by 黄万平，codeforces ID: hwpvector

前言

由于个人精力问题，所有题仅放笔者写的代码。

下面的代码默认在主函数中仅调用 `solve()`

fyc 的代码和他的绝佳讲解：https://www.luogu.com.cn/paste/f20e5eox

（本 PDF/Markdown 文件的最后会附上 fyc 的代码）

科普一些常识：

一台评测机每秒能跑 10^8 数量级的运算次数。

具体的次数，根据评测机性能不同而各不相同。笔者遇到的最好的评测机是今年 ICPC 杭州站的机子，每秒能跑 4×10^9 的 `unsigned long long` 乘法，1.5×10^9 的 `double` 乘法；通常情况每秒能跑的运算次数在 4×10^8 左右。

对于一个算法，其在最坏情况下的时间复杂度为 $O(f(n))$；若将数据范围中极大的 n 带进去，其值为 $f(n)$，可以近似的算出该算法在最极端情况下所需运算次数的数量级，和 10^8 做对比，可以知道自己的算法有没有超时。

通常情况下，我们在写算法之前，就需要算出该算法的时间复杂度，并根据时间复杂度看看自己有没有可能超时——如果超时了，就不用写了，再想想其他算法。

如果你写了一个超时的算法，可能是最终评测结果没有超时，那大概率是数据弱了。你要知道你所写的算法实际上是会超时的，最好把正解写一下。

以上。

关于额外的刷题平台，如果不想搞算法竞赛，建议前往 Leetcode 刷题；如果不知道自己想不想搞竞赛，也建议前往 Leetcode 刷题，大概率没错——如果喜欢上思考题目的感觉，想算法的过程，那再考虑打算法竞赛也不迟；若有志于打算法竞赛，建议去 Luogu 或 Codeforces 或 Atcoder

如果你数学不好，算法竞赛直接劝退。

另外笔者建议对于算法竞赛入门的同学，或是不清楚自己想不想搞、适不适合搞算法竞赛的同学，前往 Codeforces Contests Div.3 的场次，打个现场赛或是 Vitual Presentation，想想自己喜不喜欢这种思维的感觉。

算法竞赛校队大概的线：Codeforces Rating 1800 - 1900

以上。

关于零基础也能参加的算法比赛：

笔者通常以蓝桥杯为标准，比蓝桥杯还要水的算法竞赛可以说是圈钱大赛了。

下面的比赛，算法竞赛零基础也能够获得一定的成绩：

- 下半学期：蓝桥杯（已经开始报名）
- 下半学期：GPLT（需要校内选拔，校内选拔题目详见：oj.xjtuicpc.com 题库中带有 LX-X 标签的题）
- 下半学期：ICPC 校赛
- 下半学期：ICPC 省赛
- 每个季度：CSP 认证（在考研时，部分高校对于 CSP 认证超过一定分数（通常在 350 - 400 之间）的选手，可以免去机试）

以上。

Problem 1

```
void solve() {
    list<int> l; int x;
    while (cin>>x) {
        if (x==-1) break;
        l.push_front(x);
    }
    cin>>x; l.remove(x);
    for (auto v:l) cout<<v<<' ';
}
```

Problem 2

```
int a[110][110];
int maxn[110],minn[110];

void solve() {
    int n,m; cin>>n>>m;
    for (int i=1;i<=n;i++) {
```

```
        for (int j=1;j<=m;j++) cin>>a[i][j];
    }
    for (int i=1;i<=n;i++) {
        maxn[i]=-inf;
        for (int j=1;j<=m;j++) maxn[i]=max(maxn[i],a[i][j]);
    }
    for (int i=1;i<=m;i++) {
        minn[i]=inf;
        for (int j=1;j<=n;j++) minn[i]=min(minn[i],a[j][i]);
    }
    vector<array<int,3>> ans;
    for (int i=1;i<=n;i++) {
        for (int j=1;j<=m;j++) {
            if (a[i][j]==maxn[i]&&a[i][j]==minn[j]) {
                ans.push_back({a[i][j],i,j});
            }
        }
    }
    if (ans.size()==0) {
        cout<<"No answer\n";
    }
    for (auto [v,i,j]:ans) cout<<v<<' '<<i<<' '<<j<<'\n';
}
```

Problem 3

```
int a[110][110],b[110][110],c[110][110];

void solve() {
    int n,m; cin>>n>>m;
    for (int i=1;i<=n;i++) {
        for (int j=i;j<=n;j++) {
            cin>>a[i][j];
            if (i!=j) a[j][i]=a[i][j];
        }
    }
    for (int i=1;i<=n;i++) {
        for (int j=i;j<=n;j++) {
            cin>>b[i][j];
            if (i!=j) b[j][i]=b[i][j];
        }
    }
    for (int i=1;i<=n;i++) {
```

```
        for (int j=1;j<=n;j++) {
            for (int k=1;k<=n;k++) {
                c[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    for (int i=1;i<=n;i++) {
        for (int j=1;j<=n;j++) {
            cout<<c[i][j]<<' ';
        } cout<<'\n';
    }
}
```

Problem 4

```
set<int> ans;
void solve() {
    int x; cin>>x;
    for (int i=2;i*i<=x;i++) {
        int cnt=0;
        while (x%i==0) x/=i,++cnt;
        if (cnt!=0) ans.insert(i);
    } if (x!=1) ans.insert(x);
}
int main() {
    int T; cin>>T; while (T--) solve();
    for (auto v:ans) cout<<v<<' ';
    return 0;
}
```

Problem 5

```
int a[110][110],b[110][110],c[110][110];

void solve() {
    int n,maxn=0; cin>>n;
    for (int i=1;i<=n;i++) {
        int x,y,v; cin>>x>>y>>v;
        a[x][y]=v; maxn=max({maxn,x,y});
    } cin>>n;
    for (int i=1;i<=n;i++) {
```

```
        int x,y,v; cin>>x>>y>>v;
        b[x][y]=v; maxn=max({maxn,x,y});
    }
    for (int i=0;i<=maxn;i++) {
        for (int j=0;j<=maxn;j++) {
            for (int k=0;k<=maxn;k++) {
                c[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    for (int i=0;i<=maxn;i++) {
        for (int j=0;j<=maxn;j++) {
            if (c[i][j]!=0) {
                cout<<i<<' '<<j<<' '<<c[i][j]<<'\n';
            }
        }
    }
}
```

Problem 6

```
int a[110][110];

void solve() {
    for (int i=1;i<=19;i++) {
        for (int j=1;j<=19;j++) cin>>a[i][j];
    }
    for (int i=0;i<=20;i++) a[0][i]=a[i][0]=a[i][20]=a[20][i]=3;
    auto same=[](array<int,6> x) {
        return x[1]!=0&&(x[0]==0||x[5]==0)&&
        x[1]==x[2]&&x[1]==x[3]&&x[1]==x[4];
    };
    for (int i=1;i<=15;i++) {
        for (int j=1;j<=19;j++) {
            if (same({a[i-1][j],a[i][j],a[i+1][j],
                a[i+2][j],a[i+3][j],a[i+4][j]})) {
                cout<<a[i][j]<<':'<<i<<','<<j<<'\n'; return ;
            }
        }
    }
    for (int i=1;i<=19;i++) {
        for (int j=1;j<=15;j++) {
            if (same({a[i][j-1],a[i][j],a[i][j+1],
```

```
                a[i][j+2],a[i][j+3],a[i][j+4]})) {
                cout<<a[i][j]<<':'<<i<<','<<j<<'\n'; return ;
            }
        }
    }
    for (int i=1;i<=15;i++) {
        for (int j=1;j<=15;j++) {
            if (same({a[i-1][j-1],a[i][j],a[i+1][j+1],
                a[i+2][j+2],a[i+3][j+3],a[i+4][j+4]})) {
                cout<<a[i][j]<<':'<<i<<','<<j<<'\n'; return ;
            }
        }
    }
    for (int i=1;i<=15;i++) {
        for (int j=5;j<=19;j++) {
            if (same({a[i-1][j+1],a[i][j],a[i+1][j-1],
                a[i+2][j-2],a[i+3][j-3],a[i+4][j-4]})) {
                cout<<a[i][j]<<':'<<i<<','<<j<<'\n'; return ;
            }
        }
    }
    cout<<"No\n";
}
```

Problem 7

```
#define fi first
#define se second
const int N=1e6+10;
pair<int,int> a[N];

void solve() {
    int n; cin>>n;
    for (int i=1;i<=n;i++) cin>>a[i].fi>>a[i].se;
    sort(a+1,a+n+1);
    int l=a[1].fi,r=a[1].se;
    for (int i=2;i<=n;i++) {
        if (a[i].fi<=r) r=max(a[i].se,r);
        else cout<<l<<' '<<r<<'\n',l=a[i].fi,r=a[i].se;
    }
    cout<<l<<' '<<r<<'\n';
}
int main() {
```

```
    freopen("prz.in","r",stdin);
    freopen("prz.out","w",stdout);
    solve(); return 0;
}
```

Problem 8

```
map<string,int> mp;
void solve() {
    string s,ans;
    while (cin>>s) {
        for (auto c:s) {
            if (c>='a'&&c<='z') ans.push_back(c);
            else if (c>='A'&&c<='Z') ans.push_back(c-'A'+'a');
            else if (ans.size()) {
                mp[ans]=1; ans.clear();
            }
        }
        if (ans.size()) mp[ans]=1,ans.clear();
    }
    for (auto [u,v]:mp) cout<<u<<'\n';
}
```

Problem 9

```
const int N=2e5+10;
int n,a[N],t[N];

void solve() {
    cin>>n;
    for (int i=1;i<=n;i++) cin>>a[i],t[i]=a[i];
    sort(t+1,t+n+1);
    int m=unique(t+1,t+n+1)-t-1;
    for (int i=1;i<=n;i++) {
        a[i]=lower_bound(t+1,t+m+1,a[i])-t;
        a[i]=m-a[i]+1;
    }
    for (int i=1;i<=n;i++) cout<<a[i]<<' ';
}
```

Problem 10

```
void solve() {
    list<array<int,5>> l;
    int n; cin>>n;
    for (int i=1;i<=n;i++) {
        array<int,5> x;
        cin>>x[0]>>x[1]>>x[2]>>x[3]>>x[4];
        l.push_back(x);
    }
    int m; cin>>m;
    for (int i=1;i<=m;i++) {
        int x,y; cin>>x>>y;
        for (auto it=l.begin();it!=l.end();++it) {
            int id=(*it)[0],xl=(*it)[1],yl=(*it)[2],xr=(*it)[3],yr=(*it)[4]
            if (xl<=x&&x<=xr&&yl<=y&&y<=yr) {
                l.erase(it);
                l.push_front({id,xl,yl,xr,yr});
                break;
            }
        }
    }
    for (auto [id,i1,i2,i3,i4]:l) cout<<id<<' ';
}
```

Problem 11

```
void solve() {
    string s1,s2;
    getline(cin,s1); getline(cin,s2);
    int n=s1.size();
    for (int i=0;i<s2.size();i++) {
        if (s2[i]>='A'&&s2[i]<='Z') {
            int j=i%n;
            s2[i]=((s2[i]-'A')+(s1[j]-'A'))%26+'A';
        }
    }
    cout<<s2<<'\n';
}
```

Problem 12

```
#define fi first
#define se second
const int N=2e5+10;
pair<string,string> a[N];

void solve() {
    int n; cin>>n;
    for (int i=1;i<=n;i++) {
        cin>>a[i].fi>>a[i].se;
    }
    sort(a+1,a+n+1);
    for (int i=1;i<=n;i++) {
        while (a[i].fi.size()>10) a[i].fi.pop_back();
        while (a[i].se.size()>10) a[i].se.pop_back();
        cout<<a[i].fi<<' '<<a[i].se<<'\n';
    }
}
```

Problem 13

```
void solve() {
    stack<int> s;
    int n,r=0; cin>>n;
    for (int i=1;i<=n;i++) {
        int x; cin>>x;
        if (x<r) {
            if (s.empty()||s.top()!=x) {cout<<"NO\n";return ;}
            else s.pop();
        } else {
            for (int i=r+1;i<=x;i++) s.push(i);
            r=x; s.pop();
        }
    }
    cout<<"YES\n";
}
```

Problem 14

```
#define fi first
#define se second
void solve() {
    string str;
    stack<pair<string,int>> s;
    // 0: only contain a single number
    // 1: A + B
    // 2: A - B
    // 3: A * B
    // 4: A / B
    while (cin>>str) {
        if (str=="+") {
            auto v=s.top(); s.pop();
            auto u=s.top(); s.pop();
            s.push({u.fi+str+v.fi,1});
        } else if (str=="-") {
            auto v=s.top(); s.pop();
            auto u=s.top(); s.pop();
            if (v.se==1||v.se==2) v.fi="("+v.fi+")";
            s.push({u.fi+str+v.fi,2});
        } else if (str=="*") {
            auto v=s.top(); s.pop();
            auto u=s.top(); s.pop();
            if (v.se==1||v.se==2) v.fi="("+v.fi+")";
            if (u.se==1||u.se==2) u.fi="("+u.fi+")";
            s.push({u.fi+str+v.fi,3});
        } else if (str=="/") {
            auto v=s.top(); s.pop();
            auto u=s.top(); s.pop();
            if (v.se==1||v.se==2||v.se==3||v.se==4) v.fi="("+v.fi+")";
            if (u.se==1||u.se==2) u.fi="("+u.fi+")";
            s.push({u.fi+str+v.fi,4});
        }
        else s.push({str,0});
    }
    cout<<s.top().fi<<'\n';
}
```

by 方亦晨 的讲解及代码

PA 链表操作

感觉上课讲的很清楚了！有不理解链表怎么跑的可以去看一看课件。

```cpp
#include<iostream>

using std::cin;    using std::cout;
struct node {
  int val;
  node *nxt;
  node(int val) : val(val), nxt(NULL) {}
  node() : val(0), nxt(NULL) {}
};

int main() {
  int x;
  node *head = new node;
  while (1) {
    cin >> x;
    if (x == -1) break;
    node* now = new node(x);
    if (head->nxt != NULL) {
      now->nxt = head->nxt;
      head->nxt = now;
    }
    else head->nxt = now;
  }
  cin >> x;
  node *cur = head, *pre = head;
  while (cur != NULL) {
    if (cur->val == x) {
      pre->nxt = cur->nxt;
      delete cur;
      break;
    }
    if (cur == head) cur = cur->nxt;
    else cur = cur->nxt, pre = pre->nxt;
  }
  cur = head;
  while (cur != NULL) {
    cur = cur->nxt;
    if (cur != NULL) cout << cur->val << ' ';
  }
  return 0;
}
```

PB 求二维整型数组“鞍点”

按照题意模拟。读入数组，然后判断某个数是否不小于其所在行的所有数，同时不大于其所在列的所有数。

```
#include <iostream>
#include <vector>

using std::vector;
using std::cin;    using std::cout;

int main(){
  int n, m; cin >> n >> m;
  vector<vector<int> > Matrix(n, vector<int>(m, 0));
  for (int i=0; i<n; ++i) {
    for (int j=0; j<m; ++j) {
      cin >> Matrix[i][j];
    }
  }
  bool existAns = 0;
  for (int i=0; i<n; ++i) {
    for (int j=0; j<m; ++j) {
      int Flag1 = 1, Flag2 = 1;
      for (int _i=0; _i<n; ++_i) if (Matrix[i][j] > Matrix[_i][j]) Flag1 =
      for (int _j=0; _j<m; ++_j) if (Matrix[i][j] < Matrix[i][_j]) Flag2 =
      if (Flag1 && Flag2) {
        cout << Matrix[i][j] << ' ' << i+1 << ' ' << j+1 << std::endl;
        existAns = 1;
      }
    }
  }
  if (!existAns) cout << "No answer\n";
  return 0;
}
```

PC 对称矩阵乘法运算2

读入上三角元素之后就可以还原出两个矩阵。然后使用矩阵乘法的公式 $C_{i,j} = \sum A_{i,k} \times B_{k,j}$ 计算矩阵乘法即可。

```
#include <iostream>
#include <vector>

using std::vector;
using std::cin;    using std::cout;
```

```cpp
int main() {
  int n, m; cin >> n >> m;
  vector<int> A(m), B(m);
  for (int &i : A) cin >> i;
  for (int &i : B) cin >> i;
  vector<vector<int> > MatA(n, vector<int>(n)), MatB(n, vector<int>(n)), Ma
  int cur = 0;
  for (int i=0; i<n; ++i) {
    for (int j=i; j<n; ++j) {
      MatA[i][j] = MatA[j][i] = A[cur];
      MatB[i][j] = MatB[j][i] = B[cur];
      ++cur;
    }
  }
  for (int i=0; i<n; ++i) {
    for (int j=0; j<n; ++j) {
      for (int k=0; k<n; ++k) {
        MatC[i][j] += MatA[i][k] * MatB[k][j];
      }
    }
  }
  for (int i=0; i<n; ++i) {
    for (int j=0; j<n; ++j) {
      cout << MatC[i][j] << ' ';
    }
    cout << std::endl;
  }

  return 0;
}
```

PD 最小素数集

对于一个整数 p 分解质因数，只需要从 2 循环到 $\sqrt{p}$ 进行试除。如果循环结束之后 $p \neq 1$ 那么说明这个时候的 p 本身就是质数。

用 vector 存所有质因子之后排序并去重即可。

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

using std::vector;
```

```cpp
using std::cin;    using std::cout;

int main() {
  vector<int> ans;
  int N; cin >> N;
  for (int i=0; i<N; ++i) {
    int x; cin >> x;
    for (int p=2; p*p<=x; ++p) if (x%p==0) {
      ans.push_back(p);
      while (x%p==0) x/=p;
      if (x==1) break;
    }
    if (x!=1) ans.push_back(x);
  }
  std::sort(ans.begin(), ans.end());
  ans.erase(std::unique(ans.begin(), ans.end()), ans.end());
  for (int i:ans) cout << i << ' ';
  return 0;
}
```

PE 稀疏矩阵乘法运算

和 PC 类似，读入之后做乘法即可。可以默认矩阵大小为 9×9 因为输出时 0 元会被忽略。

```cpp
#include <vector>
#include <iostream>

using std::vector;
using std::cin;    using std::cout;

vector<vector<int> > matA(10, vector<int>(10)), matB(10, vector<int>(10)),
int main(){
  int N; cin >> N;
  for (int i=0; i<N; ++i) {
    int X, Y, V; cin >> X >> Y >> V;
    matA[X][Y] = V;
  }
  int M; cin>>M;
  for (int i=0; i<M; ++i){
    int X, Y, V; cin >> X >> Y >> V;
    matB[X][Y] = V;
  }
  for (int i=0; i<10; ++i) {
```

```
    for (int j=0; j<10; ++j) {
      for (int k=0; k<10; ++k) {
        matC[i][j] += matA[i][k] * matB[k][j];
      }
    }
  }
  for (int i=0; i<10; ++i) {
    for (int j=0; j<10; ++j) {
      if (matC[i][j]) {
        cout << i << ' ' << j << ' ' << matC[i][j] << '\n';
      }
    }
  }
  return 0;
}
```

PF 五子棋危险判断

枚举所有可能起点，然后判断是否存在四个连续的 1 或 2 且两边不被堵死。

代码里有了一些实现上的技巧来简化上述判断的过程。四个函数理论上也可以用一些技巧变成一个，但是懒得懒。

```
#include <iostream>
#include <vector>

using std::vector;
using std::cin;    using std::cout;

vector<vector<int> > A(21, vector<int>(21));
int check1(int x, int y) {
  // (x,y) -> (x+3,y)
  if (x+3 >= 20) return 0;
  if (A[x-1][y] && A[x+4][y]) return 0;
  return A[x][y] * A[x+1][y] * A[x+2][y] * A[x+3][y];
}
int check2(int x, int y) {
  if (y+3 >= 20) return 0;
  if (A[x][y-1] && A[x][y+4]) return 0;
  return A[x][y] * A[x][y+1] * A[x][y+2] * A[x][y+3];
}
int check3(int x, int y) {
  if (x+3 >= 20 || y-3 <= 0) return 0;
  if (A[x-1][y+1] && A[x+4][y-4]) return 0;
```

```
  return A[x][y] * A[x+1][y-1] * A[x+2][y-2] * A[x+3][y-3];
}
int check4(int x, int y) {
  if (x+3 >= 20 || y+3 >= 20) return 0;
  if (A[x-1][y-1] && A[x+4][y+4]) return 0;
  return A[x][y] * A[x+1][y+1] * A[x+2][y+2] * A[x+3][y+3];
}
int main() {
  for (int i=0; i<=20; ++i) {
    for (int j=0; j<=20; ++j) {
      A[i][j] = -1;
    }
  }
  for (int i=1; i<20; ++i) {
    for (int j=1; j<20; ++j) {
      cin >> A[i][j];
    }
  }
  int Flag = 0;
  for (int i=1; i<20; ++i) {
    for (int j=1; j<20; ++j) {
      int P = check1(i, j), Q = check2(i, j), R = check3(i, j), S = check4(
      if (P == 1 || Q == 1 || R == 1 || S == 1) {
        Flag = 1;
        cout << "1:" << i << ',' << j <<'\n';
      }
      if (P == 16 || Q == 16 || R == 16 || S == 16) {
        Flag = 1;
        cout << "2:" << i << ',' << j <<'\n';
      }
    }
  }
  if (!Flag) cout << "No\n";
  return 0;
}
```

PG 区间

注意文件输入输出。

容易证明按照区间左端点排序之后，可以合并的区间一定是连续的。所以排序之后扫一遍，然后能合并就合并，不能合并就新开一个区间。

```cpp
#include <iostream>
#include <utility>
#include <vector>
#include <algorithm>

using std::vector; using std::pair;
using std::cin;    using std::cout;

int main() {
  freopen("prz.in", "r", stdin);
  freopen("prz.out", "w", stdout);
  int N; cin >> N;
  vector<pair<int, int> > A(N), ans;
  for (auto &[l,r] : A) cin >> l >> r;
  std::sort(A.begin(), A.end());
  int nowL = A[0].first, nowR = A[0].second;
  for (int i=1; i<N; ++i) {
    if (A[i].first > nowR) {
      ans.push_back({nowL, nowR});
      nowL = A[i].first;
      nowR = A[i].second;
    }
    else nowR = std::max(nowR, A[i].second);
  }
  ans.push_back({nowL, nowR});
  for (auto [l,r] : ans) cout << l << ' ' << r << '\n';
  return 0;
}
```

PH 单词索引编排（非文件）

把所有单词拿出来排序。对于单词的识别，如果当前读进来的是一个字母，那么当前单词末尾就要加上这个字母。否则单词结束开始下一个单词。

我不记得有没有讲过 sstream 了，这个东西除了跑的比较慢以外还是很有用的。

```cpp
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
#include <algorithm>

using std::string;
using std::vector;
```

```
using std::cin;    using std::cout;

vector<string> A;
int main() {
  string S, T;
  while (cin >> S) T += S + '#';
  std::istringstream ss(T);
//  cout << T << std::endl;
  S = "";
  char c;
  while (ss >> c) {
    c = tolower(c);
    if (isalpha(c)) S += c;
    else {
      A.push_back(S);
      S = "";
    }
  }
  std::sort(A.begin(), A.end());
  A.erase(std::unique(A.begin(), A.end()), A.end());
  for (string s : A) cout << s << '\n';
  return 0;
}
```

PI 给一个整型数组编号b

偷懒用了非常经典的 lower_bound 实现离散化的做法。反正数据范围大力扫也能过。

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

using std::map;
using std::vector;
using std::cin;    using std::cout;

int main() {
  int N; cin >> N;
  vector<int> A(N), B(N);
  for (int &i : A) cin >> i;
  std::copy(A.begin(), A.end(), B.begin());
  std::sort(B.begin(), B.end());
  B.erase(std::unique(B.begin(), B.end()), B.end());
```

```
  for (int &i : A) i = B.size() - std::distance(B.begin(), std::lower_bound
  for (int i : A) cout << i << ' ';
  return 0;
}
```

PJ 窗口点击模拟b

链表模拟整个过程。点击操作可以认为是遍历链表找到第一个包含当前坐标的窗口，然后把这个窗口提到最前的过程。这都可以转化成基本的链表操作。

```
#include <iostream>

using std::cin;    using std::cout;
struct Window {
  int id, sx, sy, ex, ey;
  Window (int id, int sx, int sy, int ex, int ey) :
    id(id), sx(sx), sy(sy), ex(ex), ey(ey) {}
  Window () {}
};
struct Node {
  Window W;
  Node *nxt, *pre;
  Node () {}
  Node (Window W) : W(W), nxt(nullptr) {}
};
bool check(int X, int Y, Window W) {
  if (W.sx <= X && W.ex >= X && W.sy <= Y && W.ey >= Y) return 1;
  return 0;
}
int main() {
  Node *head = new Node;
  Node *tail = new Node;
  head -> nxt = tail;
  tail -> pre = head;
  int N; cin >> N;
  for (int i=1; i<=N; ++i) {
    Window W; cin >> W.id >> W.sx >> W.sy >> W.ex >> W.ey;
    Node *now = new Node(W);
    now -> pre = tail -> pre;
    now -> nxt = tail;
    now -> pre -> nxt = now;
    tail -> pre = now;
  }
  int M; cin >> M;
```

```
  for (int i=1; i<=M; ++i) {
    int X, Y; cin >> X >> Y;
    Node *cur = head -> nxt;
    while (cur != tail) {
      if (check(X, Y, cur -> W)) {
//        cout << (cur -> W).id << std::endl;
        Node *Tmp = cur;
        cur -> nxt -> pre = cur -> pre;
        cur -> pre -> nxt = cur -> nxt;
//        delete cur;
        cur = Tmp;
        head -> nxt -> pre = cur;
        cur -> nxt = head -> nxt;
        head -> nxt = cur;
        cur -> pre = head;
        break;
      }
      cur = cur -> nxt;
    }
  }
  Node *cur = head -> nxt;
  while (cur != tail) {
    cout << (cur -> W).id << ' ';
    cur = cur -> nxt;
  }
  return 0;
}
```

PL Vigenere加密算法

读入所有字符，然后按照题意进行加密即可。注意非字母字符占密钥一位但是不加密。

```
#include <iostream>
#include <string>
#include <cctype>

using std::string;
using std::cin;    using std::cout;
using std::getline;

int main() {
  string key, info;
  getline(cin, key);
  getline(cin, info);
```

```
  int cur = 0, keysz = key.size();
  for (char &i : info) {
    if(isalpha(i)) {
      i += key[cur % keysz] - 'A';
      if (i > 'Z') i -= ('Z' - 'A' + 1);
    }
    ++cur;
  }
  cout << info;
  return 0;
}
```

PM 电话薄排序

结构体存储姓名和电话号码，然后排序。输出时如果字符串超过 10 位，就一直 `pop_back` 直到少于 10 位。

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <cstdio>

using std::string;
using std::cin;    using std::cout;
using std::vector;

struct User {
  string name, number;
  bool operator< (User B) const {
    return name < B.name;
  }
};
int main() {
  int N; cin >> N;
  vector<User> A(N);
  for (User &i : A) cin >> i.name >> i.number;
  std::sort(A.begin(), A.end());
  for (User &i : A) {
    while (i.name.size() > 10) i.name.pop_back();
    while (i.number.size() > 10) i.number.pop_back();
    cout << i.name << ' ' << i.number << '\n';
  }
  return 0;
```

```
}
```

PN 判断出栈序列

模拟栈的运作过程。遇到一个数的时候，有三种情况：

- 当前数在栈顶，那么直接弹栈；
- 当前数在栈中，则不合法；
- 当前数不在栈内，则不断进行入栈操作直到当前数入栈，然后弹出当前数。

模拟过程直到所有数都出栈或者无解。

```
#include <iostream>
#include <stack>
#include <vector>

using std::stack;
using std::vector;
using std::cin;
using std::cout;

int main() {
  int N; cin >> N;
  vector<int> in_stack(N+1);
  stack<int> st;
  int cur = 1;
  for (int i=0; i<N; ++i) {
    int X; cin >> X;
    if (in_stack[X] && st.top() != X) {
      cout << "NO\n";
      return 0;
    }
    else if (in_stack[X]) st.pop();
    else {
      while (cur <= X) {
        st.push(cur);
        in_stack[cur] = 1;
        ++cur;
      }
      st.pop();
    }
  }
  cout << "YES\n";
```

```
  return 0;
}
```

PO 后缀式转中缀式

用一个栈维护当前所有的算式或字符。如果读入了一个数字，就压入栈中。否则就弹出栈顶的两个元素，让它们进行运算。

需要注意括号的实现。我直接判断了里面的运算符和外面的运算符的前后关系来确定是否需要加括号。

给一些 case： `a-(b-c)`，`a/(b*c)`，`a*b/c` 等。

```
#include <iostream>
#include <string>
#include <stack>
#include <utility>

using std::pair;
using std::stack;
using std::string;
using std::cin;    using std::cout;

stack<pair<string, int> > st;
int main() {
  string S;
  while (cin >> S) {
    if (S == "+") {
      string P = st.top().first; st.pop();
      string Q = st.top().first; st.pop();
      st.push({Q + S + P, 1});
    }
    else if (S == "-") {
      string P = st.top().first; int OpP = st.top().second; st.pop();
      string Q = st.top().first; st.pop();
      if (OpP == 1) P = "(" + P + ")";
      st.push({Q + S + P, 2});
    }
    else if (S == "*") {
      string P = st.top().first; int OpP = st.top().second; st.pop();
      string Q = st.top().first; int OpQ = st.top().second; st.pop();
      if (OpP == 1 || OpP == 2) P = "(" + P + ")";
      if (OpQ == 1 || OpQ == 2) Q = "(" + Q + ")";
      st.push({Q + S + P, 3});
```

```
    }
    else if (S == "/") {
      string P = st.top().first; int OpP = st.top().second; st.pop();
      string Q = st.top().first; int OpQ = st.top().second; st.pop();
      if (OpP != -1) P = "(" + P + ")";
      if (OpQ == 1 || OpQ == 2) Q = "(" + Q + ")";
      st.push({Q + S + P, 4});
    }
    else st.push({S, -1});
  }
  cout << st.top().first << '\n';
  return 0;
}
```