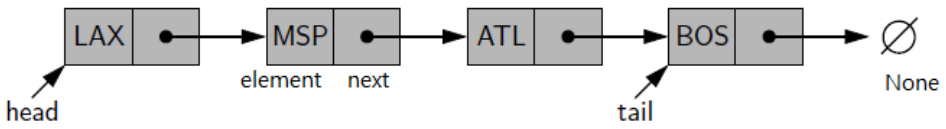
ساختار داده لیست پیوندی

ساختار داده‌هایی که بر اساس آرایه هستند دسترسی سریع به محلی دلخواه از آرایه (اصطلاحا به اندیسی دلخواه از آرایه) فراهم می‌کنند اما در این ساختارداده‌ها درج و حذف عناصر در داخل آرایه کاری زمانبر است چون مستلزم شیفت و جابجایی داده‌هاست. ساختار داده لیست پیوندی بستری را فراهم می‌کند تا بتوان عناصر را با هزینه کم حذف و اضافه کرد.

یک لیست پیوندی مجموعه‌ای از عناصر است که با زنجیره‌ای از پیوندها بصورت خطی به هم وصل شده‌اند. ساختار داده لیست پیوندی Linked List اعمالی از قبیل ایجاد یک لیست تهی، درج یک عنصر در ابتدا و انتهای لیست، درج یک عنصر بعد از یک عنصر داده شده و حذف یک عنصر از ابتدای لیست یا بعد از یک عنصر داده شده را پشتیبانی می‌کند. لیستهای پیوندی می‌توانند یک سویه، دو سویه و یا حلقوی باشند.

۱ لیست پیوندی یک سویه

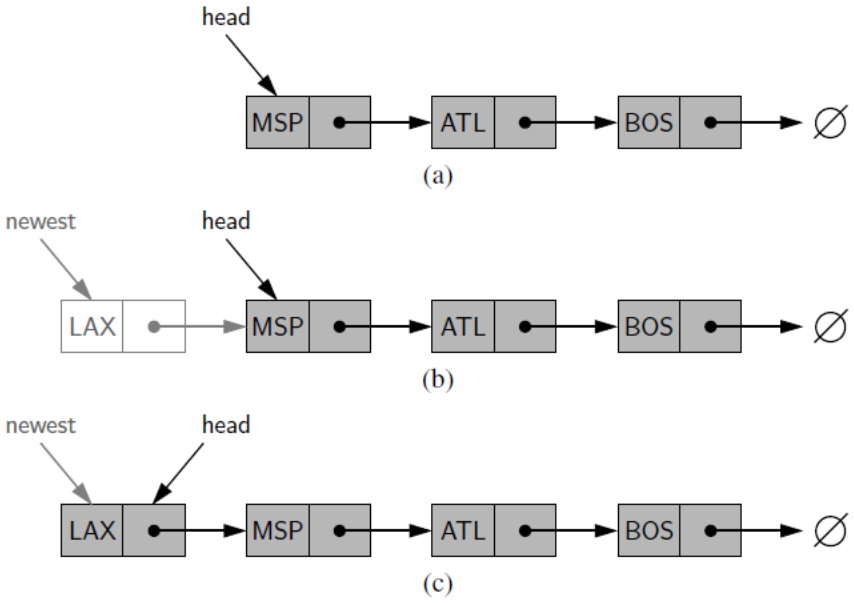
به هر عنصر لیست یک سویه، یک node گفته می‌شود که دارای دو مولفه به نامهای element و next است. مولفه element داده مرتبط به عنصر را ذخیره می‌کند. مولفه next آدرس (یا اسم) عنصر بعدی لیست را در خود جای می‌دهد که با استفاده از آن می‌توان به محل ذخیره‌شدن عنصر بعدی در حافظه دسترسی پیدا کرد. آخرین عنصر لیست در مولفه next خود مقدار تهی (در پایتون None) را دارد. شکل زیر یک نمونه از لیست پیوندی یک سویه را نشان می‌دهد. عناصر داده‌ای حاوی اسامی اختصاری چند فرودگاه است. شناسه head به اول لیست اشاره می‌کند و شناسه tail به آخر لیست.



چگونگی انجام چند عمل اصلی مربوط به لیست پیوندی در زیر آمده است.

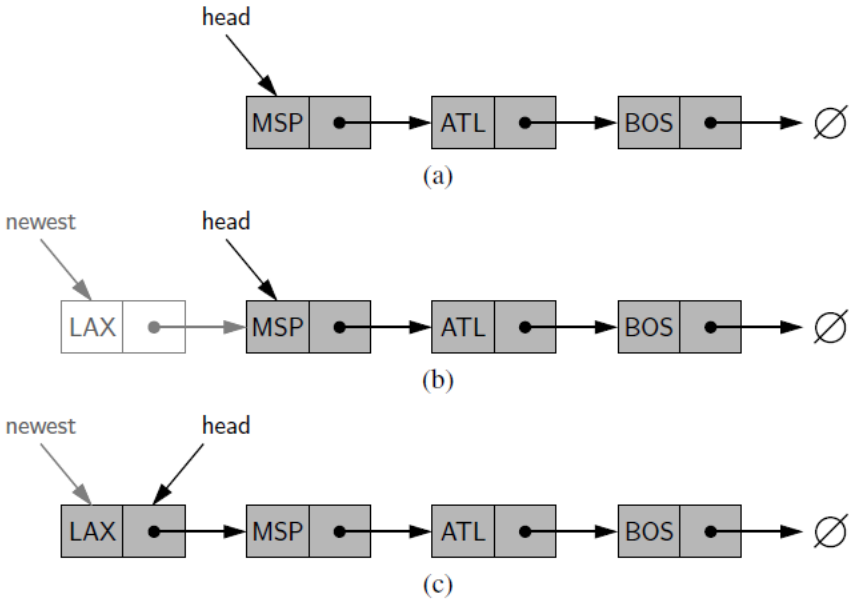
- درج در ابتدا: یک عنصر با داده e در ابتدای لیست L درج می‌شود.

```
Algorithm add_first(L,e):

newest = Node(e)
newest.next = L.head
L.head = newest
L.size = L.size + 1
```



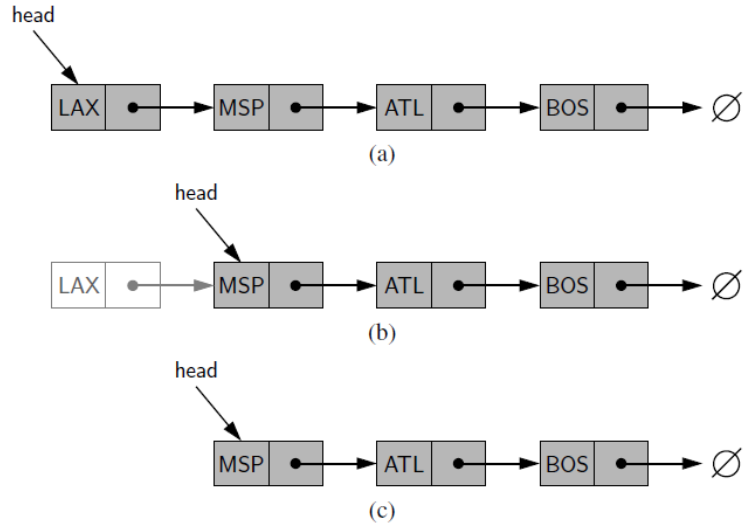
- درج در انتها: یک عنصر با داده e در انتهای لیست L درج می‌شود.

```
Algorithm add_last(L,e):

newest = Node(e)
newest.next = None
L.tail.next = newest
L.tail = newest
L.size = L.size + 1
```



- حذف از ابتدا: عنصر ابتدایی لیست حذف می‌شود.

```
Algorithm remove_first(L):

If  L.head = None
    print (error)
L.head = L.head.next
L.size = L.size - 1
```



- پیمایش و چاپ داده مربوط به عناصر لیست.

```
Algorithm print(L):

current = L.head
While(current != None):
    print(current.element)
    current = current.next
```

۱.۱ پیاده‌سازی پشته با استفاده از لیست پیوندی

پشته را می‌توان با لیست پیوندی پیاده‌سازی کرد. کد زیر یک نمونه از این را نشان می‌دهد. ساختار و اعمال پشته در کلاس LinkedStack پیاده‌سازی شده است. node های لیست پیوندی در کلاس Node تعریف شده‌اند. به عمل push دقت کنید. اینجا خبری از لیست و append نیست. می‌بینید که در این پیاده‌سازی اعمال اصلی پشته در زمان $O(1)$ قابل اجراست.

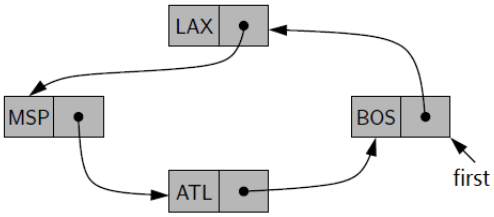
```
class LinkedStack:

  class Node:
      def __init__(self, element, next): # initialize node's fields
         self.element = element # reference to user's element
         self.next = next # reference to next node

  def __init__(self):
   self.head = None # reference to the head node
   self.size = 0 # number of stack elements

  def len(self):
   return self.size

  def is_empty(self):
    return self.size == 0

  def push(self, e):
   self.head = self.Node(e,self.head) # create and link a new node
   self. size += 1

  def top(self):
   if self.is_empty( ):
    raise Empty("Stack is empty")
   return self.head.element

  def pop(self):
    if self.is_empty( ):
      raise Empty('Stack is empty')
    answer = self.head.element
    self.head = self.head.next # bypass the former top node
    self.size -= 1
    return answer
```

پیاده‌سازی صف صف را هم می‌توان با استفاده از لیست پیوندی یک سویه پیاده‌سازی کرد. اینجا علاوه بر متغیر head که شروع لیست پیوندی را نگه‌می‌دارد به متغیر tail هم نیاز داریم تا انتهای لیست پیوندی را به سرعت پیدا کنیم و عمل enqueue را بصورت کارا پیاده‌سازی کنیم. در ای پیاده‌سازی هم اعمال اصلی صف در زمان $O(1)$

قابل اجراست.

۲ لیست پیوندی حلقوی

در لیست پیوندی یکسویه اگر اشاره‌گر انتهای لیست به ابتدای لیست اشاره کند، یک لیست پیوندی حلقوی ایجاد می‌شود. در اینجا معمولا به جای head و tail تنها یک متغیر برای اشاره به عنصر کنونی ذخیره می‌شود. این اشاره‌گر می‌تواند به معنای اول لیست باشد. در هر صورت با استفاده از این اشاره‌گر می‌توان لیست پیوندی را پیمایش کرد.



از لیست پیوندی حلقوی می‌توان برای اجرای کارها بصورت گردشی (اصطلاحا زمانبندی round robin) استفاده کرد. فرض کنید مجموعه‌ای ثابت از درخواستها داریم که می‌خواهیم به نوبت و بصورت گردشی به آنها سرویس دهی کنیم. یعنی زمانی که آخرین درخواست سرویس دهی شد، به درخواست اول برگردیم و سرویس دهی را از آنجا از سر گیریم. می‌توان متد Rotate را برای لیست حلقوی تعریف کرد که در آن محل first یکی به جلو می‌رود (مثل یک صف حلقوی.)

۳ لیست پیوندی دوسویه

فرض کنید آدرس یک عنصر از لیست پیوندی یکسویه را داریم و می‌خواهیم آن را از لیست حذف کنیم. برای این کار باید آدرس عنصر ماقبل از آن را بدانیم در غیر اینصورت باید از ابتدای لیست حرکت کنیم تا به عنصر ماقبل از آن برسیم. برای رفع این مشکل و تسریع دسترسیهای دیگر، ساختار داده لیست پیوندی دوسویه پیشنهاد شده است. همانطور که در شکل زیر می‌بینید node های لیست پیوندی دوسویه علاوه بر مولفه next مولفه prev را هم دارند که به عنصر قبلی در لیست اشاره می‌کند. برای اینکه ابتدا و انتهای لیست مجزا باشند و در برابر تغییرات نادرست حفاظت شوند، دو عنصر خالی به نامهای header و trailer به اول و آخر لیست اضافه شده است. می‌بینید که حذف عنصر با داده PVD چگونه انجام شده است. در ایجا کافی است که یک اشاره‌گر به محل عنصر PVD داشته باشیم. عنصر قبلی آن با استفاده از مولفه prev قابل دسترسی است.