

Yiyun Zhang
7/14/2016

2.2

a. array

```
8
9  #include <iostream>
10 using namespace std;
11
12 //insert
13 void insert(int x,int p, int* L){
14     int maxlength, last;
15     if(last>=maxlength){
16         cout << "list is full" << endl;
17     }
18     else if(p>last+1 || p<1){
19         cout << "position does not exists" << endl;
20     }
21     else{
22         for(int q=last;q>=p;q--){
23             L[q+1] = L[q];
24             last += 1;
25             L[p] = x;
26         }
27     }
28 }
29
30 //delete
31 void del(int p, int* L){
32     int maxlength, last;
33     if(p>=maxlength || p<0){
34         cout << "position does not exist"<< endl;
35     }
36     else{
37         last -= 1;
38         for(int q=p;q<=last;q++){
39             L[q] = L[q+1];
40         }
41     }
42 }
43
44 //locate
45 int locate(int x, int* L){
46     int last;
47     for(int q=1;q<=last;q++){
48         if(L[q] == x){
49             return q;
50         }
51     }
52     return last+1;
53 }
```

Running time:

insert: $O(1)$

delete: $O(1)$

locate: $O(n)$

b. pointer

```
2 // pointer.cpp
3 // CS260
4 //
5 // Created by Yiyun Zhang on 7/14/16.
6 // Copyright © 2016 Yiyun Zhang. All rights reserved.
7 //
8
9 #include <iostream>
10 using namespace std;
11
12
13
14 struct node {
15     int element;
16     node *next;
17 };
18
19 //insert
20 void insert(int x, node *p){
21     node *temp;
22     temp = p->next;
23     p->next = new node;
24     p->next->element = x;
25     p->next->next = temp;
26 }
27
28 //delete
29 void del(node *p){
30     p->next = p->next->next;
31 }
32
33 //locate
34 node* locate(int x, node *L){
35     node *temp;
36     temp = L;
37     while(temp->next != NULL){
38         if(temp->next->element == x){
39             return temp;
40         }
41         else{
42             temp = temp->next;
43         }
44     }
45     return temp;
46 }
47
```

Running time:

insert: $O(1)$

delete: $O(1)$

locate: $O(n)$

c. cursor-based

```
8
9  #include <iostream>
10 using namespace std;
11
12 struct node {
13     int element;
14     int *next;
15 };
16
17 bool move(int p, int q){
18     int temp;
19     node *space;
20     if(p==0){
21         cout << "cell does not exist";
22         return false;
23     }
24     else{
25         temp = q;
26         q = p;
27         p = *space[q].next;
28         space[q].next = &temp;
29         return true;
30     }
31     return false;
32 }
33
34 //insert
35 void insert(int x, int p, int* L){
36     node *space;
37     if(p==0){
38         if(move(p,*L)){
39             space[*L].element= x;
40         }
41     }
42     else{
43         if(move(p,*space[p].next)){
44             space[*space[p].next].element = x;
45         }
46     }
47 }
48
49 //delete
50 void del(int p, int *L){
51     node *space;
52     if(p==0){
53         move(*L,p);
54     }
55     else{
56         move(*space[p].next,p);
57     }
58 }
```

Running time:

insert: $O(n)$

delete: $O(n)$

2.4

```
concatenate.py
1  def concatenate(L):
2      newL = []
3      for x in L:
4          for a in x:
5              newL.append(a)
6      return newL
```

2.11

p := FIRST(L);	1
while p <> END(L) do begin	n
q := p;	
while q <> END(L) do begin	n
q := NEXT(q, L);	n-q
r := FIRST(L);	1
while r <> q do	
r := NEXT(r, L)	n-r
end;	
p := NEXT(p, L)	n-p
End;	

FIRST: $1 \cdot n \cdot n \cdot 1 = O(n^2)$ times

NEXT: $((n-r+n-q) \cdot n) + n \cdot p = O(n^3)$ times

END: $n \cdot n = O(n^2)$ times

3.10

```
level-order.py
1  class Node:
2      def __init__(self, a):
3          self.data = a
4          self.left = None
5          self.right = None
6
7  def levelorder(t):
8      h = height(t)
9      for i in range(1, h+1):
10         currentlevel(t, i)
11
12  def currentlevel(x, y):
13      if x is None:
14          return
15      if y == 1:
16          print "%d" %(x.data)
17      elif y > 1 :
18          currentlevel(x.left, y-1)
19          currentlevel(x.right, y-1)
20
21  def height(t):
22      if t is None:
23          return 0
24      else :
25          lefth = height(t.left)
26          righth = height(t.right)
27          if lefth < righth :
28              return righth+1
29          else:
30              return lefth+1
```