

Find, more, less

```
pNode find(pList p, int val) {
    DPRINT(cout << ">find val=" << val << endl);

    pNode curr = begin(p);
    while(curr!=end(p)&&curr->data!=val) curr=curr->next;

    return curr;
}

#endif
```

While without if statement

```
pset9 — ./driver — ./driver — driver — 80x24
u - unique*          O(n)      w - swap pairs  O(n)
t - show [HEAD/TAIL]           n - n nodes per line
c - clear             O(n)
Command[q to quit]: n
Enter number of nodes to show per line: 10
FRONT 1     2     3     4     5     6

Doubly Linked List(nodes:6, show:ALL,10)
f - push front        O(1)      p - pop front   O(1)
b - push back         O(1)      y - pop back   O(1)
B - push back N       O(n)      Y - pop back N  O(n)
i - push              O(n)      d - pop        O(n)
z - push sorted*      O(n)      e - pop all*   O(n)

s - sorted?            O(n)      r - reverse     O(n)
x - perfect shuffle* O(n)      a - randomize  O(n)
u - unique*           O(n)      w - swap pairs O(n)
t - show [HEAD/TAIL]           n - n nodes per line
c - clear             O(n)
Command[q to quit]: i
Enter a number to push: 99
Choose a position node: 4
FRONT 1     2     3     99    4     5     6
```

Testing find() with push command(i)

Push commands, push()

```

void push(pList p, int val, int x) {
    DPRINT(cout << ">push val=" << val << endl);
    pNode y=find(p,x);
    if(y->next==nullptr) return;
    insert(y,val);
    DPRINT(cout << "<push\n";);
}

```

Push() using both find()&insert(), implemented within 3 lines.

By using find(), we can get the target node that has same parameter val.

According to find(), if position is not found, it will return last node. And since last node's next is nullptr, push() doesn't add a new node and just returns without any actions.

Pop commands, pop_all()

```

x - perfect shuffle* O(n)      a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]           n - n nodes per line
c - clear            O(n)
Command[q to quit]: B
Enter N nodes to push back(-N for a value)??: -100000
Enter a value to push back?: 10
pushing [90000]=10
cpu: 0.006619 sec
FRONT 10      10      10      10      10
... 10 ... 10      10      10
Testing with 100,000 samples

Doubly Linked List(nodes:100000, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front   O(1)
b - push back       O(1)      y - pop back    O(1)
B - push back N    O(n)      Y - pop back N  O(n)
i - push            O(n)      d - pop          O(n)
z - push sorted*   O(n)      e - pop all*    O(n)

s - sorted?         O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)    w - swap pairs  O(n)
t - show [ALL]           n - n nodes per line
c - clear            O(n)
Command[q to quit]: e
Enter a number to pop all: 10
cpu: 0.006611 sec

The list is empty.

Doubly Linked List(nodes:0, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front   O(1)
b - push back       O(1)      y - pop back    O(1)

```

By using pop_all, we can get rid of every number that is same as input number. To fix the bug, I temporarily saved the node's next address before the erase().

```

temp=c->next;
erase(p,c);
c=temp;

```

Since erase() deletes c node, I set the c as saved node address. By this way, the bug doesn't occur anymore.

Half() and show()

```

x - perfect shuffle* O(n)      a - randomize   O(n)
u - unique*                   O(n)      w - swap pairs  O(n)
t - show [ALL]                 n - n nodes per line
c - clear                      O(n)
Command[q to quit]: t

Retry -          Command[q to quit]: t
FRONT  1       2       3       4
      5       6       7       8
      9

Doubly Linked List(nodes:9, show:ALL,4)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)      Y - pop back N O(n)
i - push          O(n)      d - pop        O(n)
z - push sorted*  O(n)      e - pop all*   O(n)

s - sorted?        O(n)      r - reverse    O(n)
x - perfect shuffle* O(n)    a - randomize  O(n)
u - unique*        O(n)      w - swap pairs O(n)
t - show [HEAD/TAIL]
c - clear          O(n)
Command[q to quit]: t
FRONT  1       2       3       4
... 5 ...
6       7       8       9

Doubly Linked List(nodes:9, show:HEAD/TAIL,4)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)      Y - pop back N O(n)
i - push          O(n)      d - pop        O(n)
z - push sorted*  O(n)      e - pop all*   O(n)

```

The length of list is 9, and 4 nodes are shown per list in HEAD/TAIL menu.

We can see that it works properly.

What about the case when length of list is even number?

```

z - push sorted*  O(n)      e - pop all*   O(n)

s - sorted?        O(n)      r - reverse    O(n)
x - perfect shuffle* O(n)    a - randomize  O(n)
u - unique*        O(n)      w - swap pairs O(n)
t - show [ALL]
c - clear          O(n)
Command[q to quit]: t
FRONT  1       2       3       4
      5       6       7       8
      9       10

Doubly Linked List(nodes:10, show:ALL,4)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)      Y - pop back N O(n)
i - push          O(n)      d - pop        O(n)
z - push sorted*  O(n)      e - pop all*   O(n)

s - sorted?        O(n)      r - reverse    O(n)
x - perfect shuffle* O(n)    a - randomize  O(n)
u - unique*        O(n)      w - swap pairs O(n)
t - show [HEAD/TAIL]
c - clear          O(n)
Command[q to quit]: t
FRONT  1       2       3       4
... 6 ...
7       8       9       10

Doubly Linked List(nodes:10, show:HEAD/TAIL,4)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)      Y - pop back N O(n)
i - push          O(n)      d - pop        O(n)

```

When list length is even number, the half node should be the first node of second half node.

We can see that it also work as intended.

There are two ways to implement half(). Both methods' time complexity is $O(n)$, but the second method is faster.

First method uses list's size and goes through halfway using for loop.

Let me test it for 20,000,000 samples.

```
s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]        n - n nodes per line
c - clear            O(n)
Command[q to quit]: B
Enter N nodes to push back(-N for a value)?: -20000000
Enter a value to push back?: 1
pushing [19990000]=1
cpu: 0.355146 sec
FRONT  1      1      1      1      1
... 1 ...
1      1      1      1      1

Doubly Linked List(nodes:20000000, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front   O(1)
b - push back       O(1)      y - pop back   O(1)
B - push back N    O(n)      Y - pop back N O(n)
i - push            O(n)      d - pop         O(n)
z - push sorted*   O(n)      e - pop all*   O(n)

s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]        n - n nodes per line
c - clear            O(n)
Command[q to quit]: q
Congratulations! It's half-time: cpu: 0.096726 sec

Cleared...
```

Joyful Coding~~

+ pset9

First method(for loop): 0.096726 sec

```
s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]        n - n nodes per line
c - clear            O(n)
Command[q to quit]: B
Enter N nodes to push back(-N for a value)?: -20000000
Enter a value to push back?: 1
pushing [19990000]=1
cpu: 0.351403 sec
FRONT  1      1      1      1      1
... 1 ...
1      1      1      1      1

Doubly Linked List(nodes:20000000, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front   O(1)
b - push back       O(1)      y - pop back   O(1)
B - push back N    O(n)      Y - pop back N O(n)
i - push            O(n)      d - pop         O(n)
z - push sorted*   O(n)      e - pop all*   O(n)

s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]        n - n nodes per line
c - clear            O(n)
Command[q to quit]: q
Congratulations! It's half-time: cpu: 0.059277 sec
```

Cleared...

Joyful Coding~~

+ pset9

Second method(rabbit and turtle): 0.059277 sec

So the second method is about 1.63 times faster than the first method.

sorted()

checked by other operations 😊

push_sorted()

```
s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [HEAD/TAIL]           n - n nodes per line
c - clear            O(n)
Command[q to quit]: t
FRONT 17      40      13      20      24
... 31 ...
27      44      19      24      9

Doubly Linked List(nodes:50, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front  O(1)
b - push back       O(1)      y - pop back   O(1)
B - push back N    O(n)      Y - pop back N O(n)
i - push            O(n)      d - pop        O(n)
z - push sorted*   O(n)      e - pop all*   O(n)

s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]           n - n nodes per line
c - clear            O(n)
Command[q to quit]: z
The operation works in sorted list only.
FRONT 17      40      13      20      24
... 31 ...
27      44      19      24      9

Doubly Linked List(nodes:50, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front  O(1)
b - push back       O(1)      y - pop back   O(1)
B - push back N    O(n)      Y - pop back N O(n)
i - push            O(n)      d - pop        O(n)
```

If we try `push_sorted()` with usorted list, it is not possible.

```
z - push sorted*   O(n)      e - pop all*   O(n)

s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]           n - n nodes per line
c - clear            O(n)
Command[q to quit]: s
Sorted in ascending order
FRONT 3      5      5      7      9
9      9      9

Doubly Linked List(nodes:8, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front  O(1)
b - push back       O(1)      y - pop back   O(1)
B - push back N    O(n)      Y - pop back N O(n)
i - push            O(n)      d - pop        O(n)
z - push sorted*   O(n)      e - pop all*   O(n)

s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*          O(n)      w - swap pairs  O(n)
t - show [ALL]           n - n nodes per line
c - clear            O(n)
Command[q to quit]: z
Enter a number to push: 5
cpu: 2.1e-05 sec
FRONT 3      5      5      5      9
9      9      9

Doubly Linked List(nodes:9, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front  O(1)
b - push back       O(1)      y - pop back   O(1)
B - push back N    O(n)      Y - pop back N O(n)
```

Since the list is sorted in ascending order, we can try `push_sorted()`.
We can see that 5 is properly inserted.

```

s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*         O(n)      w - swap pairs  O(n)
t - show [ALL]        n - n nodes per line
c - clear            O(n)
Command[q to quit]: s
Sorted in descending order
FRONT 9 9 9 9 7
      5 5 3

Doubly Linked List(nodes:8, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front   O(1)
b - push back       O(1)      y - pop back    O(1)
B - push back N    O(n)      Y - pop back N  O(n)
i - push            O(n)      d - pop          O(n)
z - push sorted*   O(n)      e - pop all*    O(n)

s - sorted?          O(n)      r - reverse      O(n)
x - perfect shuffle* O(n)    a - randomize   O(n)
u - unique*         O(n)      w - swap pairs  O(n)
t - show [ALL]        n - n nodes per line
c - clear            O(n)
Command[q to quit]: z
Enter a number to push: 8
cpu: 1.6e-05 sec
FRONT 9 9 9 9 8
      7 5 5 3

Doubly Linked List(nodes:9, show:HEAD/TAIL,5)
f - push front      O(1)      p - pop front   O(1)
b - push back       O(1)      y - pop back    O(1)
B - push back N    O(n)      Y - pop back N  O(n)
i - push            O(n)      d - pop          O(n)

```

Since the list is sorted in descending order, we can try `push_sorted()`.
We can see that 8 is inserted properly.

<pre> s - sorted? O(n) r - reverse O(n) x - perfect shuffle* O(n) a - randomize O(n) u - unique* O(n) w - swap pairs O(n) t - show [ALL] n - n nodes per line c - clear O(n) Command[q to quit]: s Sorted in ascending order FRONT 1 2 2 2 2 ... 2 ... 2 2 2 2 2 Doubly Linked List(nodes:100000, show:HEAD/TAIL,5) f - push front O(1) p - pop front O(1) b - push back O(1) y - pop back O(1) B - push back N O(n) Y - pop back N O(n) i - push O(n) d - pop O(n) z - push sorted* O(n) e - pop all* O(n) s - sorted? O(n) r - reverse O(n) x - perfect shuffle* O(n) a - randomize O(n) u - unique* O(n) w - swap pairs O(n) t - show [ALL] n - n nodes per line c - clear O(n) Command[q to quit]: z Enter a number to push: 3 cpu: 0.001376 sec FRONT 1 2 2 2 2 ... 2 ... 2 2 2 2 3 </pre>	<pre> s - sorted? O(n) r - reverse O(n) x - perfect shuffle* O(n) a - randomize O(n) u - unique* O(n) w - swap pairs O(n) t - show [ALL] n - n nodes per line c - clear O(n) Command[q to quit]: s Sorted in descending order FRONT 3 2 2 2 2 ... 2 ... 2 2 2 2 2 Doubly Linked List(nodes:100000, show:HEAD/TAIL,5) f - push front O(1) p - pop front O(1) b - push back O(1) y - pop back O(1) B - push back N O(n) Y - pop back N O(n) i - push O(n) d - pop O(n) z - push sorted* O(n) e - pop all* O(n) s - sorted? O(n) r - reverse O(n) x - perfect shuffle* O(n) a - randomize O(n) u - unique* O(n) w - swap pairs O(n) t - show [ALL] n - n nodes per line c - clear O(n) Command[q to quit]: z Enter a number to push: 1 cpu: 0.003615 sec FRONT 3 2 2 2 2 ... 2 ... 2 2 2 2 1 </pre>
<pre> Doubly Linked List(nodes:100001, show:HEAD/TAIL,5) f - push front O(1) p - pop front O(1) b - push back O(1) y - pop back O(1) </pre>	

Result of testing with 100,000 samples.

unique()

```

u - unique*      O(n)      w - swap pairs  O(n)
t - show [ALL]          n - n nodes per line
c - clear        O(n)
Command[q to quit]: z
Enter a number to push: 1
cpu: 0.003615 sec
FRONT   3      2      2      2      2
... 2 ...
2      2      2      2      1

Doubly Linked List(nodes:100001, show:HEAD/TAIL,5)
f - push front    O(1)      p - pop front   O(1)
b - push back     O(1)      y - pop back    O(1)
B - push back N   O(n)      Y - pop back N  O(n)
i - push         O(n)      d - pop         O(n)
z - push sorted*  O(n)      e - pop all*    O(n)

s - sorted?       O(n)      r - reverse     O(n)
x - perfect shuffle* O(n)  a - randomize  O(n)
u - unique*       O(n)      w - swap pairs  O(n)
t - show [ALL]          n - n nodes per line
c - clear        O(n)
Command[q to quit]: u
cpu: 0.003851 sec
FRONT   3      2      1

Doubly Linked List(nodes:3, show:HEAD/TAIL,5)
f - push front    O(1)      p - pop front   O(1)
b - push back     O(1)      y - pop back    O(1)
B - push back N   O(n)      Y - pop back N  O(n)
i - push         O(n)      d - pop         O(n)
z - push sorted*  O(n)      e - pop all*    O(n)

s - sorted?       O(n)      r - reverse     O(n)

```

Testing with one '3', 99,999 '2's, one '1'.

We can see that all the '2's are wiped out except one remainder. The existing bug is fixed by the same method

of pop_all().

```

z - push sorted*  O(n)      e - pop all*   O(n)

s - sorted?       O(n)      r - reverse     O(n)
x - perfect shuffle* O(n)  a - randomize  O(n)
u - unique*       O(n)      w - swap pairs  O(n)
t - show [ALL]          n - n nodes per line
c - clear        O(n)
Command[q to quit]: b
Enter a number to push: 9
FRONT   3      5      5      7      9
9      9      9

Doubly Linked List(nodes:8, show:HEAD/TAIL,5)
f - push front    O(1)      p - pop front   O(1)
b - push back     O(1)      y - pop back    O(1)
B - push back N   O(n)      Y - pop back N  O(n)
i - push         O(n)      d - pop         O(n)
z - push sorted*  O(n)      e - pop all*    O(n)

s - sorted?       O(n)      r - reverse     O(n)
x - perfect shuffle* O(n)  a - randomize  O(n)
u - unique*       O(n)      w - swap pairs  O(n)
t - show [ALL]          n - n nodes per line
c - clear        O(n)
Command[q to quit]: u
cpu: 1.8e-05 sec
FRONT   3      5      7      9

Doubly Linked List(nodes:4, show:HEAD/TAIL,5)
f - push front    O(1)      p - pop front   O(1)
b - push back     O(1)      y - pop back    O(1)
B - push back N   O(n)      Y - pop back N  O(n)
i - push         O(n)      d - pop         O(n)
z - push sorted*  O(n)      e - pop all*    O(n)

```

We can see that all duplicate number is wiped out.

11. shuffle()

```

b - push back      O(1)      y - pop back    O(1)
B - push back N   O(n)       Y - pop back N  O(n)
i - push          O(n)       d - pop        O(n)
z - push sorted*  O(n)       e - pop all*   O(n)

s - sorted?        O(n)       r - reverse    O(n)
x - perfect shuffle* O(n)   a - randomize  O(n)
u - unique*        O(n)       w - swap pairs O(n)
t - show [ALL]           n - n nodes per line
c - clear          O(n)
Command[q to quit]: n
Enter number of nodes to show per line: 8
FRONT 1 2 3 4 5 6

Doubly Linked List(nodes:6, show:HEAD/TAIL,8)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)       Y - pop back N O(n)
i - push          O(n)       d - pop        O(n)
z - push sorted*  O(n)       e - pop all*   O(n)

s - sorted?        O(n)       r - reverse    O(n)
x - perfect shuffle* O(n)   a - randomize  O(n)
u - unique*        O(n)       w - swap pairs O(n)
t - show [ALL]           n - n nodes per line
c - clear          O(n)
Command[q to quit]: x
cpu: 1.8e-05 sec
FRONT 4 1 5 2 6 3

Doubly Linked List(nodes:6, show:HEAD/TAIL,8)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)       Y - pop back N O(n)

```

When list length is even number

```

B - push back N   O(n)       Y - pop back N  O(n)
i - push          O(n)       d - pop        O(n)
z - push sorted*  O(n)       e - pop all*   O(n)

s - sorted?        O(n)       r - reverse    O(n)
x - perfect shuffle* O(n)   a - randomize  O(n)
u - unique*        O(n)       w - swap pairs O(n)
t - show [ALL]           n - n nodes per line
c - clear          O(n)
Command[q to quit]: y
FRONT 1 2 3 4 5

Doubly Linked List(nodes:5, show:HEAD/TAIL,8)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)       Y - pop back N O(n)
i - push          O(n)       d - pop        O(n)
z - push sorted*  O(n)       e - pop all*   O(n)

s - sorted?        O(n)       r - reverse    O(n)
x - perfect shuffle* O(n)   a - randomize  O(n)
u - unique*        O(n)       w - swap pairs O(n)
t - show [ALL]           n - n nodes per line
c - clear          O(n)
Command[q to quit]: x
cpu: 8e-06 sec
FRONT 3 1 4 2 5

Doubly Linked List(nodes:5, show:HEAD/TAIL,8)
f - push front    O(1)      p - pop front  O(1)
b - push back     O(1)      y - pop back   O(1)
B - push back N   O(n)       Y - pop back N O(n)
i - push          O(n)       d - pop        O(n)
z - push sorted*  O(n)       e - pop all*   O(n)

```

When list length is even number