دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۱۴

لیست پیوندی

محمد مصطفی رستم خانی - ۱۳۹۸/۸/۱۳

جزوه جلسه ۱۴ام مورخ ۱۳۹۸/۸/۱۳درس ساختمان‌های داده تهیه شده توسط محمد مصطفی رستم خانی. در جهت مستند کردن مطالب درس ساختمان های داده، بر آن شدیم که از دانشجویان جهت مکتوب کردن مطالب کمک بگیریم. هر دانشجو می‌تواند برای مکتوب کردن یک جلسه داوطلب شده و با توجه به کیفیت جزوه از لحاظ کامل بودن مطالب، کیفیت نوشتار و استفاده از اشکال و منابع کمک آموزشی، حداکثر یک نمره مثبت از بیست نمره دریافت کند. خواهش‌مند است نام و نام خانوادگی خود، عنوان درس، شماره و تاریخ جلسه در ابتدای این فایل را با دقت پر کنید.

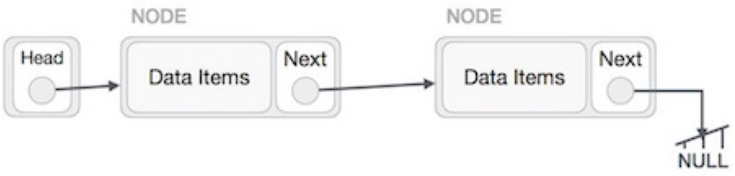
۱.۱۴ لیست پیوندی

نوعی ساختمان داده است که اعضای آن در جا های مختلفی از حافظه هستند و الزامی ندارد که پشت سر هم باشند. هر عضو از این ساختمان داده دارای حداقل دو ویژگی است. یکی مقدار آن و دیگری اشاره گری به عضو بعدی(و یا هم بعدی و هم قبلی) دنباله است. ما در این نوع ساختمان داده باید آدرس عنصر اول را نگه داریم و از آنجایی که هر عضو به عضو بعدی خود اشاره می کند و آخرین عنصر دارای اشاره گری به null است، اینگونه می توان به تمام عناصر این دنباله دسترسی پیدا کرد. این ویژگی لیست پیوندی باعث می شود که بتواند تعداد عناصر آن مختلف باشد و از آن کم کرد یا به آن اضافه کرد.گاهی اوقات اشاره گر به آخرین عنصر نیز نگه داری می شود. انواع لیست پیوندی عبارتند از:

- لبست پیوندی یک طرفه
- لیست پیوندی دو طرفه
- لیست پیوندی حلقوی

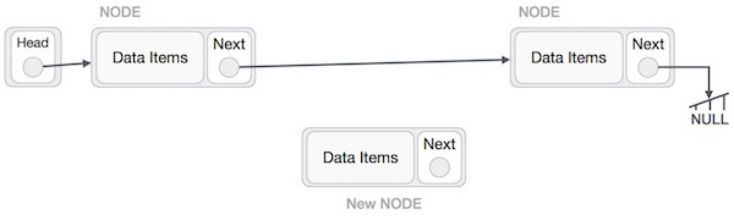
۲.۱۴ لیست پیوندی یک طرفه:

یک لیست پیوندی یک طرفه (Singly-linked list) دنباله ای از عناصر داده ای به نام گره (node) است که ترتیب خطی آنها توسط اشاره گرها تعیین می گردد. عناصر لیست تنها می توانند به ترتیب از ابتدای لیست تا انتها مورد دسترسی قرار بگیرند. هر گره آدرس گره بعدی را شامل می شود که به این صورت امکان پیمایش از یک گره به گره بعدی فراهم می شود. برای رسم لیست پیوندی گره ها به صورت مستطیل هائی پشت سرهم رسم می شوند که توسط فلش هائی بهم متصل شده اند. مقدار ثابت NULL برای علامت گذاری انتهای لیست در اشاره گر آخرین گره ذخیره می شود. لیست توسط یک اشاره گر Head که آدرس اولین گره لیست را در خود ذخیره می کند قابل دسترس است. بقیه عناصر توسط جستجوی خطی بدست می آیند.



شکل ۱.۱۴: لیست پیوندی

۳.۱۴ درج کردن:

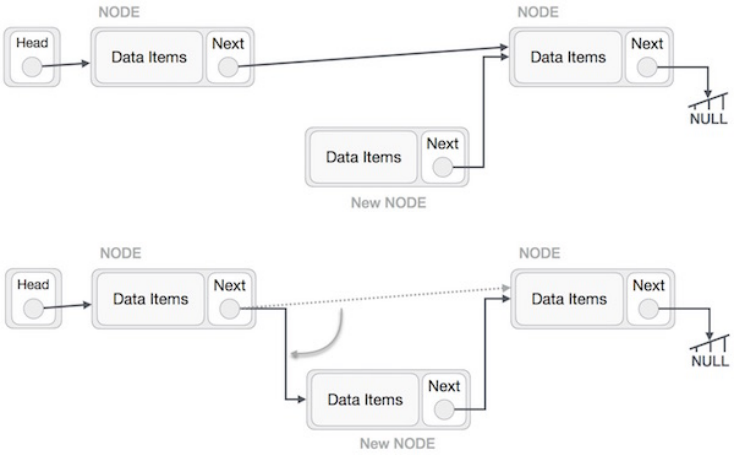


تصور کنید که می‌خواهیم یک گره B (گره جدید) را بین گره A (گره چپ) و گره C (گره راست) درج کنیم. در این صورت B باید به C به عنوان next اشاره کند:

```
NewNode.next -> RightNode;
```

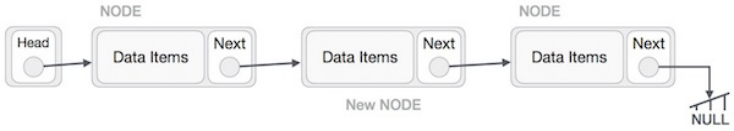
این عملیات به صورت زیر خواهد بود:

حال گره سمت چپ باید به گره جدید اشاره کند:




```
LeftNode.next -> NewNode;
```

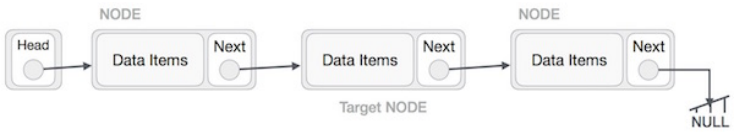
بدین ترتیب گره جدید در میان دو گره قبلی قرار می‌گیرد. لیست جدید به صورت زیر خواهد بود:



اگر بخواهیم گرهی را در ابتدای لیست اضافه کنیم نیز مراحل مشابهی را طی می‌کنیم. زمانی که می‌خواهیم گرهی را در انتهای لیست درج کنیم، گره ما قبل آخر باشد به گره جدید اشاره کند و گره جدید به یک مقدار null اشاره می‌کند.

۴.۱۴ حذف کردن

ابتدا گره هدف که می‌خواهیم حذف کنیم را با استفاده از الگوریتم‌های جستجو می‌یابیم.



گره چپ (قبلی) گره هدف اینک باید به گره بعد از گره هدف اشاره کند:

```
LeftNode.next -> TargetNode.next;
```

با این کار پیوندی که به گره هدف وجود داشت از بین می‌رود. حال با استفاده از کد زیر موردی که گره هدف به آن اشاره می‌کرد را نیز حذف می‌کنیم:

```
TargetNode.next -> NULL;
```


```
PushBack(key):
node=new node
node.key=key
node.next=nil
if tail==nil then
 | head=tail=nil
else
 | tail.next=node
 | tail=node
end
```

Algorithm 1: PushBack singly-linked list

```
PopBack():
if head==nil then
 | ERROR:Empty list
end
if head==tail then
 | head=tail=nil
else
 | p=head
 | while p.next.next != nil do
 |  | p=p.next
 | end
 | p.next=nil
 | tail=p
end
```

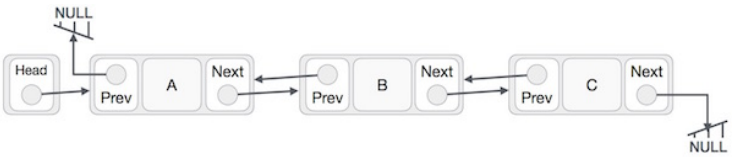
Algorithm 2: PopBack singly-linked list

```
AddAfter(node,key):
node2=new node
node2.key=key
node2.next=node.next
node.next=node2
if tail==node then
 | tail=node2;
end
```

Algorithm 3: AddAfter singly-linked list

۵.۱۴ لیست پیوندی دو طرفه

لیست پیوندی دو طرفه نوعی از لیست پیوندی است که حرکت در هر دو جهت یعنی به سمت جلو یا عقب در آن امکان‌پذیر است. یعنی هر عنصر علاوه بر آدرس عنصر بعدی به آدرس عنصر قبلی نیز اشاره می کند.




```
PushBack(key):
node=new node
node.key=key
node.next=nil
if tail==nil then
 | head=tail=node
 | node.prev=nil
else
 | tail.next=node
 | node.prev=tail
 | tail=node
end
```

Algorithm 4: PushBack Doubly-linked list

```
PopBack():
if head==nil then
 | ERROR:Empty list
end
if head==tail then
 | head=tail=nil
else
 | tail=tail.prev
 | tail.next=nil
end
```

Algorithm 5: PopBack Doubly-linked list

```
AddAfter(node,key):
node2=new node
node2.key=key
node2.next=node.next
node2.prev=node
node.next=node2
if node2.next != nil then
 | node2.next.prev=node2
end
if tail==node then
 | tail=node2
end
```

Algorithm 6: AddAfter Doubly-linked list

```
AddBefore(node,key):
node2=new node
node2.key=key
node2.next=node
node2.prev=node.prev
node.prev=node2
if node2.prev != nil then
 | node2.prev.next=node2
end
if head==node then
 | head=node2
end
```

Algorithm 7: AddAfter Doubly-linked list

[۱] [۲]. [۳].

Bibliography

[1] https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm.

[2] https://www.cs.cmu.edu/~adamchik/15-121/lectures/Linked%20Lists/linked%20lists.html#:~:targetText=A%20linked%20list%20is%20a,the%20head%20of%20the%20list.

[3] https://en.wikipedia.org/wiki/Linked_list.