

# Contents

<b>1 Basic</b>	
1.1 Template	1
1.2 .vimrc	1
1.3 Basic cmd	1
<b>2 Graph</b>	
2.1 Dijkstra	1
2.2 Kruskal 最小生成數	1
2.3 Floyd Warshell 任意點最短路	1
2.4 Floyd Warshell 有向圖最小環	1
2.5 SPFA	1
2.6 SPFA 找負環	1
2.7 拓樸排序	1
2.8 LCA 樹的最短路/找共同祖先	1
2.9 Hungarian 二分圖匹配	1
2.10 樹上兩點最遠距離	1
2.11 Max Flow	1
2.12 Min Cut Max Flow	1
<b>3 Math</b>	
3.1 尤拉函數線上	1
3.2 尤拉函數建表	1
3.3 Fibonacci 線上	1
3.4 質數	1
3.5 質數 linear	1
<b>4 Dynamic Programming</b>	
4.1 LIS	1
4.2 LCS	1
4.3 minimum Coin Change	1
4.4 Coin Change	1
4.5 2D Maximum SubArray	1
<b>5 Data Structure</b>	
5.1 Big Number	1
5.2 DisjoinSet	1
5.3 Segment Tree	1
<b>6 Others</b>	
6.1 Roman To Int	1

## 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 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
12
13 //structure
14
15 //declaration
16
17 //functions
18
19 int main(void)
20 {
21     IOS;
22
23     return 0;
24 }
25

```

### 1.2 .vimrc

```

1 |syntax on
2 |color torte
3 |set nu ts=4 sw=4 ai mouse=a bs=2 ci hls ru nocp
4 |  showmatch ar
5 |hi cursorline cterm=none ctermbg=darkred
6 |filetype plugin indent on
7
8 |so $VIMRUNTIME/mswin.vim
9 |behave mswin
10

```

### 1.3 Basic cmd

```

1 |lpr A.cpp # 列印
2 |lpq # 查詢列印 queue
3
4 |g++ A.cpp -std=gnu++14 -O2 -Wall -Wshadow -o A
5 |./A
6 |./A < input
7 |./A < input > output
8 |diff output ans
9

```

## 2 Graph

### 2.1 Dijkstra

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define MP make_pair
5 #define PII pair<int, int>
6 #define maxn 50000+5
7
8 int dis[maxn]; // 預設都是 INF
9 vector<PII> e[maxn]; // (連到的點, 邊的距離)
10
11 void dijk(int cur) // dijk(起點)
12 {
13     int d;
14     priority_queue<PII, vector<PII>, greater<PII>> q; // 放
      (距離, 點編號), 每次會拿距離最小的點出來
15     q.push( MP(0, cur) );
16
17     while (!q.empty())
18     {
19         tie(d, cur) = q.top();
20         q.pop();
21         if (dis[cur] != 1e9)
22             continue; // 如果之前就拜訪過, 無視
23
24         dis[cur] = d;
25
26         for (auto i: e[cur])
27             if (dis[i.first] == 1e9)
28             {
29                 q.push( MP(d+i.second, i.first) );
30             }
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 最小生成數

```

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[x] = find(
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         else
57         {
58             cost += tmp.cost;
59             conn(tmp.from, tmp.to);
60         }
61     }
62 }
63
64 int main(void)
65 {
66     while(cin >> n >> m)
67     {
68         //init
69         edges.clear();
70         cost = 0;
71
72         for(int i = 0; i < m; i++)
73         {

```

```

74             int a, b, c;
75             cin >> a >> b >> c;
76             edges.push_back(Edge(a, b, c));
77         }
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                     if (dis[i][j] > dis[i][k] + dis[k][j])
28                     {
29                         // 如果可以以 k 為中繼點，更新 i, j 的最短距離
30                         dis[i][j] = dis[i][k] + dis[k][j];
31                     }
32
33         if(i == j)
34             ans = min(ans, dis[i][j]);
35     }
36
37     if(ans == 1e9)
38         cout << -1 << '\n';
39     else
40         cout << ans << '\n';
41 }
42 return 0;
43 }
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 void init(void)
43 {
44     fill(dis, dis+maxn, INF);
45     for(int i = 0; i < maxn; i++)
46     {
47         e[i].clear();
48     }
49     memset(inq, false, sizeof(inq));
50 }
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 void init(void)
53 {
54     fill(dis, dis+maxn, INF);
55     for(int i = 0; i < maxn; i++)
56     {
57         e[i].clear();
58     }
59     memset(updateCount, 0, sizeof(updateCount));
60     memset(inq, false, sizeof(inq));
61 }
62
63 bool dfs(int cur)
64 {
65     vis[cur]=true;
66     if(cur==n)return true;
67
68     for(int i = 0; i < e[cur].size(); i++)
69     {
70         if(!vis[e[cur][i].first])
71             if(dfs(e[cur][i].first))
72                 return true;
73     }
74     return false;
75 }
76
77 bool check()
78 {
79     memset(vis, false, sizeof(vis));
80     for(int i = 1; i <= n; i++)
81     {
82         if(updateCount[i]>n && dfs(i))
83             return true;
84     }
85     return false;
86 }
87
88 int main(void)
89 {
90     int x, y, z;
91     while(cin >> n >> m)
92     {
93         init();
94
95         for(int i = 0; i < m; i++)
96         {
97             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             {
47                 x = f[i][x];
48                 y = f[i][y];
49             }
50     assert(f[0][x] == f[0][y]); // 走完以後, 會發現 x,y
51 // 停在 lca 的正下方一個點
52     return f[0][x];
53 }
54
55 void make_lca() {
56     dep[1] = depw[1] = 0;
57     dfs(1, 1); // 拿 1 當 root, 且 1 的父節點是 1
58     for (int i=1; i<LG; i++)
59         for (int j=1; j<=n; j++)
60             f[i][j] = f[i-1][f[i-1][j]]; // j 往上走 2^(i-1)
61 // 再往上走 2^(i-1) = 往上走 2^i 步
62 }

```

```

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)] << '
85                 \n';
86         }
87     }
88     return 0;

```

## 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 5005
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
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 100000+5
13
14 //structure
15
16 //declaration
17 int n;
18 vector<PII> e[maxn];
19 PII first;
20
21 //functions
22 void dfs(int ver, int fa, int dep)
23 {
24     if(dep > first.second)
25     {
26         first.first = ver;
27         first.second = dep;
28     }
29
30     for(auto i : e[ver])
31     {
32         if(i.first != fa)
33         {
34             dfs(i.first, ver, dep+i.second);
35         }
36     }
37     return;
38 }
39
40 int main(void)
41 {
42     IOS;
43
44     while(cin >> n)
45     {

```

```

46 //init
47 REP(i, 1, n)
48 {
49     e[i].clear();
50 }
51
52 first.second = 0;
53
54 int x, y, z;
55
56 REP(i, 1, n-1)
57 {
58     cin >> x >> y >> z;
59
60     e[x].PB(MP(y, z));
61     e[y].PB(MP(x, z));
62 }
63
64 dfs(1, 1, 0);
65
66 first.second = 0;
67
68 dfs(first.first, first.first, 0);
69
70 printf("%.1f\n", (float)(first.second)/2);
71 }
72
73 return 0;
74 }
75

```

## 2.11 Max Flow

```

1 struct edge{int t,r,opp,next;}in[?];
2 int e,first[M],gap[M],dis[M];
3 inline void add(int x,int y,int z)
4 {
5     in[e].t=y;
6     in[e].r=z;
7     in[e].opp=e+1;
8     in[e].next=first[x];
9     first[x]=e++;
10
11     in[e].t=x;
12     in[e].r=z;
13     in[e].opp=e-1;
14     in[e].next=first[y];
15     first[y]=e++;
16 }
17 void init()
18 {
19     e=0;
20     MSET(first,-1);
21     MSET(gap,0);
22     MSET(dis,0);
23     gap[st]=NODE; //num of nodes
24 }
25 int sap(int cur,int flow)
26 {
27     if(cur==ed)return flow;
28     int re=0,tmp;
29     for(int i=first[cur];~i;i=in[i].next)
30     if(dis[in[i].t] == dis[cur]-1 && in[i].r)
31     {
32         tmp = sap(in[i].t, min(in[i].r, flow));
33         re += tmp;
34         flow -= tmp;
35         in[i].r -= tmp;
36         in[in[i].opp].r += tmp;
37         if(!flow)return re;
38     }
39     if(!(--gap[dis[cur]])) dis[st]=NODE+1;
40     gap[dis[cur]]++;
41     return re;
42 }
43

```

```

44 while(dis[st]<NODE) ans += sap(st, INF);

```

## 2.12 Min Cut Max Flow

```

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 int costflow()
36 {
37     int delta,mincost=0,maxflow=0;
38     while(spfa())
39     {
40         delta=INF;
41         for(int i=ed;i!=st;i=pre[i])
42         if(in[rec[i]].r<delta)
43         delta=in[rec[i]].r;
44         for(int i=ed;i!=st;i=pre[i])
45         {
46             in[rec[i]].f+=delta;
47             in[in[rec[i]].opp].f-=delta;
48             in[rec[i]].r-=delta;
49             in[in[rec[i]].opp].r+=delta;
50         }
51         mincost+=dis[ed]*delta;
52         maxflow+=delta;
53     }
54     return mincost;
55 }

```

## 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 }

```

```

8   for (int i = 2; i < maxn; ++i)
9       if (prime[i])
10          for (int j = i*i; j < maxn; j+=i)
11              prime[j]=false;
12   }
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 totien 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
   將所有數值都 mod 10^9+7
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) //
   矩陣乘法
27 {
28     Matrix ret;
29     ret.all_0();
30     for (LL i=0; i<2; i++) {
31         for (LL j=0; j<2; j++) {
32             for (LL k=0; k<2; k++) {
33                 ret.a[i][j]+=a.a[i][k]*b.a[k][j];
34                 ret.a[i][j]%=mod;
35             }
36         }
37     }
38     return ret;
39 }
40
41 Matrix power(Matrix a, LL n) // 快速幂
42 {
43     Matrix ret;
44     ret.I();
45     if (n==0) return ret;
46     ret.X();
47     if (n==1) return ret;
48     ret=power(a, n/2);
49     ret=ret*ret;
50     if (n%2==1) ret=ret*a;
51     return ret;
52 }
53
54 LL query(LL n)
55 {
56     Matrix tmp;
57     tmp.X();
58     tmp=power(tmp, n);

```



```

59 | LL ret=tmp.a[1][0]+tmp.a[1][1]; // 因為初始的矩陣 x
    |   [0] 的兩個元的值都是 1，所以矩陣相乘的結果相當於
    |   把矩陣 A 下面的兩個元加起來
60 | ret%=mod;
61 | return ret;
62 | }
63 |
64 | int main()
65 | {
66 |     LL n;
67 |     while (cin >> n) {
68 |         cout << query(n) << endl;
69 |     }
70 |     return 0;
71 | }

```

```

5 | int arr[1500][1500];
6 |
7 | void LCS(string str1, string str2){
8 |
9 |     memset(arr, 0, sizeof(arr));
10 |
11 |     for (int i = 1; i <= str1.size(); i++)
12 |     {
13 |         for (int j = 1; j <= str2.size(); j++)
14 |         {
15 |             if (str1[i - 1] == str2[j - 1])
16 |                 arr[i][j] = arr[i - 1][j - 1] + 1;
17 |             else
18 |                 arr[i][j] = max(arr[i - 1][j], arr[i][j - 1]);
19 |         }
20 |     }
21 | }

```

### 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]<N;j++){
14 |             not_prime[i*prime[j]]=1;
15 |             if(i%prime[j]==0)
16 |                 break;
17 |         }
18 |     }
19 | }

```

## 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 |

```

### 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++)
21 |     {
22 |         // Go through all coins smaller than i
23 |         for (int j = 0; j < n; j++)
24 |             if (coins[j] <= i)
25 |             {
26 |                 int sub_res = table[i-coins[j]];
27 |                 if (sub_res != 1e9 && sub_res + 1 <
28 |                     table[i])
29 |                     table[i] = sub_res + 1;
30 |             }
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
22                 b[k] += arr[j][k];
23                 s += b[k];
24                 if(s <= 0)
25                     s = b[k];
26                 if(s > MAX)
27                     MAX = s;
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 Datastructure

### 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 struct BigNum{
11
12     int len;
13     int bgNum[size];
14     bool sign;
15
16     void reset(){
17
18         len = 1;
19         memset(bgNum, 0, sizeof(bgNum));
20     }
21
22     BigNum add(const BigNum lhs, const BigNum rhs){
23
24         BigNum sum;
25         sum.reset();
26
27         int l = std::max(rhs.len, lhs.len);
28
29         for (int i = 0; i < l; i++)
30         {
31             sum.bgNum[i] += lhs.bgNum[i] + rhs.bgNum[i];
32             if (sum.bgNum[i] >= carrySys)
33             {
34                 sum.bgNum[i + 1]++;
35                 sum.bgNum[i] -= carrySys;
36             }
37         }
38         if (sum.bgNum[l])
39             l++;
40         sum.len = l;
41
42         if (!lhs.sign && !rhs.sign)
43             sum.sign = false;
44         else
45             sum.sign = true;
46
47         return sum;
48     }
49
50     BigNum sub(const BigNum lhs, const BigNum rhs, bool s)
51     {
52         BigNum ans;
53         ans.reset();
54
55         int l = max(rhs.len, lhs.len);
56         int tmp[size];
57         memset(tmp, 0, sizeof(tmp));
58         copy(lhs.bgNum, lhs.bgNum + lhs.len, tmp);
59         for (int i = 0; i < l; i++)
60         {
61             if (tmp[i] < rhs.bgNum[i] && i != l - 1)
62             {
63                 tmp[i + 1] -= 1;
64                 tmp[i] += carrySys;
65             }
66             ans.bgNum[i] = tmp[i] - rhs.bgNum[i];
67         }
68
69         if (ans.bgNum[l - 1] < 0)
70         {

```

```

77     ans.bgNum[1 - 1] = abs(ans.bgNum[1 - 1]);
78     ans.sign = false;
79 }
80 else
81     ans.sign = true;
82
83 ans.len = 1;
84
85 while (ans.len > 1 && !ans.bgNum[ans.len - 1])
86 {
87     ans.len--;
88 }
89 ans.sign = s;
90
91 return ans;
92 }
93 void intToBigNum(ll x){
94     if(x < 0){
95         sign = false;
96         x *= -1;
97     }
98     else
99         sign = true;
100
101     reset();
102     if(x == 0)
103         return;
104
105     len = 0;
106     while(x){
107         bgNum[len++] = x % 10;
108         x /= 10;
109     }
110 }
111 void strToBigNum(char x[]){
112     reset();
113     len = strlen(x);
114     int l = 0;
115     int a = -1;
116     if(x[0] == '-'){
117         sign = false;
118         a++;
119     }
120     else{
121         sign = true;
122     }
123
124     for(int i = len-1 ; i > a ; i--){
125         bgNum[l++] = x[i] - '0';
126     }
127     if(!sign){
128         len--;
129     }
130 }
131
132 void strToBigNum(string x){
133     reset();
134     if(x[0] == '-')
135         sign = false;
136     else
137         sign = true;
138
139     reverse(x.begin(), x.end());
140     len = x.size();
141
142     if(!sign)
143         len--;
144
145     for(int i = 0 ; i < len ; i++){
146         bgNum[i] = x[i] - '0';
147     }
148 }
149
150 BigNum operator+(const BigNum &rhs){
151     BigNum a = *this;
152     BigNum b = rhs;
153
154     if(sign && rhs.sign)
155         return add(*this, rhs);
156     else if(!sign && rhs.sign){
157         a.sign = true;
158         return (a > b ? sub(a, b, false) : sub(b, a, true));
159     }
160     else if(sign && !rhs.sign){
161         b.sign = true;
162         return (a > b ? sub(a, b, true) : sub(b, a, false));
163     }
164     else{//!sign && !rhs.sign
165         return add(*this, rhs);
166     }
167 }
168
169 BigNum operator-(const BigNum &rhs){
170     BigNum a = *this;
171     BigNum b = rhs;
172
173     if(sign && rhs.sign)
174         return ((*this >= rhs) ? sub(*this, rhs, true) : sub(rhs, *this, false));
175     else if(!sign && rhs.sign){
176         b.sign = false;
177         return add(a, b);
178     }
179     else if(sign && !rhs.sign){
180         b.sign = true;
181         return add(a, b);
182     }
183     else{
184         a.sign = true;
185         b.sign = true;
186         if(a > b){
187             return sub(a, b, false);
188         }
189         else{
190             return sub(b, a, true);
191         }
192     }
193 }
194
195 // BigNum operator * (const BigNum &rhs){
196 //     // cout<< "mul" << endl;
197 //     BigNum ans;
198 //     ans.reset();
199 //     for(int i = 0 ; i < len ; i++){
200 //         for(int j = 0 ; j < rhs.Len ; j++){
201 //             int l = i + j;

```

```

228 //      ans.bgNum[L] += bgNum[i] * rhs.bgNum[j];
229 //      while(ans.bgNum[L] >= carrySys){
230
231 //      ans.bgNum[L+1] += ans.bgNum[L] / carrySys;
232 //      ans.bgNum[L] = ans.bgNum[L] % carrySys;
233 //      }
234 //  }
235 //  }
236
237 //  ans.Len = Len + rhs.Len;
238 //  if(!ans.bgNum[ans.Len-1]){
239
240 //      ans.Len--;
241 //  }
242 //  return ans;
243 //  }
244
245 friend bool operator < (const BigNum &lhs, const
    BigNum &rhs){
246
247 //  cout << lhs.Len << rhs.Len << endl;
248 if(lhs.sign < rhs.sign)
249     return true;
250 else if(lhs.sign > rhs.sign)
251     return false;
252 else{
253
254     if(lhs.len < rhs.len)
255         return true;
256     else if(lhs.len == rhs.len){
257
258         for(int i = 0 ; i < lhs.len ; i++){
259
260             if(lhs.bgNum[i] < rhs.bgNum[i])
261                 return true;
262         }
263         return false;
264     }
265     else
266         return false;
267 }
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){
295
296     return !(lhs < rhs);
297 }
298
299 friend bool operator <= (const BigNum &lhs, const
    BigNum &rhs){
300
301     return !(lhs > rhs);
302 }
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 //  char a[] = "12345";
348 //  char b[] = "-2345";
349
350 //  BigNum x = a;
351 //  BigNum y = b;
352
353 //  cout << x << " " << y << endl;
354
355 string a, b;
356
357 while(cin >> a >> b){
358
359     BigNum x, y;
360     //  cout << "aaa: ";
361     x = a;
362     y = b;
363
364     BigNum z = x - y;
365     cout << z << endl;
366     //  cout << z.Len << endl;
367 }
368
369 return 0;
370 }

```

## 5.2 DisjoinSet

```

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])); //
14         // update every child to the other father
15 }
16
17 void Union(int x, int y)
18 {
19     x = find(x);
20     y = find(y);
21
22     if(x == y)
23         return;
24
25     if(arr[x] <= arr[y])
26     {
27         arr[x] += arr[y];
28         arr[y] = x;
29     }
30     else
31     {
32         arr[y] += arr[x];
33         arr[x] = y;
34     }
35 }

```

### 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
19 // ls and rs are query range, begin and end is whole st
20 // [] range
21 int query(int now, int ls, int rs, int begin, int end)
22 {
23     int mid = (begin+end)/2;
24     int ret = 0;
25
26     if(ls <= begin && rs >= end)
27         return st[now];
28
29     // it is find max now (modify here)
30     if(ls <= mid)
31         ret = max(ret, query(now*2, ls, rs, begin, mid));
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',
2     , 10}, {'L', 50}, {'C', 100}, {'D', 500}, {'M',
3     1000}};
4
5 int romanToInt(string s){
6
7     if(s.empty())
8         return 0;
9
10    int maxDigit = -1;
11    int ans = 0;
12    for(int i = s.size()-1 ; i >= 0 ; i--){
13
14        const int current = value[s[i]];
15        if(current >= maxDigit){
16
17            ans += value[s[i]];
18            maxDigit = current;
19        }
20        else{
21
22            ans -= value[s[i]];
23        }
24    }
25    return ans;
26 }

```