

1 [src\1_基本算法](#)

- [基本算法](#)
 - [位运算](#)
 - [快速幂](#)
 - [快速乘](#)
 - [快速乘](#)
 - [排序算法](#)
 - [1快速排序](#)
 - [2归并排序](#)
 - [二分](#)
 - [整数二分算法模板](#)
 - [1区间\[l,r\]被划分成\[l,mid\]和\[mid+1,r\]时使用](#)
 - [2区间\[l,r\]被划分成\[l,mid-1\]和\[mid,r\]时使用](#)
 - [3浮点数二分算法模板](#)
 - [高精度](#)
 - [1加法](#)
 - [2减法](#)
 - [3乘法](#)
 - [4除法](#)
 - [前缀和差分](#)
 - [1一维前缀和](#)
 - [2二维前缀和](#)
 - [3一维差分](#)
 - [4二维差分](#)
 - [双指针算法](#)
 - [离散化与逆序对](#)
 - [1哈希](#)
 - [2二分](#)
 - [模拟退火](#)
 - [文件读写](#)

- [快读](#)

2 [src\2_数据结构](#)

- [数据结构](#)

- [链表](#)
 - [1单链表](#)
 - [2双链表](#)
- [栈](#)
 - [单调栈](#)
- [队列](#)
 - [单调队列](#)
- [KMP](#)
- [Trie树](#)
- [并查集](#)
- [堆](#)
- [hash表](#)
 - [1开放寻址法](#)
 - [2粒链法](#)
 - [3字符串哈希](#)
- [C++STL](#)
- [线段树](#)
- [树状数组](#)

3 [src\3_搜索](#)

- [搜索](#)

- [DFS](#)
 - [枚举](#)
 - [指数型](#)
 - [组合型](#)
 - [排列型](#)
- [BFS](#)
 - [Flood-Fill](#)

- [最短距离](#)
- [多源BFS](#)
- [最小步数](#)
- [双端队列BFS](#)

4 [src\4_动态规划](#)

- [背包问题](#)
 - [01背包问题](#)
 - [完全背包](#)
 - [多重背包](#)
 - [分组背包](#)
- [数位DP](#)
- [线性DP](#)
 - [数字三角形](#)
 - [最长上升子序列](#)
 - [最长公共子序列](#)
 - [区间DP](#)

5 [src\5_图论](#)

- [图论](#)
 - [存图](#)
 - [拓扑排序](#)
 - [最短路](#)
 - [单源最短路](#)
 - [所有边权都是正数](#)
 - [朴素Dijkstra算法](#)
 - [堆优化的Dijkstra算法](#)
 - [存在负权边](#)
 - [Bellman-Ford](#)
 - [SPFA](#)
 - [多源汇最短路](#)
 - [Floyd算法](#)

- [最小生成树](#)
 - [朴素版Prim](#)
 - [堆优化Prim](#)
 - [Kruskal](#)
- [二分图](#)
 - [染色法](#)
 - [匈牙利算法](#)
- [SPFA差分约束与判负环](#)
- [LCA](#)
 - [倍增](#)
 - [树链剖分](#)
- [有向图强连通分量SCC](#)
- [无向图的双连通分量](#)
 - [边双连通分量E-DCC](#)
 - [点双连通分量V-DCC](#)
- [欧拉回路与欧拉路径](#)
- [网络流初步](#)
 - [EK求最大流](#)
 - [dinic求最大流](#)
 - [点分裂](#)

6 [src\6_数学知识](#)

- [数学知识](#)
 - [数论](#)
 - [试除法判定质数](#)
 - [分解质因数](#)
 - [筛质数](#)
 - [试除法求约数](#)
 - [约数个数](#)
 - [约数之和](#)
 - [最大公约数](#)

- [快速幂](#)
 - [快速幂求逆元](#)
 - [线性逆元](#)
- [拓展欧几里得](#)
 - [线性同余方程](#)
- [中国剩余定理](#)
- [组合计数](#)
- [高斯消元](#)
- [简单博弈论](#)
- [容斥原理](#)
- [拓展欧拉定理](#)

基本算法

1 位运算

1.1 快速幂

```
1 int fpow(int a, int b, int c) {
2     int res = 1 % c;
3     for (; b; b >>= 1) {
4         if (b & 1) res = (long long)res * a % c;
5         a = (long long)a * a % c;
6     }
7     return res;
8 }
```

1.2 龟速乘

```
1 LL mul(LL a, LL b, LL c) {
2     LL res = 0;
3     for (; b; b >>= 1) {
4         if (b & 1) res = (res + a) % c;
5         a = (a + a) % c;
6     }
7     return res;
8 }
```

1.3 快速乘

```
1 ull mul(ull a, ull b, ull p) {
2     a %= p, b %= p;
3     ull c = (long double)a * b / p;
4     ull x = a * b, y = c * p;
5     ll res = (ll)(x % p) - (ll)(y % p);
6     if (res < 0) res += p;
7     return res;
8 }
```

2 排序算法

2.1 1快速排序

```
1 void q_sort(int l, int r) {
2     if (l >= r) return;
3     int i = l - 1, j = r + 1, x = a[l + r >> 1];
4     while (i < j) {
5         do i++; while (x > a[i]);
6         do j--; while (x < a[j]);
7         if (i < j) swap(a[i], a[j]);
8     }
9     q_sort(l, j), q_sort(j + 1, r); //此处边界不能换为(l,i),(i+1,r) 否则会死循环
10 }
```

2.2 2归并排序

```
1 void merge_sort(int l, int r) {
2     if (l >= r) return;
3     int mid = l + r >> 1;
4     merge_sort(l, mid), merge_sort(mid + 1, r);
5     int i = l, j = mid + 1, k = l;
6     while (i <= mid && j <= r)
7         if (a[i] < a[j]) b[k++] = a[i++];
8         else b[k++] = a[j++];
9     while (i <= mid) b[k++] = a[i++];
10    while (j <= r) b[k++] = a[j++];
11    for (int i = l; i <= r; i++) a[i] = b[i];
12 }
```

3 二分

3.1 整数二分算法模板

```
1 | bool check(int x) { /* ... */ } // 检查x是否满足某种性质
```

3.2 1区间[l,r]被划分成[l,mid]和[mid+1,r]时使用

```
1 | int bsearch_1(int l, int r) {  
2 |     while (l < r) {  
3 |         int mid = l + r >> 1;  
4 |         if (check(mid)) r = mid;    // check()判断mid是否满足性质  
5 |         else l = mid + 1;  
6 |     }  
7 |     return l;  
8 | }
```

3.3 2区间[l,r]被划分成[l,mid-1]和[mid,r]时使用

```
1 | int bsearch_2(int l, int r) {  
2 |     while (l < r) {  
3 |         int mid = l + r + 1 >> 1;  
4 |         if (check(mid)) l = mid;  
5 |         else r = mid - 1;  
6 |     }  
7 |     return l;  
8 | }
```

3.4 3浮点数二分算法模板

```
1 | bool check(double x) { /* ... */ } // 检查x是否满足某种性质  
2 |  
3 | double bsearch_3(double l, double r) {  
4 |     const double eps = 1e-6;    // eps 表示精度，取决于题目对精度的要求  
5 |     while (r - l > eps) {  
6 |         double mid = (l + r) / 2;  
7 |         if (check(mid)) r = mid;  
8 |         else l = mid;  
9 |     }  
10 |    return l;  
11 | }
```

4 高精度

4.1 1加法

```
1 vector<int> add(vector<int> &a,vector<int> &b){
2     vector<int> c;
3     int t=0;//进位
4     for(int i=0;i<a.size() || i<b.size();i++){
5         if(i<a.size()) t+=a[i];
6         if(i<b.size()) t+=b[i];
7         c.push_back(t%10);
8         t/=10;//进位权重下降
9     }
10    if(t) c.push_back(1);
11    return c;
12 }
```

4.2 2减法

```
1 bool cmp(vector<int> &a,vector<int> &b){
2     if(a.size()!=b.size()) return a.size()>b.size();
3     for(int i=a.size();i>0;i--){
4         if(a[i]!=b[i])
5             return a[i]>b[i];
6     }
7     return true;
8 }
9 vector<int> sub(vector<int> &a,vector<int> &b){
10    vector<int> c;
11    int t=0;//借位
12    for(int i=0;i<a.size();i++){
13        t=a[i]-t;
14        if(i<b.size()) t-=b[i];
15        c.push_back((t+10)%10);
16        if(t<0) t=1;//t<0 表示借位了
17        else t=0;//否则就是没借位
18    }
19    while(c.size()>1 && c.back()==0) c.pop_back();
20    return c;
21 }
```

4.3 3乘法

1

```
1 vector<int> mul(vector<int> &a,int &b){
2     vector<int> c;
3     for(int i=0,t=0;i<a.size() || t;i++){//进位存在或没乘完
4         if(i<a.size()) t+=a[i]*b;
5         c.push_back(t%10);
6         t/=10;
7     }
8     while(c.size()>1 && c.back()==0) c.pop_back();//去除前导零
9     return c;
10 }
```

2

```
1 // 高精乘高精
```



```

2 vector<int> mul(vector<int> &a, vector<int> &b) {
3     vector<int> c;
4     c.resize(a.size() + b.size());
5     for (int i = 0; i < a.size(); ++ i) {
6         int t = 0;
7         for (int j = 0; j < b.size(); ++ j) {
8             c[i + j] += t + a[i] * b[j];
9             t = c[i + j] / 10;
10            c[i + j] %= 10;
11        }
12        c[i + b.size()] = t;
13    }
14    while (c.size() > 1 && c.back() == 0) c.pop_back();
15    return c;
16 }

```

4.4 4除法

```

1 // 高精除低精
2 vector<int> div(vector<int> &a,int &b,int &r){
3     vector<int> c;
4     r=0;
5     for(int i=a.size()-1;i>=0;i--){
6         r=r*10+a[i];
7         c.push_back(r/b);
8         r%=b;
9     }
10    reverse(c.begin(),c.end());
11    while(c.size()>1 && c.back()==0) c.pop_back();//去除前导零
12    return c;
13 }

```

5 前缀和差分

$$S[i] = \sum_{j=1}^i A[j]$$

5.1 1一维前缀和

```

1 #include<iostream>
2 #include<cstdio>
3 using namespace std;
4 const int M=1e5+10;
5 int a[M],b[M];
6 int main(){
7     int n,m;
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++) b[i]=a[i]+b[i-1];
11    while(m--){
12        int l,r;

```

```

13         scanf("%d%d",&l,&r);
14         printf("%d\n",b[r]-b[l-1]);
15     }
16 }

```

5.2 2维前缀和

```

1  #include<iostream>
2  #include<cstdio>
3  using namespace std;
4
5  const int M=1e3+10;
6  int a[M][M],b[M][M];
7  int n,m,q,x1,y1,x2,y2;
8
9  int main(){
10     scanf("%d%d%d",&n,&m,&q);
11     for(int i=1;i<=n;i++)
12         for(int j=1;j<=m;j++)
13             scanf("%d",&a[i][j]);
14     for(int i=1;i<=n;i++)
15         for(int j=1;j<=m;j++)
16             b[i][j]=b[i-1][j]+b[i][j-1]-b[i-1][j-1]+a[i][j];
17     while(q--){
18         scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
19         printf("%d\n",b[x2][y2]-b[x1-1][y2]-b[x2][y1-1]+b[x1-1][y1-1]);
20     }
21 }

```

5.3 3一维差分

```

1  #include<iostream>
2  #include<cstdio>
3  using namespace std;
4
5  const int M=1e5+10;
6  int a[M],b[M];
7  int n,m;
8  int l,r,c;
9
10 void in(int l,int r,int c){//插入
11     b[l]+=c;
12     b[r+1]-=c;
13 }
14
15 int main(){
16     scanf("%d%d",&n,&m);
17     for(int i=1;i<=n;i++) scanf("%d",&a[i]);
18     for(int i=1;i<=n;i++) b[i]=a[i]-a[i-1]; //构造差分数组
19     for(int i=1;i<=m;i++){
20         scanf("%d%d%d",&l,&r,&c);
21         in(l,r,c);
22     }
23     for(int i=1;i<=n;i++) b[i]=b[i]+b[i-1]; //还原数组
24     for(int i=1;i<=n;i++) printf("%d ",b[i]);
25 }

```

5.4 4二维差分

```
1  #include<iostream>
2  #include<cstdio>
3
4  using namespace std;
5
6  const int M=1000+10;
7  int n,m,q;
8  int a[M][M],s[M][M];
9  int x1,y1,x2,y2,c;
10
11 void in(int x1,int y1,int x2,int y2,int c){//插入
12     s[x1][y1]+=c;
13     s[x2+1][y1]-=c;
14     s[x1][y2+1]-=c;
15     s[x2+1][y2+1]+=c;
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     for(int i=1;i<=n;i++)
24         for(int j=1;j<=m;j++)
25             in(i,j,i,j,a[i][j]); //构造
26     for(int i=1;i<=q;i++){
27         scanf("%d%d%d%d",&x1,&y1,&x2,&y2,&c);
28         in(x1,y1,x2,y2,c);
29     }
30     for(int i=1;i<=n;i++)
31         for(int j=1;j<=m;j++)
32             s[i][j]+=s[i-1][j]+s[i][j-1]-s[i-1][j-1]; //还原
33     for(int i=1;i<=n;i++)
34         for(int j=1;j<=m;j++){
35             printf("%d ",s[i][j]);
36             if(j==m) puts("");
37         }
38 }
```

6 双指针算法

```
1  for (int i = 0, j = 0; i < n; i ++ ) {
2      while (j < i && check(i, j)) j ++ ;
3      // 具体问题的逻辑
4  }
```

常见问题分类：

- (1) 对于一个序列，用两个指针维护一段区间

- (2) 对于两个序列，维护某种次序，比如归并排序中合并两个有序序列的操作

7 离散化与逆序对

7.1 1 哈希

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #include <unordered_map>
6
7  using namespace std;
8
9  typedef long long LL;
10
11 const int N = 5e5 + 10;
12 int c[N], n;
13 int a[N], b[N];
14 unordered_map<int, int> m;
15
16 int lowbit(int x) {
17     return x & -x;
18 }
19
20 void add(int x, int v) {
21     for (int i = x; i < N; i += lowbit(i))
22         c[i] += v;
23 }
24
25 int query(int x) {
26     int res = 0;
27     for (int i = x; i; i -= lowbit(i))
28         res += c[i];
29     return res;
30 }
31
32 int main() {
33     scanf("%d", &n);
34     for (int i = 1; i <= n; ++ i)
35         scanf("%d", &a[i]), b[i] = a[i];
36     sort(b + 1, b + 1 + n);
37     int cnt = 0;
38     for (int i = 1, j = 1; i <= n; ++ i) {
39         while (j <= n && b[j] == b[i]) ++ j;
40         m[b[i]] = ++ cnt;
41         i = j - 1;
42     }
43     for (int i = 1; i <= n; ++ i)
44         a[i] = m[a[i]];
45     LL ans = 0;
46     for (int i = 1; i <= n; ++ i) {
```

```

47         add(a[i], 1);
48         ans += i - query(a[i]);
49     }
50     cout << ans << endl;
51 }

```

7.2 2二分

```

1  sort(b + 1, b + 1 + n);
2  int k = unique(b + 1, b + 1 + n) - b - 1;
3  for (int i = 1; i <= n; ++ i)
4      a[i] = lower_bound(b + 1, b + 1 + k, a[i]) - b;

```

8 模拟退火

使用范围: 最优化问题, 比如DP, 贪心, 计算几何

如果函数连续性较强 (即轻微的扰动对函数值的影响较小), 退火的出解率较高

序列的邻项考虑随机交换, 如 [2424. 保龄球 - AcWing题库](https://www.acwing.com/problem/content/3170/)

```

1  #include <bits/stdc++.h> // https://www.acwing.com/problem/content/3170/
2
3  using namespace std;
4
5  #define x first
6  #define y second
7  #define toMin(a, b) a > b ? a = b : 0
8
9  typedef pair<double, double> PDD;
10
11 const int N = 110;
12 double ans = 1e8;
13 PDD a[N];
14 int n;
15
16 double rand(int l, int r) {
17     return (double)rand() / RAND_MAX * (r - l) + l;
18 }
19
20 double dist(PDD a, PDD b) {
21     auto x = a.x - b.x;
22     auto y = a.y - b.y;
23     return sqrt(x * x + y * y);
24 }
25
26 double calc(PDD p) {
27     double res = 0;
28     for (int i = 1; i <= n; ++ i) res += dist(p, a[i]);
29     toMin(ans, res);
30     return res;
31 }
32

```

```

33 void SA() {
34     PDD cur(rand(0, 10000), rand(0, 10000));
35     for (double T = 1e4; T > 1e-4; T *= 0.99) { // 不同的参数出解率不同
36         PDD np(rand(cur.x - T, cur.x + T), rand(cur.y - T, cur.y + T));
37         double dt = calc(np) - calc(cur);
38         if (exp(-dt / T) > rand(0, 1)) cur = np;
39     }
40 }
41
42 int main() {
43     cin.tie(0)->sync_with_stdio(0);
44     srand(time(0));
45
46     cin >> n;
47     for (int i = 1; i <= n; ++ i) cin >> a[i].x >> a[i].y;
48     for (int i = 1; i <= 100; ++ i) SA(); // 考场里可以考虑卡时
49     cout << int(ans + 0.5) << '\n';
50 }

```

9 文件读写

```

1 // #include <cstdlib>
2 // freopen("P2058_2.in", "r", stdin);
3 // freopen("my_ans.out", "w", stdout);
4 // fclose(stdin);
5 // fclose(stdout);

```

10 快读

1cin加速:

根据个人评测经验，这种方法会比 `scanf()` 还要快

```
1 cin.tie(0)->sync_with_stdio(0)
```

2使用 `getchar()`

```

1 template<typename T> void read(T &x) {
2     char ch = getchar(); bool flag = 0; x = 0;
3     for (; ch < '0' || ch > '9'; ch = getchar())
4         flag |= (ch == '-');
5     for (; ch >= '0' && ch <= '9'; ch = getchar())
6         x = (x << 1) + (x << 3) + ch - '0';
7     if (flag) x = -x;
8 }

```

3将字符先读到 `buff` 数组中再从 `buff` 中读入

```

1 inline char GET_CHAR() {
2     static char buf[maxn], *p1 = buf, *p2 = buf;
3     return p1 == p2 && (p2 = (p1 = buf) +
4         fread(buf, 1, maxn, stdin), p1 == p2) ? EOF: *p1 ++;
5 }
6
7 template<class T> inline void read(T &x) {
8     x = 0; int f = 0; char ch = GET_CHAR();
9     for (; ch < '0' || ch > '9'; ch = GET_CHAR()) flag |= (ch == '-');
10    for (; ch >= '0' && ch <= '9'; ch = GET_CHAR()) x = (x << 1) + (x << 3) + (ch ^
11    48);
12    x = f ? -x: x;
13 }

```

数据结构

1 链表

1.1 1单链表

```

1 #include <iostream>
2 using namespace std;
3 const int N = 100010;
4 int head, e[N], ne[N], idx;
5 void init() {
6     head = -1;
7     idx = 0;
8 }
9 void add_to_head(int x) {
10    e[idx] = x, ne[idx] = head, head = idx ++ ;
11 }
12 void add(int k, int x) {
13    e[idx] = x, ne[idx] = ne[k], ne[k] = idx ++ ;
14 }
15 void remove(int k) {
16    ne[k] = ne[ne[k]];
17 }
18 int main() {
19    int m; cin >> m;
20    init();
21    while (m -- ) {
22        int k, x; char op;
23        cin >> op;
24        if (op == 'H') {
25            cin >> x;
26            add_to_head(x);
27        } else if (op == 'D') {
28            cin >> k;
29            if (!k) head = ne[head];
30            else remove(k - 1);
31        } else {
32            cin >> k >> x;
33            add(k - 1, x);
34        }
35    }
36 }

```

```

36     for (int i = head; i != -1; i = ne[i]) cout << e[i] << ' ';
37     cout << endl;
38     return 0;
39 }

```

1.2 2双链表

```

1  #include <iostream>
2  using namespace std;
3  const int N = 100010;
4  int m;
5  int e[N], l[N], r[N], idx;
6  // 在节点a的右边插入一个数x
7  void insert(int a, int x) {
8      e[idx] = x;
9      l[idx] = a, r[idx] = r[a];
10     l[r[a]] = idx, r[a] = idx ++ ;
11 }
12 // 删除节点a
13 void remove(int a) {
14     l[r[a]] = l[a];
15     r[l[a]] = r[a];
16 }
17 int main() {
18     cin >> m;
19     // 0是左端点, 1是右端点
20     r[0] = 1, l[1] = 0;
21     idx = 2;
22     while (m -- ) {
23         string op; cin >> op;
24         int k, x;
25         if (op == "L"){
26             cin >> x;
27             insert(0, x);
28         } else if (op == "R"){
29             cin >> x;
30             insert(l[1], x);
31         } else if (op == "D") {
32             cin >> k;
33             remove(k + 1);
34         } else if (op == "IL") {
35             cin >> k >> x;
36             insert(l[k + 1], x);
37         } else {
38             cin >> k >> x;
39             insert(k + 1, x);
40         }
41     }
42     for (int i = r[0]; i != 1; i = r[i]) cout << e[i] << ' ';
43     cout << endl;
44     return 0;
45 }

```


2 栈

```
1 #include <iostream>
2 #include <cstring>
3 using namespace std;
4 const int N = 1e6 + 10;
5 int stk[N], tt = -1;
6 string op;
7 int m, k;
8 int main() {
9     cin >> m;
10    while (m --) {
11        cin >> op;
12        if (op == "push") {
13            cin >> k;
14            stk[++ tt] = k;
15        } else if (op == "query") {
16            cout << stk[tt] << endl;
17        } else if (op == "pop") tt --;
18        else cout << (tt < 0 ? "YES" : "NO") << endl;
19    }
20    return 0;
21 }
```

2.1 单调栈

```
1 #include <iostream>
2 #include <cstdio>
3
4 using namespace std;
5
6 const int N = 3e6 + 10;
7 int stk[N], tt, n;
8 int a[N], ans[N];
9
10 void push(int x) { stk[++tt] = x; }
11
12 int pop() { return stk[tt--]; }
13
14 int top() { return stk[tt]; }
15
16 int empty()
17 {
18     if (tt > 0)
19         return false;
20     return true;
21 }
22
23 void out()
24 {
25     for (int i = 1; i <= tt; i++)
26         cout << stk[i] << ' ';
27     cout << endl;
28 }
29
30 int main()
```

```

31 {
32     scanf("%d", &n);
33     for (int i = 1; i <= n; i++)
34         scanf("%d", &a[i]);
35     for (int i = n; i > 0; i--)
36     {
37         while (!empty() && a[top()] <= a[i]) pop();
38         ans[i] = empty() ? 0 : top();
39         push(i);
40     }
41     for (int i = 1; i <= n; i++) printf("%d ", ans[i]);
42     //system("pause");
43     return 0;
44 }

```

3 队列

```

1  #include <iostream>
2  using namespace std;
3  const int N = 100010;
4  int m;
5  int q[N], hh, tt = -1;
6  int main() {
7      cin >> m;
8      while (m -- ) {
9          string op;
10         int x;
11         cin >> op;
12         if (op == "push") {
13             cin >> x;
14             q[ ++ tt] = x;
15         }
16         else if (op == "pop") hh ++ ;
17         else if (op == "empty") cout << (hh <= tt ? "NO" : "YES") << endl;
18         else cout << q[hh] << endl;
19     }
20     return 0;
21 }

```

3.1 单调队列

```

1  #include <iostream>
2  #include <cstdio>
3
4  using namespace std;
5
6  const int N = 1e6 + 10;
7  int q[N], hh, tt = -1;
8  int n, k;
9  int a[N];
10
11 int main()
12 {
13     scanf("%d%d", &n, &k);
14     for (int i = 0; i < n; i++)
15     {

```

```

16     scanf("%d", &a[i]);
17     if (i - k + 1 > q[hh])
18         ++hh;
19     while (hh <= tt && a[i] <= a[q[tt]])
20         --tt;
21     q[++tt] = i;
22     if (i + 1 >= k)
23         printf("%d ", a[q[hh]]);
24 }
25 puts("");
26 hh = 0, tt = -1;
27 for (int i = 0; i < n; i++)
28 {
29     if (i - k + 1 > q[hh])
30         ++hh;
31     while (hh <= tt && a[i] >= a[q[tt]])
32         --tt;
33     q[++tt] = i;
34     if (i + 1 >= k)
35         printf("%d ", a[q[hh]]);
36 }
37 //system("pause");
38 return 0;
39 }

```

4 KMP

```

1  // KMP
2  #include <iostream>
3  #include <cstdio>
4  #include <cstring>
5
6  using namespace std;
7
8  const int N = 1e6 + 10;
9  int n, m;
10 char p[N] = "0", s[N] = "0";
11 int ne[N];
12
13 int main()
14 {
15     cin >> s + 1 >> p + 1;
16     n = strlen(p) - 1;
17     m = strlen(s) - 1;
18
19     for (int i = 2, j = 0; i <= n; i++)
20     {
21         while (j && p[i] != p[j + 1])
22             j = ne[j];
23         if (p[i] == p[j + 1])
24             j++;
25         ne[i] = j;
26     }
27
28     for (int i = 1, j = 0; i <= m; i++)

```

```

29     {
30         while (j && s[i] != p[j + 1])
31             j = ne[j];
32         if (s[i] == p[j + 1])
33             j++;
34         if (j == n)
35         {
36             printf("%d\n", i - n + 1);
37             j = ne[j];
38         }
39     }
40
41     for (int i = 1; i <= n; i++)
42         printf("%d ", ne[i]);
43     system("pause");
44     return 0;
45 }

```

5 Trie树

```

1  const int M = 1e6 + 10;
2  int son[M][26], cnt[M], idx;
3  // 插入
4  void insert(char str[])
5  {
6      int p = 0;
7      for (int i = 0; str[i]; i++)
8      {
9          int u = str[i] - 'a';
10         if (!son[p][u]) son[p][u] = ++ idx;
11         p = son[p][u];
12     }
13     cnt[p] ++;
14 }
15 // 查询
16 int query(char str[])
17 {
18     int p = 0;
19     for (int i = 0; str[i]; i++)
20     {
21         int u = str[i] - 'a';
22         if (!son[p][u]) return 0;
23         p = son[p][u];
24     }
25     return cnt[p];
26 }

```

6 并查集

```
1 void init(int n) {
2     for (int i = 1; i <= n; i++) fa[i] = i;
3 }
4
5 int find(int x) {
6     return x == fa[x] ? x : fa[x] = find(fa[x]);
7 }
8
9 void merge(int x, int y) {
10     fa[find(x)] = find(y);
11 }
```

7 堆

```
1 #include <iostream>
2 #include <cstdio>
3
4 using namespace std;
5
6 const int N = 1e6 + 10;
7 int h[N], cnt, n;
8
9 void up(int k) {
10     while (k >> 1 && h[k >> 1] > h[k]) {
11         swap(h[k], h[k >> 1]);
12         k >>= 1;
13     }
14 }
15
16 void insert(int x) {
17     h[++ cnt] = x;
18     up(cnt);
19 }
20
21 void down(int x) {
22     int t = x;
23     if (x * 2 <= cnt && h[t] > h[x * 2]) t = 2 * x;
24     if (2 * x + 1 <= cnt && h[t] > h[2 * x + 1]) t = 2 * x + 1;
25     if (t != x) swap(h[t], h[x]), down(t);
26 }
27
28 void remove() {
29     swap(h[1], h[cnt--]);
30     down(1);
31 }
32
33 int main() {
34     scanf("%d", &n);
35     int op, x;
36     while (n --) {
37         scanf("%d", &op);
38         if (op == 1) {
39             scanf("%d", &x);
```

```

40         insert(x);
41     }
42     else if (op == 2) printf("%d\n", h[1]);
43     else if (op == 3) remove();
44 }
45 return 0;
46 }

```

8 hash表

8.1 1开放寻址法

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  using namespace std;
5
6  const int N = 2e5 + 10, INF = 0x3f3f3f3f;
7  int h[N];
8  int find(int x) {
9      int t = (x % N + N) % N;
10     while (h[t] != INF && h[t] != x) {
11         t++;
12         if (t == N) t = 0;
13     }
14     return t;
15 }
16
17 int main() {
18     int n;
19     scanf("%d", &n);
20     memset(h, 0x3f, sizeof h);
21     while (n --) {
22         char op[2]; int x;
23         scanf("%s%d", op, &x);
24         int k = find(x);
25         if (op[0] == 'I') h[k] = x;
26         else {
27             if (h[k] == INF) puts("No");
28             else puts("Yes");
29         }
30     }
31 }

```

8.2 2拉链法

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4
5  using namespace std;
6
7  const int N = 1e5 + 3;
8

```

```

9  int h[N], e[N], ne[N], idx;
10
11 void insert(int x) {
12     int k = (x % N + N) % N;
13     e[idx] = x, ne[idx] = h[k], h[k] = idx ++;
14 }
15
16 bool find(int x) {
17     int k = (x % N + N) % N;
18     for (int i = h[k]; i != -1; i = ne[i]) if (e[i] == x) return true;
19     return false;
20 }
21
22 int main() {
23     int n;
24     scanf("%d", &n);
25     memset(h, -1, sizeof h);
26     while (n --) {
27         char op[2]; int x;
28         scanf("%s%d", op, &x);
29         if (op[0] == 'I') insert(x);
30         else {
31             if (find(x)) puts("Yes");
32             else puts("No");
33         }
34     }
35 }

```

8.3 3字符串哈希

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N = 1e5 + 10, P = 131;
4  #define u unsigned long long
5  u n, m, h[N], p[N];
6  char str[N];
7  u f(int l, int r) {
8     return h[r] - h[l - 1] * p[r - l + 1];
9 }
10 int main() {
11     cin >> n >> m >> str + 1; p[0] = 1;
12     for(int i = 1; i <= n; i++) {
13         h[i] = h[i - 1] * P + str[i];
14         p[i] = p[i - 1] * P;
15     }
16     while(m --) {
17         u a, b, c, d;
18         cin >> a >> b >> c >> d;
19         if(f(a, b) == f(c, d)) puts("Yes");
20         else puts("No");
21     }
22     return 0;
23 }

```

随机模数，防止 hack。

```

1  using LL = long long;
2  using i28 = __int128; // gcc win 64 only
3
4  std::mt19937 rng(std::chrono::steady_clock::now().time_since_epoch().count());
5
6  LL fpow(LL a, LL b, LL p) {
7      LL res = 1;
8      while (b) {
9          if (b & 1) res = res * a % p;
10         a = a * a % p;
11         b >>= 1;
12     }
13     return res;
14 }
15
16 bool millerRabin(LL x) {
17     if (x < 2) return false;
18     if (x == 2) return true;
19     if (x % 2 == 0) return false;
20     LL p = x - 1, q = 0;
21     while (!(p & 1)) p >>= 1, q++;
22     for (int i = 0; i < 10; i++) {
23         LL a = rng() % (x - 2) + 2;
24         LL v = fpow(a, p, x);
25         if (v == 1 || v == x - 1) continue;
26         for (int j = 0; j < q; j++) {
27             v = v * v % x;
28             if (v == x - 1) break;
29             if (j == q - 1) return false;
30         }
31     }
32     return true;
33 }
34
35 template<typename T1 = int, typename T2 = LL> // T1用来存储数据 T2用来进行预算(防止溢出)
36 struct strHash {
37     int base, n;
38     std::vector<T1> hash, pow;
39     T1 mod;
40
41     strHash(int N, int base = 131) : base(base), pow(N) {
42         mod = std::uniform_int_distribution<T1>(1e9, 2e9)(rng);
43         while (!millerRabin(mod)) mod++;
44         pow[0] = 1;
45         for (int i = 1; i < N; i++) pow[i] = (T2)pow[i - 1] * base % mod;
46     }
47
48     void init(std::string &s, int n) { // s下标从1开始
49         this->n = n;
50         hash.assign(n + 1, 0);
51         for (int i = 1; i <= n; i++) hash[i] = ((T2)hash[i - 1] * base + s[i]) % mod;
52     }
53
54     T1 get(int l, int r) { // [l, r]
55         return (hash[r] - (T2)hash[l - 1] * pow[r - l + 1] % mod + mod) % mod;
56     }

```


57 };

9 C++STL

```
1 unique, 将数组中重复的元素放到了最后
2 sort, 将数组中的元素排序
3 vector, 变长数组, 倍增的思想
4     size() 返回元素个数
5     empty() 返回是否为空
6     clear() 清空
7     front()/back()
8     push_back()/pop_back()
9     begin()/end()
10    []
11    支持比较运算, 按字典序
12 pair<int, int>
13     first, 第一个元素
14     second, 第二个元素
15     支持比较运算, 以first为第一关键字, 以second为第二关键字 (字典序)
16 string, 字符串
17     size()/length() 返回字符串长度
18     empty()
19     clear()
20     substr(起始下标, (子串长度)) 返回子串
21     c_str() 返回字符串所在字符数组的起始地址
22 queue, 队列
23     size()
24     empty()
25     push() 向队尾插入一个元素
26     front() 返回队头元素
27     back() 返回队尾元素
28     pop() 弹出队头元素
29 priority_queue, 优先队列, 默认是大根堆
30     push() 插入一个元素
31     top() 返回堆顶元素
32     pop() 弹出堆顶元素
33     定义成小根堆的方式: priority_queue<int, vector<int>, greater<int>> q;
34 stack, 栈
35     size()
36     empty()
37     push() 向栈顶插入一个元素
38     top() 返回栈顶元素
39     pop() 弹出栈顶元素
40 deque, 双端队列
41     size()
42     empty()
43     clear()
44     front()/back()
45     push_back()/pop_back()
46     push_front()/pop_front()
47     begin()/end()
48    []
49 set, map, multiset, multimap, 基于平衡二叉树 (红黑树), 动态维护有序序列
50     size()
51     empty()
```

```

52 clear()
53 begin()/end()
54 ++, -- 返回前驱和后继, 时间复杂度  $O(\log n)$ 
55
56 set/multiset
57     insert() 插入一个数
58     find() 查找一个数
59     count() 返回某一个数的个数
60     erase()
61         (1) 输入是一个数x, 删除所有x  $O(k + \log n)$ 
62         (2) 输入一个迭代器, 删除这个迭代器
63     lower_bound()/upper_bound()
64         lower_bound(x) 返回大于等于x的最小的数的迭代器
65         upper_bound(x) 返回大于x的最小的数的迭代器
66 map/multimap
67     insert() 插入的数是一个pair
68     erase() 输入的参数是pair或者迭代器
69     find()
70     [] 注意multimap不支持此操作。 时间复杂度是  $O(\log n)$ 
71     lower_bound()/upper_bound()
72 unordered_set, unordered_map, unordered_multiset, unordered_multimap, 哈希表
73 和上面类似, 增删改查的时间复杂度是  $O(1)$ 
74 不支持 lower_bound()/upper_bound(), 迭代器的++, --
75 bitset, 压位
76     bitset<10000> s;
77     ~, &, |, ^
78     >>, <<
79     ==, !=
80     []
81
82     count() 返回有多少个1
83
84     any() 判断是否至少有一个1
85     none() 判断是否全为0
86
87     set() 把所有位置成1
88     set(k, v) 将第k位变成v
89     reset() 把所有位变成0
90     flip() 等价于~
91     flip(k) 把第k位取反

```

10 C++pb_ds

洛谷 P3369 【模板】普通平衡树

```

1  #include <iostream>
2  #include <ext/pb_ds/assoc_container.hpp> // 引入树的头文件
3  #include <ext/pb_ds/tree_policy.hpp>
4
5  /**
6   * 赛时可以写万能头
7   * #include <bits/stdc++.h>
8   * #include <bits/extc++.h>
9   */
10
11 namespace pbds = __gnu_pbds;

```

```

12
13 template<typename T, typename cmp = std::less<T>>
14 using rbtree = pbds::tree<T, pbds::null_type, cmp, pbds::rb_tree_tag,
pbds::tree_order_statistics_node_update>;
15 using PII = std::pair<int, int>;
16
17 void solve() {
18     int n;
19     std::cin >> n;
20
21     rbtree<PII> tree;
22     int idx = 0;
23
24     while (n --) {
25         int op, x;
26         std::cin >> op >> x;
27
28         if (op == 1) {
29             tree.insert({x, ++ idx});
30         } else if (op == 2) {
31             tree.erase(tree.lower_bound({x, 0}));
32         } else if (op == 3) {
33             std::cout << tree.order_of_key({x, 0}) + 1 << '\n';
34         } else if (op == 4) {
35             std::cout << tree.find_by_order(x - 1)->first << '\n';
36         } else if (op == 5) {
37             std::cout << (--tree.lower_bound({x, 0}))->first << '\n';
38         } else {
39             std::cout << tree.upper_bound({x, idx})->first << '\n';
40         }
41     }
42 }
43
44 int main() {
45     std::ios::sync_with_stdio(false);
46     std::cin.tie(nullptr);
47     std::cout.tie(nullptr);
48
49     solve();
50 }

```

11 线段树

```

1  #include <bits/stdc++.h>
2
3  using LL = long long;
4
5  struct node {
6      int l, r;
7      LL val, add;
8  };
9
10 struct segmentTree {
11     #define ls(x) (x << 1)
12     #define rs(x) (x << 1 | 1)

```

```

13  #define p tr[u]
14  #define pl tr[ls(u)]
15  #define pr tr[rs(u)]
16
17  std::vector<node> tr;
18  segmentTree(int n) : tr(n << 2) {}
19
20  void pushup(int u) {
21      p.val = pl.val + pr.val;
22  }
23
24  void pushdown(int u) {
25      if (p.add) {
26          pl.val += p.add * (pl.r - pl.l + 1);
27          pr.val += p.add * (pr.r - pr.l + 1);
28          pl.add += p.add;
29          pr.add += p.add;
30          p.add = 0;
31      }
32  }
33
34  void build(int u, int l, int r, std::vector<LL> &a) {
35      p.l = l, p.r = r;
36      if (l == r) {
37          p.val = a[l];
38          return;
39      }
40      int mid = (l + r) >> 1;
41      build(ls(u), l, mid, a);
42      build(rs(u), mid + 1, r, a);
43      pushup(u);
44  }
45
46  void modify(int u, int l, int r, LL d) {
47      if (p.l >= l && p.r <= r) {
48          p.val += d * (p.r - p.l + 1);
49          p.add += d;
50          return;
51      }
52      pushdown(u);
53      int mid = (p.l + p.r) >> 1;
54      if (l <= mid) modify(ls(u), l, r, d);
55      if (r > mid) modify(rs(u), l, r, d);
56      pushup(u);
57  }
58
59  LL query(int u, int l, int r) {
60      if (p.l >= l && p.r <= r) return p.val;
61      pushdown(u);
62      int mid = (p.l + p.r) >> 1;
63      LL res = 0;
64      if (l <= mid) res += query(ls(u), l, r);
65      if (r > mid) res += query(rs(u), l, r);
66      return res;
67  }
68

```

```

69     #undef ls
70     #undef rs
71     #undef p
72     #undef pl
73     #undef pr
74 };
75
76 void solve() {
77     int n, m;
78     std::cin >> n >> m;
79     std::vector<LL> a(n + 1);
80     for (int i = 1; i <= n; ++i) std::cin >> a[i];
81     segmentTree tree(n + 10);
82     tree.build(1, 1, n, a);
83     while (m --) {
84         int op, x, y, k;
85         std::cin >> op >> x >> y;
86         if (op == 1) {
87             std::cin >> k;
88             tree.modify(1, x, y, k);
89         } else {
90             std::cout << tree.query(1, x, y) << '\n';
91         }
92     }
93 }
94
95 int main() {
96     std::ios::sync_with_stdio(false);
97     std::cin.tie(nullptr);
98     std::cout.tie(nullptr);
99
100     solve();
101 }

```

12 树状数组

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  const int N = 5e5 + 10;
6  int n, m, tr[N];
7
8  int lowbit(int x) {
9      return x & -x;
10 }
11
12 void add(int x, int v) {
13     for (; x < N; x += lowbit(x)) tr[x] += v;
14 }
15
16 int query(int x) {
17     int res = 0;
18     for (; x; x -= lowbit(x)) res += tr[x];
19     return res;

```

```

20 }
21
22 int main() {
23     scanf("%d%d", &n, &m);
24     for (int i = 1; i <= n; ++ i) {
25         int t; scanf("%d", &t);
26         add(i, t);
27     }
28     while (m --) {
29         int op, x, y;
30         scanf("%d%d%d", &op, &x, &y);
31         if (op == 1) add(x, y);
32         else printf("%d\n", query(y) - query(x - 1));
33     }
34 }

```

13 回文串

马拉车 d1: 奇数长度最长回文半径 d2: 偶数长度最长回文半径

```

1 void calc(string &s, int n, bool *pre) { // 判断前缀是否回文 字符串下标从0开始
2     vector<int> d1(n);
3     for (int i = 0, l = 0, r = -1; i < n; i++) {
4         int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
5         while (0 <= i - k && i + k < n && s[i - k] == s[i + k]) {
6             k++;
7         }
8         d1[i] = k--;
9         if (i + k > r) {
10             l = i - k;
11             r = i + k;
12         }
13     }
14     vector<int> d2(n);
15     for (int i = 0, l = 0, r = -1; i < n; i++) {
16         int k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
17         while (0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k]) {
18             k++;
19         }
20         d2[i] = k--;
21         if (i + k > r) {
22             l = i - k - 1;
23             r = i + k;
24         }
25     }
26     for (int i = 0; i < n; ++ i) {
27         int l = i - d1[i] + 1, r = i + d1[i] - 1;
28         if (!l && !pre[r]) pre[r] = true;
29
30         l = i - d2[i], r = i + d2[i] - 1;
31         if (!l && !pre[r]) pre[r] = true;
32     }
33 }

```

搜索

1 DFS

1.1 枚举

1.1.1 指数型

```
1 void calc(int x) {
2     if (x > n) {
3         for (int i: choose) cout << i << ' ';
4         cout << endl;
5         return;
6     }
7     calc(x + 1); // choose
8     choose.push_back(x); // not choose
9     calc(x + 1);
10    choose.pop_back();
11 };
```

1.1.2 组合型

```
1 void calc(int x) {
2     if (choose.size() > m || choose.size() + n - x + 1 < m)
3         return;
4     if (x > n) {
5         for (int i: choose) cout << i << ' ';
6         cout << endl;
7         return;
8     }
9     choose.push_back(x); // not choose
10    calc(x + 1);
11    choose.pop_back();
12    calc(x + 1); // choose
13 };
```

1.1.3 排列型

```
1 void calc(int x) {
2     if (x > n) {
3         for (int i = 1; i <= n; ++ i)
4             printf("%d%c", a[i], " \n"[i == n]);
5         return;
6     }
7     for (int i = 1; i <= n; ++ i)
8         if (!st[i]) {
9             a[x] = i, st[i] = true;
10            calc(x + 1);
11            a[x] = 0, st[i] = false;
12        }
13 }
```

2 BFS

2.1 Flood-Fill

可以在线性时间复杂度内, 找到某个点所在的连通块.

[1097. 池塘计数 - AcWing题库](#)

```
1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  #define x first
5  #define y second
6
7  using namespace std;
8
9  typedef pair<int, int> PII;
10
11 const int N = 1005, M = N * N;
12 char g[N][N];
13 bool st[N][N];
14 int n, m, cnt;
15 PII q[M];
16
17 void bfs(int x, int y) {
18     int hh = 0, tt = 0;
19     q[tt] = {x, y};
20     st[x][y] = true;
21
22     while (hh <= tt) {
23         auto [x, y] = q[hh++];
24         for (int i = x - 1; i <= x + 1; ++i)
25             for (int j = y - 1; j <= y + 1; ++j) {
26                 if (i == x && j == y) continue;
27                 if (i < 0 || i >= n || j < 0 || j >= m) continue;
28                 if (g[i][j] == '.' || st[i][j]) continue;
29
30                 q[++tt] = {i, j};
31                 st[i][j] = true;
32             }
33     }
34 }
35
36 int main() {
37     cin.tie(0) -> sync_with_stdio(0);
38
39     cin >> n >> m;
40     for (int i = 0; i < n; ++i) cin >> g[i];
41
42     for (int i = 0; i < n; ++i)
43         for (int j = 0; j < m; ++j)
44             if (g[i][j] == 'W' && !st[i][j]) {
```



```

45         bfs(i, j);
46         ++ cnt;
47     }
48
49     cout << cnt << '\n';
50 }

```

2.2 最短距离

要满足所有边的权重都相同

[1076. 迷宫问题 - AcWing题库](#)

```

1  #include <iostream>
2  #include <cstring>
3  #include <vector>
4  #include <algorithm>
5
6  #define x first
7  #define y second
8
9  using namespace std;
10
11 typedef pair<int, int> PII;
12
13 const int N = 1005;
14 const int dx[] = {0, -1, 0, 1};
15 const int dy[] = {-1, 0, 1, 0};
16 bool g[N][N];
17 bool st[N][N];
18 PII pre[N][N];
19 int n, idx;
20 PII q[N * N];
21
22 bool isin(int x, int y) {
23     return 0 <= x && x < n && 0 <= y && y < n;
24 }
25
26 void bfs(int x, int y) {
27     int hh = 0, tt = 0;
28     q[0] = {x, y};
29     st[x][y] = true;
30
31     while (hh <= tt) {
32         auto [x, y] = q[hh ++];
33         if (x == n - 1 && y == n - 1) return;
34
35         for (int i = 0; i < 4; ++ i) {
36             int nx = x + dx[i], ny = y + dy[i];
37             if (isin(nx, ny) && !g[nx][ny] && !st[nx][ny]) {
38                 q[++ tt] = {nx, ny};
39                 pre[nx][ny] = {x, y};
40                 st[nx][ny] = true;
41             }
42         }
43     }
44 }

```

```

44 }
45
46 void printpath(int x, int y, int cnt) {
47     if (!x && !y) { cout << "0 0\n"; return; }
48     auto [nx, ny] = pre[x][y];
49     printpath(nx, ny, cnt + 1);
50     cout << x << ' ' << y << '\n';
51 }
52
53 int main() {
54     cin.tie(0)->sync_with_stdio(0);
55
56     cin >> n;
57     for (int i = 0; i < n; ++ i)
58         for (int j = 0; j < n; ++ j)
59             cin >> g[i][j];
60
61     bfs(0, 0);
62
63     printpath(n - 1, n - 1, 0);
64 }

```

2.3 多源BFS

增加一个虚拟源点

[173. 矩阵距离 - AcWing题库](#)

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4
5  using namespace std;
6
7  typedef pair<int, int> PII;
8
9  const int N = 1010;
10 const int dx[] = {0, -1, 0, 1};
11 const int dy[] = {-1, 0, 1, 0};
12 char g[N][N];
13 int dist[N][N];
14 int n, m;
15 PII q[N * N];
16
17 bool isin(int x, int y) {
18     return 0 <= x && x < n && 0 <= y && y < m;
19 }
20
21 void bfs() {
22     memset(dist, -1, sizeof dist);
23     int hh = 0, tt = -1;
24
25     for (int i = 0; i < n; ++ i)
26         for (int j = 0; j < m; ++ j)
27             if (g[i][j] == '1') {
28                 q[++ tt] = {i, j};

```

```

29         dist[i][j] = 0;
30     }
31
32     while (hh <= tt) {
33         auto [x, y] = q[hh ++];
34         int d = dist[x][y];
35
36         for (int i = 0; i < 4; ++ i) {
37             int nx = x + dx[i], ny = y + dy[i];
38             if (isin(nx, ny) && dist[nx][ny] == -1) {
39                 dist[nx][ny] = d + 1;
40                 q[++ tt] = {nx, ny};
41             }
42         }
43     }
44 }
45
46 int main() {
47     cin.tie(0)->sync_with_stdio(0);
48
49     cin >> n >> m;
50     for (int i = 0; i < n; ++ i) cin >> g[i];
51
52     bfs();
53
54     for (int i = 0; i < n; ++ i)
55         for (int j = 0; j < m; ++ j)
56             cout << dist[i][j] << " \n"[j == m - 1];
57 }

```

2.4 最小步数

[AcWing 1107. 魔板 - AcWing](#)

[AcWing 845. 八数码 - AcWing](#)

```

1 // https://www.acwing.com/activity/content/problem/content/1475/
2 #include <iostream>
3 #include <cstring>
4 #include <algorithm>
5 #include <unordered_map>
6 #include <queue>
7
8 using namespace std;
9
10 string start = "12345678";
11 unordered_map<string, int> dist;
12 unordered_map<string, pair<char, string>> path;
13 const int change[3][8] = {
14     {7, 6, 5, 4, 3, 2, 1, 0},
15     {3, 0, 1, 2, 5, 6, 7, 4},
16     {0, 6, 1, 3, 4, 2, 5, 7}
17 };
18

```

```

19 string move(string s, int k) {
20     string res;
21     for (int i = 0; i < 8; ++ i) res += s[change[k][i]];
22     return res;
23 }
24
25 int bfs(string start, string end) {
26     queue<string> q;
27     q.push(start);
28     dist[start] = 0;
29
30     while (q.size()) {
31         string t = q.front();
32         int d = dist[t];
33         q.pop();
34
35         if (t == end) return d;
36
37         for (int i = 0; i < 3; ++ i) {
38             string u = move(t, i);
39             if (!dist.count(u)) {
40                 q.push(u);
41                 dist[u] = d + 1;
42                 path[u] = {'A' + i, t};
43             }
44         }
45     }
46
47     return -1;
48 }
49
50 void printpath(string s) {
51     if (s == start) return;
52     auto [op, str] = path[s];
53     printpath(str);
54     cout << op;
55 }
56
57 int main() {
58     cin.tie(0)->sync_with_stdio(0);
59
60     string end;
61     for (int i = 0; i < 8; ++ i) {
62         int x; cin >> x;
63         end += '0' + x;
64     }
65
66     cout << bfs(start, end) << '\n';
67     printpath(end);
68 }

```

2.5 双端队列BFS

[AcWing 175. 电路维修 - AcWing](#)

```
1  #include <iostream>      /*如果边权只有0和1就可以将bfs转化为dijkstra*/
2  #include <cstring>      /*在本题中将联通的两点看成边权为0*/
3  #include <deque>        /*如果两点中间的标准件需要旋转才能联通则看为1*/
4  #include <algorithm>    /*即转化为最短路问题*/
5                          /*容易知道奇点无法到达所以在bfs时不会出现某个点被操作两次*/
6  using namespace std;    /*该题点的坐标和标准件坐标并不同,注意变换*/
7
8  typedef pair<int, int> PII;
9
10 const int N = 510;
11 char g[N][N];
12 int dist[N][N];
13 bool st[N][N];
14 int n, m;
15
16 int bfs() {
17     memset(dist, 0x3f, sizeof dist);
18     memset(st, false, sizeof st);
19     deque<PII> q;
20     q.push_back({0, 0});
21     dist[0][0] = 0;
22
23     char cs[] = "\\//\\";
24     int dx[] = {-1, 1, 1, -1}, dy[] = {-1, -1, 1, 1};
25     int ix[] = {-1, 0, 0, -1}, iy[] = {-1, -1, 0, 0};
26
27     while (q.size()) {
28         auto [x, y] = q.front();
29         int d = dist[x][y];
30         q.pop_front();
31
32         if (st[x][y]) continue;
33         st[x][y] = true;
34
35         for (int i = 0; i < 4; ++i) {
36             int nx = x + dx[i], ny = y + dy[i];
37             if (nx < 0 || nx > n || ny < 0 || ny > m) continue;
38
39             int gx = x + ix[i], gy = y + iy[i];
40             int w = g[gx][gy] != cs[i];
41
42             if (d + w < dist[nx][ny]) {
43                 dist[nx][ny] = d + w;
44                 if (w) q.push_back({nx, ny});
45                 else q.push_front({nx, ny});
46             }
47         }
48     }
49
50     return dist[n][m];
51 }
52
```

```

53 int main() {
54     cin.tie(0)->sync_with_stdio(0);
55
56     int T; cin >> T;
57     while (T --) {
58         cin >> n >> m;
59         for (int i = 0; i < n; ++ i) cin >> g[i];
60
61         if (n + m & 1) cout << "NO SOLUTION\n";
62         else cout << bfs() << '\n';
63     }
64 }

```

3 背包问题

3.1 01背包问题

每件物品最多可以选一次

起点	$i \in [0, m], f(0, i) = 0$
转移方程	$f(i, j) = \max(f(i - 1, j), f(i - 1, j - v_i) + w_i)$
终点	$f(n, m)$

- 朴素版本

```

1 for (int i = 1; i <= n; ++ i)
2     for (int j = 0; j <= m; ++ j) {
3         f[i][j] = f[i - 1][j];
4         if (v[i] <= j) f[i][j] = max(f[i][j], f[i - 1][j - v[i]] + w[i]);
5     }
6
7 printf("%d", f[n][m]);

```

- 优化为一维

```

1 for (int i = 1; i <= n; ++ i)
2     for (int j = m; j >= v[i]; -- j)
3         f[j] = max(f[j], f[j - v[i]] + w[i]);
4
5 printf("%d", f[m]);

```

3.2 完全背包

每个物品可以选无限个

起点	$f(0, j) = 0$
转移方程	$f(i, j) = \max_{k=0}^{v_i \times k \leq j} f(i - 1, j - v_i \times k) + w_i \times k$
终点	$f(n, m)$

- 朴素版本

```

1 for (int i = 1; i <= n; ++ i)
2     for (int j = 0; j <= m; ++ j)
3         for (int k = 0; k * v[i] <= j; ++ k)
4             f[i][j] = max(f[i][j], f[i-1][j - k * v[i]] + k * w[i]);
5
6 cout << f[n][m] << endl;

```

- 一维优化

```

1 for (int i = 1; i <= n; ++ i)
2     for (int j = v[i]; j <= m; ++ j)
3         f[j] = max(f[j], f[j - v[i]] + w[i]);
4
5 cout << f[m] << endl;

```

3.3 多重背包

每个物品只能选有限个

起点	$f(0, j) = 0$
转移方程	$f(i, j) = \max_{k=0}^{s_i} f(i-1, j - v_i \times k) + w_i \times k$
终点	$f(n, m)$

- 朴素版本

```

1 for (int i = 1; i <= n; ++ i)
2     for (int j = 0; j <= m; ++ j)
3         for (int k = 0; k <= s[i] && k * v[i] <= j; ++ k)
4             f[i][j] = max(f[i][j], f[i-1][j - k * v[i]] + w[i] * k);
5
6 cout << f[n][m] << endl;

```

- 二进制优化

```

1 for (int i = 1; i <= n; ++ i) {
2     int v, w, s;
3     scanf("%d%d%d", &v, &w, &s);
4     for (int k = 1; k <= s; s -= k, k <= 1)
5         goods.push_back({k * v, k * w});
6     if (s) goods.push_back({s * v, s * w});
7 }
8 for (auto [v, w]: goods)
9     for (int j = m; j >= v; -- j)
10         f[j] = max(f[j], f[j - v] + w);
11 cout << f[m] << endl;

```

3.4 分组背包

每组物品有若干个，同一组内的物品最多只能选一个。

起点	$f(0, j) = 0$
转移方程	$f(i, j) = \max(f(i-1, j), \max_{k=1}^{s_i} f(i-1, j - v_{i,k}) + w_{i,k})$
终点	$f(n, m)$

```

1 for (int i = 1; i <= n; ++ i)
2     for (int j = m; j >= 0; -- j)
3         for (int k = 1; k <= s[i]; ++ k)
4             if (v[i][k] <= j)
5                 f[j] = max(f[j], f[j - v[i][k]] + w[i][k]);
6 printf("%d\n", f[m]);

```

4 数位DP

- 计数问题

```

1 LL f(int pos, int cnt, bool islimit, bool isnum, int digit) {
2     if (pos == s.size()) return isnum ? cnt : 0;
3     LL& u = dp[pos][cnt];
4     if (!islimit && isnum && u >= 0) return u;
5     LL res = isnum ? 0 : f(pos + 1, cnt, false, false, digit);
6     int up = islimit ? s[pos] - '0' : 9;
7     for (int d = 1 - isnum; d <= up; ++ d)
8         res += f(pos + 1, cnt + (d == digit), islimit && d == up, true, digit);
9     if (!islimit && isnum) u = res;
10    return res;
11 }
12
13 LL solve(LL x, int digit) {
14     memset(dp, -1, sizeof dp);
15     s = to_string(x);
16     return f(0, 0, true, false, digit);
17 }

```

5 线性DP

5.1 数字三角形

```

1 for (int i = 0; i <= r; ++ i)
2     for (int j = 0; j <= i + 1; ++ j)
3         f[i][j] = -INF;
4
5 f[1][1] = a[1][1];
6 for (int i = 2; i <= r; ++ i)
7     for (int j = 1; j <= i; ++ j)
8         f[i][j] = max(f[i - 1][j - 1] + a[i][j], f[i - 1][j] + a[i][j]);
9 // 可以从上向下走也可以是从左向右走
10 int res = -INF;
11 for (int i = 1; i <= r; ++ i) res = max(res, f[r][i]);
12 printf("%d\n", res);

```


5.2 最长上升子序列

- $O(n^2)$ 做法

```
1 for (int i = 1; i <= n; ++ i) {
2     f[i] = 1;
3     for (int j = 1; j <= i; ++ j)
4         if (w[i] > w[j]) f[i] = max(f[i], f[j] + 1);
5     ans = max(ans, f[i]);
6 }
```

- $O(n\log(n))$ 做法

```
1 int len = 0;
2 for (int i = 0; i < n; ++ i) {
3     int l = 0, r = len;
4     while (l < r) {
5         int mid = l + r + 1 >> 1;
6         if (q[mid] < a[i]) l = mid;
7         else r = mid - 1;
8     }
9     len = max(len, r + 1);
10    q[r + 1] = a[i];
11 }
12 printf("%d\n", len);
```

5.3 最长公共子序列

```
1 for (int i = 1; i <= n; ++ i)
2     for (int j = 1; j <= m; ++ j) {
3         f[i][j] = max(f[i - 1][j], f[i][j - 1]);
4         if (a[i] == b[j]) f[i][j] = max(f[i][j], f[i - 1][j - 1] + 1);
5     }
6 printf("%d\n", f[n][m]);
```

5.4 区间DP

- 石子合并

```
1 #include <iostream>
2 #include <cstring>
3
4 using namespace std;
5
6 const int N = 1010;
7 int n, m, f[N][N], s[N];
8
9 int main() {
10     scanf("%d", &n);
11     for (int i = 1; i <= n; ++ i) scanf("%d", &s[i]), s[i] += s[i - 1];
12
13     f[0][0] = 0;
14
15     for (int len = 2; len <= n; ++ len)
```

```

16     for (int i = 1; i + len - 1 <= n; ++ i)
17     {
18         int j = i + len - 1;
19         f[i][j] = 1e9;
20         for (int k = i; k < j; ++ k)
21             f[i][j] = min(f[i][j], f[i][k] + f[k + 1][j] + s[j] - s[i - 1]);
22     }
23
24     printf("%d\n", f[1][n]);
25 }

```

- 环形石子合并

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4
5  #define max(x, y) x < y ? x = y : 0
6  #define min(x, y) x > y ? x = y : 0
7
8  using namespace std;
9
10 const int N = 205 << 1;
11 int n, w[N], f[N][N], s[N], g[N][N];
12
13 int main() {
14     cin.tie(0)->sync_with_stdio(0), cout.tie(0);
15     cin >> n;
16     for (int i = 1; i <= n; ++ i) cin >> w[i], w[i + n] = w[i];
17
18     for (int i = 1; i <= n * 2; ++ i) s[i] = s[i - 1] + w[i];
19
20     memset(f, 0x3f, sizeof f);
21     for (int i = 0; i <= 2 * n; ++ i) f[i][i] = 0;
22
23     for (int len = 2; len <= n; ++ len)
24         for (int i = 1; i + len - 1 <= n * 2; ++ i) {
25             int j = i + len - 1;
26             for (int k = i; k < j; ++ k) {
27                 min(f[i][j], f[i][k] + f[k + 1][j] + s[j] - s[i - 1]);
28                 max(g[i][j], g[i][k] + g[k + 1][j] + s[j] - s[i - 1]);
29             }
30         }
31
32     int mv = 0x3f3f3f3f, mi = -mv;
33     for (int i = 1; i <= n; ++ i) {
34         min(mv, f[i][i + n - 1]);
35         max(mi, g[i][i + n - 1]);
36     }
37     cout << mv << endl;
38     cout << mi << endl;
39 }

```

图论

1 存图

邻接表

```
1 int h[N], e[M], ne[M], idx;
2 void add(int a, int b) { // 加边
3     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
4 }
```

2 拓扑排序

有向图才有拓扑序列（有向无环图又叫拓扑图）

性质：一个有向无环图一定至少存在一个入度为0的点

- 度数

入度：有向图中某点作为图中边的终点的次数之和

出度：顶点的出边条数称为该顶点的出度

```
1 bool topsort() {
2     int tt = -1, hh = 0;
3     for (int i = 1; i <= n; ++ i) if (!d[i]) q[++ tt] = i;
4     while (hh <= tt) {
5         int t = q[hh ++];
6         for (int i = h[t]; i != -1; i = ne[i]) {
7             int j = e[i];
8             d[j] --;
9             if (!d[j]) q[++ tt] = j;
10        }
11    }
12    return tt == n - 1;
13 }
```

3 最短路

n: 定点数; m: 边数

3.1 单源最短路

3.1.1 所有边权都是正数

3.1.1.1 朴素Dijkstra算法

$O(n^2)$ ($m, n \leq 1e5$; 即稠密图)

```
1 #include <iostream>
2 #include <cstring>
3
4 using namespace std;
5
6 const int N = 510;
7 int n, m, a, b, c;
8 int g[N][N]; // 稠密图用邻接矩阵存
```

```

9  int dist[N]; // 距离起点的距离
10 bool st[N]; // 该点距离有无确定
11
12 int dijkstra() {
13     memset(dist, 0x3f, sizeof dist); // 所有点的距离初始化为+∞
14     dist[1] = 0;
15     for (int i = 0; i < n - 1; ++ i) {
16         int t = -1;
17         // 在所有没有确定最短距离的点中找到最近的点t
18         for (int j = 1; j <= n; ++ j)
19             if (!st[j] && (t == -1 || dist[t] > dist[j]))
20                 t = j;
21         // 用t点更新其他点的最短路
22         for (int j = 1; j <= n; ++ j)
23             dist[j] = min(dist[j], dist[t] + g[t][j]);
24         st[t] = true;
25     }
26     if (dist[n] == 0x3f3f3f3f) return -1;
27     return dist[n];
28 }
29
30 int main() {
31     scanf("%d%d", &n, &m);
32     memset(g, 0x3f, sizeof g);
33     while (m --) {
34         scanf("%d%d%d", &a, &b, &c);
35         g[a][b] = min(g[a][b], c);
36         // 重边取最小即可
37     }
38     printf("%d\n", dijkstra());
39     return 0;
40 }

```

3.1.1.2 堆优化的Dijkstra算法

$O(m \log n)$ ($m, n > 1e5$; 即稀疏图)

```

1  // 堆优化dijkstra算法 --- 稀疏图
2  #include <iostream>
3  #include <queue>
4  #include <cstring>
5
6  using namespace std;
7
8  typedef pair<int, int> PII;
9
10 const int N = 1e6 + 10;
11 int n, m, a, b, c;
12 int h[N], w[N], e[N], ne[N], idx; // 稀疏图用邻接表存
13 int dist[N];
14 bool st[N];
15
16 void add(int a, int b, int c) {
17     e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++;
18 }
19

```

```

20 int dijkstra() {
21     memset(dist, 0x3f, sizeof dist);
22     dist[1] = 0;
23     priority_queue<PII, vector<PII>, greater<PII>> heap;
24     // 定义小根堆 first---距离 second---点的编号
25     heap.push({0, 1});
26     while (heap.size()) {
27         // 每次取出距离最短的点
28         auto t = heap.top();
29         heap.pop();
30         int ver = t.second, distance = t.first;
31         // 如果ver点的最短路已经确定那么continue
32         if (st[ver]) continue;
33         st[ver] = true;
34         // 更新ver点连的边的最短路
35         for (int i = h[ver]; i != -1; i = ne[i]) {
36             int j = e[i];
37             if (dist[j] > dist[ver] + w[i]) {
38                 dist[j] = dist[ver] + w[i];
39                 heap.push({dist[j], j});
40             }
41         }
42     }
43     if (dist[n] == 0x3f3f3f3f) return -1;
44     return dist[n];
45 }
46
47 int main() {
48     scanf("%d%d", &n, &m);
49     memset(h, -1, sizeof h);
50     while (m --) {
51         scanf("%d%d%d", &a, &b, &c);
52         add(a, b, c);
53         // 邻接表不用考虑重边
54     }
55     printf("%d\n", dijkstra());
56     return 0;
57 }

```

3.1.2 存在负权边

3.1.2.1 Bellman-Ford

$O(nm)$

```

1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  const int N = 510, M = 1e4 + 10;
7  struct Edge {
8      int a, b, c;
9  } edge[M];
10 int n, m, k, a, b, c;
11 int dist[N];
12 int last[N]; // 上次的最短路

```

```

13
14 void bellman_ford() {
15     memset(dist, 0x3f, sizeof dist);
16     dist[1] = 0;
17     for (int i = 0; i < k; ++ i) { // 走k次
18         memcpy(last, dist, sizeof dist); // 防止出现串联
19         for (int j = 0; j < m; j ++ ) {
20             auto e = edge[j];
21             dist[e.b] = min(dist[e.b], last[e.a] + e.c);
22             // 松弛操作
23         } // 三角不等式---dist[b]<=dist[a]+c
24     }
25 }
26
27 int main() {
28     scanf("%d%d%d", &n, &m, &k);
29     for (int i = 0; i < m; ++ i) {
30         scanf("%d%d%d", &a, &b, &c);
31         edge[i] = {a, b, c};
32     }
33     bellman_ford();
34     // 因为存在负权边可能存在dist[x]==INF-y(y比较小)
35     // 所有只要判断>一个比较大的数即可
36     if (dist[n] > 0x3f3f3f3f / 2) puts("impossible");
37     else printf("%d\n", dist[n]);
38     return 0;
39 }

```

3.1.2.2 SPFA

一般: $O(nm)$; 最坏: $O(nm)$

```

1  #include <iostream>
2  #include <cstring>
3  #include <cstdio>
4  #include <queue>
5
6  using namespace std;
7
8  const int N = 1e6 + 10;
9  int n, m, a, b, c;
10 int h[N], e[N], ne[N], idx, w[N];
11 int dist[N];
12 bool st[N];
13
14 void add(int a, int b, int c) {
15     e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++;
16 }
17
18 int spfa() {
19     memset(dist, 0x3f, sizeof dist);
20     dist[1] = 0;
21     queue<int> q;
22     q.push(1);
23     st[1] = true;
24     while (q.size()) {

```

```

25     int t = q.front();
26     q.pop();
27     st[t] = false;
28     for (int i = h[t]; i != -1; i = ne[i]) {
29         int j = e[i];
30         if (dist[j] > dist[t] + w[i]) {
31             dist[j] = dist[t] + w[i];
32             if (!st[j]) {
33                 q.push(j);
34                 st[j] = true;
35             }
36         }
37     }
38 }
39 return dist[n];
40 }
41
42 int main() {
43     scanf("%d%d", &n, &m);
44     memset(h, -1, sizeof h);
45     while (m --) {
46         scanf("%d%d%d", &a, &b, &c);
47         add(a, b, c);
48     }
49     int t = spfa();
50     if (t == 0x3f3f3f3f) puts("impossible");
51     else printf("%d\n", t);
52     return 0;
53 }

```

3.2 多源汇最短路

3.2.1 Floyd算法

$O(n^3)$

源点即起点

汇点即终点

```

1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  const int N = 210, INF = 1e9;
7  int n, m, Q;
8  int d[N][N];
9  int a, b, c;
10
11 void floyd() {
12     for (int k = 1; k <= n; ++ k)
13         for (int i = 1; i <= n; ++ i)
14             for (int j = 1; j <= n; ++ j)
15                 d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
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 <= n; ++ j)
22             if (i == j) d[i][j] = 0;
23             else d[i][j] = INF;
24     while (m --) {
25         scanf("%d%d%d", &a, &b, &c);
26         d[a][b] = min(d[a][b], c);
27     }
28     floyd();
29     while (Q --) {
30         scanf("%d%d", &a, &b);
31         int t = d[a][b];
32         if (t > INF / 2) puts("impossible");
33         else printf("%d\n", t);
34     }
35     return 0;
36 }

```

4 最小生成树

4.1 朴素版Prim

$O(n^2)$

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4
5  using namespace std;
6
7  const int N = 510, INF = 0x3f3f3f3f;
8  int n, m;
9  int g[N][N];
10 int dist[N];
11 bool st[N]; // 表示一个点是否在集合里
12 int a, b, c;
13
14 int prim() {
15     memset(dist, 0x3f, sizeof dist);
16     // 初始化成+∞
17     int res = 0;
18     for (int i = 0; i < n; ++ i) {
19         // 找到集合外距离最近的点
20         int t = -1;
21         for (int j = 1; j <= n; ++ j)
22             if (!st[j] && (t == -1 || dist[t] > dist[j]))
23                 t = j;
24         // 如果不能到达集合说明不存在最小生成树
25         if (i && dist[t] == INF) return INF;
26         // 此句写在31行前避免自环
27         if (i) res += dist[t];
28         st[t] = true; // 将t加入集合

```



```

29         // 用t更新其他点到集合的距离
30         for (int j = 1; j <= n; ++j) dist[j] = min(dist[j], g[t][j]);
31     }
32
33     return res;
34 }
35
36 int main() {
37     scanf("%d%d", &n, &m);
38     memset(g, 0x3f, sizeof g);
39     while (m --) {
40         scanf("%d%d%d", &a, &b, &c);
41         g[a][b] = g[b][a] = min(g[a][b], c);
42     }
43     int t = prim();
44     if (t == INF) puts("impossible");
45     else printf("%d\n", t);
46 }

```

4.2 堆优化Prim

$O(m \log n)$

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4  #include <queue>
5
6  using namespace std;
7
8  typedef pair<int, int> PII;
9
10 const int N = 5e3 + 5, M = 4e5 + 5, INF = 0x3f3f3f3f;
11 int st[N], dist[N];
12 int h[N], e[M], ne[M], w[M], idx;
13 int n, m;
14
15 void add(int a, int b, int c) {
16     e[++idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx;
17 }
18
19 int prim() {
20     priority_queue<PII, vector<PII>, greater<PII>> q;
21     memset(dist, 0x3f, sizeof dist);
22     int res = 0, cnt = 0;
23     dist[1] = 0;
24     q.push({dist[1], 1});
25
26     while (q.size() && cnt < n) {
27         int d = q.top().first, t = q.top().second;
28         q.pop();
29         if (st[t]) continue;
30         cnt++;
31         res += d;
32         st[t] = 1;
33         for (int i = h[t]; i; i = ne[i]) {

```

```

34         int j = e[i];
35         if (w[i] < dist[j]) {
36             dist[j] = w[i];
37             q.push({dist[j], j});
38         }
39     }
40 }
41
42 return cnt == n ? res: INF;
43 }
44
45 int main() {
46     scanf("%d%d", &n, &m);
47     while (m --) {
48         int a, b, c;
49         scanf("%d%d%d", &a, &b, &c);
50         add(a, b, c);
51         add(b, a, c);
52     }
53
54     int t = prim();
55
56     if (t == INF) puts("orz");
57     else printf("%d", t);
58 }

```

4.3 Kruskal

$O(m \log m)$

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4
5  using namespace std;
6
7  const int N = 1e6 + 10, M = 2e6 + 10, INF = 0x3f3f3f3f;
8  int n, m;
9  int p[N]; // 并查集数组
10
11 struct Edge {
12     int a, b, w;
13     bool operator < (const Edge &W) const {
14         return w < W.w;
15     } // 运算符重载, 写了可以不用写cmp
16 } edges[M];
17
18 int find(int x) {
19     return p[x] == x ? p[x] : p[x] = find(p[x]);
20 }
21
22 int kruskal() {
23     sort(edges, edges + m);
24     // 将所有点按权重从小到大排序  $O(m \log m)$ 
25     // 排序算法常数小
26     // 所以能跑的比较快

```

```

27     for (int i = 1; i <= n; ++ i) p[i] = i;
28     // 初始化并查集
29     int res = 0, cnt = 0;
30     // res表示生成树中所有边权之和
31     // cnt表示加入了多少边
32     for (int i = 0; i < m; ++ i) {
33         // 枚举每条边
34         int a = edges[i].a, b = edges[i].b, w = edges[i].w;
35         // 如果a, b不连通
36         a = find(a), b = find(b);
37         if (a != b) {
38             p[a] = b; // 将这条边加入集合中
39             res += w;
40             cnt ++;
41         }
42     }
43     // 如果有边未加入集合说明最小生成树不存在
44     if (cnt < n - 1) return INF;
45     return res;
46 }
47
48 int main() {
49     scanf("%d%d", &n, &m);
50     for (int i = 0; i < m; ++ i) {
51         int a, b, w;
52         scanf("%d%d%d", &a, &b, &w);
53         edges[i] = {a, b, w};
54     }
55     int t = kruskal();
56     if (t == INF) puts("impossible");
57     else printf("%d\n", t);
58 }

```

5 二分图

二分图：节点由两个集合组成，且两个集合内部没有边的图

性质：一个图是二分图当且仅当图中不含奇数环

5.1 染色法

$O(n+m)$

```

1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  const int N = 1e6 + 10, M = 2e6 + 10;
7  int n, m;
8  int h[N], e[M], ne[M], idx;
9  int color[N];
10 int a, b;
11

```

```

12 void add(int a, int b) {
13     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
14 }
15
16 bool dfs(int u, int c) {
17     color[u] = c;
18     // 将u染成c颜色
19     for (int i = h[u]; i != -1; i = ne[i]) {
20         int j = e[i];
21         if (!color[j]) { // 如果没有染色就染色
22             if (!dfs(j, 3 - c)) return false; // 1->2, 2->1
23         } // 相邻两点颜色不能相同
24         else if (color[j] == c) return false;
25     }
26     return true;
27 }
28
29 int main() {
30     scanf("%d%d", &n, &m);
31     memset(h, -1, sizeof h);
32     while (m --) {
33         scanf("%d%d", &a, &b);
34         add(a, b), add(b, a); // 无向图
35     }
36     bool flag = true;
37     // 遍历所有点, 如果i未染色, 将其所在的连通块染色
38     // 如果一个点染色, 那么这个点所在的连通块的颜色就已确定
39     for (int i = 1; i <= n && flag; ++ i)
40         if (!color[i] && !dfs(i, 1)) flag = false; // 如果产生矛盾则不是二分图
41     puts(flag ? "Yes" : "No");
42 }

```

5.2 匈牙利算法

$O(mn)$, 实际运行时间一般远小于 $O(mn)$ [可能是线性的也说不定]

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4
5  using namespace std;
6  // y总的奇妙比喻 (男生表示左边点, 女生表示右边点)
7  const int N = 510, M = 1e6 + 10;
8  int n1, n2, m;
9  int h[N], e[M], ne[M], idx; // 数组越界可能发生任何错误, 不只是段错误
10 int match[N]; // 记录每个女生配对的男生
11 bool st[N]; // 记录每个女生有没有遍历过
12 int a, b;
13
14 void add(int a, int b) {
15     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
16 }
17
18 bool find(int x) {
19     for (int i = h[x]; i != -1; i = ne[i]) {
20         int j = e[i];

```

```

21     if (!st[j]) {
22         st[j] = true;
23         if (match[j] == 0 || find(match[j])) { // 如果女生没有配对或能匹配别的男生则
匹配
24             match[j] = x;
25             return true;
26         }
27     }
28 }
29 return false;
30 }
31
32 int main() {
33     scanf("%d%d%d", &n1, &n2, &m);
34     memset(h, -1, sizeof h);
35     while (m --) {
36         scanf("%d%d", &a, &b);
37         add(a, b); // 存男生指向女生即可
38     }
39     int res = 0;
40     for (int i = 1; i <= n1; ++ i) {
41         memset(st, false, sizeof st);
42         if (find(i)) res ++;
43     }
44     printf("%d\n", res);
45 }

```

6 SPFA差分约束与判负环

```

1  const int N = 1e5 + 10, M = N * 3;
2  int n, m;
3  int h[N], e[M], ne[M], w[M], idx;
4  int dist[N], cnt[N], q[N];
5  bool st[N];
6
7  void add(int a, int b, int c) {
8      e[idx] = b, ne[idx] = h[a], w[idx] = c, h[a] = idx ++;
9  }
10
11 bool spfa(int s) {
12     memset(dist, 0xcf, sizeof dist);
13     int hh = 0, tt = 1;
14     q[0] = s;
15     dist[s] = 0;
16     st[s] = true;
17
18     while (hh != tt) {
19         int t = q[-- tt];
20         st[t] = false;
21         for (int i = h[t]; ~i; i = ne[i]) {
22             int j = e[i];
23             if (dist[j] < dist[t] + w[i]) {
24                 dist[j] = dist[t] + w[i];
25                 cnt[j] = cnt[t] + 1;
26                 if (cnt[j] >= n + 1) return false;

```

```

27         if (!st[j]) {
28             q[tt++] = j;
29             st[j] = true;
30         }
31     }
32 }
33 }
34
35 return true;
36 }

```

7 LCA

7.1 倍增

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4
5  using namespace std;
6
7  const int N = 5e5 + 10, M = N << 1;
8  int n, m, s;
9  int x, y;
10 int h[N], e[M], ne[M], idx;
11 int d[N], f[N][22], lg[N];
12
13 void add(int a, int b) {
14     e[idx] = b, ne[idx] = h[a], h[a] = idx++;
15 }
16
17 void dfs(int now, int fa) {
18     f[now][0] = fa, d[now] = d[fa] + 1;
19     for (int i = 1; i <= lg[d[now]]; ++i)
20         f[now][i] = f[f[now][i - 1]][i - 1];
21
22     for (int i = h[now]; ~i; i = ne[i]) {
23         int j = e[i];
24         if (j != fa) dfs(j, now);
25     }
26 }
27
28 int lca(int x, int y) {
29     if (d[x] < d[y]) swap(x, y);
30     while (d[x] > d[y])
31         x = f[x][lg[d[x]] - d[y] - 1];
32
33     if (x == y) return x;
34
35     for (int k = lg[d[x]] - 1; k >= 0; --k)
36         if (f[x][k] != f[y][k])
37             x = f[x][k], y = f[y][k];
38     return f[x][0];
39 }
40
41 int main() {

```

```

42     memset(h , -1, sizeof h);
43
44     scanf("%d%d%d", &n, &m, &s);
45
46     for (int i = 1; i <= n - 1; ++ i) {
47         scanf("%d%d", &x, &y);
48         add(x, y), add(y, x);
49     }
50
51     for (int i = 1; i <= n; ++ i)
52         lg[i] = lg[i - 1] + (1 << lg[i - 1] == i);
53
54     dfs(s, 0);
55
56     while (m --) {
57         scanf("%d%d", &x, &y);
58         printf("%d\n", lca(x, y));
59     }
60 }

```

7.2 树链剖分

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5
6  using namespace std;
7
8  const int N = 4e4 + 10, M = N << 1;
9  int h[N], e[M], ne[M], idx;
10 int fa[N], dep[N], siz[N], son[N], top[N], dfn[N], rnk[N], tot;
11 int n, m, s;
12
13 void add(int a, int b) {
14     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
15 }
16
17 void dfs1(int p, int depth, int father) {
18     fa[p] = father;
19     dep[p] = depth;
20     siz[p] = 1;
21     int mmax = -1;
22
23     for (int i = h[p]; ~i; i = ne[i]) {
24         int j = e[i];
25         if (j == father) continue;
26         dfs1(j, depth + 1, p);
27         if (siz[j] > mmax) {
28             mmax = siz[j];
29             son[p] = j;
30         }
31         siz[p] += siz[j];
32     }
33 }
34

```

```

35 void dfs2(int p, int tp) {
36     top[p] = tp;
37     dfn[p] = ++ tot;
38     rnk[tot] = p;
39
40     if (!son[p]) return;
41
42     for (int i = h[p]; ~i; i = ne[i]) {
43         int j = e[i];
44         if (j != fa[p]) {
45             if (j != son[p]) dfs2(j, j);
46             else dfs2(j, tp);
47         }
48     }
49 }
50
51 int lca(int x, int y) {
52     while (top[x] != top[y]) {
53         if (dep[top[x]] >= dep[top[y]]) x = fa[top[x]];
54         else y = fa[top[y]];
55     }
56     return dep[x] < dep[y] ? x : y;
57 }
58
59 int main() {
60     memset(h, -1, sizeof h);
61
62     scanf("%d", &n);
63
64     for (int i = 1; i <= n; ++ i) {
65         int u, v;
66         scanf("%d%d", &u, &v);
67         if (~v) add(u, v), add(v, u);
68         else s = u;
69     }
70
71     dfs1(s, 1, s);
72     dfs2(s, s);
73
74     scanf("%d", &m);
75     while (m --) {
76         int a, b;
77         scanf("%d%d", &a, &b);
78         int Lca = lca(a, b), ans = 0;
79         if (Lca == a) ans = 1;
80         else if (Lca == b) ans = 2;
81         printf("%d\n", ans);
82     }
83 }

```


8 有向图强连通分量SCC

tarjan之后不需要拓扑排序，按照强连通分量逆序即为一个拓扑序

```
1  const int N = 1e4 + 10, M = 5e5 + 10;
2  int n, m;
3  int h[N], e[M], ne[M], idx;
4  int stk[N], top;
5  bool in_stk[N];
6  int dfn[N], low[N], timestamp;
7  int dout[N];
8  int id[N], scc_cnt, siz[N];
9
10 void add(int a, int b) {
11     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
12 }
13
14 void tarjan(int u) {
15     dfn[u] = low[u] = ++ timestamp;
16     stk[++ top] = u, in_stk[u] = true;
17     for (int i = h[u]; ~i; i = ne[i]) {
18         int j = e[i];
19         if (!dfn[j]) {
20             tarjan(j);
21             low[u] = min(low[u], low[j]);
22         }
23         else if (in_stk[j]) low[u] = min(low[u], dfn[j]);
24     }
25
26     if (dfn[u] == low[u]) {
27         ++ scc_cnt;
28         int y;
29         do {
30             y = stk[top --];
31             in_stk[y] = false;
32             id[y] = scc_cnt;
33             siz[scc_cnt] ++;
34         } while (y != u);
35     }
36 }
```

9 无向图的双连通分量

9.1 边双连通分量E-DCC

极大的不含有桥的一个连通区域

```
1  int h[N], e[M], ne[M], idx;
2  int n, m;
3  int dfn[N], low[N], timestamp;
4  int stk[N], top;
5  int id[N], dcc_cnt;
6  bool is_bridge[M];
7
```

```

8 void add(int a, int b) {
9     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
10 }
11
12 void tarjan(int u, int from) {
13     dfn[u] = low[u] = ++ timestamp;
14     stk[++ top] = u;
15
16     for (int i = h[u]; ~i; i = ne[i]) {
17         int j = e[i];
18         if (!dfn[j]) {
19             tarjan(j, i);
20             low[u] = min(low[u], low[j]);
21             if (dfn[u] < low[j])
22                 is_bridge[i] = is_bridge[i ^ 1] = true;
23         } else if (i != (from ^ 1))
24             low[u] = min(low[u], dfn[j]);
25     }
26
27     if (dfn[u] == low[u]) {
28         ++ dcc_cnt;
29         int y;
30         do {
31             y = stk[top --];
32             id[y] = dcc_cnt;
33         } while (u != y);
34     }
35 }

```

9.2 点双连通分量V-DCC

极大的不含有割点的一个连通区域

```

1  const int N = 1010, M = 1010;
2  int h[N], e[M], ne[M], idx;
3  int dfn[N], low[N], timestamp;
4  int n, m, dcc_cnt, root;
5  int stk[N], top;
6  vector<int> dcc[N];
7  bool cut[N]; /*判断i是否为割点*/
8
9  void add(int a, int b) {
10     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
11 }
12
13 void tarjan(int u) {
14     dfn[u] = low[u] = ++ timestamp;
15     stk[++ top] = u;
16
17     if (u == root && h[u] == -1) {
18         dcc[++ dcc_cnt].push_back(u);
19         return;
20     }
21
22     int cnt = 0;
23     for (int i = h[u]; ~i; i = ne[i]) {

```

```

24     int j = e[i];
25     if (!dfn[j]) {
26         tarjan(j);
27         low[u] = min(low[u], low[j]);
28         if (low[j] >= dfn[u]) {
29             ++ cnt;
30             if (u != root || cnt > 1) cut[u] = true;
31             ++ dcc_cnt;
32             int y;
33             do {
34                 y = stk[top --];
35                 dcc[dcc_cnt].push_back(y);
36             } while (y != j);
37             dcc[dcc_cnt].push_back(u);
38         }
39     } else low[u] = min(low[u], dfn[j]);
40 }
41 }

```

10 欧拉回路与欧拉路径

邻接表

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4
5  using namespace std;
6
7  const int N = 1e5 + 10, M = 4e5 + 10;
8  int h[N], e[M], ne[M], idx;
9  int n, m, type, din[N], dout[N];
10 int ans[M >> 1], cnt;
11 bool used[M];
12
13 void add(int a, int b) {
14     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
15 }
16
17 void dfs(int u) {
18     for (int &i = h[u]; ~i;) { /*防止被自环图卡*/
19         if (used[i]) {
20             i = ne[i]; continue;
21         }
22
23         used[i] = true;
24         if (type == 1) used[i ^ 1] = true;
25
26         int t;
27         if (type == 1) {
28             t = i / 2 + 1;
29             if (i & 1) t = -t;
30         } else t = i + 1;
31     }

```

```

32     int j = e[i];
33     i = ne[i];
34     dfs(j);
35
36     ans[++ cnt] = t;
37 }
38 }
39
40 int main() {
41     cin.tie(0)->sync_with_stdio(0);
42     memset(h, -1, sizeof h);
43
44     cin >> type >> n >> m;
45     for (int i = 1; i <= m; ++ i) {
46         int a, b;
47         cin >> a >> b;
48         add(a, b);
49         if (type == 1) add(b, a);
50         ++ dout[a], ++ din[b];
51     }
52
53     if (type == 1) {
54         for (int i = 1; i <= n; ++ i)
55             if (din[i] + dout[i] & 1) {
56                 cout << "NO"; return 0;
57             }
58     }
59     else {
60         for (int i = 1; i <= n; ++ i)
61             if (din[i] != dout[i]) {
62                 cout << "NO"; return 0;
63             }
64     }
65
66     for (int i = 1; i <= n; ++ i)
67         if (h[i] != -1) {
68             dfs(i); break;
69         }
70
71     if (cnt < m) {
72         cout << "NO"; return 0;
73     }
74
75     cout << "YES\n";
76     for (int i = cnt; i; -- i) cout << ans[i] << ' ';
77 }

```

邻接矩阵

```

1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4
5 using namespace std;
6
7 const int N = 510, M = 1110;

```

```

8  int n = 500, m;
9  int g[N][N];
10 int ans[M], cnt;
11 int d[N];
12
13 void dfs(int u) {
14     for (int i = 1; i <= n; ++ i)
15         if (g[u][i]) {
16             -- g[u][i], -- g[i][u];
17             dfs(i);
18         }
19     ans[++ cnt] = u;
20 }
21
22 int main() {
23     cin.tie(0)->sync_with_stdio(0);
24     cin >> m;
25     while (m --) {
26         int a, b;
27         cin >> a >> b;
28         ++ g[a][b], ++ g[b][a];
29         ++ d[a], ++ d[b];
30     }
31
32     int start = 1;
33     for (int i = 1; i <= n; ++ i)
34         if (d[i] && d[i] & 1) {
35             start = i; break;
36         }
37
38     dfs(start);
39
40     for (int i = cnt; i; -- i) cout << ans[i] << '\n';
41 }

```

11 网络流初步

11.1 EK求最大流

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4
5  using namespace std;
6
7  const int N = 1000 + 10, M = 20000 + 10, INF = 1 << 29;
8  int n, m, S, T;
9  int h[N], e[M], ne[M], f[M], idx;
10 int q[N], d[N], pre[N];
11 bool st[N];
12
13 void add(int a, int b, int c) {
14     e[idx] = b, f[idx] = c, ne[idx] = h[a], h[a] = idx ++;
15     e[idx] = a, f[idx] = 0, ne[idx] = h[b], h[b] = idx ++;

```

```

16 }
17
18 bool bfs() {
19     int hh = 0, tt = -1;
20     memset(st, false, sizeof st);
21     q[++ tt] = S;
22     st[S] = true;
23     d[S] = INF;
24
25     while (hh <= tt) {
26         int t = q[hh ++];
27         for (int i = h[t]; ~i; i = ne[i]) {
28             int ver = e[i];
29             if (!st[ver] && f[i]) {
30                 st[ver] = true;
31                 d[ver] = min(d[t], f[i]);
32                 pre[ver] = i;
33                 if (ver == T) return true;
34                 q[++ tt] = ver;
35             }
36         }
37     }
38     return false;
39 }
40
41 int ek() {
42     int maxflow = 0;
43     while (bfs()) {
44         maxflow += d[T];
45         for (int i = T; i != S; i = e[pre[i] ^ 1]) {
46             f[pre[i]] -= d[T];
47             f[pre[i] ^ 1] += d[T];
48         }
49     }
50     return maxflow;
51 }
52
53 int main() {
54     scanf("%d%d%d", &n, &m, &S, &T);
55     memset(h, -1, sizeof h);
56     while (m --) {
57         int a, b, c;
58         scanf("%d%d%d", &a, &b, &c);
59         add(a, b, c);
60     }
61     printf("%d\n", ek());
62 }

```

11.2 dinic求最大流

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 const int N = 1e5 + 10, M = N << 1, INF = 0x3f3f3f3f;
6

```

```

7  int n, m, S, T;
8  int h[N], e[M], ne[M], f[M], idx;
9  int q[N], d[N], cur[N];
10
11 void add(int a, int b, int c) {
12     e[idx] = b, ne[idx] = h[a], f[idx] = c, h[a] = idx ++;
13     e[idx] = a, ne[idx] = h[b], f[idx] = 0, h[b] = idx ++;
14 }
15
16 bool bfs() {
17     int hh = 0, tt = 0;
18     memset(d, -1, sizeof d);
19     q[0] = S, d[S] = 0, cur[S] = h[S];
20     while (hh <= tt) {
21         int t = q[hh ++];
22         for (int i = h[t]; ~i; i = ne[i]) {
23             int ver = e[i];
24             if (d[ver] == -1 && f[i]) {
25                 d[ver] = d[t] + 1;
26                 cur[ver] = h[ver];
27                 if (ver == T) return true;
28                 q[++ tt] = ver;
29             }
30         }
31     }
32     return false;
33 }
34
35 int find(int u, int limit) {
36     if (u == T) return limit;
37     int flow = 0;
38     for (int i = cur[u]; ~i && flow < limit; i = ne[i]) {
39         cur[u] = i;
40         int ver = e[i];
41         if (d[ver] == d[u] + 1 && f[i]) {
42             int t = find(ver, min(f[i], limit - flow));
43             if (!t) d[ver] = -1;
44             f[i] -= t, f[i ^ 1] += t, flow += t;
45         }
46     }
47     return flow;
48 }
49
50 int dinic() {
51     int r = 0, flow;
52     while (bfs()) while (flow = find(S, INF)) r += flow;
53     return r;
54 }
55
56 int main() {
57     scanf("%d%d%d%d", &n, &m, &S, &T);
58     memset(h, -1, sizeof h);
59     while (m --) {
60         int a, b, c;
61         scanf("%d%d%d", &a, &b, &c);
62         add(a, b, c);

```

```

63     }
64     printf("%d\n", dinic());
65     return 0;
66 }

```

11.3 点分裂

```

1  #include <iostream>
2  #include <cstring>
3  #include <algorithm>
4
5  #define debug
6
7  using namespace std;
8
9  const int N = 55 << 1, M = (N * N) << 1, INF = 0x3f3f3f3f;
10 int n, m, S, T;
11 int h[N], e[M], ne[M], f[M], fs[M], idx;
12 bool g[N][N];
13 int q[N], cur[N], d[N];
14
15 void add(int a, int b, int c) {
16     e[idx] = b, ne[idx] = h[a], f[idx] = c, h[a] = idx ++;
17     e[idx] = a, ne[idx] = h[b], f[idx] = 0, h[b] = idx ++;
18 }
19
20 bool bfs() {
21     int hh = 0, tt = 0;
22     memset(d, -1, sizeof d);
23     q[0] = S, d[S] = 0, cur[S] = h[S];
24     while (hh <= tt) {
25         int t = q[hh ++];
26         for (int i = h[t]; ~i; i = ne[i]) {
27             int ver = e[i];
28             if (d[ver] == -1 && f[i]) {
29                 d[ver] = d[t] + 1;
30                 cur[ver] = h[ver];
31                 if (ver == T) return true;
32                 q[++ tt] = ver;
33             }
34         }
35     }
36     return false;
37 }
38
39 int find(int u, int limit) {
40     if (u == T) return limit;
41     int flow = 0;
42     for (int i = cur[u]; ~i && flow < limit; i = ne[i]) {
43         cur[u] = i;
44         int ver = e[i];
45         if (d[ver] == d[u] + 1 && f[i]) {
46             int t = find(ver, min(f[i], limit - flow));
47             if (!t) d[ver] = -1;
48             f[i] -= t, f[i ^ 1] += t, flow += t;
49         }
50     }
51     return flow;
52 }

```



```

50     }
51     return flow;
52 }
53
54 int dinic() {
55     int r = 0, flow;
56     while (bfs()) while (flow = find(S, INF)) r += flow;
57     return r;
58 }
59
60 void init() {
61     memset(h, -1, sizeof h);
62     memset(g, 0, sizeof g);
63     memset(f, 0, sizeof f);
64     idx = 0;
65 }
66
67 int main() {
68     while (scanf("%d%d", &n, &m) != EOF) {
69         init();
70
71         for (int i = 1; i <= m; ++ i) {
72             int a, b;
73             scanf(" (%d,%d)", &a, &b);
74             a ++, b ++;
75             add(a + n, b, INF);
76             add(b + n, a, INF);
77             g[a][b] = g[b][a] = true;
78         }
79         for (int i = 1; i <= n; ++ i) add(i, i + n, 1);
80         memcpy(fs, f, sizeof f);
81
82         int res = n;
83         for (S = n + 1; S <= 2 * n; ++ S)
84             for (T = S - n + 1; T <= n; ++ T)
85                 if (!g[S - n][T]) {
86                     memcpy(f, fs, sizeof f);
87                     res = min(res, dinic());
88                 }
89
90         printf("%d\n", res);
91     }
92 }
93 ```# 数学知识
94
95 ## 数论
96
97 ### 试除法判定质数
98
99 ```c++
100 #include <iostream>
101
102 using namespace std;
103
104 int n, a;
105

```

```

106 bool is_prime(int n) {
107     if (n <= 1) return false;
108     for (int i = 2; i <= n / i; ++ i) // 此处不要写成i*i<=n或者i<=sqrt(n)
109         if (n % i == 0) return false; // 前者会溢出后者较慢
110     return true;
111 }
112
113 int main() {
114     scanf("%d", &n);
115     while (n --) {
116         scanf("%d", &a);
117         puts(is_prime(a) ? "Yes" : "No");
118     }
119 }

```

11.4 分解质因数

```

1  #include <iostream>
2
3  using namespace std;
4
5  int n, a;
6
7  void divide(int n) {
8      for (int i = 2; i <= n / i; ++ i) {
9          if (n % i == 0) {
10             int s = 0;
11             while (n % i == 0) n /= i, s ++;
12             printf("%d %d\n", i, s);
13         }
14     } // n至多有一个大于根号n的质因子
15     if (n > 1) printf("%d 1\n", n);
16     puts("");
17 }
18
19 int main() {
20     scanf("%d", &n);
21     while (n --) {
22         scanf("%d", &a);
23         divide(a);
24     }
25 }

```

11.5 筛质数

```

1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 1e6 + 10;
6  int n;
7  int prime[N], cnt;
8  bool st[N];
9
10 void get_primes(int n) {
11     for (int i = 2; i <= n; ++ i) {

```

```

12         if (!st[i]) prime[cnt++] = i;
13         for (int j = 0; prime[j] <= n / i; ++j) {
14             st[prime[j] * i] = true;
15             if (i % prime[j] == 0) break;
16         }
17     }
18 } // 一个数只用其最小质因数筛掉
19
20 int main() {
21     scanf("%d", &n);
22     get_primes(n);
23     printf("%d\n", cnt);
24 }

```

11.6 试除法求约数

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  int n, a;
8
9  vector<int> get_divisors(int n) {
10     vector<int> res;
11     for (int i = 1; i <= n / i; ++i) {
12         if (n % i == 0) {
13             res.push_back(i);
14             if (i != n / i) res.push_back(n / i);
15         }
16     }
17     sort(res.begin(), res.end());
18     return res;
19 }
20
21 int main() {
22     scanf("%d", &n);
23     while (n --) {
24         scanf("%d", &a);
25         auto res = get_divisors(a);
26         for (auto i : res) printf("%d ", i);
27         puts("");
28     }
29 }

```

11.7 约数个数

```

1  #include <iostream>
2  #include <unordered_map>
3
4  using namespace std;
5
6  typedef long long LL;
7
8  const int MOD = 1e9 + 7;

```

```

9  int n, a;
10 unordered_map<int, int> primes;
11 LL res = 1;
12
13 int main() {
14     scanf("%d", &n);
15     while (n --) {
16         scanf("%d", &a);
17         for (int i = 2; i <= a / i; ++ i)
18             while (a % i == 0) a /= i, primes[i] ++;
19         if (a > 1) primes[a] ++;
20     }
21     for (auto i : primes) res = res * (i.second + 1) % MOD; // 约数个数其实就是质因数的
组合数
22     printf("%lld\n", res);
23 }

```

11.8 约数之和

```

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

```

11.9 最大公约数

```

1  #include <iostream>
2
3  using namespace std;
4
5  int n, a, b;
6

```

```

7  int gcd(int a, int b) {
8      return b ? gcd(b, a % b) : a;
9  }
10
11 int main() {
12     scanf("%d", &n);
13     while (n --) {
14         scanf("%d%d", &a, &b);
15         printf("%d\n", gcd(a, b));
16     }
17 }

```

12 快速幂

```

1  int q_pow(int a, int b, int c) {
2      int ans = 1;
3      while (b) {
4          if (b & 1) ans = (LL)ans * a % c;
5          a = (LL)a * a % c;
6          b >>= 1;
7      }
8      return ans;
9  }

```

12.1 快速幂求逆元

```

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

```

12.2 线性逆元

```
1  #include <iostream>
2  #include <cstdio>
3
4  using namespace std;
5
6  typedef long long LL;
7
8  const int N = 20000528 + 10;
9  int n, p;
10 int inv[N];
11
12 int main() {
13     scanf("%d%d", &n, &p);
14     inv[1] = 1;
15     for (int i = 2; i <= n; ++ i)
16         inv[i] = (LL)(p - p / i) * inv[p % i] % p;
17     for (int i = 1; i <= n; ++ i)
18         printf("%d\n", inv[i]);
19     return 0;
20 }
```

13 拓展欧几里得

```
1  int exgcd(int a, int b, int &x, int &y) {
2      if (!b) {
3          x = 1, y = 0;
4          return a;
5      }
6      int d = exgcd(b, a % b, y, x);
7      y -= a / b * x;
8      return d;
9  }
```

13.1 线性同余方程

```
1  #include <iostream>
2
3  using namespace std;
4
5  int n, a, b, m, x, y;
6
7  int exgcd(int a, int b, int &x, int &y) {
8      if (!b) {
9          x = 1, y = 0;
10         return a;
11     }
12     int d = exgcd(b, a % b, y, x);
13     y -= a / b * x;
14     return d;
15 }
16
17 int main() {
18     scanf("%d", &n);
19     while (n --) {
```

```

20     scanf("%d%d%d", &a, &b, &m);
21     int d = exgcd(a, m, x, y);
22     if (b % d == 0) printf("%d\n", x * 111 * b / d % m);
23     else printf("impossible\n");
24 }
25 }

```

14 中国剩余定理

1

```

1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  int n;
8  bool has_answer = true;
9
10 LL exgcd(LL a, LL b, LL &x, LL &y) {
11     if (!b) {
12         x = 1, y = 0;
13         return a;
14     }
15     int d = exgcd(b, a % b, y, x);
16     y -= a / b * x;
17     return d;
18 }
19
20 int main() {
21     cin >> n;
22     LL a1, m1;
23     cin >> a1 >> m1;
24
25     for (int i = 1; i < n; ++i) {
26         LL a2, m2;
27         cin >> a2 >> m2;
28
29         LL k1, k2;
30         LL d = exgcd(a1, a2, k1, k2);
31
32         if ((m2 - m1) % d) {
33             has_answer = false;
34             break;
35         }
36
37         k1 *= (m2 - m1) / d;
38         LL t = a2 / d;
39         k1 = (k1 % t + t) % t;
40
41         m1 = a1 * k1 + m1;
42         a1 = abs(a1 / d * a2);
43     }

```

```

44 |
45 |     cout << (has_answer ? (m1 % a1 + a1) % a1 : -1) << endl;
46 | }

```

2

```

1 | #include <iostream>
2 | #include <cmath>
3 |
4 | using namespace std;
5 |
6 | typedef long long LL;
7 |
8 | int n, p, e, i, d;
9 | LL a[4]; int w[] = {0, 23, 28, 33};
10 |
11 | LL exgcd(LL a, LL b, LL &x, LL &y) {
12 |     if (!b) {
13 |         x = 1, y = 0;
14 |         return a;
15 |     }
16 |     LL d = exgcd(b, a % b, y, x);
17 |     y -= a / b * x;
18 |     return d;
19 | }
20 |
21 | LL ChRe() {
22 |     int n = 1;
23 |     for (int i = 1; i <= 3; ++ i) a[i] %= w[i], n *= w[i];
24 |     LL ans = 0;
25 |     for (int i = 1; i <= 3; ++ i) {
26 |         int m = n / w[i];
27 |         LL x, y;
28 |         exgcd(m, w[i], x, y);
29 |         ans = (ans + m * x * a[i]) % n;
30 |     }
31 |     ans = (ans % n + n) % n;
32 |     ans -= d;
33 |     while (ans <= 0) ans += n;
34 |     return ans;
35 | }
36 |
37 | int main() {
38 |     scanf("%lld", &n);
39 |
40 |     while (n --) {
41 |         LL Case = 0;
42 |         while (scanf("%lld%lld%lld%lld", &a[1], &a[2], &a[3], &d) != EOF) {
43 |             if (a[1] == -1 && a[2] == -1 && a[3] == -1 && d == -1) break;
44 |             LL ans = ChRe();
45 |             printf("Case %lld: the next triple peak occurs in %lld days.\n", ++ Case,
ans);
46 |         }
47 |     }
48 | }

```


15 组合计数

公式: (a 选 b)

$$C_a^b = \frac{a!}{(a-b)!b!}$$

递推式:

$$\begin{aligned} C_a^b &= C_a^{b-1} + C_{a-1}^{b-1} \\ C_a^0 &= 1 \\ C_a^b &= 0 (a < b) \\ \sum_{i=0}^n C_i^n &= 2^n \end{aligned}$$

性质:

$$C_m^n = C_m^{m-n}$$

数据范围较小时: $a, b \leq 2e3$

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 2e3 + 10, MOD = 1e9 + 7;
6  int c[N][N];
7  int n, a, b;
8
9  void init() {
10     for (int i = 0; i < N; ++ i)
11         for (int j = 0; j <= i; ++ j)
12             if (!j) c[i][j] = 1;
13             else c[i][j] = (c[i - 1][j] + c[i - 1][j - 1]) % MOD;
14 }
15
16 int main() {
17     init();
18     scanf("%d", &n);
19     while (n --) {
20         scanf("%d%d", &a, &b);
21         printf("%d\n", c[a][b]);
22     }
23 }
```

a,b较大时: $a, b \leq 1e7$

```
1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  const int N = 1e6 + 10, MOD = 1e9 + 7;
8  int fact[N], infact[N];
```

```

9  int n, a, b;
10
11 int q_pow(int a, int b) {
12     int res = 1;
13     while (b) {
14         if (b & 1) res = (LL)res * a % MOD;
15         a = (LL)a * a % MOD;
16         b >>= 1;
17     }
18     return res;
19 }
20
21 int C(int a, int b) {
22     return (LL)fact[a] * infact[b] % MOD * infact[a - b] % MOD;
23 }
24
25 int main() {
26     fact[0] = infact[0] = 1;
27     for (int i = 1; i < N; ++ i) {
28         fact[i] = (LL)fact[i - 1] * i % MOD;
29         infact[i] = (LL)infact[i - 1] * q_pow(i, MOD - 2) % MOD;
30     }
31
32     scanf("%d", &n);
33     while (n --) {
34         scanf("%d%d", &a, &b);
35         printf("%d\n", C(a, b));
36     }
37 }

```

a,b 巨大时: $a, b \leq 1e18$ $p \leq 1e5$

```

1  #include <iostream> // 卢卡斯定理
2
3  using namespace std;
4
5  typedef long long LL;
6  int n, p;
7  LL a, b;
8
9  int q_pow(int a, int b, int p) {
10     int res = 1;
11     while (b) {
12         if (b & 1) res = (LL)res * a % p;
13         a = (LL)a * a % p;
14         b >>= 1;
15     }
16     return res;
17 }
18
19 int C(int a, int b, int p) {
20     if (b > a) return 0;
21
22     int res = 1;
23     for (int i = 1, j = a; i <= b; ++ i, -- j) {
24         res = (LL)res * j % p;

```

```

25     res = (LL)res * q_pow(i, p - 2, p) % p;
26 }
27 return res;
28 }
29
30 int lucas(LL a, LL b, int p) {
31     if (a < p && b < p) return C(a, b, p);
32     return (LL)C(a % p, b % p, p) * lucas(a / p, b / p, p) % p;
33 }
34
35 int main() {
36     scanf("%d", &n);
37     while (n --) {
38         scanf("%lld%lld%d", &a, &b, &p);
39         printf("%d\n", lucas(a, b, p));
40     }
41 }

```

朴素做法：高精

```

1  #include <vector>
2  #include <iostream>
3
4  using namespace std;
5
6  const int N = 5010;
7  int primes[N], cnt;
8  bool st[N];
9  int sum[N];
10 int a, b;
11 vector<int> res = {1};
12
13 void get_primes(int n) {
14     for (int i = 2; i <= n; ++ i) {
15         if (!st[i]) primes[cnt ++] = i;
16         for (int j = 0; primes[j] <= n / i; ++ j) {
17             st[primes[j] * i] = true;
18             if (i % primes[j] == 0) break;
19         }
20     }
21 }
22
23 int get(int n, int p) {
24     int res = 0;
25     while (n) {
26         res += n / p;
27         n /= p;
28     }
29     return res;
30 }
31
32 vector<int> mul(vector<int> &a, int b) {
33     vector<int> c;
34     int t = 0;
35     for (int i = 0; i < a.size() || t; ++ i) {
36         if (i < a.size()) t += a[i] * b;

```

```

37         c.push_back(t % 10);
38         t /= 10;
39     }
40     while (c.size() > 1 && c.back() == 0) c.pop_back();
41     return c;
42 }
43
44 int main() {
45     scanf("%d%d", &a, &b);
46     get_primes(a);
47     for (int i = 0; i < cnt; ++ i) {
48         int p = primes[i];
49         sum[i] = get(a, p) - get(a - b, p) - get(b, p);
50     }
51
52     for (int i = 0; i < cnt; ++ i)
53         for (int j = 0; j < sum[i]; ++ j)
54             res = mul(res, primes[i]);
55
56     for (int i = res.size() - 1; i >= 0; -- i) printf("%d", res[i]);
57     puts("");
58 }

```

卡特兰数

$$\begin{aligned}
 ktl_n &= \sum_{i=0}^{n-1} ktl_i * ktl_{n-1-i} \\
 ktl_1 &= ktl_0 = 1 \\
 ktl_n &= \frac{C_{2n}^n}{n+1}
 \end{aligned}$$

```

1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  const int MOD = 1e9 + 7;
8
9  int q_pow(int a, int b) {
10     int res = 1;
11     while (b) {
12         if (b & 1) res = (LL)res * a % MOD;
13         a = (LL)a * a % MOD;
14         b >>= 1;
15     }
16     return res;
17 }
18
19 int main() {
20     int n;
21     scanf("%d", &n);
22
23     int a = 2 * n, b = n;
24     int res = 1;
25

```

```

26     for (int i = a; i > a - b; -- i) res = (LL)res * i % MOD;
27     for (int i = 1; i <= b; ++ i) res = (LL)res * q_pow(i, MOD - 2) % MOD;
28     res = (LL)res * q_pow(n + 1, MOD - 2) % MOD;
29
30     printf("%d\n", res);
31 }

```

16 高斯消元

```

1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  const int N = 110;
7  const double eps = 1e-6;
8  int n;
9  double a[N][N];
10
11 bool lf_is_same(double a, double b) {
12     return fabs(a - b) < eps;
13 }
14
15 int gauss() {
16     int c, r; // c--列 r--行
17     for (c = 0, r = 0; c < n; ++ c) {
18         int t = r; // t--绝对值最大的行
19         for (int i = r; i < n; ++ i)
20             if (fabs(a[i][c]) > fabs(a[t][c])) t = i;
21         // 如果是0就continue
22         if (lf_is_same(a[t][c], 0)) continue;
23         // 将绝对值最大行放在最上方
24         for (int i = c; i < n + 1; ++ i) swap(a[t][i], a[r][i]);
25         for (int i = n; i >= c; -- i) a[r][i] /= a[r][c]; // 将改行第一个系数变为1
26         // 将下面所有行的第一个系数变为0
27         for (int i = r + 1; i < n; ++ i)
28             if (!lf_is_same(a[i][c], 0))
29                 for (int j = n; j >= c; -- j) a[i][j] -= a[r][j] * a[i][c];
30         // 行数++
31         r ++;
32     }
33     // 如果行数<n
34     if (r < n) {
35         for (int i = r; i < n; ++ i)
36             if (!lf_is_same(0, a[i][n])) return 2; // 如果出现0=非0则无解
37         return 1; // 如果出现0=0
38     }
39
40     for (int i = n - 1; i >= 0; -- i)
41         for (int j = i + 1; j < n; ++ j) a[i][n] -= a[j][n] * a[i][j]; // 将a[i][n]转
42     // 换成方程的解
43     return 0;
44 }

```

```

45 int main() {
46     scanf("%d", &n);
47     for (int i = 0; i < n; ++ i)
48         for (int j = 0; j <= n; ++ j)
49             scanf("%lf", &a[i][j]);
50
51     int t = gauss();
52     if (!t) for (int i = 0; i < n; ++ i) printf("%.2lf\n", a[i][n]);
53     else if (t == 1) puts("Infinite group solutions");
54     else puts("No solution");
55 }

```

17 简单博弈论

[Nim游戏](#)

[台阶-Nim游戏](#)

[集合-Nim游戏](#)

[拆分-Nim游戏](#)

18 容斥原理

```

1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  const int N = 20;
8  int n, m;
9  int p[N];
10
11 int main() {
12     scanf("%d%d", &n, &m);
13     for (int i = 0; i < m; ++ i) scanf("%d", &p[i]);
14
15     int res = 0;
16     for (int i = 1; i < 1 << m; ++ i) { // 不能从i=0开始因为一定要选一个集合
17         int t = 1, s = 0; // t表示所选质数乘积 s表示所选集合个数
18         for (int j = 0; j < m; ++ j)
19             if (i >> j & 1) {
20                 s ++;
21                 if ((LL)t * p[j] > n) {
22                     t = -1; break;
23                 }
24                 t *= p[j];
25             }
26         if (t == -1) continue;
27         // 奇数个+ 偶数个-
28         if (s & 1) res += n / t;

```

```

29         else res -= n / t;
30     }
31
32     printf("%d\n", res);
33 }

```

19 拓展欧拉定理

[P5091 【模板】扩展欧拉定理](#)

```

1  #include <iostream>
2
3  using namespace std;
4
5  typedef long long LL;
6
7  const int N = 20000000 + 10;
8  char b[N];
9
10 LL mul(LL a, LL b, LL m) { // 龟速乘
11     LL res = 0;
12     while (b) {
13         if (b & 1) res = (res + a) % m;
14         a = (a + a) % m;
15         b >>= 1;
16     }
17     return res;
18 }
19
20 LL q_pow(LL a, LL b, LL MOD) {
21     LL res = 1;
22     while (b) {
23         if (b & 1) res = mul(res, a, MOD);
24         a = mul(a, a, MOD);
25         b >>= 1;
26     }
27     return res;
28 }
29
30 LL phi(LL x) {
31     LL res = x;
32     for (LL i = 2; i <= x / i; i++)
33         if (x % i == 0)
34             {
35                 res = res / i * (i - 1);
36                 while (x % i == 0) x /= i;
37             }
38     if (x > 1) res = res / x * (x - 1);
39
40     return res;
41 }
42
43 int main() {
44     LL a, m; bool falg = false;
45     cin >> a >> m >> b + 1;
46     LL p = phi(m);

```

```
47     LL ans = 0;
48     for (LL i = 1; b[i]; ++ i) {
49         ans = ans * 10 + b[i] - '0';
50         if (ans >= p) {
51             falg = true;
52             ans %= p;
53         }
54     }
55     if (falg) ans += p;
56     cout << q_pow(a, ans, m) << endl;
57 }
```