双指针

常见 → 归并排序，快排

① 双指针指向两个序列（归并）

② 指向一个序列（快排）

```
for (i = 0, j = 0; i < n; i ++ )
{
    while (j < i && check(i, j)) j ++ ;

    // 每道题目的具体逻辑
}
```

模板

最核心思想：双层循环 ×

$n^2 \rightarrow 2n \Rightarrow O(n^2) \rightarrow O(n)$

朴素做法出发 ⟶ 双指针

例：最长不重复子序列

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 1e5 + 10;
int a[N],s[N];
int n;

bool check(int i,int j)
{
    if(s[a[i]]>1) return true;
    else return false;
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++ ) scanf("%d",&a[i]);
    int res = 0;
    for (int i = 0,j=0; i < n; i ++ )
    {
        s[a[i]] ++;
        while(j < n && check(i,j))
        {
            s[a[j]]--;
            j ++;
        }
        //
        res = max(res,i-j+1);
    }
    printf("%d",res);
    return 0;
```

位运算

常见操作：① n的二进制表示中，第k位是几？

$n=15=(1111)_2$

4 3 2 1 0

a. 先把第k位移到最后一位 $n>>k$ > $n>>k\ \&\ 1$

b. 看个位是~

例：10 的二进制：1010

1010 >> 0 1010

1 101

2 10

3 1

② lowbit(x) 返回x的最后一位1是多少

$x=1010 \rightarrow 10$

$x=1010 \rightarrow 1000$

lowbit(x) → $x\ \&\ -x$

$-x=\sim x+1$

$\therefore x\&-x=x\&(\sim x+1)$

$x=1010\cdots100\cdots00$

$\sim x=0101\cdots011\cdots11$

$\sim x+1=0101\cdots100\cdots00$

则 $x\&-x$

$=100\cdots00$

$=$ 最后一位1

原码 x

反码 $\sim x$

补码 $\sim x+1$

应用：统计x中1的个数：

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 100010;

int n;
int a;

int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++ )
    {
        scanf("%d", &a);
        int icnt=0;
        while(a)
        {
            a -=a&(-a);
            icnt += 1;
        }
        printf("%d ",icnt);

    }
    return 0;

}
```

离散化

整数、保序离散化

$0 \sim 10^9$（值域）

个数 10^5

$1\ 2\ 5\ 8\ \cdots\ 10^9 \Rightarrow$ 映射缩小值域

值域 >> 数的个数

例：a[] 1 3 100 2000 500000

↓ ↓ ↓ ↓ ↓

0 1 2 3 4

① a[]中有重复元素：去重

② 如何算出 a[i]，离散化后的值 [二分]

↳ 保存（a → 下标）

去重常用代码：

```
sort(alls.begin(),alls.end());
alls.erase(unique(alls.begin(), alls.end()), alls.end());
```

离散数据前缀和 → (L,R) ⇒ (L', R')

离散化后的值

↳ 映射成自然数

```
using namespace std;

typedef pair<int, int> PII;

const int N = 300010;

vector<PII> add,query;
vector<int> alls;
int a[N],s[N];

int find(int x)
{
    int l=0,r=alls.size()-1;
    while(l<r)
    {
        int mid = (l+r+1) / 2;
        if(alls[mid]<=x) l=mid;
        else r = mid -1;
    }
    return r+1;
}

int main()
{
    int n,m;
    scanf("%d%d", &n, &m);
    for (int i = 0; i < n; i ++ )
    {
        int x,c;
        scanf("%d%d", &x, &c);
        add.push_back({x,c});
        alls.push_back(x);
    }
    for (int i=0; i < m ; i++)
    {
        int l,r;
        scanf("%d%d", &l, &r);
        query.push_back({l,r});
        alls.push_back(l);
        alls.push_back(r);
    }

    sort(alls.begin(),alls.end());
    alls.erase(unique(alls.begin(),alls.end()),alls.end());

    for(auto item: add)
    {
        int x = find(item.first);
        a[x] += item.second;
    }
    for (int i = 1; i <= alls.size(); i ++ ) s[i] = a[i]+s[i-1];

    for(auto item: query)
    {
        int l = find(item.first);
        int r = find(item.second);
        cout<<s[r] -s[l-1]<<endl;
    }
    return 0;
}
```

区间合并

⇒ 贪心 !!!

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>

using namespace std;

typedef pair<int,int> PII;

vector<PII> itv;

vector<PII> merge(vector<PII>& itv)
{
    vector<PII> res;
    int st = -1e9-10, ed = -1e9-10;
    sort(itv.begin(),itv.end());
    for(auto i: itv)
    {
        if(i.first > ed)
        {
            if(st != -1e9-10) res.push_back({st,ed});
            st = i.first;
            ed = i.second;
        }
        else
        {
            ed = max(i.second,ed);
        }
    }
     res.push_back({st,ed});
    return res;
}
int main()
{
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i ++ )
    {
        int l,r;
        scanf("%d%d", &l, &r);
        itv.push_back({l,r});
    }
    auto res = merge(itv);
    printf("%d",res.size());
    return 0;


}
```