

Contents

1 Basic

- 1.1 Template
- 1.2 .vimrc
- 1.3 Basic cmd
- 1.4 Standard Library

2 Graph

- 2.1 Dijkstra
- 2.2 Kruskal 最小生成樹
- 2.3 Floyd Warshell 任意點最短路
- 2.4 Floyd Warshell 有向圖最小環
- 2.5 SPFA
- 2.6 SPFA 找負環
- 2.7 拓樸排序
- 2.8 LCA 樹的最短路/找共同祖先
- 2.9 Hungarian 二分圖匹配
- 2.10 樹上兩點最遠距離
- 2.11 Max Flow
- 2.12 Max Flow Minimum Cut
- 2.13 尤拉路徑

3 Math

- 3.1 尤拉函數線上
- 3.2 尤拉函數建表
- 3.3 Fibonacci 線上
- 3.4 質數
- 3.5 質數 linear
- 3.6 模逆元
- 3.7 擴展歐基里德
- 3.8 中國剩餘定理-韓信點兵

4 Dynamic Programming

- 4.1 LIS
- 4.2 LCS
- 4.3 Minimum Coin Change
- 4.4 Coin Change
- 4.5 2D Maximum SubArray

5 Data Structure

- 5.1 Big Number
- 5.2 Disjoin Set
- 5.3 Segment Tree

6 Others

- 6.1 Roman to Int
- 6.2 misc

7 Recent update

- 7.1 上下樓梯
- 7.2 組合數
- 7.3 遞增減可能
- 7.4 dijk 第二短路
- 7.5 物理
- 7.6 $4n+1$ prime
- 7.7 八皇后
- 7.8 線性質數表
- 7.9 二進制表示 + Dijkstra
- 7.10 Flow 例題

1 Basic

1.1 Template

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5 #define PB push_back
6 #define PII pair<int, int>
7 #define MP make_pair
8 #define all(x) x.begin(), x.end()
9 #define REP(x, y, z) for(int x = y; x <= z; x++)
10 #define REPP(x, y, z) for(int x = y; x >= z; x--)
11 #define MSET(x, y) memset(x, y, sizeof(x))
12 #define F first
13 #define S second
14 #define maxn
15
16 //structure
17
18 //declaration

```

```

19
20 //functions
21
22 int main(void)
23 {
24
25     return 0;
26 }

```

1.2 .vimrc

```

31 syntax on
42 color torte
43 set nu ts=4 sw=4 ai mouse=a bs=2 ci hls ru nocp
5     showmatch ar
54 hi cursorline cterm=none ctermbg=darkred
65 filetype plugin indent on
76
77 so $VIMRUNTIME/mswin.vim
78 behave mswin
79
80

```

1.3 Basic cmd

```

91 lpr A.cpp # 列印
92 lpq # 查詢列印 queue
103
104 g++ A.cpp -std=gnu++14 -O2 -Wall -Wshadow -o A
105 ./A
106 ./A < input
107 ./A < input > output
108 diff output ans
109
110

```

1.4 Standard Library

2 Graph

2.1 Dijkstra

```

161 #include <bits/stdc++.h>
162 using namespace std;
163
164 #define MP make_pair
165 #define PII pair<int, int>
166 #define maxn 50000+5
167
168 int dis[maxn]; // 預設都是 INF
169 vector<PII> e[maxn]; // (連到的點, 邊的距離)
170
171 void dijk(int cur) // dijk(起點)
172 {
173     int d;
174     priority_queue<PII, vector<PII>, greater<PII>> q; // 放
175     // (距離, 點編號), 每次會拿距離最小的點出來
176     q.push( MP(0, cur) );
177
178     while (!q.empty())
179     {
180         tie(d, cur) = q.top();
181         q.pop();
182         if (dis[cur] != 1e9)
183             continue; // 如果之前就拜訪過, 無視
184
185         dis[cur] = d;
186
187         for (auto i: e[cur])
188             if (dis[i.first] == 1e9)
189             {
190                 q.push( MP(d+i.second, i.first) );
191             }
192     }
193 }

```

```

31     }
32 }
33
34 void init(void)
35 {
36     fill(dis, dis+maxn, 1e9);
37
38     for(int i = 0; i < maxn; i++)
39     {
40         e[i].clear();
41     }
42 }

```

2.2 Kruskal 最小生成樹

```

1 //kruskal algorithm
2 //minimum spanning tree
3
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 #define maxn
8
9 struct Edge
10 {
11     int from, to, cost;
12
13     Edge(int _from, int _to, int _cost)
14     {
15         from = _from;
16         to = _to;
17         cost = _cost;
18     }
19
20     bool operator< (const Edge &r) const
21     {
22         return cost < r.cost;
23     }
24 };
25
26 int parent_arr[maxn];
27 int n, m, cost;
28 vector<Edge> edges;
29
30 int find(int x)
31 {
32     return parent_arr[x] < 0 ? x : (parent_arr[
33         parent_arr[x]]);
34 }
35
36 void conn(int x,int y)
37 {
38     parent_arr[find(y)]=find(x);
39 }
40
41 void kruskal_algorithm(void)
42 {
43     cost = 0;
44     memset(parent_arr, -1, sizeof(parent_arr));
45     sort(edges.begin(), edges.end());
46
47     for(int i = 0; i < m; i++)
48     {
49         Edge tmp = edges[i];
50
51         if(find(tmp.to) == find(tmp.from))
52         {
53             //不能形成環的邊
54             continue;
55         }
56
57         else
58         {
59             cost += tmp.cost;
60             conn(tmp.from, tmp.to);
61         }
62     }
63 }

```

```

60     }
61 }
62 }
63 }
64
65 int main(void)
66 {
67     while(cin >> n >> m)
68     {
69         //init
70         edges.clear();
71         cost = 0;
72
73         for(int i = 0; i < m; i++)
74         {
75             int a, b, c;
76             cin >> a >> b >> c;
77             edges.push_back(Edge(a, b, c));
78         }
79
80         kruskal_algorithm();
81
82         cout << cost << '\n';
83     }
84     return 0;
85 }

```

2.3 Floyd Warshell 任意點最短路

```

1  for (int k = 1; k <= n; k++)
2      for (int i = 1; i <= n; i++)
3          for (int j = 1; j <= n; j++)
4              if (dis[i][j] > dis[i][k] + dis[k][j])
5                  {
6                      // 如果可以以 k 為中繼點，更新 i, j 的最短距離
7                      dis[i][j] = dis[i][k] + dis[k][j];
8                  }

```

2.4 Floyd Warshell 有向圖最小環

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define maxn 100+5
5
6 int n;
7 int ans;
8 int dis[maxn][maxn];
9
10 int main(void)
11 {
12     while(cin >> n)
13     {
14         ans = 1e9;
15         for(int i = 1; i <= n; i++)
16         {
17             for(int j = 1; j <= n; j++)
18             {
19                 cin >> dis[i][j];
20                 if(!dis[i][j]) dis[i][j] = 1e9;
21             }
22         }
23
24         for (int k = 1; k <= n; k++)
25             for (int i = 1; i <= n; i++)
26                 for (int j = 1; j <= n; j++)
27                 {
28                     if (dis[i][j] > dis[i][k] + dis[k][j])
29                     {
30                         // 如果可以以 k 為中繼點，更新 i,j 的最短距離
31                         dis[i][j] = dis[i][k] + dis[k][j];
32                     }
33                 }
34     }
35     cout << ans << endl;
36     return 0;
37 }

```

```

32     }
33
34     if(i == j)
35         ans = min(ans, dis[i][j]);
36     }
37
38     if(ans == 1e9)
39         cout << -1 << '\n';
40     else
41         cout << ans << '\n';
42 }
43 return 0;
44 }

```

2.5 SPFA

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define MP make_pair
5 #define PII pair<int, int>
6 #define maxn 500+5
7
8 const int INF = 1e9; //比最大可能的距離更大
9
10 bool inq[maxn]; // inq[i] 代表 i 在 queue 裡面
11 int dis[maxn]; // 預設都是 INF
12 vector<PII> e[maxn]; // (連到的點, 邊的距離)
13
14 void spfa(int cur)
15 {
16     queue<int> q;
17     dis[cur] = 0;
18     q.push(cur);
19
20     while (!q.empty())
21     {
22         cur = q.front();
23         q.pop();
24         inq[cur] = false;
25
26         for (auto i: e[cur])
27         {
28             // 如果點 cur, 經過權重 i.S 這條邊, 走到 i.F 可
29             // 以更短, 就更新
30             if (i.second + dis[cur] < dis[i.first])
31             {
32                 dis[i.first] = dis[cur] + i.second;
33                 if (!inq[i.first])
34                 {
35                     inq[i.first] = true;
36                     q.push(i.first);
37                 }
38             }
39         }
40     }
41 }
42
43 void init(void)
44 {
45     fill(dis, dis+maxn, INF);
46     for(int i = 0; i < maxn; i++)
47     {
48         e[i].clear();
49     }
50     memset(inq, false, sizeof(inq));
51 }

```

2.6 SPFA 找負環

```

1 #include <bits/stdc++.h>
2 using namespace std;

```

```

3
4 #define MP make_pair
5 #define PII pair<int, int>
6 #define maxn 500+5
7
8 const int INF = 1e9; //比最大可能的距離更大
9
10 bool inq[maxn]; // inq[i] 代表 i 在 queue 裡面
11 int dis[maxn]; // 預設都是 INF
12 int updateCount[maxn];
13 int vis[maxn];
14 vector<PII> e[maxn]; // (連到的點, 邊的距離)
15 int n, m;
16
17 void spfa(int cur)
18 {
19     queue<int> q;
20     dis[cur] = 0;
21     q.push(cur);
22
23     while (!q.empty())
24     {
25         cur = q.front();
26         q.pop();
27         inq[cur] = false;
28
29         for (auto i: e[cur])
30         {
31             // 如果點 cur, 經過權重 i.S 這條邊, 走到 i.F 可
32             // 以更短, 就更新
33             if (i.second + dis[cur] < dis[i.first])
34             {
35                 dis[i.first] = dis[cur] + i.second;
36                 if (!inq[i.first])
37                 {
38                     // updateCount 紀錄一個點被放到 queue 幾
39                     // 次
40                     updateCount[i.first]++;
41                     if(updateCount[i.first] > n)
42                     {
43                         continue;
44                     }
45                     inq[i.first] = true;
46                     q.push(i.first);
47                 }
48             }
49         }
50     }
51 }
52
53 void init(void)
54 {
55     fill(dis, dis+maxn, INF);
56     for(int i = 0; i < maxn; i++)
57     {
58         e[i].clear();
59     }
60     memset(updateCount, 0, sizeof(updateCount));
61     memset(inq, false, sizeof(inq));
62 }
63
64 bool dfs(int cur)
65 {
66     vis[cur]=true;
67     if(cur==n)return true;
68
69     for(int i = 0; i < e[cur].size(); i++)
70     {
71         if(!vis[e[cur][i].first])
72             if(dfs(e[cur][i].first))
73                 return true;
74     }
75     return false;
76 }
77
78 bool check()
79 {
80     memset(vis, false, sizeof(vis));
81 }

```

```

77     for(int i = 1; i <= n; i++)
78         if(updateCount[i]>n && dfs(i))
79             return true;
80     return false;
81 }
82
83 int main(void)
84 {
85     int x, y, z;
86     while(cin >> n >> m)
87     {
88         init();
89
90         for(int i = 0; i < m; i++)
91         {
92             cin >> x >> y >> z;
93             e[x].push_back(MP(y, z));
94         }
95
96         spfa(1);
97
98         if(dis[n]!=INF && !check())
99             cout << dis[n] << '\n';
100         else
101             cout << "There a negative cycle or no path\n";
102     }
103     return 0;
104 }

```

2.7 拓樸排序

```

1 #include <bits/stdc++.h>
2 #define maxn 50005
3 using namespace std;
4 struct edge
5 {
6     int t,next;
7 } in[maxn*4];
8 //n vertex has n*4 maximum edges
9
10 int n,m,e,first[maxn],s[maxn],top;
11 // first 紀錄是否有固定順序
12 // s 紀錄順序
13
14 bool fail,ins[maxn],vis[maxn];
15 // vis 是否訪問
16 // ins 在做dfs的當下 那點是否被訪問過
17
18 void add(int x,int y)
19 {
20     in[e].t=y;
21     in[e].next=first[x];
22     first[x]=e++;
23 }
24 void dfs(int cur)
25 {
26     ins[cur]=vis[cur]=true;
27     for(int i=first[cur]; ~i; i=in[i].next)
28     {
29         if(!vis[in[i].t])
30             dfs(in[i].t);
31         else if(ins[in[i].t])
32             fail=true;
33     }
34     ins[cur]=false;
35     s[top++]=cur;
36 }
37 int main(void)
38 {
39     int x,y;
40     while(cin >> n >> m)
41     {
42         //init
43         e = 0;
44         top = 0;

```

```

45     fail = false;
46     memset(first, -1, sizeof(first));
47     memset(ins, false, sizeof(ins));
48     memset(vis, false, sizeof(vis));
49
50     for(int i = 1; i <= m; i++)
51     {
52         scanf("%d %d",&x,&y);
53         add(x,y);
54     }
55
56     for(int i = 1; i <= n; i++)
57         if(!vis[i])
58             dfs(i);
59
60     if(fail)
61         puts("-1");
62     else
63         for(int i = top-1; i >= 0; i--)
64             printf("%d\n",s[i]);
65 }
66 return 0;
67 }

```

2.8 LCA 樹的最短路/找共同祖先

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define maxn
5 #define LG //LG = Log2n
6 #define PB push_back
7 #define MP make_pair
8
9 int f[LG][maxn];
10 int dep[maxn]; // dep[i] 是點 i 的深度, root 深度是 0,
11 // 下一層的深度是 1...
12 int depw[maxn];
13 int n, m;
14 // if no weight
15 // int e[maxn];
16
17 //if the edge with weight
18 vector< pair<int, int> > e[maxn];
19
20 void dfs(int cur,int fa) { // 多帶一個父節點的參數, 是
21 // 在樹上 dfs 常見的技巧, 可以省去平常 dfs 需要的 vis
22 // 陣列
23     f[0][cur] = fa;
24     for (auto i: e[cur]) if (i != fa) {
25         dep[i] = dep[cur]+1;
26         dfs(i, cur);
27     }
28 }
29
30 int lca(int x,int y) {
31     // 跟 swap(x,y) 是一樣的意思
32     if (dep[x] < dep[y]) return lca(y,x);
33     // 這裡開始 dep[x] >= dep[y] 一定成立
34
35     for (int i=LG-1; i>=0; i--)
36         if (dep[x]-(1<<i) >= dep[y]) // 先想辦法把 x,y
37             // 調到同深度
38             x = f[i][x];
39     if (x==y) return x; // 如果發現同深度時, 是同一
40     // 個點就回傳找到 LCA 了
41
42     // 否則盡量想辦法往上走, 只要 x,y 同時往上走 2^i 步
43     // 還不是相同的點, 就 greedy 走
44     for (int i=LG-1; i>=0; i--)
45         if (f[i][x] != f[i][y])
46             x = f[i][x];

```

```

43     y = f[i][y];
44 }
45 assert(f[0][x] == f[0][y]); // 走完以後，會發現 x,y
    停在 lca 的正下方一個點
46 return f[0][x];
47 }
48 void make_lca() {
49     dep[1] = depw[1] = 0;
50     dfs(1, 1); // 拿 1 當 root，且 1 的父節點是 1
51     for (int i=1; i<LG; i++)
52         for (int j=1; j<=n; j++)
53             f[i][j] = f[i-1][f[i-1][j]]; // j 往上走 2^(i-1)
    再往上走 2^(i-1) = 往上走 2^i 步
54 }
55
56 int main(void)
57 {
58     while(cin >> n >> m)
59     {
60         //init
61         for(int i = 0; i < maxn; i++)
62             e[i].clear();
63
64         int x, y, z;
65
66         for(int i = 0; i < n-1; i++)
67         {
68             // if no weight
69             // cin >> x >> y;
70             // e[x].PB(y);
71             // e[y].PB(x);
72
73             cin >> x >> y >> z;
74             e[x].PB(MP(y, z));
75             e[y].PB(MP(x, z));
76         }
77
78         //make LCA
79         make_lca();
80
81         for(int i = 0; i < m; i++)
82         {
83             cin >> x >> y;
84             cout << dep[x]+dep[y]-2*dep[lca(x, y)] << '
    \n';
85         }
86     }
87     return 0;
88 }

```

2.9 Hungarian 二分圖匹配

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define MP make_pair
5 #define PB push_back
6 #define maxn 50005
7
8 struct edge
9 {
10     int t,next;
11 } in[maxn*8];
12
13 int n,m,e,first[maxn];
14 bool vis[maxn],fail,side[maxn];
15
16 void add(int x,int y)
17 {
18     in[e].t=y;
19     in[e].next=first[x];
20     first[x]=e++;
21 }
22
23 void dfs(int cur,bool tf)

```

```

24 {
25     vis[cur]=true;
26     side[cur]=tf;
27     for(int i = first[cur]; ~i; i = in[i].next)
28     {
29         if(fail)
30             return;
31         if(vis[in[i].t] && side[in[i].t]==tf)
32             fail=true;
33         else if(!vis[in[i].t])
34             dfs(in[i].t, !tf);
35     }
36 }
37
38 int main(void)
39 {
40     int x,y;
41     while(cin >> n >> m)
42     {
43         e = 0;
44         fail = false;
45         memset(first, -1, sizeof(first));
46         memset(vis, false, sizeof(vis));
47
48         for(int i = 0; i < m; i++)
49         {
50             cin >> x >> y;
51
52             add(x,y);
53             add(y,x);
54         }
55
56         for(int i = 1; i <= n; i++)
57             if(!vis[i])
58                 dfs(i, false);
59
60         if(fail)
61             puts("no");
62         else
63             puts("yes");
64     }
65     return 0;
66 }

```

2.10 樹上兩點最遠距離

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5 #define PB push_back
6 #define PII pair<int, int>
7 #define MP make_pair
8 #define IOS ios_base::sync_with_stdio(false); cin.tie
    (0)
9 #define all(x) x.begin(), x.end()
10 #define REP(x, y, z) for(int x = y; x <= z; x++)
11 #define maxn 100000+5
12
13 //structure
14
15 //declaration
16 int n;
17 vector<PII> e[maxn];
18 PII first;
19
20 //functions
21 void dfs(int ver, int fa, int dep)
22 {
23     if(dep > first.second)
24     {
25         first.first = ver;
26         first.second = dep;
27     }
28
29     for(auto i : e[ver])

```

```

30 {
31     if(i.first != fa)
32     {
33         dfs(i.first, ver, dep+i.second);
34     }
35 }
36 return;
37 }
38 }
39 int main(void)
40 {
41     IOS;
42     while(cin >> n)
43     {
44         //init
45         REP(i, 1, n)
46         {
47             e[i].clear();
48         }
49         first.second = 0;
50
51         int x, y, z;
52
53         REP(i, 1, n-1)
54         {
55             cin >> x >> y >> z;
56
57             e[x].PB(MP(y, z));
58             e[y].PB(MP(x, z));
59         }
60
61         dfs(1, 1, 0);
62
63         first.second = 0;
64
65         dfs(first.first, first.first, 0);
66
67         printf("%.1f\n", (float)(first.second)/2);
68     }
69
70     return 0;
71 }
72
73
74
75

```

2.11 Max Flow

```

1 struct Edge{
2     int t,r,opp,next; // t=邊的終點, r=剩餘流量, opp=反
3     // 向邊編號, next=下一條邊編號(鍊表)
4     Edge () {}
5     Edge (int a,int b,int c,int d) { t=a; r=b; opp=c;
6     next=d; }
7 }in[M*25];
8 int e,st,ed; // 當前邊標號, 源點, 匯點
9 int first[M],arc[M],dis[M]; // arc: bfs完以後, 每個點找
10 剩餘流找到哪條邊的標記
11 void add(int x,int y,int z) { // 加一條 x->y, 流量 z
12 的邊
13     in[e] = Edge(y,z,e+1,first[x]);
14     first[x] = e++;
15     in[e] = Edge(x,0,e-1,first[y]);
16     first[y] = e++;
17 }
18 void init() {
19     e = 0;
20     MSET(first, -1);
21     MSET(dis, 0);
22 }
23 bool bfs() {
24     int cur;
25     queue<int> q;
26     REP(i,0,n) dis[i] = 0;

```

```

23     REP(i,0,n) arc[i] = first[i];
24     dis[ed] = 1;
25     q.push(ed);
26
27     while (!q.empty()) {
28         cur = q.front();
29         q.pop();
30         for (int i=first[cur]; ~i; i=in[i].next)
31             if (in[in[i].opp].r && !dis[in[i].t]) {
32                 dis[in[i].t] = dis[cur] + 1;
33                 q.push(in[i].t);
34             }
35     }
36     return dis[st] > 0;
37 }
38 int dfs(int cur,int flow) {
39     if (cur==ed) return flow;
40     int re=0, tmp;
41     for (int i=arc[cur]; ~i; i=arc[cur]=in[i].next)
42         if (dis[in[i].t]==dis[cur]-1 && in[i].r>0) { //
43             // 如果距離編號剛好小1, 且有剩餘流量
44             tmp = dfs(in[i].t, min(in[i].r, flow));
45             re += tmp;
46             flow -= tmp;
47             in[i].r -= tmp;
48             in[in[i].opp].r += tmp;
49             if (!flow) return re;
50         }
51     return re;
52 }
53 int maxflow() {
54     int res = 0;
55     while(bfs()) res += dfs(st, 200000000);
56     return res;
57 }

```

2.12 Max Flow Minimin Cut

```

1 const int INF = 1e9;
2 struct edge{int t,r,cost,next;};
3 int dis[M],pre[M],rec[M];
4 bool inq[M];
5
6 bool spfa()
7 {
8     int cur;
9     MSET(inq,false);
10    REP(i,0,n)dis[i]=INF;
11    dis[st]=0;
12    q.push(st);
13
14    while(!q.empty())
15    {
16        cur=q.front();
17        q.pop();
18        inq[cur]=false;
19        for(int i=first[cur];~i;i=in[i].next)
20            if(in[i].r>0 && dis[cur]+in[i].cost<dis[in[i].t])
21            {
22                dis[in[i].t]=dis[cur]+in[i].cost;
23                pre[in[i].t]=cur;
24                rec[in[i].t]=i;
25                if(!inq[in[i].t])
26                {
27                    q.push(in[i].t);
28                    inq[in[i].t]=true;
29                }
30            }
31    }
32    if(dis[ed]==INF)return false;
33    return true;
34 }
35
36 int costflow()
37 {

```

```

38 int delta,mincost=0,maxflow=0;
39 while(spfa())
40 {
41     delta=INF;
42     for(int i=ed;i!=st;i=pre[i])
43         if(in[rec[i]].r<delta)
44             delta=in[rec[i]].r;
45     for(int i=ed;i!=st;i=pre[i])
46     {
47         in[rec[i]].f+=delta;
48         in[in[rec[i]].opp].f-=delta;
49         in[rec[i]].r-=delta;
50         in[in[rec[i]].opp].r+=delta;
51     }
52     mincost+=dis[ed]*delta;
53     maxflow+=delta;
54 }
55 return mincost;
56 }

```

2.13 尤拉路徑

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5 #define PB push_back
6 #define PII pair<int, int>
7 #define MP make_pair
8 #define IOS ios_base::sync_with_stdio(false); cin.tie
9 (0)
10 #define all(x) x.begin(), x.end()
11 #define REP(x, y, z) for(int x = y; x <= z; x++)
12 #define maxn 50000+5
13 #define maxm 200000+5
14 //structure
15 //declaration
16 int n, m;
17 int st;
18 vector<PII> adj[maxn];
19 vector<bool> edges;
20 int chk[maxn];
21 //functions
22 void dfs(int v)
23 {
24     for(auto i : adj[v])
25     {
26         if(edges[i.first] == true)
27         {
28             edges[i.first] = false;
29             dfs(i.second);
30             cout << i.second << ' ' << v << '\n';
31         }
32     }
33 }
34 int main(void)
35 {
36     //IOS;
37     while(cin >> n >> m)
38     {
39         //init
40         REP(i, 1, n)
41             adj[i].clear();
42         edges.clear();
43         memset(chk, 0, sizeof(chk));
44         int x, y;
45         REP(i, 1, m)

```

```

54 {
55     cin >> x >> y;
56     edges.PB(true);
57     adj[x].PB(MP(i-1, y));
58     adj[y].PB(MP(i-1, x));
59     chk[x]++;
60     chk[y]++;
61 }
62
63 //find 奇數點
64 st = 1;
65 REP(i, 1, n)
66 {
67     if(chk[i]%2 == 1)
68     {
69         st = i;
70         break;
71     }
72 }
73
74 dfs(st);
75
76 return 0;
77 }
78

```

3 Math

3.1 尤拉函數線上

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define maxn 46340 //bcz sqrt(2^31-1)~=46340.95 and
4 46341 not prime
5 bool prime[maxn];
6 void prime_table(){
7     memset(prime,true,sizeof(prime));
8     prime[0]=prime[1]=false;
9     for (int i = 2; i < maxn; ++i)
10         if (prime[i])
11             for (int j = i*i; j < maxn; j+=i)
12                 prime[j]=false;
13 }
14 int eularphi(int n)
15 {
16     if (n==0) return n;
17     int ans=n;
18     for (int i = 2; i < maxn; ++i)
19     {
20         if(prime[i] && n%i==0){
21             ans=ans/i*(i-1);
22             while(n%i==0&&n)
23                 n/=i;
24         }
25     }
26     if (n!=1){
27         ans=ans/n*(n-1);
28     }
29     return ans;
30 }
31
32 int main()
33 {
34     prime_table();
35     int in;
36     while(~scanf("%d",&in)){
37         printf("%d\n", eularphi(in));
38     }
39     return 0;
40 }

```


3.2 尤拉函數建表

```

1 // all numbers smaller than or equal to n.
2 #include<iostream>
3 using namespace std;
4 #define maxn 250000
5 // Computes and prints totient of all numbers
6 // smaller than or equal to n.
7 void eularphi_table(int n)
8 {
9     // Create and initialize an array to store
10    // phi or totient values
11    long long phi[n+1];
12    for (int i=1; i<=n; i++)
13        phi[i] = i; // indicates not evaluated yet
14        // and initializes for product
15        // formula.
16
17    // Compute other Phi values
18    for (int p=2; p<=n; p++)
19    {
20        // If phi[p] is not computed already,
21        // then number p is prime
22        if (phi[p] == p)
23        {
24            // Phi of a prime number p is
25            // always equal to p-1.
26            phi[p] = p-1;
27
28            // Update phi values of all
29            // multiples of p
30            for (int i = 2*p; i<=n; i += p)
31            {
32                // Add contribution of p to its
33                // multiple i by multiplying with
34                // (1 - 1/p)
35                phi[i] = (phi[i]/p) * (p-1);
36            }
37        }
38    }
39
40    // Print precomputed phi values
41    for (int i=1; i<=n; i++)
42        cout << "Totient of " << i << " is " << phi[i] << '\n';
43 }
44
45 int main()
46 {
47     freopen("o.out", "w", stdout); //for test
48     int n = maxn;
49     eularphi_table(n);
50     return 0;
51 }

```

3.3 Fibonacci 線上

```

1 #include <iostream>
2 #include <cstring>
3 using namespace std;
4 #define LL long long
5 //注意，f0=1, f1=1, f2=2...
6 const LL mod=1e9+7; // 避免數值過大造成 overflow，因此
7 // 將所有數值都 mod 10^9+7
8 struct Matrix {
9     LL a[2][2];
10    void all_0() // 清空矩陣
11    {
12        memset(a, 0, sizeof(a));
13    }
14    void I() // 讓矩陣變成單位方陣
15    {
16        a[0][0]=1; a[0][1]=0;

```

```

17    a[1][0]=0; a[1][1]=1;
18    }
19    void X() // 讓矩陣變成文章中的矩陣 A
20    {
21        a[0][0]=1; a[0][1]=1;
22        a[1][0]=1; a[1][1]=0;
23    }
24 };
25
26 Matrix operator*(const Matrix &a, const Matrix &b) //
27 // 矩陣乘法
28 {
29     Matrix ret;
30     ret.all_0();
31     for (LL i=0; i<2; i++) {
32         for (LL j=0; j<2; j++) {
33             for (LL k=0; k<2; k++) {
34                 ret.a[i][j]+=a.a[i][k]*b.a[k][j];
35                 ret.a[i][j]%=mod;
36             }
37         }
38     }
39     return ret;
40 }
41
42 Matrix power(Matrix a, LL n) // 快速幂
43 {
44     Matrix ret;
45     ret.I();
46     if (n==0) return ret;
47     ret.X();
48     if (n==1) return ret;
49     ret=power(a, n/2);
50     ret=ret*ret;
51     if (n%2==1) ret=ret*a;
52     return ret;
53 }
54
55 LL query(LL n)
56 {
57     Matrix tmp;
58     tmp.X();
59     tmp=power(tmp, n);
60     LL ret=tmp.a[1][0]+tmp.a[1][1]; // 因為初始的矩陣 X
61     // [0] 的兩個元的值都是 1，所以矩陣相乘的結果相當於
62     // 把矩陣 A 下面的兩個元加起來
63     ret%=mod;
64     return ret;
65 }
66
67 int main()
68 {
69     LL n;
70     while (cin >> n) {
71         cout << query(n) << endl;
72     }
73     return 0;
74 }

```

3.4 質數

```

1 #define maxn 46340
2 //bcz sqrt(2^31-1)~=46340.95 and 46341 46340 not prime
3 bool prime[maxn];
4 void prime_table(){
5     memset(prime, true, sizeof(prime));
6     prime[0]=prime[1]=false;
7     for (int i = 2; i < maxn; ++i)
8         if (prime[i])
9             for (int j = i*i; j < maxn; j+=i)
10                 prime[j]=false;
11 }

```


3.5 質數 linear

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define N 1000000
4 long long int not_prime[N];
5 vector<long long int> prime;
6 void prime_sieve(){
7     memset(not_prime,0,sizeof(not_prime));
8     not_prime[1]=1;
9     for(long long int i=2;i<N;i++){
10         if(!not_prime[i]){
11             prime.push_back(i);
12         }
13         for(long long int j=0;j<prime.size()&&i*prime[j]
14             <N;j++){
15             not_prime[i*prime[j]]=1;
16             if(i%prime[j]==0)
17                 break;
18         }
19 }

```

3.6 模逆元

```

1 #include <stdio.h>
2 //ax == 1 (mod m), 求x
3 struct gcdstruct {
4     int d;
5     int x;
6     int y;
7 };
8
9 gcdstruct EXTENDED_EUCLID(int a, int b) {
10     gcdstruct aa, bb;
11     if (b == 0) {
12         aa.d = a;
13         aa.x = 1;
14         aa.y = 0;
15         return aa;
16     } else {
17         bb = EXTENDED_EUCLID(b, a % b);
18         aa.d = bb.d;
19         aa.x = bb.y;
20         aa.y = bb.x - bb.y * (a / b);
21     }
22     return aa;
23 }
24
25
26 int inverse(int a, int m) {
27     int x;
28     gcdstruct aa;
29     aa = EXTENDED_EUCLID(a, m);
30     return aa.x;
31 }
32
33 int main() {
34     int a, m;
35     scanf("%d %d",&a,&m);
36
37     printf("%d\n",inverse(a, m));
38     getchar();
39     return 0;
40 }

```

3.7 擴展歐基里德

```

1 #include <bits/stdc++.h>
2 #define ll long long
3 //ax+by==gcd(a,b)
4 //ax+by==1
5 //ax+by==c
6 ll gcdEx(ll a, ll b, ll &x, ll &y)

```

```

7 {
8     if(b==0)
9     {
10         x = 1,y = 0;
11         return a;
12     }
13     else
14     {
15         ll r = gcdEx(b, a%b, x, y); /* r = GCD(a, b) =
16             GCD(b, a%b) */
17         ll t = x;
18         x = y;
19         y = t - a/b * y;
20         return r;
21     }
22 }
23 void extendex(ll m, ll n, ll &x, ll &y) {
24     //printf("%d %d %d %d\n",m,n,x,y );
25     x%=n;
26     y%=m;
27 }
28 int main(int argc, char const *argv[])
29 {
30     ll m,m_ans,n,n_ans,c;
31     scanf("%lld %lld %lld",&m,&n,&c);
32     m_ans=m;
33     n_ans=n;
34     gcdEx(m,n,m_ans,n_ans);
35     m_ans*=c;
36     n_ans*=c;
37     extendex(m,n,m_ans,n_ans);
38     printf("%lld %lld\n",m_ans,n_ans );
39     return 0;
40 }

```

3.8 中國剩餘定理-韓信點兵

```

1 #include <iostream>
2
3 using namespace std;
4 //参数可为负数的扩展欧几里德定理
5 void exOJLD(int a, int b, int &x, int &y) {
6     //根据欧几里德定理
7     if (b == 0) { //任意数与0的最大公约数为其本身。
8         x = 1;
9         y = 0;
10     } else {
11         int x1, y1;
12         exOJLD(b, a % b, x1, y1);
13         if (a * b < 0) { //异号取反
14             x = -x1;
15             y = a / b * y1 - x1;
16         } else { //同号
17             x = x1;
18             y = x1 - a / b * y1;
19         }
20     }
21 }
22 //剩餘定理
23 int calSYDL(int a[], int m[], int k) {
24     int N[k]; //这个可以删除
25     int mm = 1; //最小公倍数
26     int result = 0;
27     for (int i = 0; i < k; i++) {
28         mm *= m[i];
29     }
30     for (int j = 0; j < k; j++) {
31         int L, J;
32         exOJLD(mm / m[j], -m[j], L, J);
33         N[j] = m[j] * J + 1; //1
34         N[j] = mm / m[j] * L; //2 1和2这两个值应该是相
35             等的。
36         result += N[j] * a[j];
37     }
38 }

```

```

37|     return (result % mm + mm) % mm; //落在(0, mm)之间,
    这么写是为了防止result初始为负数, 本例中不可能
    为负可以直接写成: return result%mm;即可。
38| }
39|
40| int main() {
41|     int a[3] = {2,3,2}; //a[i]=n%m[i]
42|     int m[3] = {3,5,7};
43|     cout << calSYDL(a, m, 3) << endl;
44|     return 0;
45| }

```

4 Dynamic Programming

4.1 LIS

```

1| if(lis.size()==0||tmp>lis.back())
2|     lis.push_back(tmp);
3| else
4|     *lower_bound(lis.begin(),lis.end(),tmp)=tmp;

```

4.2 LCS

```

1| #include<bits/stdc++.h>
2|
3| using namespace std;
4|
5| int arr[1500][1500];
6|
7| void LCS(string str1, string str2){
8|
9|     memset(arr, 0, sizeof(arr));
10|    for (int i = 1; i <= str1.size(); i++){
11|
12|        for (int j = 1; j <= str2.size(); j++){
13|
14|            if (str1[i - 1] == str2[j - 1]){
15|
16|                arr[i][j] = arr[i - 1][j - 1] + 1;
17|            }
18|            else{
19|
20|                arr[i][j] = max(arr[i - 1][j], arr[i][j - 1]);
21|            }
22|        }
23|    }
24| }

```

4.3 Minimum Coin Change

```

1| #include <bits/stdc++.h>
2| using namespace std;
3|
4| #define maxn 100000
5| #define maxm 100000
6|
7| int coins[maxn];
8| int table[maxm];
9| int n, target;
10|
11| int minCoins(int n, int tar)
12| {
13|     table[0] = 0;
14|
15|     for (int i = 1; i <= tar; i++)
16|         table[i] = 1e9;
17|
18|     // Compute minimum coins required for all
19|     // values from 1 to V
20|     for (int i = 1; i <= tar; i++)

```

```

    {
        // Go through all coins smaller than i
        for (int j= 0; j < n; j++)
            if (coins[j] <= i)
            {
                int sub_res = table[i-coins[j]];
                if (sub_res != 1e9 && sub_res + 1 <
                    table[i])
                    table[i] = sub_res + 1;
            }
    }
31| }
32|
33| int main(void)
34| {
35|     while(cin >> n >> target)
36|     {
37|         for(int i = 0; i < n; i++)
38|         {
39|             cin >> coins[i];
40|         }
41|
42|         minCoins(n, target);
43|     }
44| }

```

4.4 Coin Change

```

1| #include<bits/stdc++.h>
2|
3| using namespace std;
4|
5| int coin[] = {1, 5, 10, 25, 50};
6| int arr[100000];
7|
8| void build(){
9|
10|    memset(arr, 0, sizeof(arr));
11|    arr[0] = 1;
12|    for (int i = 0; i < 5; i++){
13|
14|        for (int j = 1; j < 10000; j++){
15|
16|            if (j >= coin[i]){
17|
18|                arr[j] = arr[j] + arr[j - coin[i]];
19|            }
20|        }
21|    }
22| }

```

4.5 2D Maximum SubArray

```

1| #include<bits/stdc++.h>
2|
3| using namespace std;
4|
5| #define size 4
6|
7| int arr[size][size];
8|
9| int maxSubArr(){
10|
11|     int b[size];
12|     int MAX = -11111111;
13|
14|     for(int i = 0 ; i < size; i++){
15|
16|         memset(b, 0, sizeof(b));
17|         for(int j = i ; j < size ; j++){
18|
19|             int s = 0;
20|             for(int k = 0 ; k < size ; k++){

```

```

21 |         b[k] += arr[j][k];
22 |         s += b[k];
23 |         if(s <= 0)
24 |             s = b[k];
25 |         if(s > MAX)
26 |             MAX = s;
27 |     }
28 | }
29 | }
30 | }
31 | return MAX;
32 | }
33 |
34 | int main(){
35 |
36 | #ifdef DBG
37 | freopen("1.in", "r", stdin);
38 | freopen("2.out", "w", stdout);
39 | #endif
40 |
41 | for(int i = 0 ; i < size ; i++)
42 |     for(int j = 0 ; j < size ; j++)
43 |         cin >> arr[i][j];
44 |
45 | maxSubArr();
46 |
47 | return 0;
48 | }

```

5 Data Structure

5.1 Big Number

```

1 | #include<bits/stdc++.h>
2 |
3 | using namespace std;
4 |
5 | #define ll long long
6 |
7 | const int size = 1000;
8 | const int carrySys = 10;
9 |
10 |
11 | struct BigNum{
12 |
13 |     int len;
14 |     int bgNum[size];
15 |     bool sign;
16 |
17 |     void reset(){
18 |
19 |         len = 1;
20 |         memset(bgNum, 0, sizeof(bgNum));
21 |     }
22 |
23 |
24 |     BigNum add(const BigNum lhs, const BigNum rhs){
25 |
26 |         BigNum sum;
27 |         sum.reset();
28 |
29 |         int l = std::max(rhs.len, lhs.len);
30 |
31 |         for (int i = 0; i < l; i++)
32 |         {
33 |
34 |             sum.bgNum[i] += lhs.bgNum[i] + rhs.bgNum[i];
35 |             if (sum.bgNum[i] >= carrySys)
36 |             {
37 |
38 |                 sum.bgNum[i + 1]++;
39 |                 sum.bgNum[i] -= carrySys;
40 |             }
41 |         }

```

```

42 |         if (sum.bgNum[l])
43 |             l++;
44 |         sum.len = l;
45 |
46 |         if (!lhs.sign && !rhs.sign)
47 |             sum.sign = false;
48 |         else
49 |             sum.sign = true;
50 |
51 |         return sum;
52 |     }
53 |
54 |     BigNum sub(const BigNum lhs, const BigNum rhs, bool s
55 |     ){
56 |
57 |         BigNum ans;
58 |         ans.reset();
59 |
60 |         int l = max(rhs.len, lhs.len);
61 |         int tmp[size];
62 |         memset(tmp, 0, sizeof(tmp));
63 |         copy(lhs.bgNum, lhs.bgNum + lhs.len, tmp);
64 |         for (int i = 0; i < l; i++)
65 |         {
66 |
67 |             if (tmp[i] < rhs.bgNum[i] && i != l - 1)
68 |             {
69 |
70 |                 tmp[i + 1] -= 1;
71 |                 tmp[i] += carrySys;
72 |             }
73 |             ans.bgNum[i] = tmp[i] - rhs.bgNum[i];
74 |         }
75 |
76 |         if (ans.bgNum[l - 1] < 0)
77 |         {
78 |
79 |             ans.bgNum[l - 1] = abs(ans.bgNum[l - 1]);
80 |             ans.sign = false;
81 |         }
82 |         else
83 |             ans.sign = true;
84 |
85 |         ans.len = l;
86 |
87 |         while (ans.len > 1 && !ans.bgNum[ans.len - 1])
88 |         {
89 |
90 |             ans.len--;
91 |         }
92 |
93 |         ans.sign = s;
94 |
95 |         return ans;
96 |     }
97 |     void intToBigNum(ll x){
98 |
99 |         if(x < 0){
100 |             sign = false;
101 |             x *= -1;
102 |         }
103 |         else
104 |             sign = true;
105 |
106 |         reset();
107 |         if(x == 0)
108 |             return;
109 |
110 |         len = 0;
111 |         while(x){
112 |
113 |             bgNum[len++] = x % 10;
114 |             x /= 10;
115 |         }
116 |     }
117 |     void strToBigNum(char x[]){

```

```

118     reset();
119     len = strlen(x);
120     int l = 0;
121     int a = -1;
122     if(x[0] == '-'){
123
124         sign = false;
125         a++;
126     }
127     else{
128
129         sign = true;
130     }
131
132     for(int i = len-1 ; i > a ; i--){
133
134         bgNum[l++] = x[i] - '0';
135     }
136     if(!sign){
137
138         len--;
139     }
140 }
141
142 void strToBigNum(string x){
143
144     reset();
145
146     if(x[0] == '-')
147         sign = false;
148     else
149         sign = true;
150
151     reverse(x.begin(), x.end());
152     len = x.size();
153     if(!sign)
154         len--;
155
156     for(int i = 0 ; i < len ; i++){
157
158         bgNum[i] = x[i] - '0';
159     }
160 }
161
162 BigNum operator+(const BigNum &rhs){
163
164     BigNum a = *this;
165     BigNum b = rhs;
166
167     if(sign && rhs.sign)
168         return add(*this, rhs);
169     else if(!sign && rhs.sign){
170
171         a.sign = true;
172         return (a > b ? sub(a, b, false) : sub(b, a, true));
173     }
174     else if(sign && !rhs.sign){
175
176         b.sign = true;
177         return (a > b ? sub(a, b, true) : sub(b, a, false));
178     }
179     else if(!sign && !rhs.sign)
180         return add(*this, rhs);
181 }
182
183 BigNum operator-(const BigNum &rhs){
184
185     BigNum a = *this;
186     BigNum b = rhs;
187
188     if(sign && rhs.sign)
189         return ((*this >= rhs) ? sub(*this, rhs, true) : sub(rhs, *this, false));
190     else if(!sign && rhs.sign){
191
192         b.sign = false;
193         return add(a, b);
194     }
195     else if(sign && !rhs.sign){
196
197         b.sign = true;
198         return add(a, b);
199     }
200     else{
201
202         a.sign = true;
203         b.sign = true;
204         if(a > b){
205
206             return sub(a, b, false);
207         }
208         else{
209
210             return sub(b, a, true);
211         }
212     }
213 }
214
215 // BigNum operator * (const BigNum &rhs){
216 //
217 //     cout<< "mul" << endl;
218 //     BigNum ans;
219 //     ans.reset();
220 //     for(int i = 0 ; i < len ; i++){
221 //
222 //         for(int j = 0 ; j < rhs.Len ; j++){
223 //
224 //             int l = i + j;
225 //             ans.bgNum[l] += bgNum[i] * rhs.bgNum[j];
226 //             while(ans.bgNum[l] >= carrySys){
227 //
228 //                 ans.bgNum[l+1] += ans.bgNum[l] / carrySys;
229 //                 ans.bgNum[l] = ans.bgNum[l] % carrySys;
230 //             }
231 //         }
232 //     }
233 //
234 //     ans.Len = len + rhs.Len;
235 //     if(!ans.bgNum[ans.Len-1]){
236 //
237 //         ans.Len--;
238 //     }
239 //     return ans;
240 // }
241
242 friend bool operator < (const BigNum &lhs, const BigNum &rhs){
243
244     // cout << lhs.Len << rhs.Len << endl;
245     if(lhs.sign < rhs.sign)
246         return true;
247     else if(lhs.sign > rhs.sign)
248         return false;
249     else{
250
251         if(lhs.len < rhs.len)
252             return true;
253         else if(lhs.len == rhs.len){
254
255             for(int i = 0 ; i < lhs.len ; i++){
256
257                 if(lhs.bgNum[i] < rhs.bgNum[i])
258                     return true;
259             }
260             return false;
261         }
262         else
263             return false;
264     }
265 }

```

```

268 }
269
270 friend bool operator > (const BigNum &lhs, const
    BigNum &rhs){
271
272     if(lhs.sign > rhs.sign)
273         return true;
274     else if(lhs.sign < rhs.sign)
275         return false;
276     else{
277
278         if (lhs.len > rhs.len)
279             return true;
280         else if (lhs.len == rhs.len){
281
282             for (int i = 0; i < lhs.len; i++){
283
284                 if (lhs.bgNum[i] > rhs.bgNum[i])
285                     return true;
286             }
287             return false;
288         }
289         else
290             return false;
291     }
292 }
293
294 friend bool operator >= (const BigNum &lhs, const
    BigNum &rhs){
296
297     return !(lhs < rhs);
298 }
299
300 friend bool operator <= (const BigNum &lhs, const
    BigNum &rhs){
301
302     return !(lhs > rhs);
303 }
304 BigNum operator = (const BigNum &rhs){
305
306     len = rhs.len;
307     copy(rhs.bgNum, rhs.bgNum+rhs.len, bgNum);
308     sign = rhs.sign;
309 }
310
311 friend ostream& operator<<(ostream &out, const BigNum
    &num){
312
313     if(!num.sign){
314
315         cout << "-";
316     }
317
318     out << num.bgNum[num.len-1];
319     for(int i = num.len-2 ; i >= 0 ; i--)
320         out << num.bgNum[i];
321     return out;
322 }
323
324 BigNum(){ reset(); }
325 BigNum(int x){ reset(); intToBigNum(x); }
326 BigNum(ll x){ reset(); intToBigNum(x); }
327 BigNum(string x){
328
329     reset();
330     strToBigNum(x);
331 }
332 BigNum(char x[]){
333
334     reset();
335     strToBigNum(x);
336 }
337
338 };
339
340 int main(){

```

```

341
342 #ifdef DBG
343 freopen("1.in", "r", stdin);
344 freopen("2.out", "w", stdout);
345 #endif // DEBUG
346
347
348 // char a[] = "12345";
349 // char b[] = "-2345";
350
351 // BigNum x = a;
352 // BigNum y = b;
353
354 // cout << x << " " << y << endl;
355
356 string a, b;
357
358 while(cin >> a >> b){
359
360     BigNum x, y;
361     // cout << "aaa: ";
362     x = a;
363     y = b;
364
365     BigNum z = x - y;
366     cout << z << endl;
367     // cout << z.len << endl;
368 }
369
370
371 return 0;
372 }

```

5.2 Disjoin Set

```

1 #define SIZE 10000
2
3 int arr[SIZE];
4
5 void init(int n) // give a initial length
6 {
7     for(int i=0; i<n; i++)
8         arr[i] = -1;
9 }
10
11 int find(int x)
12 { // find the father point
13     return arr[x] < 0 ? x : (arr[x] = find(arr[x])); //
    update every child to the other father
14 }
15
16 void Union(int x, int y)
17 {
18     x = find(x);
19     y = find(y);
20
21     if(x == y)
22         return;
23
24     if(arr[x] <= arr[y])
25     {
26         arr[x] += arr[y];
27         arr[y] = x;
28     }
29     else
30     {
31         arr[y] += arr[x];
32         arr[x] = y;
33     }
34 }

```

5.3 Segment Tree

```

1 #define SIZE 100000
2
3 int st[SIZE];
4 int st_val[SIZE];
5
6 void st_build(int *st, int *st_val, int now, int ls,
7             int rs)
8 {
9     if(ls == rs)
10        st[now] = st_val[ls];
11    else
12    {
13        st_build(st, st_val, now*2, ls, (ls+rs)/2);
14        st_build(st, st_val, now*2+1, (ls+rs)/2+1, rs);
15        st[now] = max(st[now*2], st[now*2+1]);
16    }
17
18 // ls and rs are query range, begin and end is whole st
19 // [] range
20 int query(int now, int ls, int rs, int begin, int end)
21 {
22     int mid = (begin+end)/2;
23     int ret = 0;
24
25     if(ls <= begin && rs >= end)
26         return st[now];
27
28 // it is find max now (modify here)
29 if(ls <= mid)
30     ret = max(ret, query(now*2, ls, rs, begin, mid));
31
32 if(rs > mid)
33     ret = max(ret, query(now*2+1, ls, rs, mid+1, end));
34
35 return ret;
36 }

```

6 Others

6.1 Roman to Int

```

1 unordered_map<char, int> value{{'I', 1}, {'V', 5}, {'X', 10}, {'L', 50}, {'C', 100}, {'D', 500}, {'M', 1000}};
2
3 int romanToInt(string s){
4
5     if(s.empty())
6         return 0;
7
8     int maxDigit = -1;
9     int ans = 0;
10    for(int i = s.size()-1 ; i >= 0 ; i--){
11
12        const int current = value[s[i]];
13        if(current >= maxDigit){
14
15            ans += value[s[i]];
16            maxDigit = current;
17        }
18        else{
19
20            ans -= value[s[i]];
21        }
22    }
23    return ans;
24 }

```

6.2 misc

1| STL func 用法

```

2 priority_queue<int> p;
3 priority_queue<int> //大到小, 預設
4 priority_queue<int, vector<int>, greater<int>> //小到
5 大
6 map:
7
8 查找不重複資料數量: 用map實作, 直接讀map.size()
9 map.count(key)用來判斷key有無對應val, 有1無0
10
11 vector:
12
13 vector大小相關
14 預開: v.reserve(N);
15 vector<int> bar (5,0); 五個0
16 myvector.capacity()//看目前最大容量
17
18 insert:
19 v.insert (v.begin()+i, av.begin(), av.end()); //把整個
20 av插入v[i]左邊
21 it = myvector.begin();
22 it = myvector.insert ( it , 200 ); //插入完後指向插
23 完的左邊
24 v.insert (v.begin(), 2, 300); //插兩個300 在v.begin()
25 左邊
26 list:(隨機存取慢, 增減資料快)
27 insert: 與vector相同
28
29 sort: 1.sort() (預設小到大), 可自放cmp
30 bool mycomparison (double first, double second)
31 { return ( int(first)<int(second) ); }
32 unique: 1.unique() 合併重複項, 可自定義(cmp)
33 bool same_integral_part (double first, double
34 second)
35 { return ( int(first)==int(second) ); }
36
37 merge: first.merge(second); //second會清空, 會排
38 序, 可自定義排序方式
39
40 string:
41 string str="We think in generalities, but we live in
42 details.";
43 string str2 = str.substr (3,5); // "think"
44 size_t pos = str.find("Live"); // position of
45 "Live" in str
46 string str3 = str.substr (pos); // get from "
47 Live" to the end
48 cout << str2 << ' ' << str3 << '\n';
49 'think live in details.'
50
51 math function 用法:
52 #define PI 3.14159265
53 atan2 (y,x) * 180 / PI; //給x,y座標回傳角度
54 cos ( x * PI / 180.0 ); //cos(x)==?
55 log();
56 log10();

```

7 Recent update

7.1 上下樓梯

```

1 //uva 1648
2 //設往上x次, 則往下n-x次(n次移動)
3 //Level=xu-(n-x)d大等0
4 //所以(u+d)大等nd min(x)=nd/(u+d)
5 //否則x=nd/(u+d)+1

```

```

6 //整除時多上少下一次 答案u+d
7 #include <bits/stdc++.h>
8 using namespace std;
9 int main(int argc, char const *argv[])
10 {
11     int n,m,t1,t2;
12     while(~scanf("%d %d",&n,&m)){
13         std::vector<int> up;
14         std::vector<int> down;
15         for (int i = 0; i < m; ++i)
16         {
17             scanf("%d %d",&t1,&t2);
18             up.push_back(t1);
19             down.push_back(t2);
20         }
21         int minn=(1<<31-1);
22         for (int i = 0; i < up.size(); ++i)
23         {
24             if ((n*down[i])%(up[i]+down[i])==0)
25             {
26                 minn=min(minn,up[i]+down[i]);
27             }
28             else{
29                 int tmp =(n*down[i])/(up[i]+down[i])+1;
30                 minn=min(minn,tmp*up[i]-(n-tmp)*down[i]);
31             }
32         }
33         printf("%d\n",minn );
34     }
35     return 0;
36 }

```

7.2 組合數

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define maxn 1000000000000000
5 #define MP make_pair
6 #define PLILI pair<long long int,long long int>
7 ll m;
8 ll compose(ll n, ll k){
9     ll x=1;
10     for(int i=1;i<=k;i++)
11     {
12         if(x/i>m/(n-i+1)) return m+1; //直接視為超大數(
13         //超越m)
14         x*=n-i+1; //促使bin_serach往右走
15         x/=i; //怕OF 原本形式為x*(n-i+1)>m
16         //L13,14玄學，一定能除盡
17     }
18     return x;
19 }
20 ll bio(ll m,ll k){
21     ll l = 2*k;
22     ll r = m;
23     ll mid = (l+r)/2;
24     while(l<=r){
25         // printf("call compose where l=%d mid=%d r=%d k==%d\n",l,mid,r,k);
26         ll tmp =compose(mid,k);
27         if (tmp<m)
28         {
29             l=mid+1;
30             mid=(l+r)/2;
31         }
32         else if (tmp>m){
33             r=mid-1;
34             mid=(l+r)/2;
35         }
36         else{
37             return mid;
38         }
39     }

```

```

39 }
40 return -1;
41 }
42 }
43
44 int main(int argc, char const *argv[])
45 {
46     int t;
47     scanf("%d",&t);
48     while(t--){
49         scanf("%lld",&m);
50         vector<PLILI> v;
51         for (ll i = 1; ; ++i)
52         {
53             if (compose(i*2,i)>m)
54                 break;
55             ll tmp = bio(m,i);
56
57             if (tmp==-1)
58                 continue;
59             v.push_back(MP(tmp,i));
60             if ((tmp-i)!=i)
61             {
62                 v.push_back(MP(tmp,tmp-i));
63             }
64         }
65         sort(v.begin(), v.end());
66         printf("%d\n",v.size() );
67         for (int i = 0; i < v.size(); ++i)
68         {
69             if(i==0)
70                 printf("%lld,%lld",v[i].first,v[i].second);
71             else
72                 printf(" (%lld,%lld)",v[i]. first,v[i].second);
73         }
74         printf("\n");
75     }
76 }
77 return 0;
78 }

```

7.3 遞增減可能

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 #define PROBLEM "1650"
6 #define mod 1000000007
7
8 string str;
9 long long dp[1005][1005];
10 long long sum[1005][1005];
11 int len = 0;
12
13 void build(){
14
15     len = str.size()+1 ;
16
17     dp[1][1] = 1;
18     sum[1][1] = 1;
19
20     for(int i = 2 ; i <= len ; i++){
21
22         for(int j = 1 ; j <= i ; j++){
23
24             if(str[i-2] == 'D' || str[i-2] == '?'){
25
26                 dp[i][j] = (dp[i][j] + sum[i-1][i-1] - sum[i-1][j-1]) % mod;
27                 // for(int k = j ; k < i ; k++){
28
29                     // dp[i][j] += dp[i-1][k];
30                     // dp[i][j] %= mod;
31                 // }

```



```

32     }
33     if(str[i-2] == 'I' || str[i-2] == '?'){
34         dp[i][j] = (dp[i][j] + sum[i-1][j-1]) % mod;
35     }
36     sum[i][j] = (dp[i][j] + sum[i][j-1]) % mod;
37 }
38 }
39 }
40 }
41
42 int main(){
43
44     #ifdef DBG
45     freopen("UVA" PROBLEM ".in", "r", stdin);
46     freopen("UVA" PROBLEM ".out", "w", stdout);
47     #endif
48
49     while(cin >> str){
50         memset(dp, 0, sizeof(dp));
51         memset(sum, 0, sizeof(sum));
52         build();
53         cout << (sum[len][len]+mod)%mod << endl;
54     }
55
56     return 0;
57 }
58 }

```

7.4 dijk 第二短路

```

1 //dijk第二長路徑
2 #include <bits/stdc++.h>
3 using namespace std;
4 #define maxn 105
5 #define MP make_pair
6 #define PII pair<int,int>
7 int dis[maxn][maxn][2];
8 vector<PII> e[maxn];
9
10 void dijk(int cur){
11
12     int d, st = cur;
13     priority_queue<PII, vector<PII>, greater<PII> > q;
14
15     q.push( MP(0, cur) );
16
17     while(!q.empty()){
18         tie(d, cur) = q.top();
19         q.pop();
20         if (dis[st][cur][0]==1e9){
21             dis[st][cur][0]=d;
22         }
23         else{
24             if (d<dis[st][cur][0])
25             {
26                 dis[st][cur][1]=dis[st][cur][0];
27                 dis[st][cur][0]=d;
28             }
29             else if (dis[st][cur][0]<d && d<dis[st][cur][1])
30             {
31                 dis[st][cur][1]=d;
32             }
33             else{
34                 continue;
35             }
36         }
37
38         for (auto i : e[cur])
39         {
40             q.push(MP(d+i.second,i.first));
41         }
42     }
43 }
44
45 void init(void){
46     for (int i = 0; i < maxn; ++i)

```

```

47 {
48     for (int j = 0; j < maxn; ++j)
49     {
50         dis[i][j][0]=1e9;
51         dis[i][j][1]=1e9;
52     }
53 }
54
55 for (int i = 0; i < maxn; ++i)
56 {
57     e[i].clear();
58 }
59 }
60
61 int main(int argc, char const *argv[])
62 {
63     int dots,roads,tes=1;
64     int x,y,d;
65     while(~scanf("%d %d",&dots,&roads)){
66         printf("Set #%d\n",tes++ );
67         init();
68         for (int i = 0; i < roads; ++i)
69         {
70             scanf("%d %d %d",&x,&y,&d);
71             e[x].push_back(MP(y,d));
72             e[y].push_back(MP(x,d));
73         }
74
75         for (int i = 0; i < dots; ++i)
76         {
77             dijk(i);
78         }
79
80         // for (int i = 0; i < dots; ++i)
81         // {
82         //     printf("%d %d == %d\n",0,i,dis[0][i] );
83         // }
84         int ques,q;
85         scanf("%d",&ques);
86         for (int i = 0; i < ques; ++i)
87         {
88             scanf("%d %d",&x,&y);
89
90             if (dis[x][y][1] == 1e9)
91             {
92                 printf("? \n");
93             }
94             else{
95                 printf("%d \n",dis[x][y][1] );
96             }
97         }
98     }
99     return 0;
100 }
101 }

```

7.5 物理

```

1 #include<cstdio>
2 #include<cmath>
3 using namespace std;
4 const double g=9.81;
5 //k 拉力常數 L 繩長 s 高度 m 重量
6 int main(){
7     double k,l,s,m;
8     while(~scanf("%Lf%Lf%Lf%Lf",&k,&l,&s,&m)&&(k||l||s||m)){
9         double dtl=(m*g+sqrt(m*m*g*g+2.0*k*l*m*g))/k;
10         if(s>l+dtl) puts("Stuck in the air.");
11         else{
12             double v=2.0*g*s;
13             if(s>l) v-=k*(s-l)*(s-l)/m;
14             if(v<=100.0) puts("James Bond survives.");
15             else puts("Killed by the impact.");
16         }

```

```

17 |     }
18 |     return 0;
19 | }
20 |     row[i] = L[left] = R[right] = false;
21 |     }
22 | }
23 | }

```

7.6 4n+1prime

```

1 // h_prime 4n+1是質數 求範圍內兩質相乘之合數總和
2 #include <bits/stdc++.h>
3 using namespace std;
4 #define maxn 1000001
5 bool vis[maxn+5];
6 bool h_semi[maxn+5];
7 int prefix_sum[maxn+5];
8 long long int pri[maxn+5],pn=0;
9 void h_table(){
10     memset(vis,false,sizeof(vis));
11     memset(h_semi,false,sizeof(h_semi));
12     for (int i = 5; i <= maxn; i += 4){
13         if (vis[i]) continue;
14         pri[pn++] = i;
15         for (int j = i; j <= maxn; j += i){
16             vis[j] = true;
17         }
18     }
19     for (int i = 0; i < pn; ++i)
20     {
21         if (pri[i]*pri[i]>maxn)
22             break;
23         for (int j = i; j < pn; ++j)
24         {
25             if (pri[i]*pri[j]>maxn)
26                 break;
27             h_semi[pri[i]*pri[j]]=true;
28         }
29     }
30     for (int i = 1; i <=maxn; ++i)
31         prefix_sum[i]=prefix_sum[i-1]+(h_semi[i]?1:0);
32 }
33
34
35
36 int main(int argc, char const *argv[])
37 {
38     h_table();
39     int in;
40
41     while(~scanf("%d",&in)&&in){
42         printf("%d %d\n",in,prefix_sum[in]);
43     }
44     return 0;
45 }

```

7.7 八皇后

```

1 // N_Queen(0,num);
2 int Queen[37000][14];
3 int tmp[14];
4 int cnt;
5 bool row[14], L[27], R[27];
6
7 void N_Queen(int k, int Num){
8     if(k == Num){
9         for(int j = 0; j < Num; j++){
10             Queen[cnt][j] = tmp[j];
11             cnt++;
12             return;
13         }
14         for(int i = 0; i < Num; i++) {
15             int right= k+i, left= k-i+Num-1;
16             if(!row[i] && !L[left] && !R[right]) {
17                 row[i] = L[left] = R[right] = true;
18                 tmp[k]=i;
19                 N_Queen(k+1, Num);

```

7.8 線性質數表

```

1
2
3 void primeTable()
4 {
5     is_notp.reset();
6     is_notp[0] = is_notp[1] = 1;
7     for (int i = 2; i < N; i++)
8     {
9         if (!is_notp[i])
10         {
11             p.push_back(i);
12         }
13         for (int j = 0; j < (int)p.size() && i * p[j] <
14             N; j++)
15         {
16             is_notp[i * p[j]] = 1;
17             if (i % p[j] == 0)
18             {
19                 break;
20             }
21         }
22     }
23 }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41 }

```

7.9 二進制表示 + Dijkstra

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5 #define PB push_back
6 #define PII pair<int, int>
7 #define MP make_pair
8 #define all(x) x.begin(), x.end()
9 #define REP(x, y, z) for(int x = y; x <= z; x++)
10 #define REPP(x, y, z) for(int x = y; x >= z; x--)
11 #define F first
12 #define S second
13 #define MSET(x, y) memset(x, y, sizeof(x))
14 #define maxn 25
15 #define maxm 100+5
16
17 //structure
18 struct tool
19 {
20     int t;
21     int a1, a2, b1, b2;

```

```

22 };
23
24 struct Item
25 {
26     int cost;
27     int status;
28
29     Item(int cost, int status): cost(cost), status(
        status){}
30
31     bool operator< (const Item& rs) const
32     {
33         return cost > rs.cost;
34     }
35 };
36 //declaration
37 int n, m;
38 tool tools[maxm];
39 bool vis[(1<<20)+10];
40 //functions
41
42 int main(void)
43 {
44     #ifdef DBG
45     freopen("in.in", "r", stdin);
46     freopen("out.out", "w", stdout);
47     #endif
48
49     int cases = 1;
50
51     while(~scanf("%d %d", &n, &m))
52     {
53         if(!n && !m)
54             break;
55
56         REP(i, 0, m)
57         {
58             tools[i].t = tools[i].a1 = tools[i].a2 =
                tools[i].b1 = tools[i].b2 = 0;
59         }
60
61         string x, y;
62
63         REP(i, 0, m-1)
64         {
65             cin >> tools[i].t >> x >> y;
66             REP(j, 0, n-1)
67             {
68                 if ('+' == x[n-1-j]) { tools[i].a1 =
                    tools[i].a1 | (1 << j); }
69                 if ('-' == x[n-1-j]) { tools[i].a2 =
                    tools[i].a2 | (1 << j); }
70                 if ('+' == y[n-1-j]) { tools[i].b1 =
                    tools[i].b1 | (1 << j); }
71                 if ('-' == y[n-1-j]) { tools[i].b2 =
                    tools[i].b2 | (1 << j); }
72             }
73         }
74
75         //dis
76         priority_queue<Item> q;
77
78         q.push(Item(0, (1<<n)-1));
79
80         int ans = -1;
81         MSET(vis, false);
82
83         while(!q.empty())
84         {
85             Item tmp = q.top();
86             q.pop();
87
88             if(tmp.status == 0)
89             {
90                 ans = tmp.cost;
91                 break;
92             }

```

```

93         if(vis[tmp.status]) continue;
94         vis[tmp.status] = true;
95
96         for(int i = 0; i < m; i++)
97         {
98             if((tmp.status & tools[i].a1) == tools[
                i].a1 && (~tmp.status & tools[i].a2
                ) == tools[i].a2)
99             {
100                 int uu = tmp.status | tools[i].b1;
101                 uu = uu & ~tools[i].b2;
102                 q.push(Item(tmp.cost+tools[i].t, uu
                    ));
103             }
104         }
105     }
106
107     printf("Product %d\n", cases++);
108     if(ans == -1)
109         printf("Bugs cannot be fixed.\n");
110     else
111         printf("Fastest sequence takes %d seconds.\n", ans);
112     puts("");
113 }
114
115 return 0;
116 }

```

7.10 Flow 例題

```

1 //Dinic版
2 #include <cstdio>
3 #include <cstring>
4 #include <string>
5 #include <map>
6 #include <vector>
7 #include <queue>
8 #include <bits/stdc++.h>
9 using namespace std;
10
11 /// 結點下標從0開始，注意maxn
12 struct Dinic
13 {
14     static const int maxn = 1e3 + 5;
15     static const int INF = 0x3f3f3f3f;
16     struct Edge
17     {
18         int from, to, cap, flow;
19     };
20
21     int n, m, s, t; // 結點數，邊數（包括反向弧），源
        點編號和匯點編號
22     vector<Edge> edges; // 邊表。edges[e]和edges[e^1]互
        為反向弧
23     vector<int> G[maxn]; // 鄰接表，G[i][j]表示結點i的第j
        條邊在e數組中的序號
24     bool vis[maxn]; // BFS使用
25     int d[maxn]; // 從起點到i的距離
26     int cur[maxn]; // 當前弧下標
27
28     void init(int n)
29     {
30         this->n = n;
31         edges.clear();
32         for (int i = 0; i < n; i++)
33             G[i].clear();
34     }
35     void AddEdge(int from, int to, int cap)
36     {
37         edges.push_back((Edge){from, to, cap, 0});
38         edges.push_back((Edge){to, from, 0, 0});
39         m = edges.size();
40         G[from].push_back(m - 2);

```

```

41     G[to].push_back(m - 1);
42 }
43 bool BFS()
44 {
45     memset(vis, 0, sizeof(vis));
46     queue<int> Q;
47     Q.push(s);
48     vis[s] = 1;
49     d[s] = 0;
50     while (!Q.empty())
51     {
52         int x = Q.front();
53         Q.pop();
54         for (int i = 0; i < G[x].size(); i++)
55         {
56             Edge &e = edges[G[x][i]];
57             if (!vis[e.to] && e.cap > e.flow)
58             { // 只考慮殘量網絡中的弧
59                 vis[e.to] = 1;
60                 d[e.to] = d[x] + 1;
61                 Q.push(e.to);
62             }
63         }
64     }
65     return vis[t];
66 }
67 int DFS(int x, int a)
68 {
69     if (x == t || a == 0)
70         return a;
71     int flow = 0, f;
72     for (int &i = cur[x]; i < G[x].size(); i++)
73     { // 從上次考慮的弧
74         Edge &e = edges[G[x][i]];
75         if (d[x] + 1 == d[e.to] && (f = DFS(e.to, min(a,
76             e.cap - e.flow))) > 0)
77         {
78             e.flow += f;
79             edges[G[x][i] ^ 1].flow -= f;
80             flow += f;
81             a -= f;
82             if (a == 0)
83                 break;
84         }
85     }
86     return flow;
87 }
88 int Maxflow(int s, int t)
89 {
90     this->s = s;
91     this->t = t;
92     int flow = 0;
93     while (BFS())
94     {
95         memset(cur, 0, sizeof(cur));
96         flow += DFS(s, INF);
97     }
98     return flow;
99 }
100 vector<int> Mincut()
101 { // 在Maxflow之後調用
102     vector<int> ans;
103     for (int i = 0; i < edges.size(); i++)
104     {
105         Edge &e = edges[i];
106         if (vis[e.from] && !vis[e.to] && e.cap > 0)
107             ans.push_back(i);
108     }
109     return ans;
110 };
111
112 const int UP = 100 + 5;
113 int recep[UP], device[UP];
114 bool can[UP * 4][UP * 4];
115 map<string, int> ids;
116 Dinic dc;
117
118 int id(string s)
119 {
120     if (ids.count(s))
121         return ids[s];
122     int i = ids.size();
123     return ids[s] = i;
124 }
125
126 void floyd()
127 {
128     int n = ids.size();
129     for (int i = 0; i < n; i++)
130         can[i][i] = true;
131     for (int k = 0; k < n; k++)
132     {
133         for (int i = 0; i < n; i++)
134         {
135             for (int j = 0; j < n; j++)
136             {
137                 can[i][j] |= can[i][k] && can[k][j];
138             }
139         }
140     }
141 }
142
143 int main()
144 {
145     int T, n, m, k;
146     char s[24 + 5], s2[24 + 5];
147     scanf("%d", &T);
148     while (T--)
149     {
150         ids.clear();
151         memset(can, false, sizeof(can));
152         scanf("%d", &n);
153         for (int i = 0; i < n; i++)
154         {
155             scanf("%s", s);
156             recep[i] = id(s);
157         }
158         scanf("%d", &m);
159         for (int i = 0; i < m; i++)
160         {
161             scanf("%s%s", s2, s);
162             device[i] = id(s);
163         }
164         scanf("%d", &k);
165         for (int i = 0; i < k; i++)
166         {
167             scanf("%s%s", s, s2);
168             can[id(s)][id(s2)] = true;
169         }
170
171         floyd();
172
173         int V = ids.size();
174
175         dc.init(V + 2);
176         for (int i = 0; i < m; i++)
177         {
178             dc.AddEdge(V, device[i], 1);
179         }
180         for (int i = 0; i < n; i++)
181         {
182             dc.AddEdge(recep[i], V + 1, 1);
183         }
184
185         for (int i = 0; i < m; i++)
186         {
187             for (int t = 0; t < n; t++)
188             {
189                 if (!can[device[i]][recep[t]])
190                     continue;
191                 dc.AddEdge(device[i], recep[t], 1);
192             }
193         }
194     }
195 }

```

```
194|    }
195|    printf("%d\n", m - dc.Maxflow(V, V + 1));
196|    if (T)
197|        printf("\n");
198|    }
199|    return 0;
200| }
```