

(坐标统计)输入n个整点在平面上的坐标。对于每个点,可以控制所有位于它左下方的点(即x、y坐标都比它小),它可以控制的点的数目称为"战斗力"。依次输出每个点的战斗力,最后输出战斗力最高的点的编号(如果若干个点的战斗力并列最高,输出其中最大的编号)

#### 输入样例:

3

1 1

2 2

3 4

#### 输出多少?

输出样例:

0

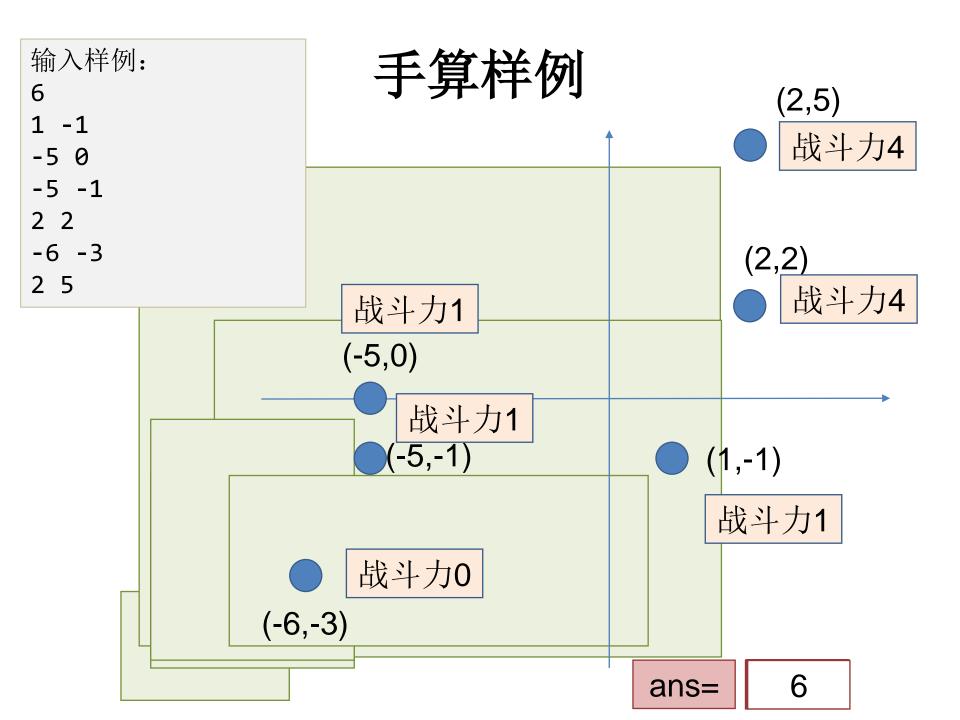
1

7

3

```
输入样例:
6
1 -1
-5 0
-5 -1
2 2
-6 -3
2 5
```

```
输出多少?
输出样例:
1
1
4
0
4
6
```



```
5分钟
```

```
#include <iostream>
                         完善程序
   using namespace std;
   const int SIZE=100;
                                                      识别变量
   int x[SIZE],y[SIZE],f[SIZE];
   int n,i,j,max_f,ans;
                                                  常见变量名
   int main(){
                                                 翻译循环变量
7
       cin>>n;
                                              根据变量名的英文推断
       for(i=1;i<=n;i++) cin>>x[i]>>y[i];
8
       max f=0;
       for(i=1;i<=n;i++){</pre>
10 🗎
           f[i] = (1)
11
                                               2
                                                    找出关键语句
12 =
           for(j=1;j<=n;j++){
              if(x[j]<x[i]&&____(2)___
13
                                                控制结构(for, if)
14
                                              常见算法的基本操作
15
                                               函数参数、返回值
           if(____(4)___){
16 🖹
              max_f=f[i];
17
               (5) ;
18
19
20
                                               3
                                                   理解代码段作用
       for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
21
       cout<<ans<<endl;
22
                                                翻译解释代码段
23
       return 0;
24
```

```
#include <iostream>
                         完善程序
   using namespace std;
   const int SIZE=100;
   int x[SIZE],y[SIZE],f[SIZE];
   int n,i,j,max f,ans;
   int main(){
                                             n表示点的数量
       cin>>n;
8
       for(i=1;i<=n;i++) cin>>x[i]>>y[i];
                                         x[i]表示第i个点的横坐标
       max f=0;
       for(i=1;i<=n;i++){
10 E
          f[i] = (1)
11
                                         y[i]表示第i个点的纵坐标
12 =
          for(j=1;j<=n;j++){
              if(x[j]<x[i]&&_</pre>
13
14
                                         f[i]表示i号点的战斗力值
15
16 🖹
                                         max_f表示最大战斗力值
              max_f=f[i];
17
                  (5) ;
18
19
                                         ans表示目前最大战斗力
20
                                               的点的编号
21
       for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
22
       cout<<ans<<endl;
23
       return 0;
```

24

```
#include <iostream>
                             完善程序
    using namespace std;
    const int SIZE=100;
    int x[SIZE],y[SIZE],f[SIZE];
    int n,i,j,max_f,ans;
    int main(){
7
        cin>>n;
        for(i=1;i<=n;i++) cin>>x[i]>>y[i];
8
 9
        max f=0;
        for(i=1;i<=n;i++){</pre>
10 🖨
            f[i] = (1)
11
12 E
            for(j=1;j<=n;j++){
                if(x[j]<x[i]&&____(2)___
13
14
15
16 🖹
                max_f=f[i];
17
                 (5) ;
18
19
20
21
        for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
22
        cout<<ans<<endl;
23
        return 0;
24
```

将每个点i的横纵坐标分 别读取数组x[i]和y[i]

```
#include <iostream>
                              完善程序
    using namespace std;
    const int SIZE=100;
    int x[SIZE],y[SIZE],f[SIZE];
    int n,i,j,max_f,ans;
    int main(){
7
        cin>>n;
        for(i=1;i<=n;i++) cin>>x[i]>>y[i];
8
9
        max f=0;
10 🖨
        for(i=1;i<=n;i++){
            f[i]=___(1)_
11
12 =
            for(j=1;j<=n;j++){
                 if(x[j]<x[i]&&___</pre>
13
14
15
             if(
                      (4)
16 🖹
17
                 max_f=f[i];
                      (5) ;
18
19
20
        for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
21
22
        cout<<ans<<endl;
23
        return 0;
24
```

对每个点i 都将位于它**左下方的点** 的个数统计到f[i]中 表示"战斗力"

```
#include <iostream>
                          完善程序
    using namespace std;
    const int SIZE=100;
    int x[SIZE],y[SIZE],f[SIZE];
    int n,i,j,max_f,ans;
    int main(){
7
       cin>>n;
8
       for(i=1;i<=n;i++) cin>>x[i]>>y[i];
       max f=0;
10 E
       for(i=1;i<=n;i++){
           f[i] = (1)
11
12 =
           for(j=1;j<=n;j++){
               if(x[j]<x[i]&&__</pre>
13
                                 (2)
                                                   打擂台法
14
15
                                            如果当前点"战斗力"
           if(
                 (4)
16 🖹
                                               超过擂主值max f
               max_f=f[i];
17
18
                    (5) ;
19
                                               更新擂主值max_f
20
                                                并更新擂主编号
21
       for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
22
       cout<<ans<<endl;
23
       return 0;
```

24

```
#include <iostream>
                          完善程序
    using namespace std;
    const int SIZE=100;
    int x[SIZE],y[SIZE],f[SIZE];
    int n,i,j,max f,ans;
   int main(){
                                            将每个点i的横纵坐标
       cin>>n;
                                                输入到x[i]和y[i]
       for(i=1;i<=n;i++) cin>>x[i]>>y[i];
8
       max f=0;
       for(i=1;i<=n;i++){
10 🗎
                                                   双重循环
           f[i] = (1)
11
           for(j=1; j<=n; j++){</pre>
12 E
                                                统计每个点i的
               if(x[j]<x[i]&&_</pre>
13
                                              左下方点的个数f[i]
14
15
                   (4)
16 🖹
                                                   打擂台法
              max_f=f[i];
17
                 (5)____;
18
                                              维护擂主值和编号
19
20
                                                输出每个点ffil
       for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
21
       cout<<ans<<endl;
22
                                            并输出最大值编号ans
23
       return 0;
24
```

```
完善程序
    #include <iostream>
    using namespace std;
    const int SIZE=100;
4
    int x[SIZE],y[SIZE],f[SIZE];
    int n,i,j,max f,ans;
    int main(){
7
        cin>>n;
8
        for(i=1;i<=n;i++) cin>>x[i]>>y[i];
                                                战斗力值初始化为0
9
        max f=0;
        for(i=1;i<=n;i++){
10 🖨
            f[i]=___(1)_
11
12 E
            for(j=1;j<=n;j++){
                if(x[j]<x[i]&&____(2)___
13
14
15
16 🖨
               max_f=f[i];
17
                 (5)
18
19
20
21
        for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
        cout<<ans<<endl;
22
23
        return 0;
24
```

```
#include <iostream>
                             完善程序
    using namespace std;
    const int SIZE=100;
4
    int x[SIZE],y[SIZE],f[SIZE];
    int n,i,j,max_f,ans;
    int main(){
7
        cin>>n;
8
        for(i=1;i<=n;i++) cin>>x[i]>>y[i];
9
        max f=0;
        for(i=1;i<=n;i++){</pre>
10 🖨
            f[i] = (1)
11
12 E
            for(j=1;j<=n;j++){
                if(x[j]<x[i]&&____(2)
13
14
15
16 🖨
                max_f=f[i];
17
                  (5) ;
18
19
20
21
        for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
22
        cout<<ans<<endl;
23
        return 0;
24
```

点j需要同时满足 横纵坐标值都比点i小

```
#include <iostream>
                            完善程序
    using namespace std;
    const int SIZE=100;
    int x[SIZE],y[SIZE],f[SIZE];
4
    int n,i,j,max f,ans;
    int main(){
7
        cin>>n;
8
        for(i=1;i<=n;i++) cin>>x[i]>>y[i];
9
        max f=0;
        for(i=1;i<=n;i++){
10 🖨
            f[i] = (1)
11
12 E
            for(j=1;j<=n;j++){
                if(x[j]< x[i]&&___(2)
13
14
15
16 🖨
                max_f=f[i];
17
                 (5) ;
18
19
20
21
        for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
22
        cout<<ans<<endl;
23
        return 0;
24
```

如果符合条件

在左下方

那么f[i]就自增1

```
#include <iostream>
                            完善程序
    using namespace std;
    const int SIZE=100;
    int x[SIZE],y[SIZE],f[SIZE];
    int n,i,j,max f,ans;
    int main(){
7
        cin>>n;
8
        for(i=1;i<=n;i++) cin>>x[i]>>y[i];
9
        max f=0;
        for(i=1;i<=n;i++){
10 =
            f[i] = (1)
11
12 =
            for(j=1;j<=n;j++){
                if(x[j]<x[i]&&____(2)_
13
14
15
            if( (4) ){
16 🖹
                max_f=f[i];
17
                 (5) ;
18
19
20
21
        for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
22
        cout<<ans<<endl;
23
        return 0;
```

24

打擂台 条件是目前点的武力值f[i] 大于等于擂主值max\_f

思考: 等号可不可省略

如果若干个点的战斗力并列 最高,输出其中最大的编号

```
#include <iostream>
                             完善程序
    using namespace std;
    const int SIZE=100;
    int x[SIZE],y[SIZE],f[SIZE];
4
    int n,i,j,max_f,ans;
    int main(){
7
        cin>>n;
8
        for(i=1;i<=n;i++) cin>>x[i]>>y[i];
9
        max f=0;
        for(i=1;i<=n;i++){</pre>
10 🖨
            f[i] = (1)
11
12 E
            for(j=1;j<=n;j++){
                if(x[j] < x[i] & (2)
13
14
15
16 🖹
                max_f=f[i];
17
                 (5)
18
19
20
21
        for(i=1;i<=n;i++) cout<<f[i]<<endl;</pre>
22
        cout<<ans<<endl;
23
        return 0;
24
```

观察22行输出了ans 结合题意 要求输出 "战斗力最高点的编号" 可知ans需要维护最大值 编号

(**子矩阵**)输入一个n1\*m1的矩阵a,和n2\*m2的矩阵b,问a中是否存在子矩阵和b相等。若存在,输出所有子矩阵左上角的坐标;若不存在输出"There is no answer"。

#### 输入样例:

- 4 4
- 1 2 3 4
- 3 4 1 4
- 2 2 3 4
- 1 4 1 4
- 2 3
- 2 3 4
- 4 1 4

1,2

1	2	3	4
3	4	1	4
2	2	3	4
1	4	1	4

2	3	4
4	1	4

输	出多少?
输	出样例:
1	2
3	2

3,2

#### 输入样例:

3 3

1 2 3

3 4 1

2 2 3

2 2

1 3

2 4

#### 输出多少?

输出样例:

There is no answer

```
1 #include <iostream>
 2 using namespace std;
 3 const int SIZE = 50;
 4 int n1, m1, n2, m2, a[SIZE][SIZE], b[SIZE][SIZE];
 5 int main(){
        int i, j, k1, k2;
        bool good, haveAns;
        cin >> n1 >> m1;
        for(i = 1; i <= n1; i++)
            for(j = 1; j \leftarrow m1; j++)
10
                cin >> a[i][j];
11
12
        cin >> n2 >> m2;
        for(i = 1; i <= n2; i++)
13
            for(j = 1; j \leftarrow m2; j++)
14
15
                  (1) \quad ;
        haveAns = false:
16
17
        for(i = 1; i \leftarrow n1 - n2 + 1; i++)
            for(j = 1; j \leftarrow (2)_{j++}
18 ₽
19
                 ___(3)___;
                 for(k1 = 1; k1 \leftarrow n2; k1++)
20
                     for (k2 = 1; k2 \leftarrow (4); k2++)
21 □
                         if(a[i + k1 - 1][j + k2 - 1] != b[k1][k2])
22
                              good = false;
23
24
                 if(good){
25 ₱
                     cout << i << ' ' << j << endl;</pre>
26
                     _(5)___;
27
28
29
        if(!haveAns)
30
            cout << "There is no answer" << endl;</pre>
31
32
        return 0;
33 L
```

1 识别变量

常见变量名 翻译循环变量 根据变量名的英文推断

2 找出关键语句

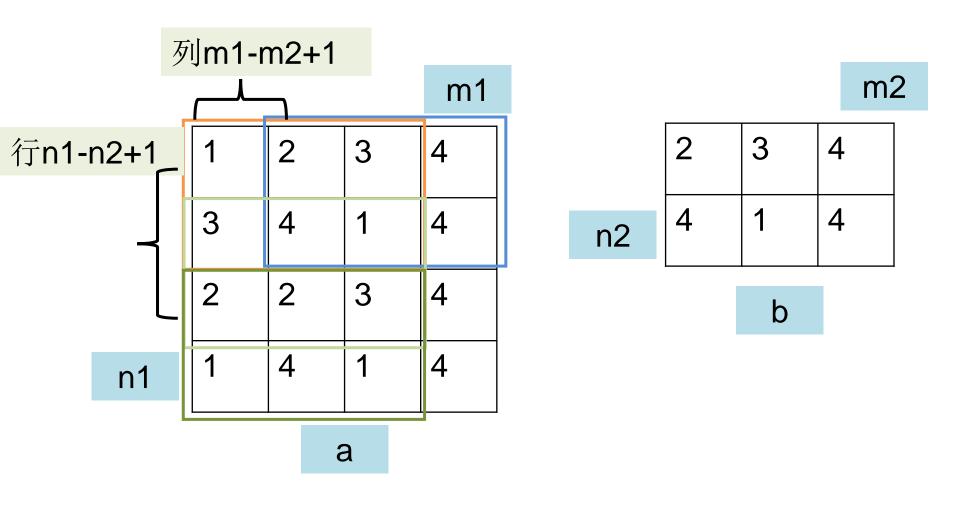
控制结构(for, if) 常见算法的基本操作 函数参数、返回值

3 理解代码段作用

翻译解释代码段

```
1 #include <iostream>
 2 using namespace std;
                                                                      解释变量的作用
 3 const int SIZE = 50;
 4 int n1, m1, n2, m2, a[SIZE][SIZE], b[SIZE][SIZE];
 5 int main(){
      int i, j, k1, k2;
                                                        n1,m1表示a矩阵行、列数
      bool good, haveAns;
      cin >> n1 >> m1;
      for(i = 1; i <= n1; i++)
                                                        n2,m2表示b矩阵行、列数
          for(j = 1; j <= m1; j++)
10
             cin >> a[i][j];
11
12
      cin >> n2 >> m2;
13
      for(i = 1; i <= n2; i++)
                                       haveANS表示是否存在b是a的子矩阵
          for(j = 1; j \leftarrow m2; j++)
14
             (1) \quad ;
15
      haveAns = false:
16
                                                 good表示a的当前子矩阵是否和
      for(i = 1; i \leftarrow n1 - n2 + 1; i++)
17
          for(j = 1; j \leftarrow (2)_{j++}
18 ∮
                                                                 b矩阵相同
             (3)
19
             for(k1 = 1; k1 \leftarrow n2; k1++)
20
21 ₽
                 for(k2 = 1; k2 \leftarrow (4); k2++){
22
                    if(a[i + k1 - 1][j + k2 - 1] != b[k1][k2])
23
                        good = false;
24
                                                             i,j,k1,k2循环变量
25 ∮
             if(good){
                 cout << i << ' ' << j << endl;</pre>
26
27
28
29
      if(!haveAns)
30
          cout << "There is no answer" << endl;</pre>
32
      return 0;
33 <sup>L</sup>
```

### 变量解释



```
1 #include <iostream>
2 using namespace std;
3 const int SIZE = 50;
                                                                         关键语句
4 int n1, m1, n2, m2, a[SIZE][SIZE], b[SIZE][SIZE];
5 int main(){
      int i, j, k1, k2;
      bool good, haveAns;
 8
      cin >> n1 >> m1;
      for(i = 1; i <= n1; i++)
                                                                     循环枚举:
10
          for(j = 1; j \leftarrow m1; j++)
11
              cin >> a[i][j];
                                                                 i和j在枚举什么?
      cin >> n2 >> m2;
12
      for(i = 1; i \le n2; i++)
13
                                                              k1和k2又在循环什么
          for(j = 1; j <= m2; j++)
14
15
              (1) \quad ;
      haveAns = false;
16
      for(i = 1; i <= n1 - n2 + 1; i++)
17
          for(j = 1; j \leftarrow (2)_{j++}
18 ₽
19
              (3) ;
              for(k1 = 1; k1 \leftarrow n2; k1++)
20
                 for(k2 = 1; k2 \leftarrow (4)_{ }; k2++){}
21 □
22
                     if(a[i + k1 - 1][j + k2 - 1] != b[k1][k2])
                         good = false;
23
24
25 ₽
              if(good){
                                                  i,i枚举子矩阵的左上角坐标
26
                 cout << i << ' ' << j << endl;</pre>
                   (5)___;
27
28
                                            k1,k2循环查看子矩阵中各坐标
29
      if(!haveAns)
30
31
          cout << "There is no answer" << endl;
32
      return 0;
33 <sup>[</sup> }
```

```
1 #include <iostream>
2 using namespace std;
 3 const int SIZE = 50;
4 int n1, m1, n2, m2, a[SIZE][SIZE], b[SIZE][SIZE];
5 int main(){
        int i, j, k1, k2;
        bool good, haveAns;
        cin >> n1 >> m1;
 8
        for(i = 1; i <= n1; i++)
10
            for(j = 1; j \leftarrow m1; j++)
11
                cin >> a[i][j];
        cin >> n2 >> m2;
12
        for(i = 1; i \le n2; i++)
13
            for(j = 1; j \leftarrow m2; j++)
14
15
                 ____(1)___;
        haveAns = false;
16
        for(i = 1; i \leftarrow n1 - n2 + 1; i++)
17
            for(j = 1; j \leftarrow (2)_{j++}
18 =
19
                 (3) ;
                 for(k1 = 1; k1 \leftarrow n2; k1++)
20
                     for(k2 = 1; k2 \leftarrow (4)_{ }; k2++){}
21 □
22
                          if(a[i + k1 - 1][j + k2 - 1] != b[k1][k2])
                              good = false;
23
24
25 ₽
                if(good){
                     cout << i << ' ' << j << endl;</pre>
26
27
28
29
        if(!haveAns)
30
            cout << "There is no answer" << endl;</pre>
31
32
        return 0;
33 <sup>[</sup> }
```

#### 关键语句

good=true说明什 么?

说明找到a矩阵中和b矩阵匹配情况。

题目中有"若不存在则输出……" 因此需要记录存在 匹配情况以便后续 分类输出。

```
1 #include <iostream>
2 using namespace std;
 3 const int SIZE = 50;
 4 int n1, m1, n2, m2, a[SIZE][SIZE], b[SIZE][SIZE];
 5 int main(){
        int i, j, k1, k2;
        bool good, haveAns;
 8
       cin >> n1 >> m1:
       for(i = 1; i <= n1; i++)
            for(j = 1; j \leftarrow m1; j++)
10
                cin >> a[i][j];
11
12
        cin >> n2 >> m2;
13
        for(i = 1; i <= n2; i++)
            for(j = 1; j \leftarrow m2; j++)
14
15
       haveAns = false:
16
        for(i = 1; i <= n1 - n2 + 1; i++)
17
            for(j = 1; j \leftarrow (2) ; j++){
18 ₽
19
                 ___(3)___;
                 for(k1 = 1; k1 \leftarrow n2; k1++)
20
                     for(k2 = 1; k2 \leftarrow (4) ; k2++){
21₽
                         if(a[i + k1 - 1][j + k2 - 1] != b[k1][k2])
22
                              good = false;
23
24
25 ₽
                 if(good){
26
                     cout << i << ' ' << j << endl;</pre>
27
                     (5) ;
28
29
30
        if(!haveAns)
            cout << "There is no answer" << endl;</pre>
31
32
        return 0;
33 <sup>L</sup>
```

#### 代码段作用

输入矩阵a、 矩阵b

枚举每个子矩阵进行匹配

输出子矩阵左 上角坐标

没有匹配的情况单独考虑

(二叉查找树)二叉查找树具有如下性质:每个节点的值都大于其左子树上所有节点的值、小于其右子树上所有节点的值。试判断一棵树是否为二叉查找树。输入的第一行包含一个整数 n,表示这棵树有 n 个顶点,编号分别为 1,2,...,n其中编号为 1 的为根结点。之后的第 i 行有三个数 value,left\_child,right\_child,分别表示该节点关键字的值、左子节点的编号、右子节点的编号;如果不存在左子节点或右子节点,则用 0 代替。输出 1 表示这棵树是二叉查找树,输出 0 则表示不是。

输入样例:

3

20 2 3

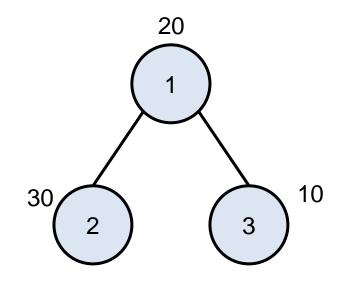
30 0 0

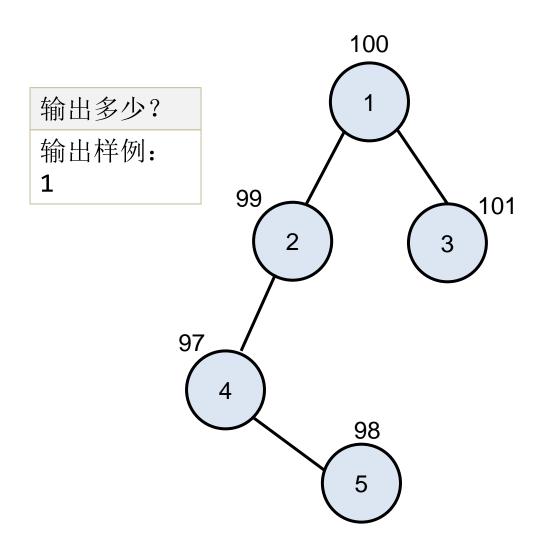
10 0 0

输出多少?

输出样例:

0





```
#include <iostream>
    using namespace std;
    const int SIZE=100;
    const int INFINITE=1000000;
    struct node {int left child, right child, value;};
    node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
        if(root==0) return 1;
8
        int cur=a[root].value;
10
        if( (cur>lower_bound) && ( (1) ) &&
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is bst( (2) , __(3) , __(4) __)==1) )
            return 1;
13
14
        return 0;
16 ☐ int main(){
        int i,n;
17
        cin>>n;
18
        for(i=1;i<=n;i++)
19
20
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
        cout<<is bst( (5) , -INFINITE, INFINITE);</pre>
21
        return 0;
```

1 识别变量

常见变量名 翻译循环变量 根据变量名的英文推断

2 找出关键语句

控制结构(for, if) 常见算法的基本操作 函数参数、返回值

3 理解代码段作用

翻译解释代码段

```
a[i]表示第i号结点
   #include <iostream>
   using namespace std;
                                                      每个结点有左儿子编号、
   const int SIZE=100;
                                                      右儿子编号、值三个属性
   const int INFINITE=1000000;
   struct node {int left child, right child, value;};
   node a[SIZE];
                                                      is bst判断以root为根的
7 ☐ int is_bst(int root, int lower_bound, int upper_bound) {
                                                       子树是否是二叉查找树
      if(root==0) return 1;
      int cur=a[root].value;
10
      if( (cur>lower_bound) && ( (1) ) &&
                                                         root指根结点编号
      (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
      (is bst( (2) , __(3) , __(4)__)==1) )
          return 1;
13
                                                      lower_bound表示结点值
14
      return 0;
                                                           应满足的下限
16 ☐ int main(){
                                                      upper_bound表示结点值
      int i,n;
17
      cin>>n;
18
                                                           应满足的上限
      for(i=1;i<=n;i++)
19
20
          cin>>a[i].value>>a[i].left_child>>a[i].right_child;
      cout<<is_bst(___(5)___, -INFINITE, INFINITE);</pre>
21
      return 0;
                                                       cur是本层结点root的值
```

```
#include <iostream>
    using namespace std;
    const int SIZE=100;
    const int INFINITE=1000000;
    struct node {int left_child, right_child, value;};
    node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
        if(root==0) return 1;
8
        int cur=a[root].value;
10
        if( (cur>lower bound) && ( (1) ) &&
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is bst( (2) , (3) , (4) )==1) )
13
            return 1;
14
        return 0;
15 L
16 ☐ int main(){
        int i,n;
17
18
        cin>>n;
        for(i=1;i<=n;i++)
19
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
20
        cout<<is bst( (5) , -INFINITE, INFINITE);</pre>
21
        return 0;
```

定义结构体node 包含其左儿子编号、右儿 子编号、值三个属性 定义结构体数组a并输入

```
#include <iostream>
    using namespace std;
    const int SIZE=100;
    const int INFINITE=1000000;
    struct node {int left child, right child, value;};
    node a[SIZE];
7 int is_bst(int root, int lower_bound, int upper_bound) {
8
        if(root==0) return 1;
        int cur=a[root].value;
        if( (cur>lower_bound) && ( (1) ) &&
10
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is bst( (2) , (3) , (4) )==1) )
13
            return 1;
14
        return 0;
15 L
16 ☐ int main(){
        int i,n;
17
        cin>>n;
18
19
        for(i=1;i<=n;i++)
20
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
        cout<<is_bst(___(5)___, -INFINITE, INFINITE);</pre>
21
        return 0;
```

is\_bst函数是递归调用 三个参数分别表示本层结点 编号、上限值、下限值

```
#include <iostream>
    using namespace std;
    const int SIZE=100;
    const int INFINITE=1000000;
    struct node {int left child, right child, value;};
    node a[SIZE];
7 □ int is bst(int root, int lower_bound, int upper_bound) {
       if(root==0) return 1;
8
        int cur=a[root].value;
10
        if( (cur>lower bound) && ( (1) ) &&
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
                       (3) (4) )==1) )
        (is bst(
13
            return 1:
14
       return 0;
16 ☐ int main(){
        int i,n;
17
        cin>>n;
18
       for(i=1;i<=n;i++)
19
20
           cin>>a[i].value>>a[i].left_child>>a[i].right_child;
        cout<<is bst( (5) , -INFINITE, INFINITE);</pre>
21
       return 0;
```

is\_bst函数有三个返回语句

首先当**root==0** 叶子结点上一层没有左孩子 或右孩子结点 结束递归直接返回给上一层**1** 

其次当满足if中条件时返回1 否则返回0

#include <iostream>

# 完善程序

```
using namespace std;
   const int SIZE=100;
   const int INFINITE=1000000;
   struct node {int left child, right child, value;};
   node a[SIZE];
                                                            观察if成立的条件
7□ int is_bst(int root, int lower_bound, int upper_bound) {
                                                         ① 需要结点值大于下限
       if(root==0) return 1;
8
                                                         ② 需要结点值小于上限
       int cur=a[root].value;
       if( (cur>lower bound) && ( (1) ) &&
10
                                                       ③ 调用is_bst函数,将左子
       (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
                                                       结点编号传入,需要返回1
       (is_bst(___(2)___,
                                                       ④调用is_bst函数,将右子
          return 1;
13
       return 0;
14
                                                       结点编号传入,需要返回1
15 L
16 ☐ int main(){
       int i,n;
17
                                                       哪种二叉树遍历顺序
       cin>>n;
18
       for(i=1;i<=n;i++)
19
20
          cin>>a[i].value>>a[i].left_child>>a[i].right_child;
                                                          前序
                                                                    根左右
       cout<<is_bst(___(5)___, -INFINITE, INFINITE);</pre>
21
       return 0;
                                                          中序
                                                                    左根右
                                                                    左右根
                                                          后序
```

```
#include <iostream>
   using namespace std;
                                                     第一个is_bst函数调用
   const int SIZE=100;
                                                  将本层结点左子结点编号传入
   const int INFINITE=1000000;
                                                  该左子结点值下限只要保持和
   struct node {int left child, right child, value;};
   node a[SIZE];
                                                 |父节点一样就行(思考为什么)
7 ☐ int is_bst(int root, int lower_bound, int upper_bound)
                                                   上限调整为不超过本层结点
      if(root==0) return 1;
                                                   (该左子结点父亲) 的值cur
      int cur=a[root].value;
      if( (cur>lower bound) && ( (1) ) &&
      (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
      (is bst( (2)
                                                     第二个is_bst函数调用
13
          return 1:
                                                  将本层结点右子结点编号传入
14
      return 0;
                                                  该右子结点值上限只要保持和
16 ⊟ int main(){
                                                 父节点一样就行(同样的原因)
      int i,n;
17
      cin>>n;
18
                                                   下限调整为不小于本层结点
      for(i=1;i<=n;i++)
19
                                                   (该左子结点父亲)的值cur
20
          cin>>a[i].value>>a[i].left_child>>a[i].right_chi
      cout<<is_bst(____(5)____, -INFINITE, INFINITE);</pre>
21
      return 0;
```

数的入口,从根结点开始

#include <iostream>

```
using namespace std;
   const int SIZE=100;
   const int INFINITE=1000000;
   struct node {int left child, right child, value;};
                                                             定义结构体数组
   node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
       if(root==0) return 1;
                                                            递归调用is bst函数
       int cur=a[root].value;
       if( (cur>lower_bound) && (____(1)____) &&
LØ
                                                        检查每节点值是否都大于其
11
       (is_bst(a[root].left_child, lower_bound, cur)==1) &&
                                                        左子树上所有节点值、小于
       (is_bst(___(2)___, ___(3)___, ___(4)___)==1) )
L3
L4
L5
                                                          其右子树上所有节点值
          return 1:
       return 0;
                                                         如果是返回1,否则返回0
16 ☐ int main(){
17
       int i,n;
       cin>>n;
L8
                                                         将每个node的值、左儿子
19
20
21
       for(i=1;i<=n;i++)
                                                        编号和右儿子编号依次输入
          cin>>a[i].value>>a[i].left_child>>a[i].right_child
       cout<<is bst( (5) , -INFINITE, INFINITE);</pre>
       return 0;
                                                         主函数内递归调用is_bst函
```

```
#include <iostream>
    using namespace std;
                                                该层结点的值cur需要小于参数
    const int SIZE=100;
                                                  传入的上限upper bound
   const int INFINITE=1000000;
 5
    struct node {int left_child, right_child, value;};
    node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
8
        if(root==0) return 1;
9
        int cur=a[root].value;
        if( (cur>lower_bound) && (____(1)____) &&
10
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is_bst(___(2)___, ___(3)___, ___(4)___)==1) )
12
13
            return 1;
14
        return 0;
15
16 □ int main(){
        int i,n;
17
        cin>>n;
18
19
        for(i=1;i<=n;i++)
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
20
        cout<<is_bst(____(5)____, -INFINITE, INFINITE);</pre>
21
22
        return 0;
23
```

第一个参数应传入该层结点的 右子结点编号 即a[root].right\_child

```
#include <iostream>
    using namespace std;
    const int SIZE=100;
    const int INFINITE=1000000;
    struct node {int left child, right child, value;};
 5
    node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
8
        if(root==0) return 1;
9
        int cur=a[root].value;
        if( (cur>lower_bound) && (____(1)____) &&
10
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is_bst(___(2)___, ___(3)___, ___(4)___)==1) )
12
            return 1;
13
14
        return 0;
15
16 □ int main(){
        int i,n;
17
        cin>>n;
18
19
        for(i=1;i<=n;i++)
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
20
        cout<<is_bst(____(5)____, -INFINITE, INFINITE);</pre>
21
22
        return 0;
23
```

第二个参数应传入右儿子下限 右儿子不能小于父亲值 所以下限应是**cur** 

```
#include <iostream>
    using namespace std;
    const int SIZE=100;
    const int INFINITE=1000000;
 5
    struct node {int left child, right child, value;};
    node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
8
        if(root==0) return 1;
9
        int cur=a[root].value;
        if( (cur>lower_bound) && (____(1)____) &&
10
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is_bst(___(2)___, ___(3)___, ___(4)___)==1) )
12
13
            return 1;
14
        return 0;
15
16 □ int main(){
17
        int i,n;
        cin>>n;
18
19
        for(i=1;i<=n;i++)
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
20
        cout<<is_bst(____(5)____, -INFINITE, INFINITE);</pre>
21
22
        return 0;
23
```

第三个参数应传入右儿子上限 右儿子只需要继承父亲上限 下层沿用本层upper\_bound

```
#include <iostream>
    using namespace std;
    const int SIZE=100;
    const int INFINITE=1000000;
 5
    struct node {int left child, right child, value;};
    node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
8
        if(root==0) return 1;
9
        int cur=a[root].value;
        if( (cur>lower_bound) && (____(1)____) &&
10
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is_bst(___(2)___, ___(3)___, ___(4)___)==1) )
12
13
            return 1;
14
        return 0;
15
16 □ int main(){
        int i,n;
17
        cin>>n;
18
19
        for(i=1;i<=n;i++)
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
20
        cout<<is_bst(____(5)____, -INFINITE, INFINITE);</pre>
21
22
        return 0;
23
```

```
#include <iostream>
    using namespace std;
    const int SIZE=100;
   const int INFINITE=1000000;
 5
    struct node {int left child, right child, value;};
    node a[SIZE];
7 □ int is_bst(int root, int lower_bound, int upper_bound) {
8
        if(root==0) return 1;
9
        int cur=a[root].value;
        if( (cur>lower_bound) && (____(1)____) &&
10
        (is_bst(a[root].left_child, lower_bound, cur)==1) &&
11
        (is_bst(___(2)___, ___(3)___, ___(4)___)==1) )
12
13
            return 1;
14
        return 0;
                                                从结点1(也即根结点)开始
15
                                                        递归调用
16 □ int main(){
17
        int i,n;
        cin>>n;
18
19
        for(i=1;i<=n;i++)
            cin>>a[i].value>>a[i].left_child>>a[i].right_child;
20
        cout<<is_bst(____(5)____, -INFINITE, INFINITE);</pre>
21
22
        return 0;
23
```

(双链表)对于一个1到n的排列Q(即1到n中每一个数在Q中出现了恰好一次),令 $q_i$ 为

第i个位置之后第一个比 $Q_i$  值更大的位置,如果不存在这样的位置,则 $q_i = n + 1$ 。

举例来说,如果n = 5且Q为15423,则q为26656。

下列程序读入了排列Q,使用双向链表求解了答案。试补全程序。

数据范围 1 ≤ *n* ≤ 10<sup>5</sup>

# 手算样例

输入样例:

4

1 2 3 4

输出多少?

输出样例:

2 3 4 5

# 手算样例

输入样例:

8

1 3 7 4 6 2 8 5

输出多少?

输出样例:

2 3 7 5 7 7 9 9

```
#include(iostream>
                                              识别变量
   using namespace std;
   const int N=10010;
                                          常见变量名
   int n:
                                         翻译循环变量
   int L[N], R[N], a[N];
6 ☐ int main() {
                                     根据变量名的英文推断
      cin>>n;
      for(int i=1; i<=n; ++i) {
8 🖹
          int x;
10
          cin>>x;
                                      2
                                            找出关键语句
           (1) ;
11
12
                                        控制结构(for, if)
       for(int i=1; i<=n; ++i) {
13 白
          R[i] = ___(2) ___;
                                      常见算法的基本操作
14
15
          L[i] = i-1;
                                       函数参数、返回值
16
       for(int i=1; i<=n; ++i) {
17 白
          L[___(3) ___] = L[a[i]];
18
19
          R[L[a[i]]] = R[___(4) ___];
20
21
       for(int i=1; i<=n; ++i)
                                      3
                                           理解代码段作用
          cout<<___(5) ____<<" ";
22
23
       cout<<endl:
                                        翻译解释代码段
24
      return 0:
```

```
#include<iostream>
   using namespace std;
   const int N=10010;
   int n;
    int L[N], R[N], a[N];
6 ☐ int main() {
       cin>>n;
       for(int i=1; i<=n; ++i) {
8 🗎
           int x;
           cin>>x;
10
             (1) ;
11
12
       for(int i=1; i<=n; ++i) {
13 🖨
           R[i] = ____(2) ____;
14
           L[i] = i-1;
15
16
17 白
       for(int i=1; i<=n; ++i) {
           L[__(3)_] = L[a[i]];
18
           R[L[a[i]]] = R[___(4) ___];
19
20
21
       for(int i=1; i<=n; ++i)
           cout<<____(5) ____<<" ";
22
23
       cout<<endl;
24
       return 0;
```

#### 解释变量的作用

n	表示序列长度					
X	表示第i个数字					
a[x]	表示数字 <b>x</b> 在序列中的位 置					
L[i]	表示位置i的前驱					
R[i]	表示位置i的后继					

输入:

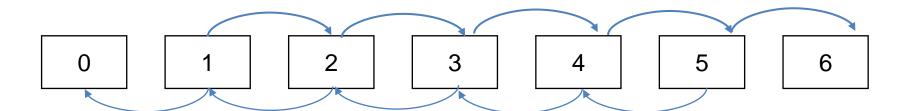
5

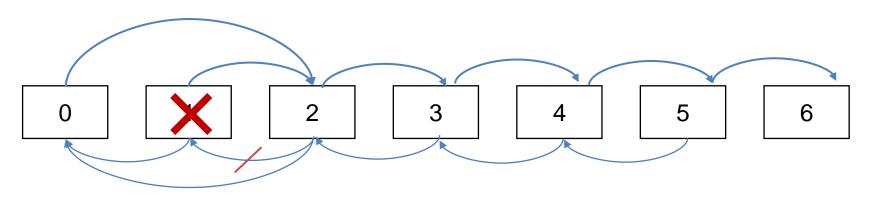
15423

输出:

26656

i	0	1	2	3	4	5	6
а		1	4	5	3	2	
L		0	1	2	3	4	
R		2	3	4	5	6	



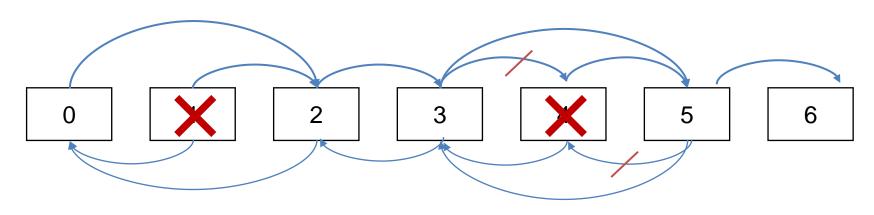


删除当前最小数字1

1的位置a[i]=1

从链表中删除a[i]

a[i]的后继的前驱变成a[i]的前驱

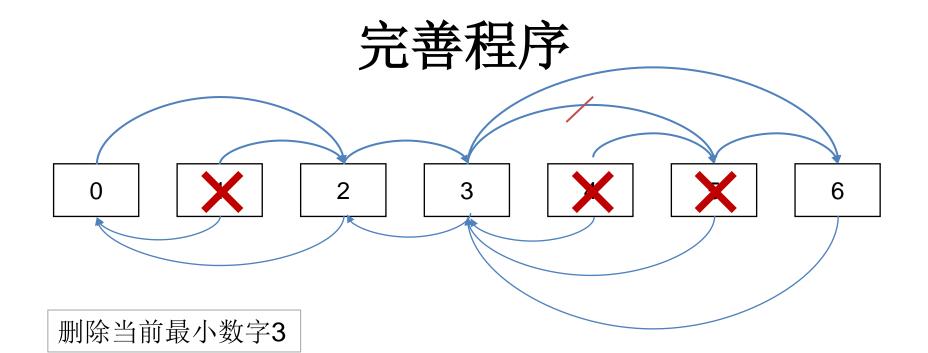


删除当前最小数字2

2的位置a[2]=4

从链表中删除a[i]

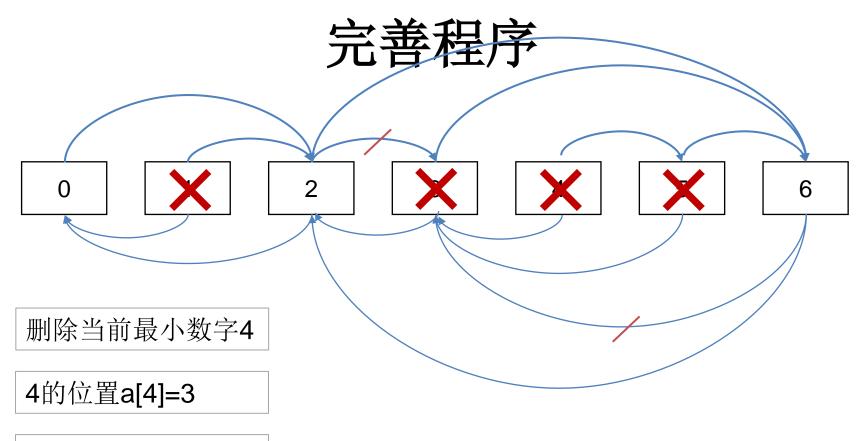
a[i]的后继的前驱变成a[i]的前驱



3的位置a[3]=5

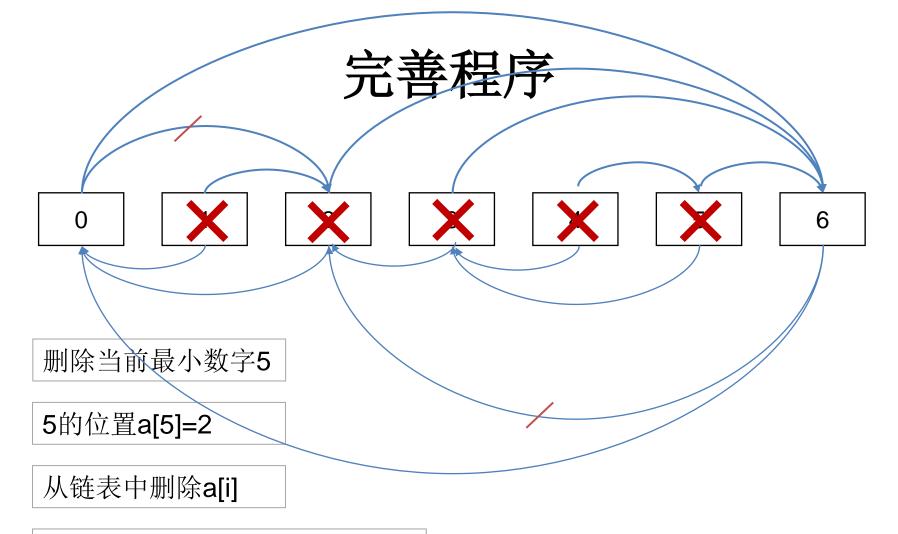
从链表中删除a[i]

a[i]的后继的前驱变成a[i]的前驱



从链表中删除a[i]

a[i]的后继的前驱变成a[i]的前驱



a[i]的后继的前驱变成a[i]的前驱

输出: 

```
#include(iostream>
   using namespace std;
   const int N=10010;
   int n:
    int L[N], R[N], a[N];
6 ☐ int main() {
        cin>>n;
        for(int i=1; i<=n; ++i) {
8 🖹
9
            int x;
            cin>>x;
10
             (1) ;
11
12
        for(int i=1; i<=n; ++i) {
13 🗎
14
           R[i] = ____(2) ____;
15
           L[i] = i-1;
16
        for(int i=1; i<=n; ++i) {
17 白
            L[___(3) ___] = L[a[i]];
18
           R[L[a[i]]] = R[___(4) ___];
19
20
21
        for(int i=1; i<=n; ++i)
           cout<<____(5) ____<<" ";
22
23
        cout<<endl:
24
        return 0:
```

#### 关键语句

a[i]=x还是a[x]=i 如果a[i]=x为什么不 直接cin>>a[i]

记录数字x在序列 中的位置

```
#include<iostream>
    using namespace std;
   const int N=10010;
    int n;
    int L[N], R[N], a[N];
6 ☐ int main() {
7
       cin>>n;
8 点
        for(int i=1; i<=n; ++i) {
9
            int x;
           cin>>x;
10
11
              (1) ;
12
13 白
        for(int i=1; i<=n; ++i) {
14
           R[i] = ___(2) ___;
15
           L[i] = i-1;
16
        for(int i=1; i<=n; ++i) {
17 白
18
            L[___(3) ___] = L[a[i]];
           R[L[a[i]]] = R[ (4) ];
19
20
        for(int i=1; i<=n; ++i)
21
           cout<< __(5) __
22
        cout<<endl;
23
24
        return 0;
25
```

关键语句

右侧第一个比位置i 上数字大的位置 a[i]、L[i]还是R[i]?

```
#include(iostream>
    using namespace std;
   const int N=10010;
    int n;
    int L[N], R[N], a[N];
6 ☐ int main() {
       cin>>n;
       for(int i=1; i<=n; ++i) {
9
           int x;
           cin>>x;
10
             (1) ;
11
12
       for(int i=1; i<=n; ++i) {
13 白
           R[i] = ___(2) ___;
14
           L[i] = i-1;
15
16
        for(int i=1; i<=n; ++i) {
17 白
           L[___(3) ___] = L[a[i]];
18
           R[L[a[i]]] = R[ (4) ];
19
20
21
       for(int i=1; i<=n; ++i)
           cout<<___(5) ____<<" ";
22
       cout<<endl;
23
24
       return 0;
25
```

关键语句

位置i的后继为i+1

位置i的前驱为i-1

```
#include(iostream>
    using namespace std;
   const int N=10010;
    int n;
    int L[N], R[N], a[N];
6 ☐ int main() {
       cin>>n;
       for(int i=1; i<=n; ++i) {
           int x;
           cin>>x;
10
             (1)
11
12
13 白
       for(int i=1; i<=n; ++i) {
           R[i] = ___(2) ___;
14
           L[i] = i-1;
15
16
       for(int i=1; i<=n; ++i) {
17 白
           L[___(3) ___] = L[a[i]];
18
           R[L[a[i]]] = R[ (4) ];
19
20
       for(int i=1; i<=n; ++i)
21
           cout<<____(5) ____<<" ";
22
       cout<<endl;
23
       return 0:
24
25
```

关键语句

L[a[i]]数字i所在位 置在序列的前驱

R[L[a[i]]数字i所在 位置序列位置的前 驱的后继

```
#include<iostream>
                                         代码段作用
   using namespace std;
3 const int N=10010;
   int n:
   int L[N], R[N], a[N];
6 ☐ int main() {
       cin>>n;
       for(int i=1; i<=n; ++i) {
8 🖹
                                            输入记录x出
          int x;
                                              现的位置
10
          cin>>x;
           ____(1) _____;
11
12
       for(int i=1; i<=n; ++i) {
13 白
                                              构建静态
14
          R[i] = ___(2) ___;
                                               双链表
15
          L[i] = i-1;
16
17 白
       for(int i=1; i<=n; ++i) {
          L[___(3) ___] = L[a[i]];
18
                                              删除节点
          R[L[a[i]]] = R[___(4) ___];
19
20
21
       for(int i=1; i<=n; ++i)
                                              输出后继
          cout<<___(5) ___<<" ";
22
23
       cout<<end1:
24
       return 0;
```

```
1 #include<iostream>
   using namespace std;
3 const int N=10010;
   int n:
   int L[N], R[N], a[N];
6 ☐ int main() {
       cin>>n;
                                      输入数字x出现的位置
       for(int i=1; i<=n; ++i) {
8 🖹
9
           int x;
           cin>>x;
10
11
12
       for(int i=1; i<=n; ++i) {
13 🗎
           R[i] = ___(2) ___;
14
15
           L[i] = i-1;
16
       for(int i=1; i<=n; ++i) {
17 白
           L[__(3)_] = L[a[i]];
18
19
           R[L[a[i]]] = R[___(4) ___];
20
21
       for(int i=1; i<=n; ++i)
           cout<<___(5) ____<<" ";
22
23
       cout<<endl;
24
       return 0;
25 L
```

```
#include<iostream>
   using namespace std;
 3 const int N=10010;
   int n:
   int L[N], R[N], a[N];
6 ☐ int main() {
7
       cin>>n;
8 🖹
       for(int i=1; i<=n; ++i) {
9
           int x;
10
           cin>>x;
11
12
                                              创建双链表
        for(int i=1; i<=n; ++i) {
13 🖹
14
           R[i] =
15
           L[i] = i-1;
16
        for(int i=1; i<=n; ++i) {
17 白
           L[__(3)_] = L[a[i]];
18
19
           R[L[a[i]]] = R[___(4) ___];
20
21
        for(int i=1; i<=n; ++i)
           cout<<___(5) ____<<" ";
22
23
       cout<<endl;
24
       return 0;
25 L
```

```
#include<iostream>
    using namespace std;
    const int N=10010;
    int n:
    int L[N], R[N], a[N];
6 ☐ int main() {
7
        cin>>n;
8 🖹
        for(int i=1; i<=n; ++i) {
9
            int x;
10
            cin>>x;
11
12
13 白
        for(int i=1; i<=n; ++i) {
            R[i] =
14
15
            L[i] = i-1;
16
                                              从小到大删除数字
        for(int i=1; i<=n; ++i) {
17 白
18
            L
                            ] = L[a[i]];
19
            R[L[a[i]]] = R[
                                           ];
20
21
        for(int i=1; i<=n; ++i)
            cout<<____(5) ____<<" ";
22
23
        cout<<endl;
24
        return 0;
25
```

```
#include<iostream>
    using namespace std;
    const int N=10010;
    int n:
    int L[N], R[N], a[N];
 6 ☐ int main() {
 7
        cin>>n;
 8 🖹
        for(int i=1; i<=n; ++i) {
 9
            int x;
10
            cin>>x;
11
12
13 白
        for(int i=1; i<=n; ++i) {
14
            R[i] =
15
            L[i] = i-1;
16
        for(int i=1; i<=n; ++i) {
17 白
18
            LI
                              [] = L[a[i]];
19
            R[L[a[i]]] = R[
                                            ];
20
        for(int i=1; i<=n; ++i)
21
                                                    输出i的后继
                                   <<" ";
22
            cout
23
        cout<<endl;
24
        return 0;
25
```

kkcoding.net

## 作业: 完善程序

626 坐标统计

229 子矩阵

2007 静态双链表