دانشکده علوم ریاضی و آمار

مدرس: دکتر مجتبی رفیعی نیمسال اول ۱۴۰۰-۱۴۰۱

ساختمان داده‌ها و الگوریتم‌ها

جلسه ۲۲

نگارنده: حسنا اتقیا

۲۰ آبان ۱۴۰۰

فهرست مطالب

نکته: همواره بهترین پیاده سازی برای یک مجموعه متناهی، متناسب با کاربرد و وابسته به عملیات مورد پشتیبانی است.

۱ قرارداد

۱)برای عناصر یک مجموعه پویای متناهی از اصطلاح عنصر یا شی استتفاده می‌کنیم.
۲) در ادامه برای اختصار، از اصطلاح مجموعه یا مجموعه پویا به جای مجموعه پویای متناهی استفاده می‌کنیم.
به عبارت دیگر در این درس، تمامی مجموعه‌هایی که تعریف می‌کنیم را متناهی فرض می‌کنیم.

۲ مجموعه پویا

۱.۲ عناصر

هر عنصر یا شی از یک مجموعه حاوی گردایه‌ای از خصیصه‌هاست که به طور معمول یکی از آنها به عنوان کلید برای آن شی در نظر گرفته می‌شود. لازم به ذکر است که این امکان نیز وجود دارد که چندین خصیصه کلیدی برای یک شی در نظر گرفته شود و ضرورتی بر تک کلیده بودن خصیصه ها وجود ندارد.

۲.۲ عملیات روی مجموعه پویا

به طور کلی عملیات روی مجموعه پویا را می‌توان در دو رده زیر بیان کرد:

- **پرسمان‌های بازیابی**[1]: که اطلاعاتی را در مورد مجموعه برمی‌گرداند.
- **پرسمان‌های بروزرسانی**[2]: که سبب ایجاد تغییر (حذف و اضافه عناصر مجموعه) در مجموعه می‌شوند.

برخی از متداول‌ترین این عملیات‌ها در ادامه آورده شده است:

- **Search(S،k)**: یک پرسمان بازیابی است و به صورت زیر عمل می‌کند:
  - اگر شی با کلید kدر مجموعه S وجود داشته باشد، آن شی را برمی‌گرداند.
  - در غیر اینصورت $Null$ برمی‌گرداند.
- **Insert(S,x)**: یک پرسمان بروزرسانی است و عنصر x را به مجموعه S اضافه می‌کند.
- **Delete(S،x)**: یک پرسمان بروزرسانی است و عنصر x را از مجموعه S حذف می‌کند.
- **Minimum(S)**: یک پرسمان بازیابی است و شی (یا شی‌هایی) از مجموعه S را که کلید آنها کمترین مقدار را دارد، برمی‌گرداند.
- **Maximum(S)**: یک پرسمان بازیابی است و شی (یا شی‌هایی) از مجموعه S را که کلید آنها بیشترین مقدار را دارد، برمی‌گرداند.
- **Successor(S،x)**: یک پرسمان بازیابی است که کوچکترین شی بزرگتر از شی x (براساس کلید شی) در مجموعه S را برمی‌گرداند. لازم به ذکر است که اگر شی x ماکزیمم باشد، Null را برمی‌گرداند.
- **Predecessor(S،x)**: یک پرسمان بازیابی استکه بزرگترین شی کوچکتر از شی x (براساس کلید شی) در مجموعه S را برمی‌گرداند. لازم به ذکر است که اگر عنصر x مینیمم باشد، Null رابرمی‌گرداند.

اطلاعات تکمیلی: برای مرتب کردن مجموعه S (فاقد تکرار) با n عنصر با استفاده از توابع بالا، می‌توان به صورت زیر عمل کرد:

- یکبار فراخوانی $x_1 = Minimum(S)$،
- $n-1$ بار فراخوانی $Successor$ به صورت زیر:

$$x_2 = Successor(S, x_1)$$
$$x_3 = Successor(S, x_2)$$
$$.$$
$$.$$
$$.$$
$$x_i = Successor(S, x_{i-1})$$

[1] Retrieval queries
[2] Update queries

.

.

.

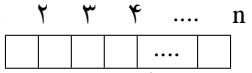
$$x_n = Successor(S, x_{n-1})$$

در ادامه به معرفی برخی از داده ساختارهای پایه نظیر: لیست پیوندی، صف، پشته و درخت ریشه دار خواهیم پرداخت.

۳ لیست پیوندی

۱.۳ لیست

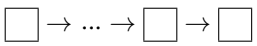
یک داده ساختار است که اشیای آن در یک ترتیب خطی (منطبق با اندیس لیست) قرار گرفته‌اند.



نمایش گرافیکی لیست

۲.۳ لیست پیوندی

یک داده ساختار است که اشیای آن در یک ترتیب خطی (از طریق اشاره‌گر) قرار گرفته‌اند.



نمایش گرافیکی لیست پیوندی

سه نوع مشهور از لیست‌های پیوندی عبارتند از:

- لیست پیوندی یک طرفه[۳]
- لیست پیوندی دو طرفه[۴]
- لیست پیوندی دایره‌ای[۵]

در ادامه؛ ابتدا لیست پیوندی دوطرفه را معرفی می‌کنیم و سپس با گذاشتن قیودی بر روی آن، لیست‌های پیوندی دیگر را معرفی می‌کنیم.
لیست پیوندی دوطرفه: هر نود در این لیست شامل سه بخش زیر است:

- اشاره گر به نود قبلی که آن را با prev نشان می‌دهیم،
- اشاره گر به نود بعدی که آن را با next نشان می‌دهیم،
- نوع داده‌ای برای نگهداری یک عنصر از مجموعه که آن را با element نشان می‌دهیم.

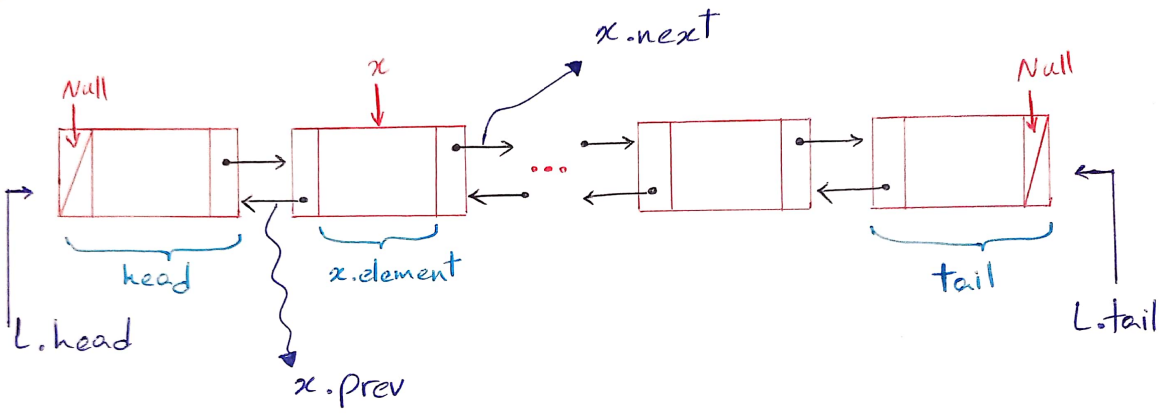
next	element	prev

جدول ۱: نمایش یک نود در لیست پیوندی دوطرفه

[3] Singly linked list
[4] Doubly linked list
[5] Circular linked list

Record node{
element: datatype,
prev: pointer to node,
next: pointer to node }

شکل زیر یک نمای کلی از لیست L را که شامل چندین نود است، نشان می‌دهد:

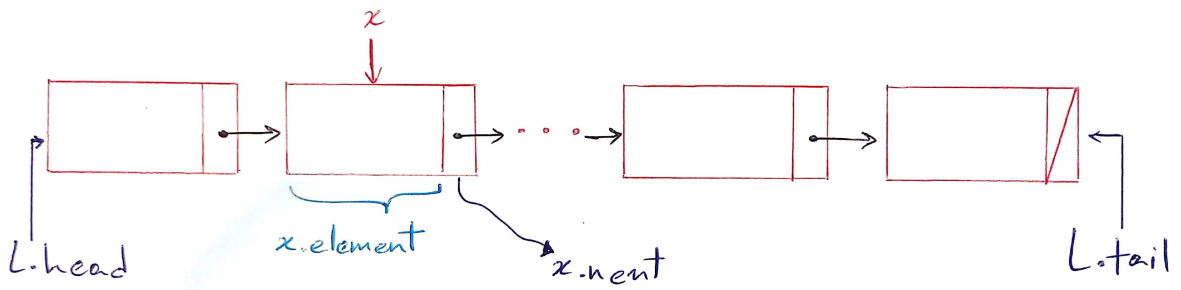


Record List{
size: integer,
head: pointer to node,
tail: pointer to node }

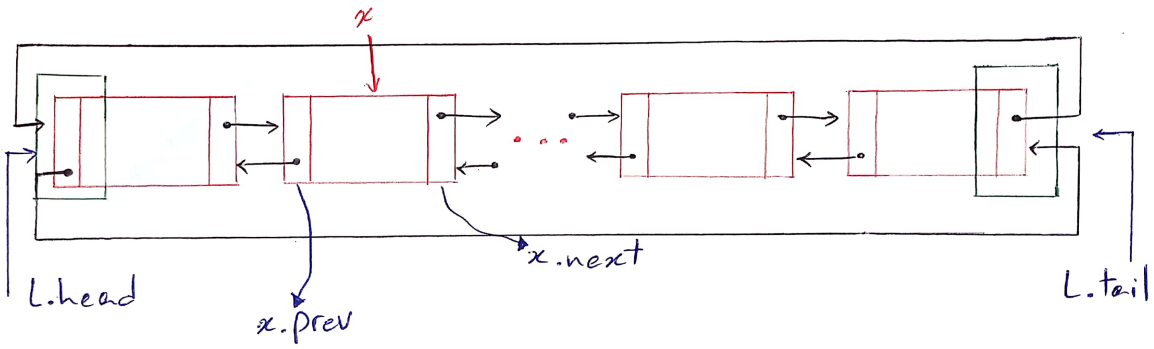
v برخی نکات در رابطه با لیست پیوندی دوطرفه:

- ۱- اگر L.head=Null باشد، بدین معناست که لیست ما تهی است.
- ۲- اگر x.next=Null باشد، عنصر x در ترتیب خطی تشکیل دهنده لیست، عنصر آخر است.
- ۳- اگر x.prev=Null باشد، عنصر x در ترتیب خطی تشکیل دهنده لیست، عنصر اول است.

لیست پیوندی یک طرفه: شبیه به لیست پیوندی دوطرفه است با این تفاوت که هر نود آن تنها شامل اشاره‌گر به نود بعدی است و اشاره‌گر به نود قبلی ندارد.



لیست پیوندی دایره ای: این نوع لیست قابل تعریف برای لیست‌های پیوندی یک طرفه و دوطرفه است. با اینحال، در ادامه تنها به لیست پیوندی دایره ای برای حالت دوطرفه می پردازیم.



سوال: آیا در این ساختار نیاز هست tail در لیست نگهداری شود؟ خیر، از طریق $head$ به صورت زیر می‌توان به آن دسترسی داشت:

$$L.tail = L.head.prev$$

۳.۳ عملیات روی لیست‌های پیوندی

کلیه عملیاتی که در این بخش معرفی می‌شوند بر پایه لیست پیوندی دوطرفه تشریح شده است، با این حال به‌راحتی قابل تبدیل به دیگر نوع‌ها نیز می‌باشد.

- List-Search(L،k): یک پرسمان بازیابی است که برای پیدا کردن اولین عنصری که مقدار کلید آن برابر k است مورد استفاده قرار می‌گیرد. اگر عنصر در لیست موجود باشد، آن عنصر برگردانده می‌شود، در غیر اینصورت مقدار Null خروجی خواهد بود.

Algorithm 1 List-Search(L, k)

```
1: x = L.head
2: while x ≠ Null and x.key ≠ k do
3:     x = x.next
4: Return x
```

پیچیدگی زمانی الگوریتم فوق در بدترین حالت O(n) است،جایی که n تعداد نودهای لیست پیوندی L است.

- List-Insert(L،x): یک پرسمان بروزرسانی است که عنصر x را به ابتدای لیست L اضافه می‌کند.

Algorithm 2 List-Insert(L, x)

```
1: x.next = L.head
2: if (L.head ≠ Null) then
3:     L.head.prev = x
4: L.head = x
5: x.prev = Null
```

پیچیدگی زمانی الگوریتم فوق O(۱) است.