

ICPC Templates

Shanda

April 1, 2021

Contents

1	基础算法	2
1.1	快速排序	2
1.1.1	模板	2
1.1.2	第 k 个数	2
1.2	归并排序	3
1.2.1	模板	3
1.2.2	逆序对数量	3
1.3	二分	4
1.3.1	数的范围	4
1.3.2	数的三次方根	5
1.4	高精度运算	5
1.4.1	加法	5
1.4.2	减法	6
1.4.3	乘法	7
1.4.4	除法	8
1.5	前缀与差分	8
1.5.1	前缀和	8
1.5.2	子矩阵的和	9
1.5.3	差分	9
1.5.4	差分矩阵	10
1.6	双指针算法	11
1.6.1	最长连续不重复子序列	11
1.6.2	数组元素的目标和	11
1.6.3	判断子序列	12
1.7	位运算	12
1.7.1	二进制中 1 的个数	12
1.8	离散化	13
1.8.1	区间和	13
1.9	区间合并	14
1.9.1	区间合并	14
2	数据结构	15
2.1	单链表	15
2.2	双链表	16
2.3	模拟栈	17
2.4	模拟队列	18
2.5	单调栈	19

2.6	单调队列	20
2.7	kmp	20
2.8	Trie 树	21
2.9	最大异或树	22
2.10	并查集	23
2.10.1	合并集合	23
2.10.2	连通块中点的数量	23
2.10.3	食物链	24
2.11	堆	25
2.11.1	堆排序	25
2.11.2	模拟堆	26
2.12	哈希表	27
2.12.1	模拟散列表	27
2.12.2	字符串哈希	28
3	搜索与图论	29
3.1	DFS	29
3.1.1	排列数字	29
3.1.2	n 皇后问题	29
3.2	BFS	30
3.2.1	走迷宫	30
3.2.2	八重码	31
3.3	树与图的 DFS	32
3.3.1	树的重心	32
3.4	树与图的 BFS	33
3.4.1	图中点的层次	33
3.5	拓扑排序	34
3.5.1	有向图的拓扑排序	34
3.6	Dijkstra	35
3.6.1	求最短路 1	35
3.6.2	求最短路 2	36
3.7	Bellman-Ford	37
3.7.1	有边数限制的最短路	37
3.8	SPFA	37
3.8.1	SPFA 求最短路	37
3.8.2	SPFA 判断负环	38
3.9	Floyd	39
3.9.1	Floyd 求最短路	39
3.10	最小生成树	40
3.10.1	Kruskal	40
3.10.2	Prim	41
3.11	二分图	42
3.11.1	染色法判定二分图	42
3.11.2	匈牙利算法求二分最大匹配	43
4	数论	44
4.1	质数	44
4.1.1	试除法判定质数	44
4.1.2	分解质因数	44

4.1.3	筛质数	45
4.2	约数	46
4.2.1	试除法求约数	46
4.2.2	约数个数	46
4.2.3	约数之和	47
4.2.4	最大公约数	47
4.3	欧拉函数	48
4.3.1	欧拉函数	48
4.3.2	筛法求欧拉函数	48
4.4	快速幂	49
4.4.1	快速幂	49
4.4.2	快速幂求逆元	50
4.5	扩展欧几里得算法	50
4.5.1	扩展欧几里得算法	50
4.5.2	线性同余方程	51
4.6	中国剩余定理	52
4.6.1	表达整数的奇怪方式	52
4.7	高斯消元	53
4.7.1	高斯消元解线性方程组	53
4.7.2	高斯消元解异或线性方程组	53
4.8	求组合数	54
4.8.1	上三角法	54
4.8.2	小费马定理与逆元法	54
4.8.3	卢卡斯定理	55
4.8.4	高精度	56
4.8.5	卡特兰数	57
4.9	容斥原理	58
4.9.1	能被整除的数	58
4.10	博弈论	59
4.10.1	Nim 游戏	59
4.10.2	台阶-Nim 游戏	59
4.10.3	集合-Nim 游戏	60
4.10.4	拆分-Nim 游戏	60
5	动态规划	61
5.1	背包问题	61
5.1.1	01 背包问题	61
5.1.2	完全背包问题	62
5.1.3	多重背包问题	62
5.1.4	多重背包问题 2	62
5.1.5	分组背包问题	63
5.2	线性 DP	64
5.2.1	数字三角形	64
5.2.2	最长上升子序列	64
5.2.3	最长上升子序列 2	65
5.2.4	最长公共子序列	65
5.2.5	最短编辑距离	66
5.3	区间 DP	66
5.3.1	石子合并	66

5.4	计数类 DP	67
5.4.1	整数划分	67
5.5	数位统计 DP	67
5.5.1	计数问题	67
5.6	状态压缩 DP	68
5.6.1	最短 Hamilton 路径	68
5.6.2	蒙德里安的梦想	69
5.7	树形 DP	69
5.7.1	没有上司的舞会	69
5.8	记忆化搜索	70
5.8.1	滑雪	70

1 基础算法

1.1 快速排序

1.1.1 模板

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4 #define N 100010
5 int num[N];
6 int n;
7 int quick_sort(int num[], int l, int r){
8     if(l >= r) return 0;
9     int mid = num[l+r >> 1], i = l-1, j = r + 1;
10    while(i < j){
11        while(num[++i] < mid);
12        while(num[--j] > mid);
13        if(i < j) swap(num[i],num[j]);
14    }
15    quick_sort(num, l, j);
16    quick_sort(num, j + 1, r);
17 }
18 int main(){
19     scanf("%d", &n);
20     for(int i = 0 ; i < n ; i++) scanf("%d", &num[i]);
21     quick_sort(num, 0, n-1);
22     for(int i = 0; i < n; i++) printf("%d ", num[i]);
23     return 0;
24 }
```

1.1.2 第 k 个数

```
1 #include <iostream>
2 using namespace std;
3 #define N 100010
4
5 int num[N];
6 int n,k;
7
8 int quick_sort(int num[], int l, int r,int k){
9     if(l>=r) return 0;
10    int x = num[l + r>>1], i = l-1, j= r+1;
11    while(i < j){
12        while(num[++i] < x);
13        while(num[--j] > x);
14        if(i < j) swap(num[i],num[j]);
15    }
16
17    int sk = j - l + 1;
18    if(k <= sk) quick_sort(num, l, j, k);
19    else quick_sort(num, j+1, r, k - sk);
20 }
```

```
21
22 int main(){
23     scanf("%d %d", &n, &k);
24     for(int i = 0; i < n; i++) scanf("%d", &num[i]);
25     quick_sort(num, 0, n-1, k);
26     printf("%d", num[k-1]);
27     return 0;
28 }
```

1.2 归并排序

1.2.1 模板

```
1  #include<iostream>
2  using namespace std;
3  #define N 100010
4
5  int q[N];
6  int n;
7
8
9  int merge_sort(int q[], int l, int r){
10     if(l>=r) return 0;
11     int mid = l+r >>1;
12     merge_sort(q,l,mid);
13     merge_sort(q,mid+1,r);
14     int i=l,j=mid+1,k=0,tmp[r-l+1];
15     while(i<=mid&&j<=r){
16         if(q[i]<=q[j]) tmp[k++] = q[i++];
17         else tmp[k++] = q[j++];
18     }
19     while(i<=mid) tmp[k++] = q[i++];
20     while(j<=r) tmp[k++] = q[j++];
21     for(int i=0,j=l;j<=r;i++,j++) q[j] = tmp[i];
22 }
23
24 int main(){
25     scanf("%d", &n);
26     for(int i = 0 ; i < n; i++) scanf("%d", &q[i]);
27     merge_sort(q,0,n-1);
28     for(int i = 0; i < n ; i++) printf("%d ", q[i]);
29     return 0;
30 }
```

1.2.2 逆序对数量

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4
5  #define N 100010
6
```

```
7  int q[N];
8  int n;
9  long long res=0;
10
11 int merge_sort(int q[], int l, int r){
12     if(l>=r) return 0;
13     int mid = l + r >> 1;
14     merge_sort(q,l,mid);
15     merge_sort(q,mid+1,r);
16     int i=l,j=mid+1,k=0,tmp[r-l+1];
17     while(i<=mid&&j<=r){
18         if(q[i] <= q[j]) tmp[k++] = q[i++];
19         else {
20             tmp[k++] = q[j++];
21             res += (mid - i + 1);
22         }
23     }
24     while(i<=mid) tmp[k++] = q[i++];
25     while(j<=r) tmp[k++] = q[j++];
26     for(int i=0,j=1;j<=r;i++,j++) q[j] = tmp[i];
27 }
28
29 int main(){
30     scanf("%d", &n);
31     for(int i= 0; i< n ; i++) scanf("%d", &q[i]);
32     merge_sort(q, 0, n-1);
33     printf("%ld",res);
34     return 0;
35 }
```

1.3 二分

1.3.1 数的范围

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4  #define N 100010
5
6
7  int n,q,k;
8  int a[N];
9
10 int lower_find(int a[], int l, int r, int k){
11     while(l < r){
12         int mid = l + r >> 1;
13         if(k <= a[mid]) r = mid;
14         else l = mid+1;
15     }
16     return a[l] == k ? l : -1;
17 }
18
```

```
19 int upper_find(int a[], int l, int r, int k){
20     while(l < r){
21         int mid = l+r+1 >> 1;
22         if(k >= a[mid]) l = mid;
23         else r = mid -1;
24     }
25     return a[l] == k ? l : -1;
26 }
27
28 int main(){
29     scanf("%d %d",&n, &q);
30     for(int i= 0 ;i< n;i++) scanf("%d", &a[i]);
31     for(int i=0 ;i < q;i++) {
32         scanf("%d", &k);
33         int res = lower_find(a, 0, n-1, k);
34         if(res == -1) printf("%d %d", -1, -1);
35         else{
36             printf("%d ", res);
37             res = upper_find(a, 0, n-1, k);
38             printf("%d", res);
39         }
40         if(i != q - 1)
41             printf("\n");
42     }
43 }
```

1.3.2 数的三次方根

```
1 #include <iostream>
2 using namespace std;
3 # define N 100010
4
5 double n;
6
7 int main(){
8     scanf("%lf", &n);
9     double l = -10000, r = 10000, x=0;
10    while(r - l >= 10e-8){
11        x = (l + r) / 2;
12        if(x * x * x >=n) r = x;
13        else l = x;
14    }
15    printf("%lf", x);
16 }
```

1.4 高精度运算

1.4.1 加法

```
1 #include <iostream>
2 #include <algorithm>
3 #include <string>
```



```

4  #include <vector>
5  using namespace std;
6
7  vector<int> add(vector<int> &A, vector<int> &B){
8      vector<int> C;
9      int t = 0;
10     for(int i=0; i < A.size() || i < B.size() ; i++){
11         if(i < A.size()) t += A[i];
12         if(i < B.size()) t += B[i];
13         C.push_back(t % 10);
14         t /= 10;
15     }
16     if(t) C.push_back(1);
17     return C;
18 }
19
20 int main(){
21     vector<int> A,B;
22     string a,b;
23     cin >> a >> b;
24     for(int i = a.size() - 1; i>=0; i--) A.push_back(a[i] - '0');
25     for(int i = b.size() - 1; i>=0;i--) B.push_back(b[i] - '0');
26     auto C = add(A,B);
27     for(int i = C.size() - 1; i>=0;i--) printf("%d", C[i]);
28     return 0;
29 }

```

1.4.2 减法

```

1  #include<iostream>
2  #include<string>
3  #include<algorithm>
4  #include<vector>
5  using namespace std;
6
7
8  // A >= B
9  bool cmp(vector<int> &A, vector<int> &B){
10     if(A.size() > B.size()) return true;
11     else if(A.size() < B.size()) return false;
12     else{
13         for(int i = A.size() - 1; i >= 0; i--)
14             if(A[i] != B[i]) return A[i] > B[i];
15         return true;
16     }
17 }
18
19 vector<int> sub(vector<int> &A, vector<int> &B){
20     vector<int> C;
21     int t = 0;
22     for(int i = 0; i < A.size(); i++){
23         t = A[i] - t; // 减掉进位

```

```

24     if(i < B.size()) t -= B[i];
25     C.push_back((t+10)%10);
26     if(t<0) t=1;
27     else t=0;
28 }
29 while(C.size() > 1 && C.back() == 0) C.pop_back();
30 return C;
31 }
32
33 int main(){
34     string a, b;
35     vector<int> A, B;
36     cin >> a >> b;
37     for(int i = a.size() - 1; i>=0; i--) A.push_back(a[i] - '0');
38     for(int i = b.size() - 1; i>=0; i--) B.push_back(b[i] - '0');
39     vector<int> C;
40     if(cmp(A,B)){
41         C = sub(A,B);
42     }
43     else{
44         printf("%c", '-');
45         C = sub(B,A);
46     }
47     for(int i = C.size() - 1; i>=0 ; i--) printf("%d", C[i]);
48     return 0;
49 }

```

1.4.3 乘法

```

1  #include<iostream>
2  #include<vector>
3  #include<algorithm>
4  #include<string>
5  using namespace std;
6
7
8  vector<int> mul(vector<int> &A, int b){
9      vector<int> C;
10     int t = 0;
11     for(int i = 0; i < A.size() || t; i++){
12         if(i < A.size()) t += A[i] * b;
13         C.push_back(t % 10);
14         t /= 10;
15     }
16     while(C.size() > 1 && C.back() == 0) C.pop_back();
17     return C;
18 }
19
20
21 int main(){
22     string a;
23     int b;

```

```

24     vector<int> A,B;
25     cin>>a>>b;
26     for(int i = a.size() - 1 ;i >= 0 ; i --) A.push_back(a[i] - '0');
27     // for(int i = b.size() - 1 ;i >= 0 ; i ++) B.push_back(b[i] - '0');
28     auto C = mul(A,b);
29     for(int i = C.size() - 1; i>=0; i--) printf("%d" ,C[i]);
30 }

```

1.4.4 除法

```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <algorithm>
5  using namespace std;
6
7
8  vector<int> div(vector<int> &A, int b, int &r){
9      vector<int> C;
10     int t = 0;
11     for(int i = A.size() -1 ; i >=0 ;i --){
12         r = r * 10 + A[i];
13         C.push_back(r / b);
14         r %= b;
15     }
16     reverse(C.begin(),C.end());
17     while(C.size() > 1 && C.back() == 0) C.pop_back();
18     return C;
19 }
20
21
22 int main(){
23     string a;
24     int b, r=0;
25     vector<int> A;
26     cin >> a >> b;
27     for(int i = a.size() - 1 ; i>=0; i--) A.push_back(a[i] - '0');
28     auto C = div(A,b,r);
29     for(int i =C.size() -1 ; i>=0; i--) printf("%d", C[i]);
30     printf("\n%d", r);
31     return 0;
32 }

```

1.5 前缀与差分

1.5.1 前缀和

```

1  #include<iostream>
2  using namespace std;
3  #define N 100010
4  int a[N];
5  int s[N];

```

```

6  int n,m,l,r;
7  int main(){
8      scanf("%d %d",&n,&m);
9      for(int i = 1; i<=n; i++) scanf("%d", &a[i]);
10     for(int i =1 ;i<=n;i++) S[i] = S[i-1] + a[i];
11     while(m--){
12         scanf("%d %d", &l, &r);
13         printf("%d\n",S[r] - S[l-1]);
14     }
15 }

```

1.5.2 子矩阵的和

```

1  #include<iostream>
2  using namespace std;
3
4  #define N 1010
5
6  int a[N][N];
7  int S[N][N];
8
9  int n,m,q,x1,x2,y1,y2;
10
11 int main(){
12     scanf("%d %d %d",&n,&m, &q);
13     for(int i = 1; i<=n;i++)
14         for(int j=1; j<=m; j++)
15             scanf("%d", &a[i][j]);
16
17     for(int i = 1;i<=n;i++)
18         for(int j = 1; j<=m;j++)
19             S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + a[i][j];
20
21
22     while(q--){
23         scanf("%d %d %d %d", &x1,&y1, &x2, &y2);
24         printf("%d\n", S[x2][y2] - S[x1-1][y2] - S[x2][y1-1] + S[x1-1][y1-1]);
25     }
26     return 0;
27 }

```

1.5.3 差分

```

1  #include <iostream>
2  using namespace std;
3
4  #define N 100010
5
6  int a[N], b[N];
7  int n,m,l,r,c;
8

```

```

9  int insert(int b[],int l, int r , int c){
10     b[l] += c;
11     b[r+1] -= c;
12 }
13
14 int main(){
15     scanf("%d %d", &n, &m);
16     for(int i=1; i<=n;i++) scanf("%d", &a[i]);
17     for(int i=1; i<=n;i++) insert(b,i,i,a[i]);
18     while(m--){
19         scanf("%d %d %d",&l, &r, &c);
20         insert(b,l,r,c);
21     }
22     for(int i=1;i<=n;i++)
23         b[i] += b[i-1];
24     for(int i=1;i<=n;i++) printf("%d ", b[i]);
25 }

```

1.5.4 差分矩阵

```

1  #include <iostream>
2  using namespace std;
3  #define N 1010
4
5
6  int a[N][N], b[N][N];
7
8  int n,m,q,x1,x2,y1,y2,c;
9
10 int insert(int x1, int x2, int y1 ,int y2, int c){
11     b[x1][y1] += c;
12     b[x1][y2+1] -= c;
13     b[x2+1][y1] -=c;
14     b[x2+1][y2+1] +=c;
15     return 0;
16 }
17
18 int main(){
19     scanf("%d %d %d",&n,&m,&q);
20     for(int i = 1;i <=n ;i++)
21         for(int j = 1; j<=m; j++){
22             scanf("%d", &a[i][j]);
23             insert(i,i,j,j,a[i][j]);
24         }
25     while(q--){
26         scanf("%d %d %d %d %d", &x1,&y1,&x2,&y2,&c);
27         insert(x1,x2,y1,y2,c);
28     }
29
30     for(int i = 1; i<=n ;i++)
31         for(int j=1; j<=m; j++)
32             b[i][j] = b[i-1][j] + b[i][j-1] - b[i-1][j-1] + a[i][j];

```

```
33
34     for(int i = 1;i<=n;i++){
35         for(int j = 1 ; j<=m; j++) {
36             printf("%d ", b[i][j]);
37         }
38         printf("\n");
39     }
40 }
```

1.6 双指针算法

1.6.1 最长连续不重复子序列

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4
5
6  #define N 100010
7
8  int n,a[N];
9  int cnt[N];
10
11 int main(){
12     scanf("%d", &n);
13     int res = 0;
14     for(int i=0; i<n;i++) scanf("%d", &a[i]);
15
16     for(int i = 0,j=i ; i< n; i++){
17         ++cnt[a[i]];
18         while(cnt[a[i]] > 1) --cnt[a[j++]];
19         res = max(res, i-j + 1);
20     }
21
22     printf("%d", res);
23
24     return 0;
25 }
```

1.6.2 数组元素的目标和

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4
5  #define N 100010
6
7  int n,m,x;
8  int a[N], b[N];
9
10 int main(){
11     scanf("%d %d %d", &n, &m, &x);
```

```
12     for(int i = 0 ; i < n; i++) scanf("%d", &a[i]);
13     for(int i = 0 ; i < m; i++) scanf("%d", &b[i]);
14     int i = n-1, j = 0;
15     while(i >= 0 || j < m){
16         if(i>=0 && a[i] + b[j] > x) i--;
17         else if(j < m && a[i] + b[j] < x) j++;
18         else if(i>=0 && j < m && a[i]+b[j] == x){
19             printf("%d %d", i ,j);
20             break;
21         };
22     }
23     return 0;
24 }
```

1.6.3 判断子序列

```
1  #include <iostream>
2  #include <algorithm>
3  #include <string>
4  #include <map>
5  using namespace std;
6  #define N 100010
7
8  int a[N], b[N];
9  int n, m;
10 int main(){
11     scanf("%d %d", &n, &m);
12     for(int i = 0 ; i < n ; i++) scanf("%d", &a[i]);
13     for(int i = 0 ; i < m ; i++) scanf("%d", &b[i]);
14     int i = 0, j = 0;
15     for(; j < m; j++){
16         if(i < n && a[i] == b[j]) i++;
17     }
18     if(i == n) printf("Yes");
19     else printf("No");
20
21     return 0;
22 }
```

1.7 位运算

1.7.1 二进制中 1 的个数

```
1  #include <iostream>
2  using namespace std;
3
4  #define N 100010
5
6  int lowbit(int x){
7     return x & -x;
8 }
9
```

```
10 int n;
11
12 int main(){
13     scanf("%d", &n);
14     int x;
15     for(int i = 0 ; i < n; i++ ){
16         scanf("%d", &x);
17         int cnt = 0;
18         while(x){
19             x = x - lowbit(x);
20             cnt++;
21         }
22         printf("%d ",cnt);
23     }
24     return 0;
25 }
```

1.8 离散化

1.8.1 区间和

假定有一个无限长的数轴，数轴上每个坐标上的数都是 0。现在，我们首先进行 n 次操作，每次操作将某一位置 x 上的数加 c 。接下来，进行 m 次询问，每个询问包含两个整数 l 和 r ，你需要求出在区间 $[l, r]$ 之间的所有数的和。

```
1  #include<iostream>
2  #include<algorithm>
3  #include<vector>
4  using namespace std;
5
6  #define N 300000 + 10
7  typedef pair<int,int> PII;
8  vector<PII> opers,query;
9  vector<int> alls;
10 int a[N];
11
12 int n, m, x,c, l,r;
13
14 int binary_search(int k){
15     int l = 0 , r = alls.size() - 1;
16     while(l < r){
17         int mid = l + r >> 1;
18         if(k <= alls[mid]) r = mid;
19         else l = mid + 1;
20     }
21     return alls[l] == k ? l+1 : -1;
22 }
23
24 int main(){
25     scanf("%d %d", &n, &m);
26     for(int i = 0 ; i < n; i++){
27         scanf("%d %d", &x, &c);
```



```

28     ops.push_back({x,c});
29     alls.push_back(x);
30 }
31
32 for(int i= 0; i < m; i++){
33     scanf("%d %d", &l, &r);
34     query.push_back({l,r});
35     alls.push_back(l);
36     alls.push_back(r);
37 }
38 sort(alls.begin(),alls.end());
39 alls.erase(unique(alls.begin(),alls.end()),alls.end());
40
41 for(auto p : ops){
42     int index = binary_search(p.first);
43     a[index] += p.second;
44 }
45
46 for(int i = 1; i<= alls.size(); i++) a[i] = a[i-1] + a[i];
47
48 for(auto p : query){
49     int l = binary_search(p.first);
50     int r = binary_search(p.second);
51     printf("%d\n", a[r] - a[l-1]);
52 }
53
54 }

```

1.9 区间合并

1.9.1 区间合并

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  typedef pair<int,int> PII;
7
8  int n;
9  int l, r;
10 vector<PII> segs;
11
12 int merge(vector<PII> &segs){
13     sort(segs.begin(),segs.end());
14     int st = -2e9, ed = -2e9;
15     vector<PII> res;
16     for(auto seg : segs){
17         if(ed < seg.first){
18             if(st!=-2e9) res.push_back({st,ed});
19             st = seg.first;
20             ed = seg.second;

```

```
21     }
22     else{
23         ed = max(ed, seg.second);
24     }
25 }
26 if(st != -2e9) res.push_back({st,ed});
27 return res.size();
28 }
29
30 int main(){
31     scanf("%d",&n);
32     while(n--){
33         scanf("%d %d", &l,&r);
34         segs.push_back({l,r});
35     }
36     int res = merge(segs);
37     printf("%d", res);
38     return 0;
39 }
```

2 数据结构

2.1 单链表

```
1
2 #include <iostream>
3 #include <string>
4 using namespace std;
5 const int N = 100010;
6 string c;
7 int head,idx,el[N],ne[N],k,x,m;
8 void init(){
9     head = -1;
10    idx = 0;
11 }
12 void add_head(int x){
13     el[idx] = x, ne[idx] = head, head = idx++;
14 }
15 void add(int k, int x){
16     el[idx] = x; ne[idx] = ne[k], ne[k] = idx++;
17 }
18 void del_head(){
19     head = ne[head];
20 }
21 void del(int k){
22     ne[k] = ne[ne[k]];
23 }
24 int main(){
25     cin >> m;
26     init();
27     while(m--){
28         cin >> c;
```

```
29     if(c == "H"){
30         cin >> x;
31         add_head(x);
32     }
33     if(c == "D"){
34         cin >> k;
35         if(!k) del_head();
36         else del(k-1);
37     }
38     if(c == "I"){
39         cin >> k >> x;
40         add(k-1,x);
41     }
42 }
43 for(int i = head; i!=-1;i=ne[i]){
44     cout << el[i] << ' ';
45 }
46 return 0;
47 }
```

2.2 双链表

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  const int N = 100010;
6
7  string oper;
8  int m,k,x,idx,l[N],r[N],el[N];
9
10 void init(){
11     // 0是左节点, 1是右节点;
12     r[0] = 1, l[1]=0;
13     idx = 2;
14 }
15 //在第k个数的右边添加新节点
16 void add(int k, int x){
17     el[idx] = x, l[idx] = k, r[idx] = r[k], l[r[k]] = idx, r[k] = idx ++;
18 }
19 //删除第k个节点
20 void del(int k){
21     r[l[k]] = r[k];
22     l[r[k]] = l[k];
23 }
24 int main(){
25     cin >> m;
26     init();
27     while(m--){
28         cin >> oper;
29         if(oper == "L"){
30             cin >> x;
```

```
31         add(0,x);
32     }
33     if(oper == "R"){
34         cin >> x;
35         add(1[1],x);
36     }
37     if(oper == "D"){
38         cin >> k;
39         del(k+1);
40     }
41     if(oper == "IL"){
42         cin >> k >> x;
43         add(1[k+1], x);
44     }
45     if(oper == "IR"){
46         cin >> k >> x;
47         add(k+1, x);
48     }
49 }
50 for(int i = r[0]; i!=1; i = r[i]){
51     cout << el[i] << ' ';
52 }
53 return 0;
54 }
```

2.3 模拟栈

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  const int N = 100010;
6
7  int m;
8  int s[N], t=0, x;
9  string oper;
10
11 void init(){
12     t = 0;
13 }
14
15 void push(int x){
16     s[++t] = x;
17 }
18
19 void pop(){
20     --t;
21 }
22
23 string empty(){
24     return t == 0 ? "YES" : "NO";
25 }
```

```
26
27 int query(){
28     return s[t];
29 }
30
31
32 int main(){
33     cin >> m;
34     init();
35     while(m--){
36         cin >> oper;
37         if(oper == "push"){
38             cin >> x;
39             push(x);
40         }
41         if(oper == "pop"){
42             pop();
43         }
44         if(oper == "empty"){
45             cout << empty() << endl;
46         }
47         if(oper == "query"){
48             cout << query() << endl;
49         }
50     }
51 }
```

2.4 模拟队列

```
1  #include<iostream>
2  #include<string>
3  #include<algorithm>
4  using namespace std;
5  const int N = 100010;
6
7  int q[N], h,t,m,x;
8  string o;
9
10 void init(){
11     h = 0, t = -1;
12 }
13
14 void push(int x){
15     q[++t] = x;
16 }
17
18 void pop(){
19     ++h;
20 }
21
22 int query(){
23     return q[h];
24 }
```

```
24 }
25
26 string empty(){
27     return h > t ? "YES" : "NO";
28 }
29
30 int main(){
31     cin >> m;
32     init();
33     while(m--){
34         cin >> o;
35         if(o == "push"){
36             cin >> x;
37             push(x);
38         }
39         if(o == "pop"){
40             pop();
41         }
42         if(o == "empty"){
43             cout << empty() << endl;
44         }
45         if(o == "query"){
46             cout << query() << endl;
47         }
48     }
49 }
```

2.5 单调栈

```
1  #include <iostream>
2  using namespace std;
3
4  const int N = 100010;
5
6  int s[N], a[N], t=0,n;
7
8
9  int main(){
10     scanf("%d", &n);
11     for(int i = 0 ; i < n; i++) scanf("%d", &a[i]);
12     for(int i = 0; i < n; i++){
13         while(t && s[t] >= a[i]) t--;
14         if(t){
15             printf("%d ",s[t]);
16         }
17         else{
18             printf("%d ",-1);
19         }
20         s[++t] = a[i];
21     }
22     return 0;
23 }
```

2.6 单调队列

```

1  #include <iostream>
2  using namespace std;
3
4  const int N = 1000010;
5  int n,k,q[N],a[N];
6  int h=0,t=-1;
7
8
9  int main(){
10     scanf("%d %d",&n,&k);
11     for(int i = 0 ; i < n;i++) scanf("%d", &a[i]);
12
13     for(int i = 0; i< n;i++) {
14         while(h<=t && q[h] < i - k + 1) h++; // 去除不在滑动窗口内的下标;
15         while(h<=t && a[q[t]]>= a[i]) t--; // 单调队列中所有元素小于当前元素;
16         q[++t] = i; // 队列储存的是下标;
17         if(i+1>=k){ // 滑动窗口未完全进入不输出;
18             printf("%d ", a[q[h]]);
19         }
20     }
21     puts("");
22     h=0, t=-1;
23     for(int i = 0; i< n;i++) {
24         while(h<=t && q[h] < i - k + 1) h++;
25         while(h<=t && a[q[t]]<= a[i]) t--;
26         q[++t] = i;
27         if(i+1>=k){
28             printf("%d ", a[q[h]]);
29         }
30     }
31
32     return 0;
33 }

```

2.7 kmp

```

1  #include <iostream>
2  using namespace std;
3
4  const int N = 1000010;
5
6  int ne[N],n,m;
7  char p[N],s[N];
8
9  int main(){
10     scanf("%d %s %d %s",&n, p+1, &m, s+1);
11     for(int i = 2 ,j = 0; i <=n;i++){
12         while(j && p[i] != p[j+1]) j = ne[j];
13         if(p[i] == p[j+1]) j++;
14         ne[i] = j;

```

```

15     }
16     for(int i = 1, j = 0; i <= m; i++){
17         while(j && s[i] != p[j+1]) j = ne[j];
18         if(s[i] == p[j+1]) j++;
19         if(j == n){
20             j = ne[j];
21             printf("%d ", i-n);
22         }
23     }
24 }

```

2.8 Trie 树

```

1  #include<iostream>
2  using namespace std;
3
4  const int N = 2e4 + 10;
5
6  char oper[2], str[N];
7  int cnt[N], n, s[N][26], idx = 0;
8
9  void insert(char * str){
10     int p = 0;
11     for(int i = 0; str[i]; i++){
12         int c = str[i] - 'a';
13         if(!s[p][c]) s[p][c] = ++idx;
14         p = s[p][c];
15     }
16     ++cnt[p];
17 }
18
19 int query(char * str){
20     int p = 0;
21     for(int i = 0; str[i]; i++){
22         int c = str[i] - 'a';
23         if(!s[p][c]) return 0;
24         p = s[p][c];
25     }
26     return cnt[p];
27 }
28
29 int main(){
30     scanf("%d", &n);
31     while(n--){
32         scanf("%s %s", oper, str);
33         // cout << oper << endl << str << endl;
34         if(oper[0] == 'I'){
35             // printf("%d\n", insert())
36             insert(str);
37         }
38         if(oper[0] == 'Q'){
39             printf("%d\n", query(str));

```



```
40     }
41 }
42 return 0;
43 }
```

2.9 最大异或树

```
1  #include <iostream>
2  using namespace std;
3  const int N = 1e5 * 31 + 10;
4
5
6  int n, son[N][2], idx=0;
7
8  void insert(int x){
9      int p = 0;
10     for(int i = 31; i >= 0; i--){ // 贪心, 从整数最高位开始构造Tried树;
11         int b = x >> i & 1;
12         if(!son[p][b]) son[p][b] = ++idx;
13         p = son[p][b];
14     }
15 }
16
17 int query(int x){
18     int p = 0, res=0;
19     for(int i = 31; i >= 0; i--){
20         int b = x >> i & 1;
21         if(!son[p][!b]){
22             res = 2 * res;
23             p = son[p][b];
24         }
25         else{
26             res = 2 * res + 1;
27             p = son[p][!b];
28         }
29     }
30     return res;
31 }
32
33
34 int main(){
35     scanf("%d", &n);
36     int res = 0, x=0;
37     for(int i = 0; i < n; i++){
38         scanf("%d", &x);
39         insert(x);
40         res = max(res, query(x));
41     }
42     printf("%d", res);
43
44     return 0;
45 }
```

2.10 并查集

2.10.1 合并集合

```
1  #include <iostream>
2  using namespace std;
3  const int N = 1e5 * 31 + 10;
4
5
6  int n, son[N][2], idx=0;
7
8  void insert(int x){
9      int p = 0;
10     for(int i = 31; i >= 0; i--){ // 贪心, 从整数最高位开始构造Tried树;
11         int b = x >> i & 1;
12         if(!son[p][b]) son[p][b] = ++idx;
13         p = son[p][b];
14     }
15 }
16
17 int query(int x){
18     int p = 0, res=0;
19     for(int i = 31; i >= 0; i--){
20         int b = x >> i & 1;
21         if(!son[p][!b]){
22             res = 2 * res;
23             p = son[p][b];
24         }
25         else{
26             res = 2 * res + 1;
27             p = son[p][!b];
28         }
29     }
30     return res;
31 }
32
33
34 int main(){
35     scanf("%d", &n);
36     int res = 0, x=0;
37     for(int i = 0; i < n; i++){
38         scanf("%d", &x);
39         insert(x);
40         res = max(res, query(x));
41     }
42     printf("%d", res);
43
44     return 0;
45 }
```

2.10.2 连通块中点的数量

```
1  #include <iostream>
```

```
2 using namespace std;
3
4 const int N = 100010;
5 char oper[3];
6 int p[N], s[N], n, m, x, y;
7
8 int find(int x){
9     if(p[x] != x) p[x] = find(p[x]);
10    return p[x];
11 }
12
13
14 int main(){
15     scanf("%d %d", &n, &m);
16     for(int i = 1; i <= n; i++) p[i] = i, s[i] = 1;
17     while(m--){
18         scanf("%s", oper);
19         if(oper[0] == 'C'){
20             scanf("%d%d", &x, &y);
21             int px = find(x), py = find(y);
22             if(px != py){
23                 s[py] += s[px];
24                 p[px] = py;
25             }
26         }
27         if(oper[0] == 'Q'){
28             if(oper[1] == '1'){
29                 scanf("%d%d", &x, &y);
30                 if(find(x) == find(y)) printf("Yes\n");
31                 else printf("No\n");
32             }
33             if(oper[1] == '2'){
34                 scanf("%d", &x);
35                 printf("%d\n", s[find(x)]);
36             }
37         }
38     }
39 }
```

2.10.3 食物链

```
1 #include <iostream>
2 using namespace std;
3
4 const int N = 50010;
5
6
7 int n, k, d, x, y, res = 0, dist[N], p[N];
8
9 int find(int x){
10     if(p[x] != x){
11         int t = find(p[x]);
```

```

12     dist[x] += dist[p[x]];
13     p[x] = t;
14 }
15 return p[x];
16 }
17
18 int main(){
19     scanf("%d%d",&n,&k);
20     for(int i = 1;i<=n;i++) p[i] = i, dist[i] = 0;
21     while(k--){
22         scanf("%d%d%d",&d,&x,&y);
23         if( x > n || y > n) res++;
24         else{
25             if(d==1){
26                 int px = find(x), py = find(y);
27                 if(px == py && (dist[x] - dist[y]) % 3) res++;
28                 else if(px!=py){
29                     p[px] = py;
30                     dist[px] = dist[y] - dist[x];
31                 }
32             }
33             else{
34                 int px = find(x), py =find(y);
35                 if(px == py && (dist[x] - dist[y] - 1) %3) res++;
36                 else if(px!=py){
37                     p[px] = py;
38                     dist[px] = dist[y] + 1 -dist[x];
39                 }
40             }
41         }
42     }
43     printf("%d",res);
44 }

```

2.11 堆

2.11.1 堆排序

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  const int N = 100010;
6
7  int n,m,h[N];
8
9
10
11 void down(int u){
12     int t = u;
13     if(2*u<=n&&h[2*u] < h[t]) t = 2*u;
14     if(2*u+1<=n&&h[2*u+1] < h[t]) t = 2*u+1;

```

```

15     if(u!=t){
16         swap(h[u],h[t]);
17         down(t);
18     }
19 }
20
21
22 int main(){
23     scanf("%d%d",&n,&m);
24     for(int i = 1;i<=n;i++) scanf("%d",&h[i]);
25     for(int i = n/2;i--){
26         down(i);
27     }
28     while(m--){
29         printf("%d ", h[1]);
30         swap(h[1],h[n]);
31         n--;
32         down(1);
33     }
34     return 0;
35 }

```

2.11.2 模拟堆

```

1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  const int N = 100010;
5
6  char o[3];
7  // string o;
8  int idx=0,n,sz=0,x,k,h[N],hp[N],ph[N];
9
10
11 void heap_swap(int u, int d){
12     swap(ph[hp[u]],ph[hp[d]]);
13     swap(h[u],h[d]);
14     swap(hp[u],hp[d]);
15 }
16
17 int down(int u){
18     int t = u;
19     if(2*u<=sz && h[2*u] < h[t]) t = 2*u;
20     if(2*u+1<=sz && h[2*u+1] < h[t]) t = 2*u+1;
21     if(u!=t){
22         heap_swap(u,t);
23         down(t);
24     }
25 }
26
27
28 int up(int u){

```

```

29     while (u/2 && h[u] < h[u/2]){
30         heap_swap(u/2,u);
31         u >>=1;
32     }
33 }
34
35
36 int main(){
37     scanf("%d", &n);
38     while(n--){
39         scanf("%s", o);
40         if(!strcmp(o,"I")){ // 插入一个数
41             scanf("%d", &x);
42             h[++sz] = x;
43             hp[sz] = ++idx;
44             ph[idx] = sz;
45             up(sz);
46         }
47         if(!strcmp(o,"PM")){ // 输出当前集合中的最小值
48             printf("%d\n", h[1]);
49         }
50         if(!strcmp(o,"DM")){ // 删除当前集合的最小值
51             heap_swap(1,sz--);
52             down(1);
53         }
54         if(!strcmp(o,"D")){ // 删除第k个插入的数
55             scanf("%d", &k);
56             int u = ph[k];
57             heap_swap(u, sz--);
58             up(u);
59             down(u);
60         }
61         if(!strcmp(o,"C")){ // 修第k个插入的数
62             scanf("%d%d",&k, &x);
63             h[ph[k]] = x;
64             up(ph[k]);
65             down(ph[k]);
66         }
67     }
68 }

```

2.12 哈希表

2.12.1 模拟散列表

```

1 #include<iostream>
2 #include <string.h>
3 using namespace std;
4
5 const int N = 100010;
6 char oper[2];
7 int idx=0,n,x, e1[N],ne[N],h[N], mod =100003;

```

```

8
9 void insert(int x){
10     int t = (x % mod + mod) % mod;
11     el[idx] = x, ne[idx] = h[t], h[t] = idx++;
12 }
13 bool query(int x){
14     int t = (x % mod + mod) % mod;
15     for(int i = h[t]; i != -1; i = ne[i]){
16         if(el[i] == x) return true;
17     }
18     return false;
19 }
20
21 int main(){
22     scanf("%d", &n);
23     memset(h, -1, sizeof(h));
24     while(n--){
25         scanf("%s%d", oper, &x);
26         if(oper[0] == 'I'){
27             insert(x);
28         }
29         if(oper[0] == 'Q'){
30             bool is_exisit = query(x);
31             if(is_exisit)
32                 printf("%s\n", "Yes");
33             else
34                 printf("%s\n", "No");
35         }
36     }
37 }

```

2.12.2 字符串哈希

```

1 #include<iostream>
2 using namespace std;
3 const int N = 100010;
4
5 typedef unsigned long long ULL;
6 ULL p[N], h[N];
7 int n, m, l1, r1, l2, r2, P=1331;
8 char str[N];
9
10 ULL get(int l, int r){
11     return h[r] - h[l-1] * p[r-l+1];
12 }
13
14 int main(){
15     scanf("%d%d", &n, &m);
16     scanf("%s", str+1);
17     p[0] = 1;
18     for(int i=1; i <=n; i++){
19         p[i] = p[i-1] * P;

```

```
20     h[i] = h[i-1] * P + str[i];
21 }
22 while(m--){
23     scanf("%d%d%d", &l1, &r1, &l2, &r2);
24     if(get(l1, r1) == get(l2, r2)) puts("Yes");
25     else puts("No");
26 }
27 }
```

3 搜索与图论

3.1 DFS

3.1.1 排列数字

```
1  #include<iostream>
2  using namespace std;
3
4  const int N = 8;
5
6  int n, p[N], st[N];
7
8  void dfs(int u){
9      if(u == n){
10         for(int i = 0; i < n; i++) printf("%d ", p[i]);
11         puts("");
12     }
13
14     for(int i = 1; i <= n; i++){
15         if(!st[i]){
16             st[i] = true;
17             p[u] = i;
18             dfs(u+1);
19             st[i] = false;
20         }
21     }
22 }
23
24 int main(){
25     scanf("%d", &n);
26     dfs(0);
27 }
```

3.1.2 n 皇后问题

```
1  #include <iostream>
2  using namespace std;
3
4  const int N = 20;
5
6  char q[N][N];
```



```

7  bool dg[N],udg[N],col[N];
8  int n;
9
10 void dfs(int u){
11     if( u == n){
12         for(int i = 0 ; i < n; i++) puts(q[i]);
13         puts("");
14     }
15
16     for(int i = 0 ; i < n; i++){
17         if(!col[i] && !dg[u+i] && !udg[n-u+i]){
18             q[u][i] = 'Q';
19             col[i] = dg[u+i] = udg[n-u+i] = true; // 用截距表示对角线与反对角线
20             dfs(u+1);
21             col[i] = dg[u+i] = udg[n-u+i] = false;
22             q[u][i] = '.';
23         }
24     }
25 }
26
27 int main(){
28     scanf("%d",&n);
29     for(int i = 0 ; i<n;i++)
30         for(int j = 0; j< n; j++)
31             q[i][j] = '.';
32     dfs(0);
33 }

```

3.2 BFS

3.2.1 走迷宫

```

1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  typedef pair<int,int> PII;
6  const int N = 110;
7
8  int n,m,h=0,t=-1,d[N][N],g[N][N],dx[4]={-1,0,1,0},dy[4]={0,1,0,-1};
9  PII q[N*N];
10
11 int bfs(PII u){
12     memset(d,-1,sizeof(d));
13     d[u.first][u.second] = 0;
14     q[++t] = u;
15     while(h<=t){
16         auto e = q[h++];
17         for(int i = 0 ; i < 4;i++){
18             int x = e.first + dx[i], y = e.second + dy[i];
19             if(x <= n && x >=1 && y<=m && y>=1 && d[x][y] == -1 && !g[x][y]){
20                 d[x][y] = d[e.first][e.second] + 1;

```

```

21         q[++t] = {x,y};
22     }
23 }
24 }
25 return d[n][m];
26 }
27
28
29 int main(){
30     scanf("%d%d",&n,&m);
31     for(int i = 1 ; i <= n ; i++)
32         for(int j = 1; j<= m; j++)
33             scanf("%d",&g[i][j]);
34
35     printf("%d",bfs({1,1}));
36 }

```

3.2.2 八重码

```

1  #include <iostream>
2  #include <queue>
3  #include <string>
4  #include <unordered_map>
5  using namespace std;
6
7  int dx[4] = {-1,0,1,0},dy[4] = {0,1,0,-1};
8  queue<string> q;
9  string u;
10 unordered_map<string,int> d;
11
12
13 int bfs(string u){
14     q.push(u);
15     d[u] = 0;
16     string end = "12345678x";
17     while(!q.empty()){
18         auto t = q.front();
19         q.pop();
20         if(t == end) return d[t];
21         int k = t.find('x');
22         int a = k / 3, b = k % 3;
23         int dist = d[t];
24         for(int i = 0 ; i < 4;i++){
25             int x = a + dx[i], y = b + dy[i];
26             if(x>=0 && x < 3 && y>=0 && y < 3){
27                 swap(t[x * 3 + y], t[k]);
28                 if(!d.count(t)){
29                     d[t] = dist + 1;
30                     q.push(t);
31                 }
32                 swap(t[x * 3 + y], t[k]);
33             }

```

```
34     }
35
36     }
37     return -1;
38 }
39
40
41 int main(){
42     char c;
43     for(int i = 0 ; i < 9;i++){
44         scanf("%c ",&c);
45         u+=c;
46     }
47     // cout << u << endl;
48     printf("%d",bfs(u));
49 }
```

3.3 树与图的 DFS

3.3.1 树的重心

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  const int N = 100010, M = 2 * N; // 无向图边数比有向图边数多两倍;
6
7  int h[N], el[M], ne[M], idx = 0, n,x,y;
8  int ans = N;
9  bool st[N];
10
11
12 void insert(int x ,int y){
13     el[idx] = y, ne[idx] = h[x], h[x] = idx ++; //头插法
14 }
15
16 // 计算以u为根节点的树的节点数（包括u自身）
17 int dfs(int u){
18     st[u] = true;
19     int sum = 1, res = 0; // 当前树的节点数，连通子图的节点数最大值;
20     for(int i = h[u]; i != -1; i = ne[i]){
21         if(!st[el[i]]){
22             int s = dfs(el[i]);
23             sum += s;
24             res = max(res, s);
25         }
26     }
27
28     res = max(res, n - sum); // 比较剩余父连通图的节点数
29     ans = min(ans, res);
30     return sum;
31 }
```

```

32
33
34 int main(){
35     scanf("%d", &n);
36     memset(h,-1,sizeof(h));
37     for(int i = 0 ; i < n-1 ; i++){// n-1条边, 题意
38         scanf("%d%d",&x,&y);
39         insert(x,y);
40         insert(y,x);// 无向图插两条边
41     }
42     dfs(1);
43     printf("%d", ans);
44 }

```

3.4 树与图的 BFS

3.4.1 图中点的层次

```

1  #include <iostream>
2  #include <string.h>
3  #include <queue>
4  using namespace std;
5  const int N = 100010, M = 100010;
6  int n,m,h[N],e1[N], ne[M], idx=0, x, y ,d[N];
7  queue<int> q;
8
9
10 void insert(int x, int y){
11     e1[idx] = y, ne[idx] = h[x], h[x] = idx ++;
12 }
13
14 int bfs(int u){
15     q.push(u);
16     memset(d, -1, sizeof(d));
17     d[u] = 0;
18     while(!q.empty()){
19         int t = q.front();
20         q.pop();
21         for(int i = h[t]; i!=-1 ; i = ne[i]){
22             if(d[e1[i]] == -1){
23                 d[e1[i]] = d[t] + 1;
24                 q.push(e1[i]);
25             }
26         }
27     }
28     return d[n];
29 }
30
31 int main(){
32     scanf("%d%d",&n,&m);
33     memset(h, -1,sizeof(h));
34     for(int i = 0 ; i < m ; i++){

```

```

35     scanf("%d%d",&x,&y);
36     insert(x,y);
37 }
38 printf("%d", bfs(1));
39 }

```

3.5 拓扑排序

3.5.1 有向图的拓扑排序

```

1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  const int N = 100010, M = 100010;
6  int idx=0,el[M],ne[M],h[N],x,y,n,m,degree[N];
7  int q[M], hh = 0, tt= -1;
8
9  void insert(int x, int y){
10     el[idx] = y, ne[idx] = h[x], h[x] = idx ++;
11 }
12
13 int topsort(){
14     for(int i = 1 ; i <= n ; i++){
15         if(!degree[i]) q[++tt] = i;
16     }
17     while(hh<=tt){
18         int t = q[hh++];
19         for(int i = h[t]; i != -1; i = ne[i]){
20             int j = el[i];
21             --degree[j];
22             if(!degree[j]){
23                 q[++tt] = j;
24             }
25         }
26     }
27     return tt == n-1;
28 }
29
30 int main(){
31     memset(h,-1,sizeof(h));
32     memset(degree,0,sizeof(degree));
33     scanf("%d%d",&n,&m);
34     for(int i = 0; i < m; i++){
35         scanf("%d%d",&x,&y);
36         insert(x,y);
37         ++degree[y];
38     }
39     if(topsort()){
40         for(int i = 0; i<=tt;i++){
41             printf("%d ", q[i]);
42         }

```

```
43     }
44     else{
45         printf("%d",-1);
46     }
47 }
```

3.6 Dijkstra

3.6.1 求最短路 1

```
1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  const int N = 510, M = 100010;
6  int d[N][N], dist[M], n, m, x, y, z;
7  bool st[N];
8
9
10 int dijkstra(int u){
11     memset(dist, 0x3f, sizeof dist);
12     dist[u] = 0;
13     for(int i = 0; i < n; i++){
14         int t = -1;
15         for(int j = 1; j <= n; j++){
16             if(!st[j] && ((t == -1) || dist[t] > dist[j]))
17                 t = j;
18         }
19
20         st[t] = true;
21
22         for(int j = 1; j <= n; j++){
23             dist[j] = min(dist[j], dist[t] + d[t][j]);
24         }
25     }
26     if(dist[n] == 0x3f3f3f3f) return -1;
27     else return dist[n];
28 }
29
30 int main(){
31     memset(d, 0x3f, sizeof d);
32     scanf("%d%d", &n, &m);
33     for(int i = 0; i < m; i++){
34         scanf("%d%d%d", &x, &y, &z);
35         d[x][y] = min(d[x][y], z);
36     }
37     int res = dijkstra(1);
38     printf("%d", res);
39     return 0;
40 }
```

3.6.2 求最短路 2

```
1  #include <iostream>
2  #include <vector>
3  #include <queue>
4  #include <string.h>
5  using namespace std;
6
7  const int N = 150010,M = 150010;
8
9  typedef pair<int,int> PII;
10 int dist[N],el[M],ne[M],w[M],h[N],n,m,x,y,z,idx=0;
11 bool st[N];
12 priority_queue<PII,vector<PII>,greater<PII>> q;
13
14 void add(int x, int y, int z){
15     el[idx] = y, w[idx] = z, ne[idx] = h[x], h[x] = idx++;
16 }
17
18 int dijkstra(int u){
19     memset(dist, 0x3f, sizeof dist);
20     dist[u] = 0;
21     q.push({dist[u], u});
22     while(q.size()){
23         auto t = q.top();
24         q.pop();
25         int ver = t.second, d = t.first;
26         if(st[ver]) continue;
27         st[ver] = true;
28         for(int i = h[ver]; i != -1; i = ne[i]){
29             int y = el[i];
30             if(dist[y] > d + w[i]){
31                 dist[y] = d + w[i];
32                 q.push({dist[y],y});
33             }
34         }
35     }
36     if(dist[n] == 0x3f3f3f3f) return -1;
37     return dist[n];
38 }
39
40
41 int main(){
42     scanf("%d%d",&n,&m);
43     memset(h, -1, sizeof h);
44     for(int i = 0; i < m;i++){
45         scanf("%d%d%d",&x,&y,&z);
46         add(x,y,z);
47     }
48     printf("%d",dijkstra(1));
49 }
```

3.7 Bellman-Ford

3.7.1 有边数限制的最短路

```

1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  const int N = 510, M = 10010;
6  int n,m,k,dist[N],backup[N];
7
8  struct Edge{
9      int x,y,w;
10 }edges[M];
11
12 int bellman_ford(int u){
13     memset(dist,0x3f,sizeof dist);
14     dist[u] = 0;
15     for(int i = 0 ; i< k; i++){
16         memcpy(backup,dist,sizeof backup);
17         for(int j = 0; j < m; j++){
18             int x = edges[j].x,y = edges[j].y, w = edges[j].w;
19             dist[y] = min(dist[y], backup[x] + w);
20         }
21     }
22     if(dist[n] > 0x3f3f3f3f / 2) return -1;
23     return dist[n];
24 }
25
26 int main(){
27     scanf("%d%d%d",&n,&m,&k);
28     for(int i = 0 ; i < m ; i++){
29         scanf("%d%d%d",&edges[i].x,&edges[i].y,&edges[i].w);
30     }
31     int res = bellman_ford(1);
32     if(res == -1) printf("impossible");
33     else printf("%d", res);
34     return 0;
35 }

```

3.8 SPFA

3.8.1 SPFA 求最短路

```

1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  const int N = 100010, M = 100010;
6  int idx=0,el[M], w[M], ne[M], h[N], dist[N], n, m, x, y, z;
7  int hh=0,tt=-1,q[N];
8  bool st[N];
9

```



```

10 void add(int x, int y, int z){
11     el[idx] = y, w[idx] = z, ne[idx] = h[x], h[x] = idx++;
12 }
13
14 int spfa(int u){
15     memset(dist, 0x3f3f3f, sizeof dist);
16     dist[u] = 0;
17     q[++tt] = u;
18     while(hh <= tt){
19         int t = q[hh++];
20         st[t] = false;
21         for(int i = h[t]; i != -1; i = ne[i]){
22             int j = el[i];
23             if(dist[j] > dist[t] + w[i]){
24                 dist[j] = dist[t] + w[i];
25                 if(!st[j]){
26                     q[++tt] = j;
27                     st[j] = true;
28                 }
29             }
30         }
31     }
32     if(dist[n] == 0x3f3f3f3f) return -1;
33     return dist[n];
34 }
35
36 int main(){
37     memset(h, -1, sizeof h);
38     scanf("%d%d", &n, &m);
39     for(int i = 0; i < m; i++){
40         scanf("%d%d%d", &x, &y, &z);
41         add(x, y, z);
42     }
43     int res = spfa(1);
44     if(res == -1) puts("impossible");
45     else printf("%d", res);
46 }

```

3.8.2 SPFA 判断负环

```

1  #include <iostream>
2  #include<string.h>
3  #include <queue>
4  using namespace std;
5
6  const int N = 100010, M = 100010;
7
8  int n,m,x,y,z,idx=0, h[N],el[M],ne[M],w[M],dist[N],cnt[N];
9  bool st[N];
10 queue<int> q;
11
12 void add(int x, int y, int z){

```

```

13     el[idx] = y, w[idx] = z ,ne[idx] = h[x], h[x] = idx ++;
14 }
15
16 bool spfa(){
17     for(int i = 1; i<=n; i++){
18         q.push(i);
19         st[i] = true;
20     }
21     while(q.size()){
22         int t = q.front();
23         q.pop();
24         st[t] = false;
25         for(int i = h[t]; i!=-1; i=ne[i]){
26             int j = el[i];
27             if(dist[j] > dist[t] + w[i]){
28                 dist[j] = dist[t] + w[i];
29                 cnt[j] = cnt[t] + 1;
30                 if(cnt[j] >=n) return true;
31                 if(!st[j]){
32                     q.push(j);
33                     st[j] = true;
34                 }
35             }
36         }
37     }
38     return false;
39 }
40
41 int main(){
42     memset(h, -1, sizeof h);
43     scanf("%d%d", &n, &m);
44     for(int i = 0 ; i < m; i++){
45         scanf("%d%d%d", &x, &y, &z);
46         add(x,y,z);
47     }
48     if(spfa()) puts("Yes");
49     else puts("No");
50 }

```

3.9 Floyd

3.9.1 Floyd 求最短路

```

1 #include <iostream>
2 #include <string.h>
3 using namespace std;
4
5 const int N = 210, M = 20010, INF = 0x3f3f3f3f;
6 int n, m, k, d[N][N], x, y, z;
7
8
9 void floyd(){

```

```

10     for(int k = 1; k <= n; k++)
11         for(int i = 1; i <= n; i++)
12             for(int j = 1; j <= n; j++)
13                 d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
14 }
15
16
17 int main(){
18     scanf("%d%d%d", &n, &m, &k);
19
20     for(int i = 1; i <= n; i++)
21         for(int j = 1; j <= n; j++){
22             if(i == j) d[i][j] = 0;
23             else d[i][j] = INF;
24         }
25
26     for(int i = 0; i < m; i++){
27         scanf("%d%d%d", &x, &y, &z);
28         d[x][y] = min(d[x][y], z);
29     }
30
31     floyd();
32
33     while(k -- ){
34         scanf("%d%d", &x, &y);
35         if(d[x][y] > INF / 2) puts("impossible");
36         else printf("%d\n", d[x][y]);
37     }
38     return 0;
39 }
40 }

```

3.10 最小生成树

3.10.1 Kruskal

```

1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4  const int N = 1e5 + 10, M = 2e5 + 10;
5  int n, m, x, y, z, p[N];
6
7  struct Edge{
8      int x, y, z;
9      bool operator < (const Edge &e) const{
10         return z < e.z;
11     }
12 }edges[M];
13
14 int find(int x){
15     if(p[x] != x) p[x] = find(p[x]);
16     return p[x];

```

```

17 }
18
19 int main(){
20     scanf("%d%d",&n,&m);
21     for(int i = 0 ; i < m; i++){
22         scanf("%d%d%d", &x, &y, &z);
23         edges[i] = {x,y,z};
24     }
25
26     for(int i = 1; i<= N; i++) p[i] =i;
27
28     sort(edges , edges + m);
29
30     int cnt = 0, res = 0;
31
32     for(int i = 0; i<m; i++){
33         int x = edges[i].x, y = edges[i].y, z = edges[i].z;
34
35         int px = find(x), py = find(y);
36
37         if(px != py){
38             p[px] = py;
39             cnt ++;
40             res += z;
41         }
42     }
43
44     if(cnt != n-1) puts("impossible");
45     else printf("%d",res);
46 }

```

3.10.2 Prim

```

1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  const int N = 520, M = 2e105 + 10, INF = 0x3f3f3f3f;
6  int d[N][N], dist[N], n, m, x, y, z;
7  bool st[N];
8
9  int prime(){
10     memset(dist, 0x3f, sizeof dist);
11     int res = 0;
12     for(int i = 0 ; i<n ;i++){
13         int t = -1;
14         for(int j = 1; j<=n ;j++){
15             if(!st[j] && ((t == -1 ) || dist[t] > dist[j])){
16                 t = j;
17             }
18         }
19         st[t] = true;

```

```

20     if(i && dist[t] == INF) return -1;
21     if(i) res+= dist[t];
22
23     for(int j = 1; j<=n; j++){
24         dist[j] = min(dist[j],d[t][j]);
25     }
26 }
27 return res;
28 }
29
30 int main(){
31     scanf("%d%d",&n,&m);
32     memset(d, 0x3f, sizeof d);
33     for(int i = 0 ; i < m ;i ++){
34         scanf("%d%d%d", &x, &y, &z);
35         d[x][y] = d[y][x] = min(d[x][y],z);
36     }
37     int res = prime();
38     if(res == -1) puts("impossible");
39     else printf("%d", res);
40 }

```

3.11 二分图

3.11.1 染色法判定二分图

```

1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  const int N = 1e5 + 10, M = 2e5 + 10;
6
7  int idx=0,el[M],ne[M],h[N],color[N],n,m,x,y;
8
9  void add(int x, int y){
10     el[idx] = y, ne[idx] = h[x], h[x] = idx++;
11 }
12
13
14 bool dfs(int u, int c){
15     color[u] = c;
16     for(int i = h[u]; i != -1 ; i = ne[i]){
17         int j = el[i];
18         if(!color[j]){
19             if(!dfs(j, 3-c)) return false;
20         }
21         else if(color[j] == c) return false;
22     }
23     return true;
24 }
25
26 int main(){

```

```

27     memset(h, -1, sizeof h);
28     scanf("%d%d", &n, &m);
29     while(m --){
30         scanf("%d%d", &x, &y);
31         add(x, y), add(y, x);
32     }
33
34     bool flag = true;
35     for(int i = 1 ; i <= n ; i++){
36         if(!color[i]){
37             if(!dfs(i, 1)){
38                 flag = false;
39                 break;
40             }
41         }
42     }
43     if(flag) puts("Yes");
44     else puts("No");
45 }

```

3.11.2 匈牙利算法求二分最大匹配

```

1  #include<iostream>
2  #include<string.h>
3  using namespace std;
4
5  const int N = 510, M = 1e5 + 10;
6  int n1, n2, m, x, y, match[N], el[M], ne[M], h[N], idx=0;
7  bool st[N];
8  void add(int x, int y){
9      el[idx] = y, ne[idx] = h[x], h[x] = idx ++;
10 }
11
12
13 bool find(int u){
14     for(int i = h[u]; i != -1; i = ne[i]){
15         int j = el[i];
16         if(!st[j]){
17             st[j] = true;
18             if(match[j] == 0 || find(match[j])){
19                 match[j] = u;
20                 return true;
21             }
22         }
23     }
24     return false;
25 }
26
27 int main(){
28     memset(h, -1, sizeof h);
29     scanf("%d%d%d", &n1, &n2, &m);
30     for(int i = 0 ; i < m; i++){

```

```
31     scanf("%d%d",&x,&y);
32     add(x,y);
33 }
34
35 int res = 0;
36
37 for(int i = 1 ; i<=n1; i++){
38     memset(st, false, sizeof st);
39     if(find(i)){
40         res ++;
41     }
42 }
43 printf("%d\n",res);
44 return 0;
45
46 }
```

4 数论

4.1 质数

4.1.1 试除法判定质数

```
1  #include<iostream>
2  using namespace std;
3  const int N = 110;
4  int m,x;
5  bool st[N];
6
7  bool is_prime(int x){
8      if(x < 2) return false;
9      for(int i = 2; i <= x/i ; i++){
10         if(x % i == 0) return false;
11     }
12     return true;
13 }
14
15 int main(){
16     scanf("%d", &m);
17     while(m -- ){
18         scanf("%d", &x);
19         if(is_prime(x)) puts("Yes");
20         else puts("No");
21     }
22 }
```

4.1.2 分解质因数

```
1  #include<iostream>
2  using namespace std;
3  const int N = 110;
```

```
4 int n, x;
5
6
7 void get_prime(int x){
8     for(int i = 2; i <= x/i ; i++){
9         if(x % i == 0){
10             int s = 0;
11             while(x % i == 0){
12                 x /= i;
13                 s++;
14             }
15             cout << i << ' ' << s << endl;
16         }
17     }
18     if(x > 1) cout << x << ' ' << 1 << endl;
19     cout << endl;
20 }
21
22 int main(){
23     scanf("%d",&n);
24     while(n --){
25         scanf("%d", &x);
26         get_prime(x);
27     }
28     return 0;
29 }
```

4.1.3 筛质数

```
1 #include<iostream>
2 using namespace std;
3
4 const int N = 1e6 +10;
5 int primes[N];
6 bool st[N];
7 int cnt;
8
9 void get_primes(int n){
10     for(int i = 2; i <= n ; i++){
11         if(!st[i]) primes[cnt++] = i;
12         for(int j = 0 ; primes[j] <= n/i ; j++){
13             st[primes[j] * i] = true;
14             if(i % primes[j] == 0) break;
15         }
16     }
17 }
18
19 int main(){
20     int n;
21     scanf("%d", &n);
22     get_primes(n);
23     printf("%d\n",cnt);
24 }
```



```
24 }
```

4.2 约数

4.2.1 试除法求约数

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 vector<int> get_divisor(int x) {
5     vector<int> res;
6     for (int i = 1; i <= x / i; ++ i )
7         if (x % i == 0) {
8             res.push_back(i);
9             if (i != x / i) res.push_back(x / i);
10        }
11    sort(res.begin(), res.end());
12    return res;
13 }
14
15 int main() {
16     int n;
17     cin >> n;
18
19     while (n -- ) {
20         int x;
21         cin >> x;
22         auto res = get_divisor(x);
23
24         for (auto x : res) cout << x << ' ';
25         cout << endl;
26     }
27     return 0;
28 }
```

4.2.2 约数个数

```
1 #include<iostream>
2 #include<unordered_map>
3 using namespace std;
4 const int MOD = 1e9+7;
5 typedef long long LL;
6 unordered_map<int,int> primes;
7 int n = 0;
8
9 int main(){
10     scanf("%d",&n);
11     LL res = 1;
12     while(n--){
13         int x;
14         scanf("%d",&x);
15         for(int i = 2 ; i <= x / i ; i++){
```

```
16         while(x % i == 0){
17             x /= i;
18             primes[i]++;
19         }
20     }
21     if( x > 1) primes[x] ++;
22 }
23 for(auto p : primes) res = res * (p.second + 1) % MOD;
24 printf("%lld\n", res);
25 return 0;
26
27 }
```

4.2.3 约数之和

```
1  #include <iostream>
2  #include <unordered_map>
3  using namespace std;
4
5  const int MOD = 1e9 + 7;
6  typedef long long LL;
7  int n;
8  unordered_map<int, int> primes;
9
10 int main(){
11     scanf("%d",&n);
12     while(n--){
13         int x;
14         scanf("%d", &x);
15         for(int i = 2 ; i <= x/i ; i++){
16             while(x % i == 0){
17                 x /= i ;
18                 primes[i] ++;
19             }
20         }
21         if(x > 1) primes[x] ++;
22     }
23     LL res = 1;
24     for(auto x : primes){
25         int p = x.first, m = x.second;
26         LL t = 1;
27         while(m --) t = (t * p + 1) % MOD;
28         res = res * t % MOD;
29     }
30     printf("%lld", res);
31     return 0;
32 }
```

4.2.4 最大公约数

```
1  #include<iostream>
```

```
2 #include<algorithm>
3 using namespace std;
4
5 int gcd(int a, int b){
6     if(a < b) swap(a,b);
7     return b ? gcd(b, a%b) : a;
8 }
9
10 int n;
11
12 int main(){
13     scanf("%d",&n);
14     while(n--){
15         int a, b;
16         scanf("%d%d",&a, &b);
17         printf("%d\n", gcd(a,b));
18     }
19     return 0;
20 }
```

4.3 欧拉函数

4.3.1 欧拉函数

```
1 #include<iostream>
2 using namespace std;
3 const int N = 1e4 + 10;
4 int primes[N], n, phi[N];
5
6 int main(){
7     scanf("%d", &n);
8     while(n -- ){
9         int x;
10        scanf("%d", &x);
11        int res = x;
12        for(int i = 2 ; i <= x / i ; i++){
13            if(x % i == 0){
14                res = res / i * (i-1); // 防止溢出;
15                while(x % i == 0) x /= i;
16            }
17        }
18        if( x > 1) res = res / x * (x-1);
19        printf("%d\n", res);
20    }
21 }
```

4.3.2 筛法求欧拉函数

```
1 #include<iostream>
2 using namespace std;
3
4 const int N = 1e6 + 10;
```

```

5
6 typedef long long LL;
7 int phi[N], primes[N], cnt;
8 bool st[N];
9
10 LL get_eulers(int n){
11     phi[1] = 1;
12     for(int i = 2; i <= n ; i++){
13         if(!st[i]){
14             primes[cnt++] = i;
15             phi[i] = i - 1;
16         }
17         for(int j = 0 ; primes[j] <= n / i ; j++){
18             int t = primes[j] * i;
19             st[t] = true;
20             if(i % primes[j] == 0){
21                 phi[t] = phi[i] * primes[j];
22                 break;
23             }
24             phi[t] = phi[i] * (primes[j] - 1);
25         }
26     }
27     LL res = 0;
28     for(int i = 1; i <= n ; i++){
29         res += phi[i];
30     }
31     return res;
32 }
33
34 int main(){
35     int n;
36     scanf("%d", &n);
37     printf("%lld\n", get_eulers(n));
38 }

```

4.4 快速幂

4.4.1 快速幂

```

1 #include<iostream>
2 using namespace std;
3
4 int n, a, b ,p;
5 typedef long long LL;
6
7 LL qmi(int a,int b, int p){
8     LL res = 1;
9     while(b){
10         if(b & 1){
11             res = (LL)res * a % p;
12         }
13         a = (LL) a * a % p;

```

```

14     b >>= 1;
15 }
16 return res;
17 }
18
19 int main(){
20     scanf("%d", &n);
21     while(n--){
22         scanf("%d%d%d", &a, &b, &p);
23         printf("%lld\n", qmi(a,b,p));
24     }
25 }

```

4.4.2 快速幂求逆元

```

1 #include<iostream>
2 using namespace std;
3
4 typedef long long LL;
5
6 LL pmi(int a, int b, int p){
7     LL res = 1;
8     while(b){
9         if(b&1){
10             res = (LL)res * a % p;
11         }
12         a = (LL) a * a % p;
13         b >>=1;
14     }
15     return res;
16 }
17
18 int main(){
19     int n ;
20     scanf("%d",&n);
21     while(n -- ){
22         int a, p;
23         scanf("%d%d",&a,&p);
24         LL res = pmi(a, p -2 , p);
25         if(a % p) printf("%lld\n", res);
26         else puts("impossible");
27     }
28
29 }

```

4.5 扩展欧几里得算法

4.5.1 扩展欧几里得算法

```

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

```

```

4  /*
5  (a,b) = (b, a%b) ==>
6  ax + by = d
7  (a - a//b * b)x + by = d
8  ax + b(y - a //b *x) = d
9  y' = y - a//b *x
10 */
11 int exgcd(int a, int b, int &x, int &y){
12     if(!b){
13         x = 1, y = 0;
14         return a;
15     }
16     int d = exgcd(b,a%b,y,x);
17     y -= (a/b)*x;
18     return d;
19 }
20
21 int main(){
22     int n;
23     scanf("%d",&n);
24     while(n--){
25         int a, b, x ,y;
26         scanf("%d%d",&a,&b);
27         exgcd(a,b,x,y);
28         printf("%d %d\n",x,y);
29     }
30     return 0;
31 }

```

4.5.2 线性同余方程

```

1  /*
2      a * x === b % m
3      ==> ax = my + b
4      ==> ax - my = b
5      ==> ax + my' = b
6      gcd(a, m) | b 则有解
7      x = x0 * b / d % m 相当于倍增
8  */
9  #include<bits/stdc++.h>
10 using namespace std;
11
12 using LL = long long;
13
14 int exgcd(int a, int b, int &x, int &y) {
15     if (!b) {
16         x = 1, y = 0;
17         return a;
18     }
19     int d = exgcd(b, a % b, y, x);
20     y -= a / b * x;
21     return d;

```

```

22 }
23
24 int main() {
25     int n;
26     cin >> n;
27     while (n -- ) {
28         int a, b, m;
29         cin >> a >> b >> m;
30         int x, y;
31         int d = exgcd(a, m, x, y);
32         if (b % d) cout << "impossible" << endl;
33         else cout << (LL)b / d * x % m << endl;
34     }
35 }

```

4.6 中国剩余定理

4.6.1 表达整数的奇怪方式

```

1  #include<iostream>
2  using namespace std;
3
4  typedef long long LL;
5
6  LL exgcd(LL a, LL b, LL &x, LL &y){
7      if(!b){
8          x = 1, y = 0;
9          return a;
10     }
11     LL d = exgcd(b, a%b, y, x);
12     y -= (a/b) * x;
13     return d;
14 }
15
16 int main(){
17     int n;
18     LL a1,m1;
19     scanf("%d",&n);
20     scanf("%lld%lld", &a1, &m1);
21     bool has_ans = true;
22     for(int i = 0 ; i < n - 1 ;i++){
23         LL a2,m2,k1,k2;
24         scanf("%lld%lld", &a2, &m2);
25         LL d = exgcd(a1,a2,k1,k2);
26         if((m2-m1) % d){
27             has_ans = false;
28             break;
29         }
30         int t = a2 / d;
31         k1 += (m2-m1) / d;
32         k1 = (k1 % t + t) % t;
33     }

```

```

34     m1 = a1 * k1 + m1;
35     a1 = abs(a1 / d * a2);
36 }
37 if(has_ans) printf("%lld\n", (m1 % a1 + a1) % a1);
38 else printf("%d\n", -1);
39 return 0;
40 }

```

4.7 高斯消元

4.7.1 高斯消元解线性方程组

4.7.2 高斯消元解异或线性方程组

```

1  #include<iostream>
2  using namespace std;
3  const int N = 110;
4
5  int n,a[N][N];
6
7
8  int solve(){
9      int r, c;
10     for(r=0,c=0; c < n; c++){
11         int t = r;
12         for(int i = r ; i < n ; i++){
13             if(a[i][c]){
14                 t = i;
15                 break;
16             }
17
18             if(!a[t][c]) continue;
19
20             for(int i = c; i < n + 1; i++) swap(a[t][i], a[r][i]);
21
22             for(int i = r + 1; i < n; i++)
23                 if(a[i][c]){
24                     for(int j = n ; j >= c; j --)
25                         a[i][j] ^= a[r][j];
26                 }
27             r ++;
28         }
29         if(r < n){
30             for(int i = r ; i < n; i++)
31                 if(a[i][n]) return 2;
32             return 1;
33         }
34         for(int i = n - 1 ; i >= 0 ; i--){
35             for(int j = i + 1; j < n ; j++){
36                 a[i][n] ^= a[i][j] & a[j][n];
37             }
38         }
39         return 0;
40     }

```



```
38 }
39
40 int main(){
41     cin >> n;
42     for(int i = 0 ; i < n ; i++)
43         for(int j = 0 ; j < n + 1; j++)
44             cin >> a[i][j];
45
46     int t = solve();
47     if(t == 0){
48         for(int i = 0; i < n ; i++)
49             cout << a[i][n] << endl;
50     }
51     else if(t == 1) puts("Multiple sets of solutions");
52     else puts("No solution");
53     return 0;
54 }
```

4.8 求组合数

4.8.1 上三角法

```
1  #include<iostream>
2  using namespace std;
3  const int N = 2010, mod = 1e9 + 7;
4
5  int C[N][N], n;
6
7  void solve(){
8      for(int a = 0 ; a < N; a++){
9          for(int b = 0 ; b <= a ; b++){
10             if(!b) C[a][b] = 1;
11             else{
12                 C[a][b] = (C[a-1][b] + C[a-1][b-1]) % mod;
13             }
14         }
15     }
16
17     int main(){
18         cin >> n;
19         int a, b;
20         solve();
21         while(n--){
22             cin >> a >> b;
23             cout << C[a][b] << endl;
24         }
25         return 0;
26     }
```

4.8.2 小费马定理与逆元法

```
1  #include<iostream>
2  #include<algorithm>
3  #include<string.h>
4  using namespace std;
5
6  typedef long long LL;
7
8  const int N = 1e5 + 10, mod = 1e9 + 7;
9
10 LL qmi(LL a, LL k, LL p){
11     LL res = 1;
12     while(k){
13         if(k & 1) res = (LL) res * a % p;
14         a = (LL) a * a % p;
15         k >>= 1;
16     }
17     return res;
18 }
19
20 int main(){
21     int n;
22     cin >> n;
23     LL fact[N], infact[N];
24     fact[0] = infact[0] = 1;
25     for(int i = 1 ; i < N ; i ++){
26         fact[i] = fact[i-1] * i % mod;
27         infact[i] = infact[i-1] * qmi(i, mod - 2, mod) % mod;
28     }
29     LL a, b;
30     while(n--){
31         cin >> a >> b;
32         cout << fact[a] * infact[b] % mod * infact[a-b] % mod << endl;
33     }
34     return 0;
35 }
```

4.8.3 卢卡斯定理

```
1  #include<iostream>
2  using namespace std;
3
4  typedef long long LL;
5
6  LL qmi(LL a ,LL k, LL p){
7     LL res = 1;
8     while(k){
9         if(k & 1) res = (LL) res * a % p;
10         a = (LL) a * a % p;
11         k >>= 1;
12     }
13     return res;
14 }
```

```

15
16 LL C(LL a, LL b, LL p){
17     LL res = 1;
18     for(int i = 1, j = a ; i <= b; i++, j--){
19         res = (LL)res * j % p;
20         res = (LL) res * qmi(i, p-2 ,p) % p;
21     }
22     return res;
23 }
24
25 LL lucica(LL a, LL b, LL p){
26     if(a < p && b < p) return C(a,b,p);
27     return C(a % p, b % p, p) * lucica(a / p, b / p, p) % p;
28 }
29
30 int main(){
31     int n;
32     LL a,b,p;
33     scanf("%d", &n);
34     while(n--){
35         scanf("%lld%lld%lld", &a,&b,&p);
36         printf("%lld\n", lucica(a,b,p));
37     }
38 }

```

4.8.4 高精度

```

1  #include <iostream>
2  #include<vector>
3  using namespace std;
4
5  const int N = 5010;
6  typedef long long LL;
7
8  int primes[N],sum[N], cnt=0;
9  bool st[N];
10
11 int get(int n, int p){
12     int res = 0;
13     while(n){
14         res += n / p;
15         n /= p;
16     }
17     return res;
18 }
19
20 void get_primes(int n){
21     for(int i = 2 ; i <= n; i++){
22         if(!st[i]) primes[cnt++] = i;
23         for(int j = 0 ; primes[j] <= n / i ; j++){
24             st[primes[j] * i] = true;
25             if(i % primes[j] == 0) break;

```

```

26     }
27 }
28 }
29
30 vector<int> mul(vector<int> &a , int b){
31     vector<int> c ;
32     int t = 0;
33     for(int i = 0; i < a.size(); i++){
34         t += a[i] * b;
35         c.push_back(t%10);
36         t /= 10;
37     }
38     while(t) {
39         c.push_back(t % 10);
40         t /= 10;
41     }
42     return c;
43 }
44
45 int main(){
46     int a, b;
47     scanf("%d%d",&a,&b);
48     get_primes(a);
49     for(int i = 0 ; i < cnt; i++){
50         int p = primes[i];
51         sum[i] = get(a,p) - get(a-b,p) - get(b,p);
52     }
53     vector<int> res;
54     res.push_back(1);
55     for(int i = 0 ; i < cnt; i++)
56         for(int j = 0; j < sum[i]; j++){
57             int p = primes[i];
58             res = mul(res, p);
59         }
60
61     for(int i = res.size() -1; i>=0 ; i--)
62         printf("%d",res[i]);
63     return 0;
64 }

```

4.8.5 卡特兰数

```

1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  typedef long long LL;
6  const LL mod = 1e9 + 7;
7
8
9  typedef long long LL;
10

```

```

11
12 LL qmi(LL a, LL k, LL p){
13     LL res = 1;
14     while(k){
15         if(k & 1) res = (LL) res * a % p ;
16         a = (LL) a * a % p;
17         k >>= 1;
18     }
19     return res;
20 }
21
22 int main(){
23     int n;
24     scanf("%d", &n);
25     int a = 2 *n, b = n;
26     LL res =1 ;
27     for(int i = a; i > a- b ; i--) res = (LL) res * i % mod;
28     for(int i = 1; i <= b ; i++) res = (LL) res * qmi(i, mod - 2, mod) % mod;
29     res = (LL) res * qmi(n+1, mod - 2, mod) % mod;
30     printf("%lld\n",res);
31     scanf("%d", &n);
32
33 }

```

4.9 容斥原理

4.9.1 能被整除的数

```

1  #include<iostream>
2  using namespace std;
3
4  const int N = 20;
5  typedef long long LL;
6  int p[N];
7  int main(){
8      int n, m;
9      scanf("%d%d",&n,&m);
10     int res = 0;
11     for(int i = 0 ; i < m; i++) scanf("%d", &p[i]);
12     for(int i = 1 ; i < 1 << m; i++){
13         int t = 1, cnt = 0;
14         for(int j = 0 ; j < m; j++){
15             if(i >> j & 1){
16                 if((LL)t * p[j] > n){
17                     t = -1;
18                     break;
19                 }
20                 cnt++;
21                 t *= p[j];
22             }
23         }
24         if(t != -1){

```

```
25         if(cnt % 2) res += n / t;
26         else res -= n / t;
27     }
28 }
29 printf("%d\n", res);
30 return 0;
31 }
```

4.10 博弈论

4.10.1 Nim 游戏

```
1  #include<iostream>
2  using namespace std;
3
4
5  int main(){
6      int n;
7      scanf("%d",&n);
8      int x;
9      scanf("%d", &x);
10     for(int i = 0; i < n -1; i++){
11         int y;
12         scanf("%d",&y);
13         x ^=y;
14     }
15     if(x) puts("Yes");
16     else puts("No");
17     return 0;
18 }
```

4.10.2 台阶-Nim 游戏

```
1  #include<iostream>
2  using namespace std;
3
4
5  int main(){
6      int n;
7      scanf("%d",&n);
8      int res = 0;
9      for(int i = 1; i <= n ; i ++){
10         int x;
11         scanf("%d",&x);
12         if(i%2) res ^= x;
13     }
14     if(res) puts("Yes");
15     else puts("No");
16     return 0;
17 }
```

4.10.3 集合-Nim 游戏

```
1  #include<iostream>
2  #include<algorithm>
3  #include<unordered_set>
4  #include<string.h>
5
6  using namespace std;
7
8  const int N = 110, M = 10010;
9  int s[N], f[M], n, k;
10
11
12 int sg(int x){
13     if(f[x] != -1) return f[x];
14
15     unordered_set<int> S;
16     for(int i = 0; i < k; i++){
17         if(x >= s[i]) S.insert(sg(x - s[i]));
18     }
19     for(int i = 0; i < n; i++){
20         if(!S.count(i)) {
21             return f[x] = i;
22         }
23     }
24 }
25
26 int main(){
27     memset(f, -1, sizeof f);
28     scanf("%d", &k);
29     for(int i = 0 ; i < k; i ++ ) scanf("%d", &s[i]);
30     int res = 0;
31     scanf("%d", &n);
32     for(int i = 0 ; i < n; i ++ ) {
33         int x;
34         scanf("%d", &x);
35         res ^= sg(x);
36     }
37     if(res) puts("Yes");
38     else puts("No");
39     return 0;
40 }
```

4.10.4 拆分-Nim 游戏

```
1  #include<iostream>
2  #include<unordered_set>
3  #include<algorithm>
4  #include<string.h>
5  using namespace std;
6  const int N = 110;
7  int s[N], f[N], n;
```

```
8 int sg(int x){
9     if(f[x]!=-1) return f[x];
10
11     unordered_set<int> S;
12
13     for(int i = 0 ; i < x; i++)
14         for(int j = 0 ; j <= i; j++)
15             S.insert(sg(i) ^ sg(j));
16
17     for(int i = 0; ; i++)
18         if(!S.count(i)) return f[x] = i;
19 }
20
21 int main(){
22     scanf("%d", &n);
23     int res = 0;
24     memset(f, -1, sizeof f);
25     for(int i = 0 ; i < n; i++){
26         int x;
27         scanf("%d", &x);
28         res ^= sg(x);
29     }
30     if(res) puts("Yes");
31     else puts("No");
32 }
```

5 动态规划

5.1 背包问题

5.1.1 01 背包问题

```
1 #include<iostream>
2 using namespace std;
3
4 const int N = 1010, M = 1010;
5 int n,m,f[M], v[N], w[N];
6
7 int main(){
8     cin >> n >> m;
9     for(int i = 1; i <=n; i++){
10         cin >> v[i] >> w[i];
11     }
12
13     for(int i = 1; i <=n ;i++)
14         for(int j = m; j >= v[i]; j--)
15             f[j] = max(f[j], f[j - v[i]] + w[i]);
16     cout << f[m] << endl;
17     return 0;
18 }
```


5.1.2 完全背包问题

```

1  #include<iostream>
2  using namespace std;
3
4
5  const int N = 1010, M = 1010;
6  // int n,m, f[N][M], v[N], w[N];
7  int n,m, f[M], v[N], w[N];
8
9  int main(){
10     cin >> n >> m;
11     for(int i = 1 ; i<= n ; i++) cin >> v[i] >> w[i];
12
13     for(int i = 1; i<=n ; i++)
14         for(int j = v[i]; j <=m; j++)
15             f[j] = max(f[j], f[j - v[i]] + w[i]);
16
17     // for(int i = 1; i <= n; i++)
18     // for(int j = 0; j <= m; j++)
19     // for(int k = 0; k * v[i] <= j; k++){
20     // f[i][j] = max(f[i][j], f[i-1][j - k*v[i]] + k * w[i]);
21     // }
22
23     cout << f[m] << endl;
24 }
```

5.1.3 多重背包问题

```

1  #include<iostream>
2  using namespace std;
3  const int N = 110, M = 110;
4  int n, m , f[N][M], v[N], w[N], s[N];
5
6  int main(){
7     cin >> n >> m;
8     for(int i = 1 ; i <=n ; i++) cin >> v[i] >> w[i] >> s[i];
9
10    for(int i = 1; i<=n; i++)
11        for(int j = 0; j <=m ; j++)
12            for(int k = 0; k <=s[i] && k * v[i] <= j; k++)
13                f[i][j]= max(f[i][j], f[i-1][j - k * v[i]] + k * w[i]);
14    cout << f[n][m] << endl;
15 }
```

5.1.4 多重背包问题 2

```

1  #include<iostream>
2  using namespace std;
3
4  const int N = 10010, M = 2010;
```

```
5
6 int n,m,cnt=0,f[M], v[N], w[N];
7
8 int main(){
9     cin >> n >> m;
10    for(int i = 1 ; i <=n; i++){
11        int a,b,s;
12        cin >> a >> b >> s;
13        int k = 1;
14        while(k <= s){
15            v[++cnt] = k * a;
16            w[cnt] = k * b;
17            s -= k;
18            k *=2;
19        }
20        if(s > 0){
21            v[++cnt] = s * a;
22            w[cnt] = s * b;
23        }
24    }
25
26    for(int i = 1; i <= cnt; i ++){
27        for(int j = m; j >= v[i]; j--){
28            f[j] = max(f[j], f[j - v[i]] + w[i]);
29        }
30    }
31    cout << f[m] <<endl;
32    return 0;
33 }
```

5.1.5 分组背包问题

```
1 #include<iostream>
2 using namespace std;
3 const int N = 110, M = 110;
4 int n,m,f[M],w[N],v[N],s[N];
5
6 int main(){
7     cin >> n >> m;
8     for(int i = 1; i<=n; i++){
9         cin >> s[i];
10        for(int j = 1; j <=s[i]; j++) cin >> v[j] >> w[j];
11        for(int j = m; j >=0; j--){
12            for(int k = 1; k <= s[i]; k++){
13                if(j >= v[k])
14                    f[j] = max(f[j], f[j - v[k]] + w[k]);
15            }
16        }
17        cout << f[m] << endl;
18        return 0;
19    }
```

5.2 线性 DP

5.2.1 数字三角形

```
1  #include<iostream>
2  using namespace std;
3
4  const int N = 510;
5
6  int n,a[N][N], f[N][N], INF = 1e9;
7
8
9  int main(){
10     cin >> n;
11     for(int i = 1 ; i <=n; i++)
12         for(int j = 1; j <=i; j++)
13             cin >> a[i][j];
14     for(int i = 1; i <=n; i++)
15         for(int j = 0; j <= i+1; j++)
16             f[i][j] = -INF;
17     f[1][1] = a[1][1];
18     for(int i = 1; i <=n ; i++)
19         for(int j = 1; j <= i; j++)
20             f[i][j] = max(f[i-1][j-1] + a[i][j], f[i-1][j] + a[i][j]);
21
22     int res = -INF;
23     for(int i = 1; i<=n; i++)
24         res = max(res, f[n][i]);
25
26     cout << res <<endl;
27     return 0;
28 }
```

5.2.2 最长上升子序列

```
1  #include<iostream>
2  using namespace std;
3  const int N = 1010;
4  int a[N],f[N];
5
6  int main(){
7     int n;
8     cin >> n;
9     int res = 1;
10     for(int i = 1 ; i<=n; i++) cin >> a[i];
11     for(int i = 1; i <=n ; i++){
12         f[i] =1;
13         for(int j = 1; j < i; j++){
14             if(a[i] > a[j]) f[i] = max(f[i], f[j] + 1);
15         }
16         res = max(res, f[i]);
17     }
18     cout << res << endl;
```

```
19     return 0;
20 }
```

5.2.3 最长上升子序列 2

```
1  #include<iostream>
2  using namespace std;
3
4  const int N = 100010;
5  int a[N], q[N], n;
6
7  int main(){
8      scanf("%d", &n);
9      for(int i = 0; i < n; i++) scanf("%d", &a[i]);
10     q[0] = -1e9;
11     int len = 0;
12     for(int i = 0; i < n; i++){
13         int l = 0, r = len;
14         while(l < r){
15             int mid = l + r + 1 >>1;
16             if(q[mid] < a[i]) l = mid;
17             else r = mid -1;
18         }
19         len = max(len, r+1);
20         q[r+1] = a[i];
21     }
22     cout << len << endl;
23 }
```

5.2.4 最长公共子序列

```
1  #include<iostream>
2  using namespace std;
3
4
5  const int N = 1010, M = 1010;
6
7  char a[N], b[M];
8  int n,m,f[N][M];
9
10 int main(){
11     scanf("%d%d", &n, &m);
12     scanf("%s%s", a + 1, b + 1);
13
14     for(int i = 1; i <= n ;i ++){
15         for(int j =1 ; j <=m ;j ++){
16             f[i][j] = max(f[i-1][j], f[i][j-1]);
17             if(a[i] == b[j])
18                 f[i][j] = max(f[i][j], f[i-1][j-1] + 1);
19         }
20     }
21     cout << f[n][m] <<endl;
```

21 }

5.2.5 最短编辑距离

```

1  #include<iostream>
2  using namespace std;
3
4  const int N = 1010, M = 1010;
5  int n, m, f[N][M];
6  char a[N], b[M];
7
8  int main(){
9      scanf("%d%s", &n, a + 1);
10     scanf("%d%s", &m, b + 1);
11
12     for(int i = 1; i <= n; i++) f[i][0] = i;
13     for(int i = 1; i <= m; i++) f[0][i] = i;
14
15     for(int i = 1; i <= n; i++)
16         for(int j = 1; j <= m; j++){
17             f[i][j] = min(f[i-1][j] + 1, f[i][j-1] + 1);
18             if(a[i] == b[j])
19                 f[i][j] = min(f[i][j], f[i-1][j-1]);
20             else
21                 f[i][j] = min(f[i][j], f[i-1][j-1] + 1);
22         }
23     printf("%d\n", f[n][m]);
24     return 0;
25 }
```

5.3 区间 DP

5.3.1 石子合并

```

1  #include<iostream>
2  using namespace std;
3
4  const int N = 310;
5  int n, m, f[N][N], s[N];
6
7  int main(){
8      cin >> n;
9      for(int i = 1; i <= n; i++) cin >> s[i];
10     for(int i = 1; i <= n; i++) s[i] += s[i-1];
11     for(int len = 2; len <= n; len++){
12         for(int l = 1; l + len - 1 <= n; l++){
13             int r = l + len - 1;
14             f[l][r] = 1e9;
15             for(int k = l; k < r; k++){
16                 f[l][r] = min(f[l][r], f[l][k] + f[k+1][r] + s[r] - s[l-1]);
17             }
18         }
19     }
```

```

19     cout << f[1][n] << endl;
20     return 0;
21 }

```

5.4 计数类 DP

5.4.1 整数划分

```

1  /*
2  f[i][j] = f[i-1][j] + f[i-1][j-i] + f[i-1][j-2*i] + ... f[i-1][j-s * i];
3  f[i][j-i] = f[i-1][j-i] + f[i-1][j-2*i] + ... f[i-1][j - s * i];
4  f[i][j] = f[i-1][j] + f[i][j-i];
5  */
6  #include<iostream>
7  using namespace std;
8  const int N = 1010, mod = 1e9 + 7;
9  int n, f[N];
10 int main(){
11     scanf("%d", &n);
12     f[0] = 1;
13     for(int i = 1; i <=n ; i++)
14         for(int j = i; j <=n; j++)
15             f[j] = (f[j] + f[j - i]) % mod;
16     printf("%d\n", f[n]);
17     return 0;
18 }

```

5.5 数位统计 DP

5.5.1 计数问题

```

1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  const int N = 100;
5  int dp[N][N][2][2], digit[N];
6
7  /*
8   x: 统计的数字
9   l: 第几位
10  lead: 前导0
11  sum: 当前整数包含x的个数
12  limit: 是否达到上限
13  */
14 int f(int x, int l, int sum, bool lead, bool limit){
15     // 当前数字统计完了
16     if(!l) return sum;
17     // 记忆化搜索
18     if(dp[l][sum][lead][limit] != -1) return dp[l][sum][lead][limit];
19     int res = 0;
20     int k = limit ? digit[l] : 9;
21     for(int i = 0; i <= k; i++){

```

```

22     bool ne_limit = limit && (i == k);
23     if(lead && !i) // 含前导0, 跳过继续搜
24         res += f(x, l-1, sum, true, ne_limit);
25     else // 不含前导零, 判断当前位是否为x
26         res += f(x, l-1, sum + (i == x), false, ne_limit);
27 }
28 // 返回第1~l位中的统计结果
29 return dp[l][sum][lead][limit] = res;
30 }
31
32 int solve(int n, int x){
33     int len = 0;
34     memset(dp, -1, sizeof dp);
35     while(n){
36         digit[++len] = n % 10;
37         n /= 10;
38     }
39     return f(x, len, 0, true, true);
40 }
41
42 int main(){
43     int a, b;
44     while(scanf("%d%d", &a, &b) && a && b){
45         if(a > b) swap(a, b);
46         for(int i = 0; i <= 9; i++)
47             printf("%d ", solve(b,i) - solve(a-1,i));
48         puts("");
49     }
50     return 0;
51 }

```

5.6 状态压缩 DP

5.6.1 最短 Hamilton 路径

```

1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  const int N = 21, M = 1 << N;
5  int dp[M][N], a[N][N], n;
6
7  int main(){
8      scanf("%d", &n);
9      for(int i = 0; i < n; i++)
10         for(int j = 0; j < n; j++)
11             scanf("%d", &a[i][j]);
12     memset(dp, 0x3f, sizeof dp);
13     dp[1][0] = 0;
14     for(int i = 0; i < 1 << n; i++)
15         for(int j = 0; j < n; j++)
16             if(i >> j & 1){
17                 for(int k = 0; k < n; k++)

```

```

18         if(i ^ 1 << j >> k & 1){
19             dp[i][j] = min(dp[i][j], dp[i ^ 1 << j][k] + a[k][j]);
20         }
21     }
22     printf("%d\n", dp[(1 << n) - 1][n-1]);
23     return 0;
24 }

```

5.6.2 蒙德里安的梦想

```

1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  const int N = 12, M = 1 << N;
5  int n,m;
6  long long f[N][M];
7  bool st[M];
8
9  int main(){
10     while(scanf("%d%d", &n, &m) && (n || m)){
11         memset(f, 0, sizeof f);
12         memset(st, 0, sizeof st);
13         for(int i = 0; i < 1 << n; i++){
14             int cnt = 0;
15             st[i] = true;
16             for(int j = 0; j < n; j++){
17                 if(i >> j & 1){
18                     if(cnt & 1) st[i] = false;
19                     cnt = 0;
20                 }
21                 else cnt++;
22             }
23             if(cnt & 1) st[i] = false;
24         }
25         f[0][0] = 1;
26         for(int i = 1; i <= m; i++){
27             for(int j = 0; j < 1 << n; j++){
28                 for(int k = 0; k < 1 << n; k++){
29                     if((j & k) == 0 && st[j | k])
30                         f[i][j] += f[i-1][k];
31             }
32             printf("%lld\n", f[m][0]);
33         }
34     }
35     return 0;
36 }

```

5.7 树形 DP

5.7.1 没有上司的舞会

```

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

```



```

4  const int N = 6010;
5  int dp[N][2], happy[N], el[N], ne[N], h[N], idx = 0, n;
6  bool has_father[N];
7
8  void insert(int x, int y){
9      el[idx] = y, ne[idx] = h[x], h[x] = idx ++;
10 }
11
12 int dfs(int u){
13     dp[u][1] = happy[u]; // 初始化, 只选u的快乐值
14     for(int i = h[u]; i != -1; i = ne[i]){
15         int j = el[i];
16         dfs(j);
17         dp[u][0] += max(dp[j][0], dp[j][1]);
18         dp[u][1] += dp[j][0];
19     }
20     return max(dp[u][0], dp[u][1]);
21 }
22
23 int main(){
24     memset(h, -1, sizeof h);
25     scanf("%d", &n);
26     for(int i = 1; i <= n; i++) scanf("%d", &happy[i]);
27     for(int i = 0; i < n-1; i++){
28         int l, k;
29         scanf("%d%d", &l, &k);
30         insert(k, l); // k是l的祖先, 符合头插法
31         has_father[l] = true;
32     }
33     int root = 1;
34     while(has_father[root]) root++;
35     printf("%d\n", dfs(root));
36     return 0;
37 }

```

5.8 记忆化搜索

5.8.1 滑雪

```

1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  const int N = 310, M=310;
5  int h[N][M], f[N][M], n, m;
6  int dx[4] = {-1,0,1,0}, dy[4] = {0,1,0,-1};
7
8  int dfs(int x, int y){
9      int &v = f[x][y];
10     if(v != -1) return v;
11     v = 1;
12     for(int i=0; i < 4; i++){
13         int a = x + dx[i], b = y + dy[i];

```

```
14         if(a<=n && a >= 1 && b <=m && b >=1 && h[x][y] > h[a][b])
15             v = max(v, dfs(a,b) + 1);
16     }
17     return v;
18 }
19
20 int main(){
21     memset(f, -1, sizeof f);
22     scanf("%d%d", &n, &m);
23     for(int i = 1; i <= n; i++)
24         for(int j = 1; j <=m; j++)
25             scanf("%d", &h[i][j]);
26
27     int res = 0;
28     for(int i = 1; i <=n; i++)
29         for(int j = 1; j<=m; j++)
30             res = max(res, dfs(i,j));
31     printf("%d\n", res);
32 }
```