

Lesson3

双指针算法

- 两类
 - 两个指针指向两个序列
 - 两个指针指向一个序列
- 一般写法

```
1 for (i=0, j=0; i<n; i++) {  
2     while (j < i && check(i, j)) j++;  
3     //每道题目具体逻辑  
4 }
```

- 核心思想
对朴素算法进行优化（单调性），时间复杂度优化为 $O(n)$ ，常数为2，最坏情况下 $O(2n)$
先想暴力算法，再通过单调性进行优化， $O(n^2) \rightarrow O(n)$
- e1: 输出字符串中每个单词

```
1 #include <iostream>  
2 #include <cstring>  
3  
4 using namespace std;  
5  
6 int main(void) {  
7     char str[1000];  
8     cin.getline(str, 1010);  
9  
10    for (int i=0; str[i]; i++) {  
11        if (str[i] == ' ') continue;  
12  
13        int j = i;  
14        while (j<strlen(str) && str[j]!=' ') j++;  
15  
16        for (int k=i; k<j; k++) cout << str[k];  
17        puts("");  
18        i = j;  
19    }  
20  
21    return 0;  
22 }
```

- e2: 最长不重复子序列

```
1 #include <iostream>  
2  
3 using namespace std;
```

```

4
5  const int N = 1e5+10;
6
7  int a[N], s[N], n;
8  int res;
9
10 int main(void) {
11     scanf("%d", &n);
12
13     for (int i=0; i<n; i++) scanf("%d", &a[i]);
14
15     for (int i=0, j=0; i<n; i++) {
16         s[a[i]]++;
17
18         while (j<=i && s[a[j]]>1) {
19             s[a[j]]--;
20             j++;
21         }
22
23         res = max(res, i-j+1);
24     }
25
26     cout << res;
27
28     return 0;
29 }

```

位运算

- **n的二进制表示中第k位是什么:** $n \gg k \& 1$
- 个位 (最后一位) 是第0位, 从个位开始
- 先把第k位移至最后一位 (个位) (右移运算 $n \gg k$)
- 求个位的值
- 结合1, 2步, 得公式 $n \gg k \& 1$

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main(void) {
6      int a = 10;
7
8      for (int i=31; i>=0; i--) cout << (a>>i&1);
9
10     return 0;
11 }

```

- **lowbit (x)** 返回x的最后一位 (最右边) 1的位置, 主要用于树状数组
 - $x=1010$, $\text{lowbit}(x) = 10$
 - $x=101000$, $\text{lowbit}(x) = 101000$
 - $\text{lowbit}(x) = x \& -x = x \& (\sim x + 1)$

- 应用：统计x中1的个数

```
1  #include <iostream>
2
3  using namespace std;
4
5  int lowbit(int x) {
6      return x & -x;
7  }
8
9  int main(void) {
10     int n;
11     scanf("%d", &n);
12
13     while (n--) {
14         int x;
15         scanf("%d", &x);
16
17         int res = 0;
18         while (x) x -= lowbit(x), res++;
19
20         printf("%d ", res);
21     }
22
23     return 0;
24 }
```

离散化 (整数离散化)

- 适用于值域大，个数少的序列，如值域 $0 \sim 10^9$ ，个数 10^5
- 重复元素的处理：去重，库函数`all.erase(unique(all.begin(), all.end()), all.end())`
- 如何算出`a[i]`中离散化后的值是多少 (二分)
- 对数组下标进行映射

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  typedef pair<int, int> PII;
8
9  const int N = 300010;
10
11 int n, m;
12 int a[N], s[N];
13
14 vector<int> all;
15 vector<PII> add, query;
16
17 //去重函数
18 vector<int>::iterator unique(vector<int> &a) {
```

```

19     int j = 0;
20     for (int i=0; i<a.size(); i++)
21         if (!i || a[i]!=a[i-1])
22             a[j++] = a[i];
23
24     return a.begin()+j;
25 }
26
27 //二分
28 int find(int x) {
29     int l = 0, r = all.size()-1;
30     while (l < r) {
31         int mid = l+r>>1;
32         if (all[mid] >= x) r = mid;
33         else l = mid+1;
34     }
35
36     return r+1;
37 }
38
39 int main(void) {
40     scanf("%d%d", &n, &m);
41
42     while (n--) {
43         int x, c;
44         scanf("%d%d", &x, &c);
45         add.push_back({x, c});
46         all.push_back(x);
47     }
48
49     while (m--) {
50         int l, r;
51         scanf("%d%d", &l, &r);
52         query.push_back({l, r});
53         all.push_back(l), all.push_back(r);
54     }
55
56     sort(all.begin(), all.end());
57     // all.erase(unique(all.begin(), all.end()), all.end());
58     all.erase(unique(all), all.end());
59
60     //处理插入
61     for (auto item: add) {
62         int x = find(item.first);
63         a[x] += item.second;
64     }
65
66     //预处理前缀和
67     for (int i=1; i<=all.size(); i++) s[i] = s[i-1]+a[i];
68
69     //处理查询
70     for (auto item: query) {
71         int l = find(item.first), r = find(item.second);
72         printf("%d\n", s[r]-s[l-1]);
73     }
74
75     return 0;
76 }

```

区间 (大多数贪心) 合并

- 按区间左端点排序
- 扫描所有区间, 把所有可能有交集的区间进行合并
 - 维护两个端点st (start) , ed(end)
 - 3种情况
 - 包含
st, ed不变
 - 有交
更新ed
 - 不包含
更新st, ed (新区间)

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  typedef pair<int, int> PII;
8
9  const int N = 100010;
10
11 int n;
12 vector<PII> segs;
13
14 void merge(vector<PII> &segs) {
15     vector<PII> res;
16
17     sort(segs.begin(), segs.end());
18
19     int st = -2e9, ed = -2e9;
20     for (auto seg: segs)
21         if (ed < seg.first) {
22             if (ed != -2e9) res.push_back({st, ed});
23             st = seg.first, ed = seg.second;
24         }
25         else ed = max(ed, seg.second);
26
27     if (st != -2e9) res.push_back({st, ed});
28
29     segs = res;
30 }
31
32 int main(void) {
33     scanf("%d", &n);
34
35     for (int i=0; i<n; i++) {
36         int l, r;
```

```
37         scanf("%d%d", &l, &r);
38         segs.push_back({l, r});
39     }
40
41     merge(segs);
42
43     cout << segs.size();
44
45     return 0;
46 }
```