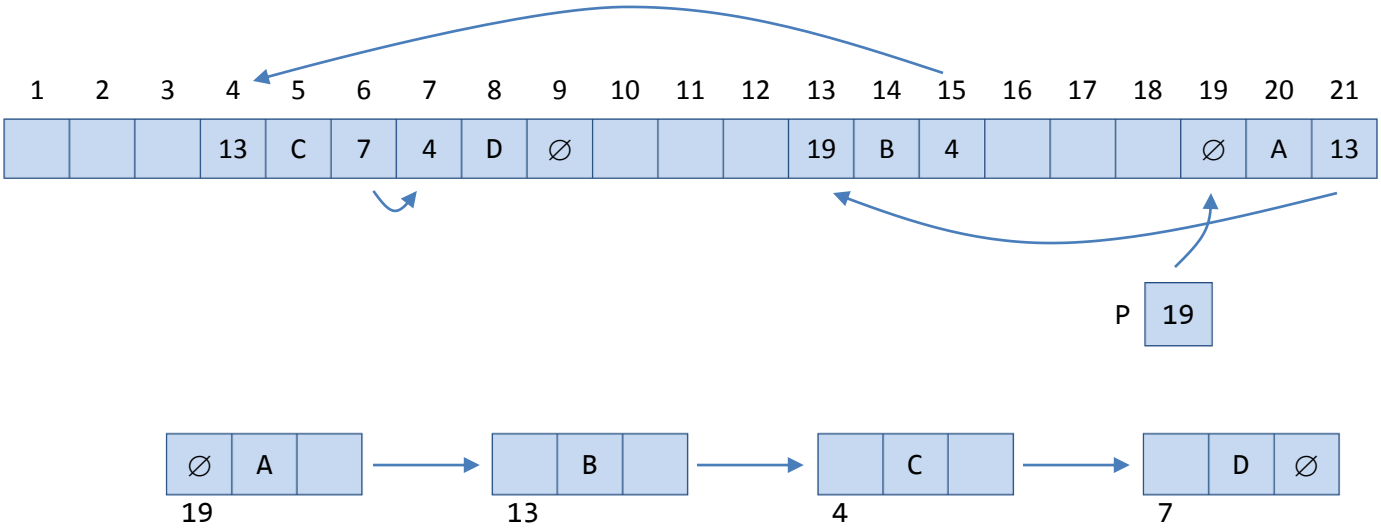
پیاده سازی لیست پیوندی به کمک آرایه یک بعدی



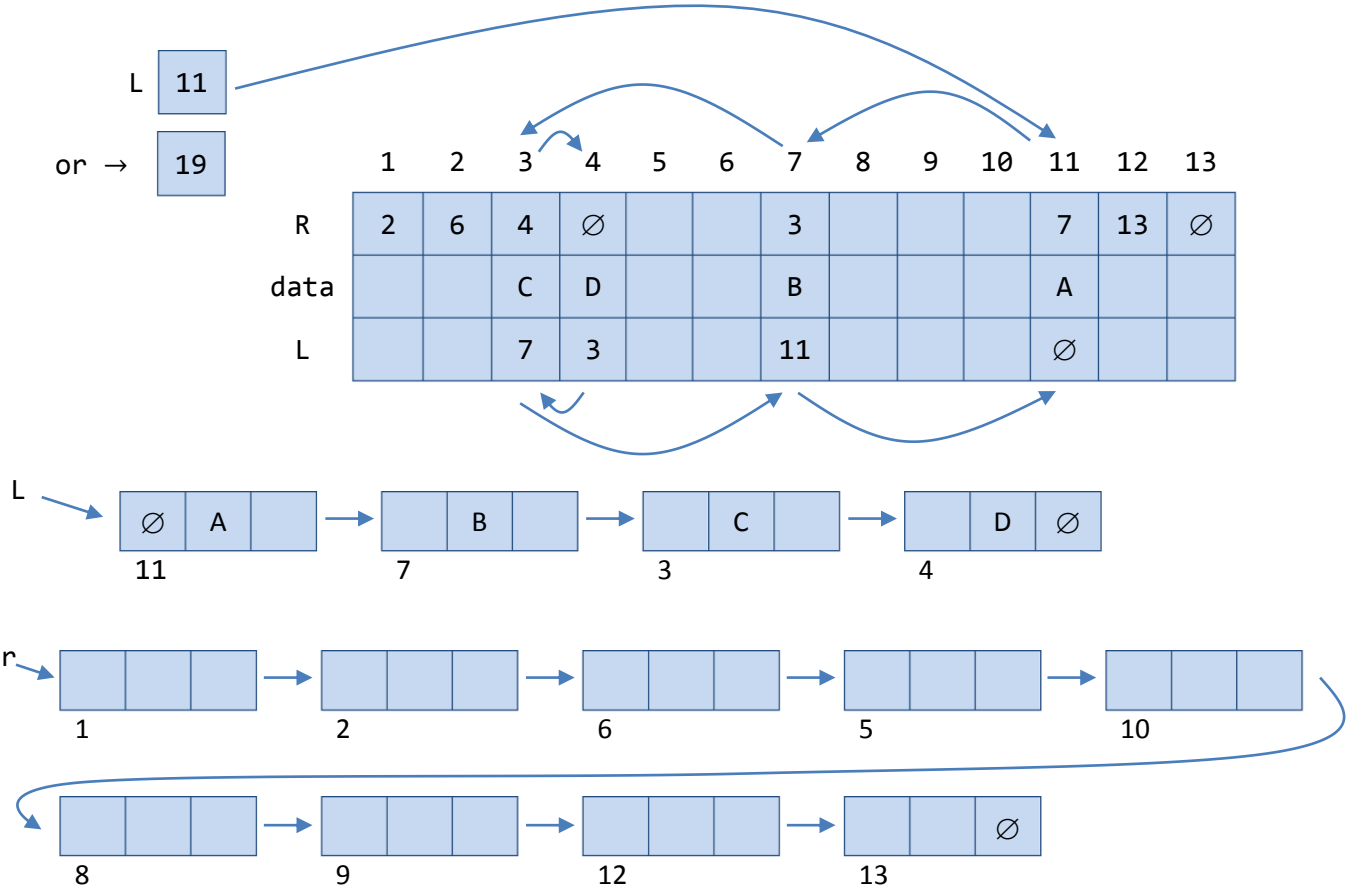
استفاده از دو لیست پیوندی در یک آرایه



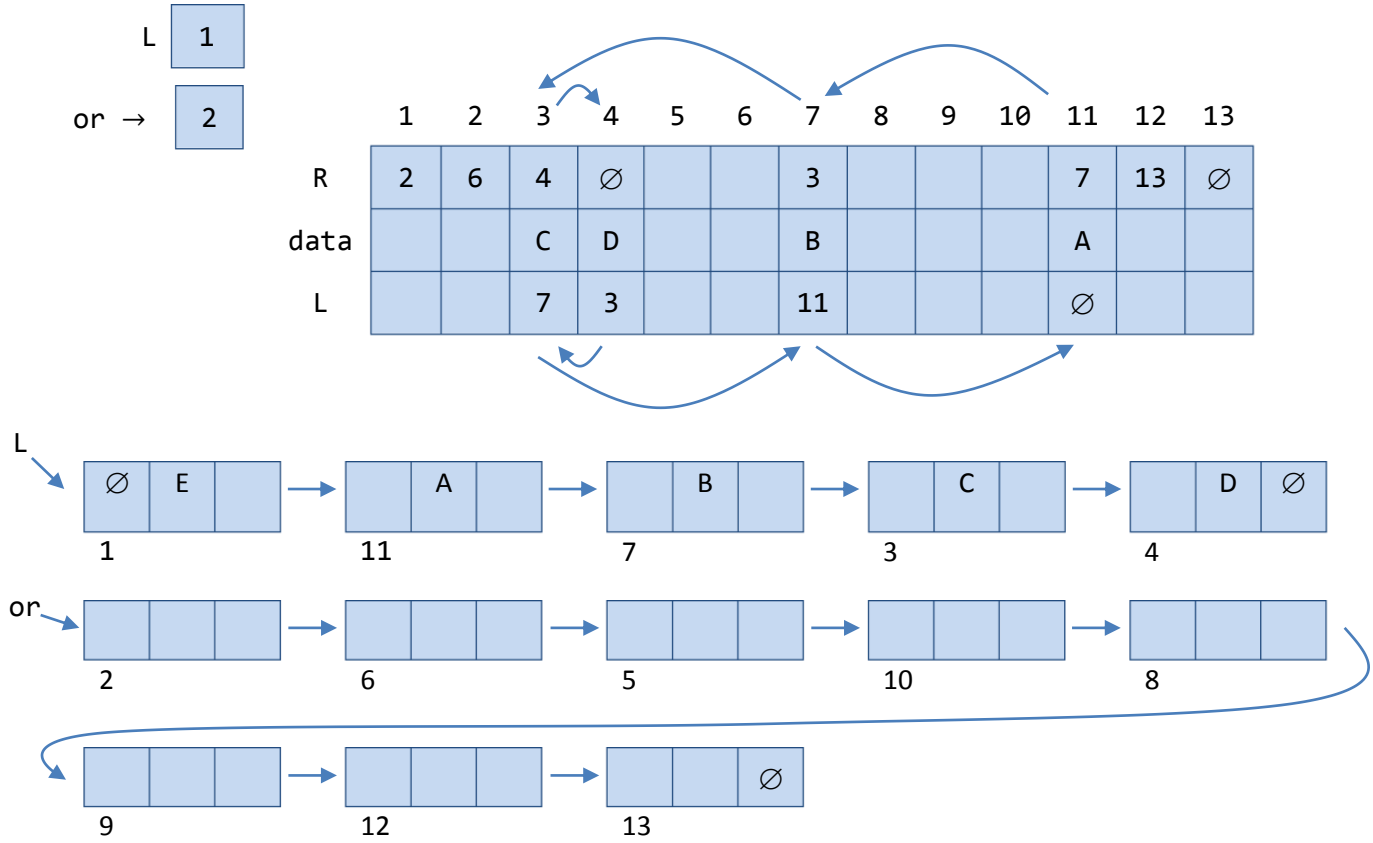
پیاده سازی لیست پیوندی به کمک آرایه یک بعدی



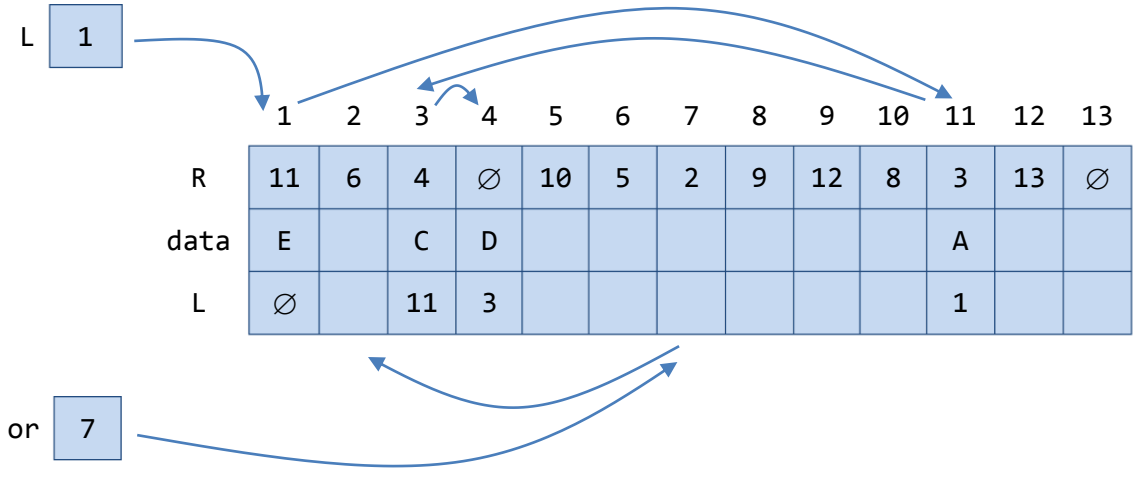
مدیریت فضای آزاد در پیاده سازی

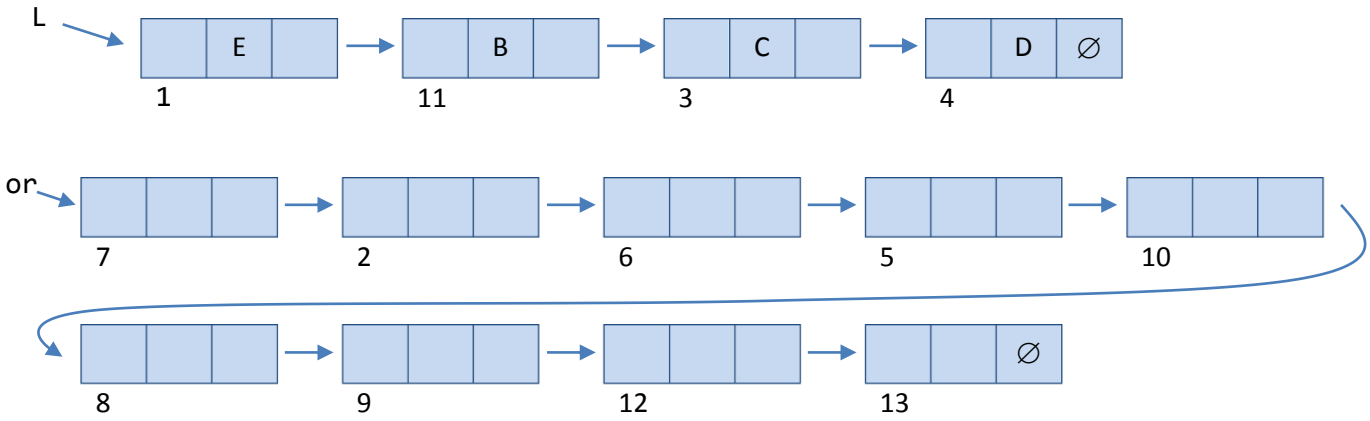


درج E به عنوان ابتدای لیست



پس از حذف عنصر B از لیست





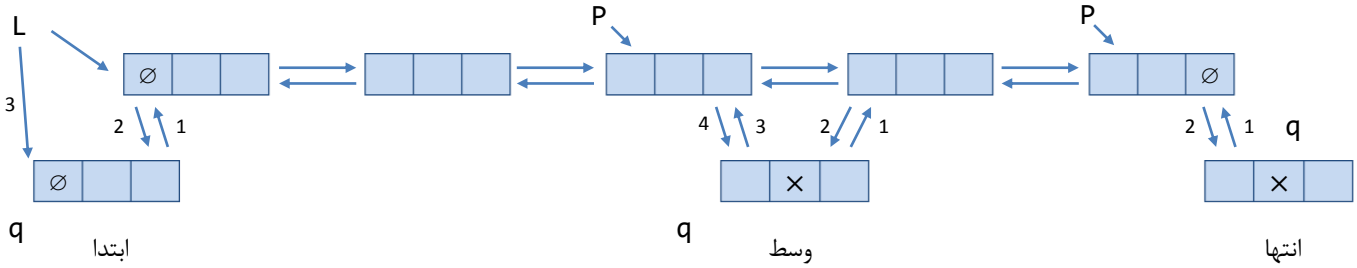
مقایسه ی این روش با استفاده از اشاره گر به عهده دانشجو

الگوریتم های لیست دوطرفه

درج

۱. درج در ابتدا
۲. درج در وسط
۳. درج در انتها

نکته: درج در لیست دوطرفه با حداکثر چهار متغیر اشاره گر انجام خواهد شد.




```
q → Left = NULL;
(1) q → Right = L;
(2) L → Left = q;
(3) L = q
```

درج در ابتدا

```
new(q);
q → data = x;
(1) q → Right = P → Right;
(2) P → Right → Left = q;
(3) q → Left = P;
(4) P → Right = q
```

درج در وسط

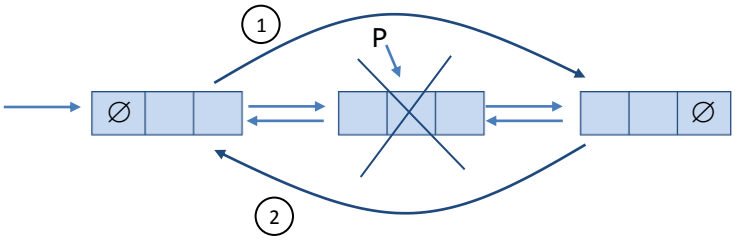
```
new(q);
q → data = x;
q → Left = P;
P → Right = q
```

درج در انتها

O(1)

اگوریتم حذف: حذف نیز به سه حالت حذف از وسط از انتها و حذف از ابتدا بررسی می شود.

نکته: حذف یک عنصر از لیست پیوندی دوطرفه با حداکثر ۲ متغیر در اشاره گر انجام می شود.



حذف از وسط

```
① P → Left → Right = P → Right;
② P → Right → Left = P → Left;
   dispose(p);
```

حذف از انتها

```
P → Left → Right = NULL;
dispose(p);
```

یا

```
P → Right;
```

حذف از ابتدا

```
P = P → Right; dispose(p → Left);
P → Left = NULL;
```

خصوصیات لیست پیوندی دوطرفه

- ✓ ۱- عناصر لزوما پشت سر هم نیستند
- ✓ ۲- دسترسی ترتیبی
- ✓ ۳- پویا
- ✓ ۴- جستجو O(n)
- ✓ ۵- درج
  - مکان معلوم O(1)
  - مکان نامعلوم O(n)
- ✓ ۶- درج در ابتدا O(1)
- ✓ ۷- حذف
  - مکان معلوم O(1)
  - مکان نامعلوم O(n)

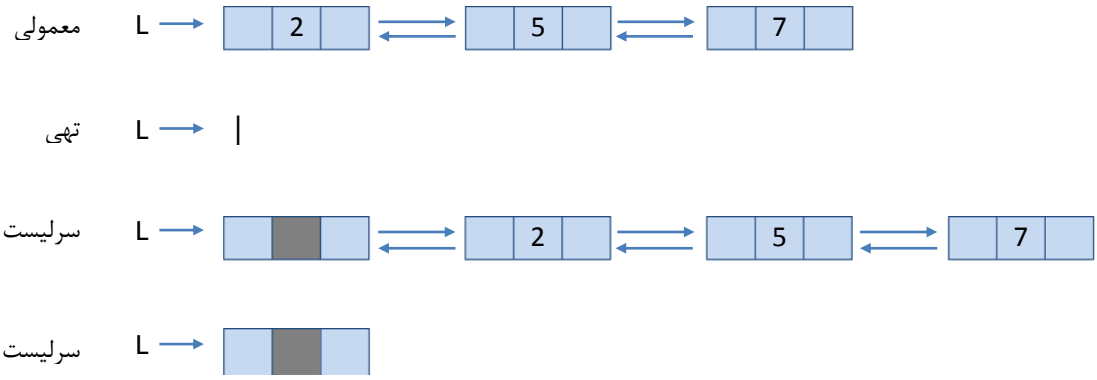
✓ ۸- حذف از ابتدا O(1)

× ۹- وابستگی به عنصر ابتدایی

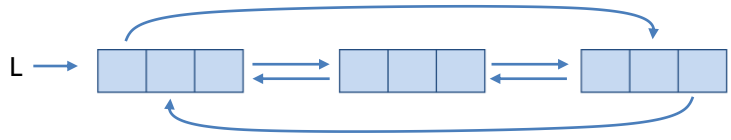
× ۱۰- تفاوت الگوریتم های ابتدا

نکته: در اینجا می توان از مفهوم لیست های سرلیست دار استفاده کرد به این شکل که یک سلول که بخش داده ای آن حایز اهمیت نیست به عنوان سرلیست به لیست اضافه می کنیم. مزایای آن دقیقا مشابه لیست یکطرفه است.

مثال: < 2 , 5 , 7 >

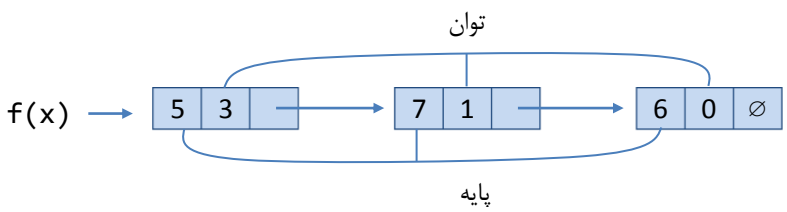


لیست دوطرفه با مشکل حذف از انتها یا درج در انتها مواجه است و آن را با زمان O(n) انجام می دهد. برای رفع این مشکل لیست دوطرفه دوار نیز قابل تعریف است به این شکل که عنصر انتهایی لیست از طریق اشره گر right به عنصر ابتدایی لیست و از طریق اشاره گر Left به عنصر انتهایی اشاره می کند. در این ساختار کلیه عملیات درج و حذف ابتدا و انتها با زمان اجرای O(1) خواهد بود.

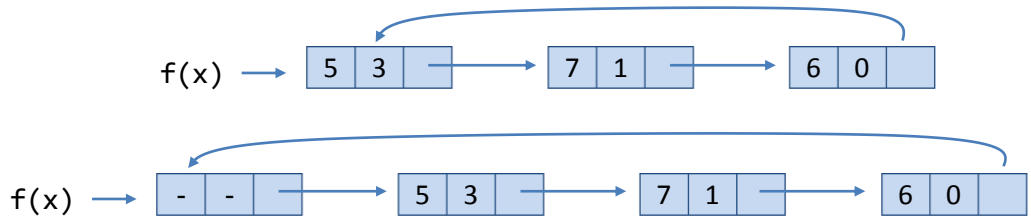


پیاده سازی چندجمله ای

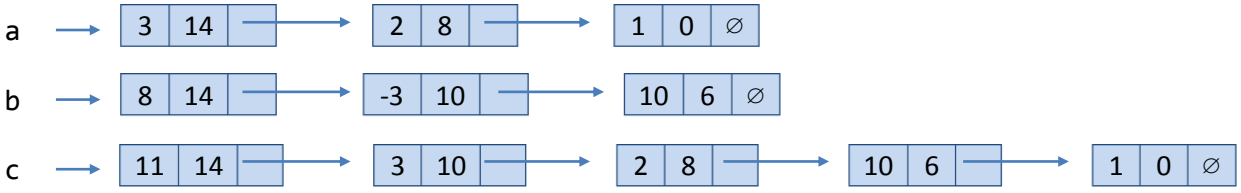
$f(x) = 5x^3 + 7x - 6$



روش دیگر از پیاده سازی چندجمله ای



جمع چندجمله ای

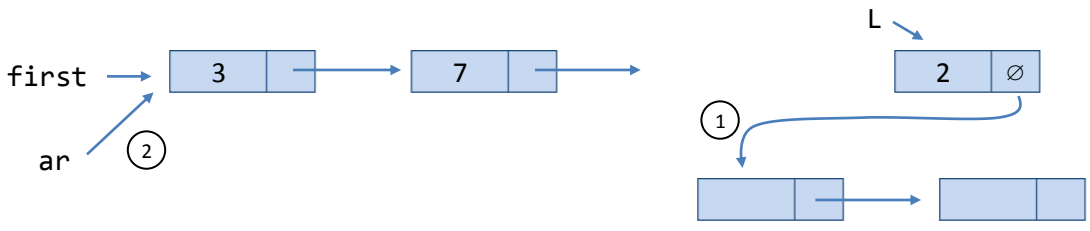


زمان اجرا: O(n+m) حداقل مقایسه: min(n,m) حداکثر مقایسه: n+m

حذف کل لیست پیوندی

- لیست پیوندی یک طرفه
- لیست پیوندی یک طرفه حلقوی

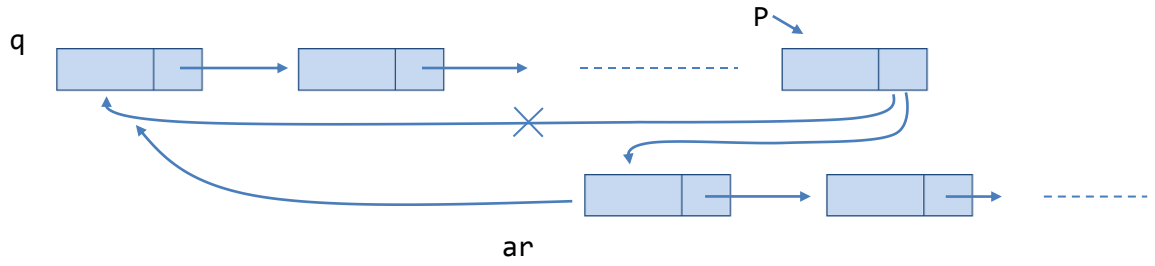
در لیست پیوندی یک طرفه می بایست تا انتهای لیست رفت سپس انتهای آن را به ar اضافه کرد.



```
L = first;
while (Lazy --> next != NULL)
    L = L --> next;
L --> next;
ar = first;
```

O(n)

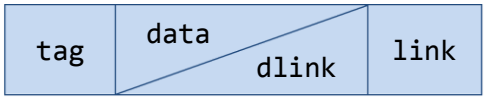
در لیست پیوندی حلقوی چون دسترسی به انتها دارد سپس زمان اجرای آن O(1) می باشد.




```
List* q = P --> next;
P --> next = args;
ar = q;
```

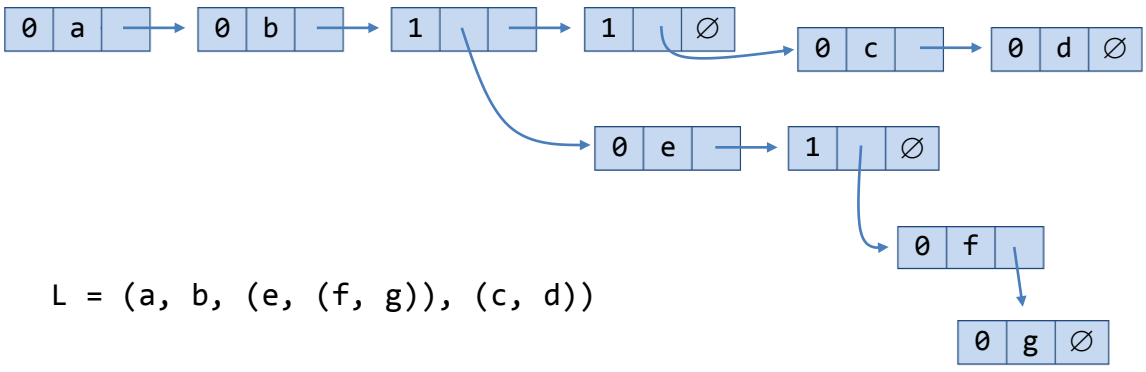
O(1)

لیست پیوندی عمومی



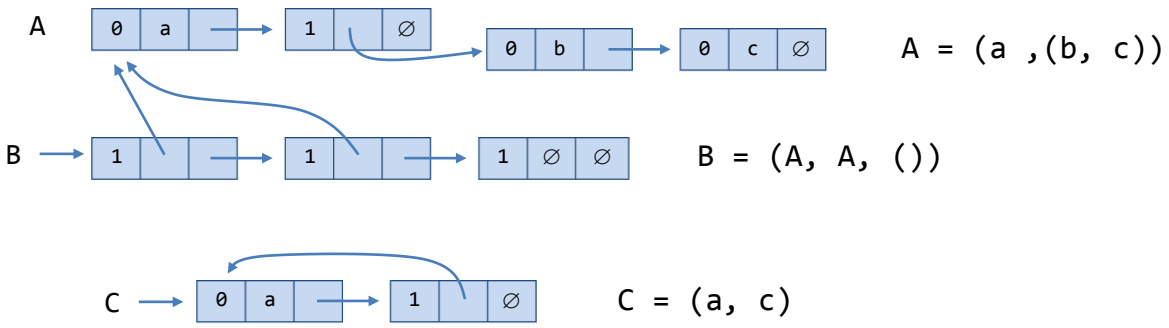
اگر tag = 0 بخش وسطی به صورت data در نظر گرفته می شود.

اگر tag = 1 بخش وسطی به صورت dlink در نظر گرفته می شود.



L = (a, b, (e, (f, g)), (c, d))

نکته: یک لیست عمومی می تواند مشابه A = () طول صفر داشته باشد.


A
0 a
1 ∅
0 b
0 c ∅
A = (a ,(b, c))
B
1
1
1 ∅ ∅
B = (A, A, ())
C
0 a
1 ∅
C = (a, c)