

Contents

1 Setting

1.1 /.vimrc

```

1 syntax on
2 color torte
3 set nu ts=4 sw=4 ai mouse=a bs=2 ci hls ru nosp
   showmatch ar fencs=utf-8
4 set guifont=Consolas:h10
5 filetype plugin indent on
6 so $VIMRUNTIME/mswin.vim
7 behave mswin
8
9 autocmd CursorMoved * exe printf('match VisualNOS /\V
   \<%s\>/', escape(expand('<word>'), '/\'))
10 autocmd CursorMovedi * exe printf('match VisualNOS /\V
   \<%s\>/', escape(expand('<word>'), '/\'))
11
12 map <F5> :r ~/sample.cpp<CR>
13 map <F9> :call Compile()<CR>
14 map! <F9> <ESC>:call Compile()<CR>
15 map <F10> :call Run()<CR>
16 map! <F10> <ESC>:call Run()<CR>
17
18 func! Compile()
19     exec "w"
20     exec "!g++ -Wall -Wshadow -std=gnu++0x % -o %< 2>
       Log.txt"
21     exe "cg Log.txt"
22     cw 5
23 endfunc
24
25 func! Run()
26     exec "!.%<" # "!%<" if windows
27 endfunc
28
29 cd ~/Desktop # C:\Users\???\Desktop

```

2 Basic

2.1 Builtin

```

1 — Built-in Function: int __builtin_ffs (T x)
2
3 Returns one plus the index of the least significant 1-
   bit of x, or if x is zero, returns zero.
4 返回右起第一个 '1' 的位置。
5
6 — Built-in Function: int __builtin_clz (T x)
7
8 Returns the number of leading 0-bits in x, starting at
   the most significant bit position. If x is 0, the
   result is undefined.
9 返回左起第一个 '1' 之前0的个数。
10
11 — Built-in Function: int __builtin_ctz (T x)
12
13 Returns the number of trailing 0-bits in x, starting at
   the least significant bit position. If x is 0, the
   result is undefined.
14 返回右起第一个 '1' 之后的0的个数。
15
16 — Built-in Function: int __builtin_popcount (T x)
17
18 Returns the number of 1-bits in x.
19 返回 '1' 的个数。
20
21 — Built-in Function: int __builtin_parity (T x)

```

```

22
23 Returns the parity of x, i.e. the number of 1-bits in x
   modulo 2.
24 返回 '1' 的个数的奇偶性。
25
26 T is unsigned, unsigned long, unsigned long long

```

2.2 BinarySearch

```

1 lower_bound(a, a+n, k); //最左邊 ≥ k 的位置
2 upper_bound(a, a+n, k); //最左邊 > k 的位置
3 upper_bound(a, a+n, k) - 1; //最右邊 ≤ k 的位置
4 lower_bound(a, a+n, k) - 1; //最右邊 < k 的位置
5 [lower_bound, upper_bound) //等於 k 的範圍
6 equal_range(a, a+n, k);

```

2.3 int128

```

1 istream &operator >> (istream &is, __int128 &x) {
2     char buf[30];
3     is >> buf;
4     bool minus = false;
5     int len = strlen(buf);
6     x = 0;
7     for (int i=0; i<len; i++) {
8         if (i==0 && buf[i]=='-') minus = true;
9         else x = x*10 + buf[i] - 48;
10    }
11    if (minus) x*=-1;
12    return is;
13 }
14 ostream &operator << (ostream &os, __int128 &x) {
15     vector<int> v;
16     __int128 tmp = x;
17     bool minus = tmp < 0;
18     if (minus) tmp *= -1;
19
20     while(tmp > 0) {
21         v.push_back(tmp%10);
22         tmp/=10;
23     }
24     if (minus) os << "-";
25     for (int i=(int)v.size()-1; i>=0; i--) os << v[i];
26     return os;
27 }

```

2.4 Mergesort

```

1 long long sol(int L, int R) {
2     if (R - L <= 1) return 0;
3     int M = (R + L) / 2;
4     long long ans = sol(L, M) + sol(M, R);
5     int i = L, j = M, k = L;
6     while (i < M || j < R) {
7         if (i >= M)
8             buf[k] = arr[j++];
9         else if (j >= R)
10            buf[k] = arr[i++];
11        else {
12            if (arr[i]<=arr[j])
13                buf[k] = arr[i++];
14            else {
15                buf[k] = arr[j++];
16                ans += M - i;
17            }
18        }
19        k++;
20    }
21    for (int k = L; k < R; k++) arr[k] = buf[k];
22    return ans;
23 }

```

2.5 ThreeSearch

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define N 20
4 int t,n,i,j;
5 struct happy{
6     double a,b,c;
7 }h[N];
8 double f2(double x,double a,double b,double c){return a
    *(x-b)*(x-b)+c;}
9 double f(double x){
10     double ans=0;
11     for(int i=0;i<n;i++){
12         ans=max(ans,f2(x,h[i].a,h[i].b,h[i].c));
13     }
14     // cout<<ans<<'\n';
15     return ans;
16 }
17 int main(){
18     cin.tie(NULL);
19     for(cin>>t;i<t;i++){
20         for(cin>>n,j=0;j<n;j++){
21             cin>>h[j].a>>h[j].b>>h[j].c;
22             double L=0,R=300,M,MM;
23             while(R-L>1e-9){
24                 M=L+(R-L)/3;
25                 MM=(M+R)/2;
26                 // cout<<L<< ' '<<M<< ' '<<MM<< ' '<<R<<'\n';
27                 if(f(M)>f(MM))L=M;
28                 else R=MM;
29             }
30             cout<<fixed<<setprecision(5)<<f(L)<<'\n';
31         }
32     }

```

3 Data and Structure

3.1 Disjoint Set

```

1 void init(){for (int i = 0; i < N; i++)p[i] = i;}
2 int find(int x){return x == p[x] ? x : p[x]=find(p[x])
    ;}
3 void Union(int a, int b){p[find(a)] = find(b);}

```

3.2 Segment Tree

```

1 int bulit(int L,int R,int x) {
2     if(L==R)return heap[x - 1]=arr[L];
3     int M=(L+R)>>1;
4     return heap[x-1]=bulit(L, M, (x << 1))+bulit(M + 1, R
        , (x << 1) + 1);
5 }
6 void modify(int L,int R,int x,int a,int b,int mo) {
7     if(b<L||R<a)return;
8     if(L==R){heap[x-1]+=mo; return;}
9     int M=(L+R)>>1;
10    modify(L,M,(x<<1),a,b,mo);
11    modify(M+1,R,(x<<1)+1,a,b,mo);
12    heap[x - 1] += mo;
13    return;
14 }
15 int quest(int L,int R,int x,int a,int b) {
16     if(b<L||R<a)return 0;
17     if(a<=L&&R<=b)return heap[x - 1];
18     int M=(L+R)>>1;
19     return quest(L,M,(x<<1),a,b)+quest(M+1,R,(x<<1)+1,a,b
        );
20 }

```

3.3 Treap

```

1 struct Treap{
2     Treap *l, *r;
3     int val, key, pri;
4     Treap(int _val, int _key) :
5         val(_val), key(_key), l(NULL), r(NULL), pri(rand())
6         {}
7     Treap(){};
8 };
9 Treap* merge(Treap* a, Treap* b){
10     if (!a || !b)return a ? a : b;
11     if (a->pri > b->pri){
12         a->r = merge(a->r, b);
13         return a;
14     }else{
15         b->l = merge(a, b->l);
16         return b;
17     }
18 }
19 void split(Treap* t, int k, Treap *&a, Treap *&b){
20     if (!t)a = b = NULL;
21     else if (t->key <= k){
22         a = t;
23         split(t->r, k, a->r, b);
24     }else {
25         b = t;
26         split(t->l, k, a, b->l);
27     }
28     return;
29 }
30 Treap* insert(Treap* t, int k){
31     Treap *tl, *tr;
32     split(t, k, tl, tr);
33     return merge(tl, merge(new Treap(k, ti++), tr));
34 }
35 Treap* remove(Treap* t, int k){
36     Treap *tl, *tr;
37     split(t, k - 1, tl, tr);
38     split(t, k, tl, tr);
39     return merge(tl, tr);

```

4 DP

4.1 CounterLine

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N=1<<15;
4 int n,m,cur;
5 long long int dp[2][N];
6
7 void update(int a,int b){
8     if(b&(1<<m)){
9         dp[cur][b^(1<<m)]+=dp[1-cur][a];
10    }
11 }
12
13 int main(){
14     while(cin>>n>>m){
15         if((n*m)&1){
16             cout<<"0\n";
17             continue;
18         }
19         if(n==1||m==1){
20             cout<<"1\n";
21             continue;
22         }
23         if(n<m)swap(n,m);
24         memset(dp,0,sizeof(dp));
25         cur=0;
26         dp[0][(1<<m)-1]=1;
27         for(int i=0;i<n;i++){

```

```

28     for(int j=0;j<m;j++){
29         cur^=1;
30         memset(dp[cur],0,sizeof(dp[cur]));
31         for(int k=0;k<(1<m);k++){
32             update(k,k<1);
33             if(i&&!(k&(1<m-1)))update(k,(k<1)^
34                 ^((1<m)^1);
35             if(j&&!(k&1))update(k,(k<1)^3);
36         }
37     }
38     cout<<dp[cur][(1<m)-1]<<'\\n';
39 }
40 }

```

4.2 LCS

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main() {
5     int n, m;
6     vector<int>a, b, dp[2];
7     cin >> n >> m;
8     a.resize(n);
9     b.resize(m);
10    for(int i=0;i<a.size();i++){
11        cin>>a[i];
12    }
13    for(int i=0;i<b.size();i++){
14        cin>>b[i];
15    }
16    dp[0].resize(m+1);
17    dp[1].resize(m+1);
18    for(int i=1;i<=n;i++){
19        for(int j=1;j<=m;j++){
20            if(a[i-1]==b[j-1])dp[i&1][j]=dp[(i&1)^1][j-1]+1;
21            else dp[i&1][j]=max(dp[i&1][j-1],dp[(i&1)^1][j]);
22        }
23    }
24    cout<<dp[n&1][m]<<'\\n';
25 }

```

4.3 LIS

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int n;
6     while(cin>>n){
7         vector<int>v;
8         for(int i=0,x;i<n;i++){
9             cin>>x;
10            if(!v.size()||x>v.back())v.push_back(x);
11            else *lower_bound(v.begin(), v.end(),x)=x;
12        }
13        cout<<v.size()<<'\\n';
14    }
15 }

```

4.4 TSP

```

1 void btb(int &x){
2     x=0;
3     for(int i=0,j=1;i<n;i++,j*=2)x+=b[i]*j;
4     return;
5 }
6 int main(){
7     memset(dp,0,sizeof(dp));
8     for(int i=1,st;i<=n;i++){//st:state

```

```

9         for(int jj=0;jj<n;jj++)b[n-jj-1]=(jj<i);
10        do{
11            btb(st);
12            for(int x=0;x<n;x++){
13                if(!b[x])continue;
14                if(i==1)dp[x][st]=dis[x][0];
15                for(int y=0;y<n;y++){
16                    if(x!=y&&b[y]&&(dp[x][st]==0||dp[x][st]>dp[y][st-(1<x)]+dis[y][x])){
17                        dp[x][st]=dp[y][st-(1<x)]+dis[y][x];
18                    }
19                }
20            }
21            while(next_permutation(b,b+n));
22        }
23        cout<<dp[0][(1<n)-1]<<'\\n';
24    }

```

5 Graph

5.1 Articulation Point

```

1 vector<int>v[N],bcc[N];//clear
2 LL dep[N],low[N],bccno[N],time_cnt,bcc_cnt;//set dep low -1 else 0
3 bitset<N>is_AP;//0
4 struct Edge{int s,t};
5 stack<Edge>st;//clear
6 int dfs(int s,int fa){
7     int child=0;
8     dep[s]=low[s]=time_cnt++;
9     for(auto t:v[s]){
10        Edge e=(Edge){s,t};
11        if(dep[t]==-1){
12            st.push(e);
13            child++;
14            dfs(t,s);
15            low[s]=min(low[s],low[t]);
16            if(dep[s]<=low[t]){
17                is_AP[s]=1;
18                bcc_cnt++;
19                bcc[bcc_cnt].clear();
20                while(1){
21                    Edge x=st.top(); st.pop();
22                    if(bccno[x.s]!=bcc_cnt){
23                        bcc[bcc_cnt].push_back(x.s);
24                        bccno[x.s]=bcc_cnt;
25                    }
26                    if(bccno[x.t]!=bcc_cnt){
27                        bcc[bcc_cnt].push_back(x.t);
28                        bccno[x.t]=bcc_cnt;
29                    }
30                    if(x.s==s&&x.t==t)break;
31                }
32            }else if(low[s]>dep[t]){
33                st.push(e);
34                low[s]=dep[t];
35            }
36        }
37    }
38    if(fa<0&&child==1)is_AP[s]=0;
39    return low[s];
40 }

```

5.2 BellmanFord

```

1 void bellman_ford(int s){
2     d[s]=0;
3     p[s]=s;
4     for(int i=0;i<n*1;i++){

```

```

5   for(int ss=0;ss<n;ss++){
6       for(auto:tt:v[ss]){
7           if(d[ss]+w[ss][tt]<d[tt]){
8               d[tt]=d[ss]+w[ss][tt];
9               p[tt]=ss;
10          }
11      }
12  }
13  }
14  }
15  void has_negative_cycle(){
16      for(int s=0;s<n;s++){
17          for(int j=0;j<n;j++){
18              if(d[s]+w[s][j]<d[j])return true;
19          }
20      }
21      return false;
22  }

```

5.3 Bipartite

```

1  #include <iostream>
2  #include <vector>
3  #include <stack>
4  #include <cstring>
5
6  #define S 50050
7
8  using namespace std;
9
10 vector<int> map[S];
11 int visit[S];
12 bool valid;
13
14 void check(int start) {
15     stack<int> st;
16     st.push(start);
17     visit[start] = 1;
18
19     while(valid && !st.empty()) {
20         int cur = st.top();
21         st.pop();
22
23         for(int i = 0; i < map[cur].size(); i++) {
24             int next = map[cur][i];
25
26             if(visit[next] == -1) {
27                 st.push(next);
28
29                 if(visit[cur] == 1) visit[next] = 2;
30                 else visit[next] = 1;
31             }
32             else if(visit[cur] == visit[next]) valid = false;
33         }
34     }
35 }
36
37 int main() {
38     int n, m;
39     cin >> n >> m;
40
41     for(int i = 0; i < m; i++) {
42         int a, b;
43         cin >> a >> b;
44
45         map[a].push_back(b);
46         map[b].push_back(a);
47     }
48
49     // -1 : not visit, 1 : tsudere, 2 : proud
50     memset(visit, -1, sizeof(visit));
51     valid = true;
52
53     for(int i = 1; i <= n; i++) {
54         if(valid && visit[i] == -1) {

```

```

55         check(i);
56     }
57 }
58
59 if(valid) cout << "yes" << endl;
60 else cout << "no" << endl;
61
62 return 0;
63 }

```

5.4 dijkstra

```

1  void dijkstra(int s){
2      //set vis[]=0 d[]=inf
3      priority_queue<Node>pq;
4      d[s]=0;
5      p[s]=s;
6      pq.push(Node(s,0));
7      while(!pq.empty()){
8          while(!pq.empty()&&vis[pq.top().p])pq.top();
9          if(pq.empty())break;
10         vis[pq.top().p]=1;
11         Node k=pq.top(); pq.pop();
12         for(auto t:v[k.p]){
13             if(d[k.p]+w[k.p][t]<d[t]){
14                 d[t]=d[k.p]+w[k.p][t];
15                 p[t]=k.p;
16                 pq.push((Node){t,d[t]});
17             }
18         }
19     }
20 }

```

5.5 Convex Hull

```

1  struct loc {
2      int x, y;
3      loc() {}
4      loc(int x, int y): x(x), y(y) {}
5      bool operator <(const loc& b)const {return x != b.x ?
6          x < b.x : y < b.y;}
7      bool operator ==(const loc& b)const {return x == b.x
8          && y == b.y;}
9      loc operator -(const loc& b)const {return loc(x - b.x
10         , y - b.y);}
11      int cross(const loc& b)const {return x * b.y - y * b.x;}
12      int dis(loc a, loc b) {return (x - b.x) * (x - b.x) +
13         (y - b.y) * (y - b.y);}
14 };
15 vector<loc>p, p1;
16 int n;
17 void convexhull() {
18     sort(p.begin(), p.end());
19     p.erase(unique(p.begin(), p.end()), p.end());
20     p1.clear();
21     p1.resize(p.size());
22     int m = 0;
23     for (int i = 0; i < p.size(); i++) {
24         while (m > 1 && (p1[m - 1] - p1[m - 2]).cross(p[i]
25             - p1[m - 2]) <= 0)m--;
26         p1[m++] = p[i];
27     }
28     int k = m;
29     for (int i = p.size() - 2; i >= 0; i--) {
30         while (m > k && (p1[m - 1] - p1[m - 2]).cross(p[i]
31             - p1[m - 2]) <= 0)m--;
32         p1[m++] = p[i];
33     }
34     if (n > 1)m--;
35     p1.resize(m);
36 }

```

5.6 Dinic

```

1 struct dinic{
2     struct Edge{int v,f,re;}; //residual flow
3     int n, s, t, level[M], now[M];
4     vector<Edge> e[M];
5     void init(int _n, int _s, int _t){
6         n = _n; s = _s; t = _t;
7         for (int i = 0; i <= n; i++)e[i].clear();
8     }
9     void add_edge(int u, int v, int f){
10         e[u].push_back({ v, f, e[v].size() });
11         e[v].push_back({ u, f, e[u].size() - 1 });
12     }
13     bool bfs(){
14         fill(level, level + n + 1, -1);
15         queue<int> q;
16         q.push(s); level[s] = 0;
17         while (!q.empty()){
18             int u = q.front(); q.pop();
19             for (auto it : e[u]){
20                 if (it.f > 0 && level[it.v] == -1){
21                     level[it.v] = level[u] + 1;
22                     q.push(it.v);
23                 }
24             }
25         }
26         return level[t] != -1;
27     }
28     int dfs(int u, int nf){
29         if (u == t)return nf;
30         int res = 0;
31         while (now[u] < e[u].size()){
32             Edge &it = e[u][now[u]];
33             if (it.f>0 && level[it.v] == level[u] + 1){
34                 int tf = dfs(it.v, min(nf, it.f));
35                 res += tf; nf -= tf; it.f -= tf;
36                 e[it.v][it.re].f += tf;
37                 if (nf == 0)return res;
38             }
39             else now[u]++;
40         }
41         if (!res)level[u] = -1;
42         return res;
43     }
44     int flow(int res = 0){
45         while (bfs()){
46             int temp;
47             memset(now, 0, sizeof(now));
48             while (temp = (dfs(s, INF))){
49                 res += temp;
50             }
51         }
52         return res;
53     }
54 };

```

5.7 FloydWarshall

```

1 #include <iostream>
2
3 #define INF 1e9
4 #define LL long long
5
6 using namespace std;
7
8 int main() {
9     int n;
10
11     while(cin >> n) {
12         LL dis[n][n];
13         LL ans = INF;
14
15         for(int i = 0; i < n; i++)
16             for(int j = 0; j < n; j++) {

```

```

17                 cin >> dis[i][j];
18                 if(dis[i][j] == 0) dis[i][j] = INF;
19             }
20
21         for(int i = 0; i < n; i++) {
22             for(int j = 0; j < n; j++) {
23                 if(i == j) continue;
24                 ans = min(ans, dis[i][j] + dis[j][i]);
25                 for(int k = 0; k < n; k++) {
26                     dis[i][j] = min(dis[i][j], dis[i][k]
27                                     + dis[k][j]);
28
29                     ans = min(ans, dis[i][j] + dis[k][i]
30                             + dis[j][k]);
31                 }
32             }
33
34             if(ans == INF) cout << -1 << endl;
35             else cout << ans << endl;
36         }
37         return 0;
38     }

```

5.8 KM

```

1 bool match(int i) {
2     vx[i] = true;
3     for (int j = 1; j <= n; j++) {
4         if ((fabs(Lx[i] + Ly[j] - w[i][j]) < 1e-9) && !vy[j]) {
5             vy[j] = 1;
6             if (!Left[j] || match(Left[j])) {
7                 Left[j] = i;
8                 return true;
9             }
10         }
11     }
12     return false;
13 }
14 void update() {
15     double a = 1e30;
16     for (int i = 1; i <= n; i++) {
17         if (vx[i])for (int j = 1; j <= n; j++) {
18             if (!vy[j])a = min(a, Lx[i] + Ly[j] - w[i][j]);
19         }
20     }
21     for (int i = 1; i <= n; i++) {
22         if (vx[i])Lx[i] -= a;
23         if (vy[i])Ly[i] += a;
24     }
25 }
26 void KM() { //reset Lx Ly Left
27     for (int i = 1; i <= n; i++) {
28         Left[i] = Lx[i] = Ly[i] = 0;
29         for (int j = 1; j <= n; j++) {
30             Lx[i] = max(Lx[i], w[i][j]);
31         }
32     }
33     for (int i = 1; i <= n; i++) {
34         while (1) {
35             vx.reset(); vy.reset();
36             if (match(i))break;
37             update();
38         }
39     }
40 }

```

5.9 Longest Common Ancestor

```

1 void preprocess() {
2     for (int i = 1; i <= 25; i++) {

```

```

3   for (int j = 1; j <= n; j++) {
4       if (par[j][i - 1] == -1 || par[par[j][i - 1]][i - 1] == -1) continue;
5       par[j][i] = par[par[j][i - 1]][i - 1];
6   }
7   }
8 }

```

5.10 MST

```

1 #include <iostream>
2 #include <vector>
3 #include <stack>
4 #include <cstring>
5 #include <algorithm>
6
7 #define LL long long
8 #define MAX 1e11
9 #define S 50050
10 using namespace std;
11
12 int n, m;
13 int sum;
14
15 typedef struct {
16     int a, b, l;
17 } edge;
18 bool cmp(edge l, edge r) { return l.l < r.l; }
19
20 vector<edge> v;
21
22 typedef struct {
23     int d;
24     LL l;
25 } node;
26
27 vector<node> map[S];
28
29 int disjoint[S];
30
31 int root(int x) {
32     if(disjoint[x] < 0) return x;
33     else {
34         disjoint[x] = root(disjoint[x]);
35         return disjoint[x];
36     }
37 }
38
39 bool same(int a, int b) {
40     return root(a) == root(b);
41 }
42
43 void connect(int a, int b) {
44     // cout << "CONNECT " << a << " " << b << endl;
45     int ra = root(a);
46     int rb = root(b);
47
48     disjoint[ra] += disjoint[rb];
49     disjoint[rb] = ra;
50 }
51
52 void kruskal() {
53     int remain = n - 1;
54     for(auto i : v) {
55         if(remain == 0) break;
56
57         if(!same(i.a, i.b)) {
58             connect(i.a, i.b);
59
60             map[i.a].push_back((node){i.b, i.l});
61             map[i.b].push_back((node){i.a, i.l});
62
63             sum += i.l;
64             remain--;
65         }
66     }

```

```

67 }
68
69 bool book[S];
70
71 void dfs(int start) {
72     stack<int> st;
73     st.push(start);
74
75
76     memset(book, false, sizeof(book));
77
78     while(!st.empty()) {
79         int cur = st.top();
80         // cout << cur << endl;
81         st.pop();
82
83         book[cur] = true;
84
85         for(int i = 0; i < map[cur].size(); i++) {
86             int next = map[cur][i].d;
87             if(!book[next]) {
88                 st.push(next);
89             }
90         }
91     }
92 }
93
94 void init() {
95     memset(disjoint, -1, sizeof(disjoint));
96     sum = 0;
97 }
98
99 bool check() {
100     for(int i = 1; i <= n; i++)
101         if(!book[i]) return false;
102
103     return true;
104 }
105
106 int main() {
107     init();
108
109     cin >> n >> m;
110
111     for(int i = 0; i < m; i++) {
112         edge tmp;
113         cin >> tmp.a >> tmp.b >> tmp.l;
114
115         v.push_back(tmp);
116     }
117
118     sort(v.begin(), v.end(), cmp);
119
120     kruskal();
121     dfs(1);
122
123     if(!check()) cout << -1 << endl;
124     else cout << sum << endl;
125
126     return 0;
127 }

```

5.11 SPFA

```

1 #include <iostream>
2 #include <vector>
3 #include <stack>
4 #include <queue>
5 #include <cstring>
6
7 #define S 50050
8 #define MAX 1e11
9 #define LL long long
10
11 using namespace std;
12

```

```

13 typedef struct {
14     int d;
15     LL l;
16 } XXX;
17 vector<XXX> map[S];
18
19 LL lon[S];
20 int cnt[S];
21 int n, m;
22 bool cycle;
23 bool inqueue[S];
24
25 void dfs(int start) {
26     stack<int> st;
27     st.push(start);
28
29     bool book[S];
30     memset(book, false, sizeof(book));
31
32     while(!st.empty()) {
33         int cur = st.top();
34         // cout << cur << endl;
35         st.pop();
36         lon[cur] = -MAX;
37         book[cur] = true;
38
39         for(int i = 0; i < map[cur].size(); i++) {
40             int next = map[cur][i].d;
41             if(!book[next]) st.push(next);
42         }
43     }
44 }
45
46 void spfa(int start) {
47     memset(inqueue, false, sizeof(inqueue));
48     for(int i = 0; i < S; i++) lon[i] = MAX;
49     cycle = false;
50
51     queue<int> q;
52     q.push(start);
53     lon[start] = 0;
54     inqueue[start] = true;
55
56     while(!q.empty()) {
57         int cur = q.front();
58         q.pop();
59         inqueue[cur] = false;
60         // cout << "AT: " << cur << " " << cnt[cur] <<
61         // endl;
62         cnt[cur]++;
63         if(cnt[cur] > n) {
64             dfs(cur);
65             return;
66         }
67
68         for(int i = 0; i < map[cur].size(); i++) {
69             int next = map[cur][i].d;
70
71             if(lon[next] > lon[cur] + map[cur][i].l) {
72                 lon[next] = lon[cur] + map[cur][i].l;
73                 if(!inqueue[next] && cnt[cur] <= n) {
74                     q.push(next);
75                     inqueue[next] = true;
76                 }
77             }
78         }
79     }
80 }
81
82 int main() {
83     cin >> n >> m;
84
85     for(int i = 0; i < m; i++) {
86         int a, b;
87         LL c;

```

```

89         cin >> a >> b >> c;
90
91         map[a].push_back((XXX) {b, c});
92     }
93
94     spfa(1);
95
96     if(lon[n] >= MAX || lon[n] <= -MAX) cout << "QAQ"
97     << endl;
98     else cout << lon[n] << endl;
99
100    return 0;

```

5.12 SumOfDistanceInTree

```

1 #include <bits/stdc++.h>
2 #pragma comment(linker, "/STACK:10240000,10240000")//递
   归太深，导致爆栈，所以使用扩栈语句
3 using namespace std;
4
5 const int N = 100009;
6 int dp[N] = {}, num[N];
7 vector<int> p[N];
8 bool f[N] = {};
9
10 void dfs(int s, int depth)
11 {
12     int len = p[s].size();
13     f[s] = 1;
14     num[s] = 1;
15     dp[1] += depth;
16     for(int i=0; i<len; i++)
17     {
18         if(!f[p[s][i]])
19         {
20             dfs(p[s][i], depth+1);
21             num[s] += num[p[s][i]];
22         }
23     }
24 }
25
26 void solve(int s, int n)
27 {
28     int len = p[s].size();
29     f[s] = 1;
30     for(int i=0; i<len; i++)
31     {
32         if(!f[p[s][i]])
33         {
34             dp[p[s][i]] = dp[s]+n-num[p[s][i]]*2;
35             solve(p[s][i], n);
36         }
37     }
38 }
39
40 int main()
41 {
42     int n;
43     scanf("%d", &n);
44     for(int i=1; i<n; i++)
45     {
46         int a, b;
47         scanf("%d%d", &a, &b);
48         p[a].push_back(b);
49         p[b].push_back(a);
50     }
51     dfs(1, 0);
52     memset(f, 0, sizeof(f));
53     solve(1, n);
54     for(int i=1; i<=n; i++)
55         printf("%d\n", dp[i]);
56     return 0;
57 }

```


5.13 TopologicalSort

```

1 #include <iostream>
2 #include <stack>
3 #include <vector>
4 #include <cstring>
5
6 #define S 50050
7
8 using namespace std;
9
10 vector<int> map[S];
11 stack<int> ans;
12 int state[S];
13 bool head[S];
14 bool valid;
15 int n, m;
16
17 void dfs(int cur) {
18     state[cur] = 1;
19
20     for(auto next : map[cur])
21         if(!state[next]) dfs(next);
22     else if(state[next] == 1) {
23         valid = false;
24         return ;
25     }
26
27     state[cur] = 2;
28
29     ans.push(cur);
30 }
31
32 void topology_sort() {
33     for(int i = 1; i <= n; i++)
34         if(valid && head[i]) dfs(i);
35
36     if(!valid) {
37         cout << -1 << endl;
38         return ;
39     }
40
41     while(!ans.empty()) {
42         cout << ans.top() << endl;
43         ans.pop();
44     }
45 }
46
47 int main() {
48     cin >> n >> m;
49
50     memset(head, true, sizeof(head));
51
52     for(int i = 0; i < m; i++) {
53         int a, b;
54         cin >> a >> b;
55
56         head[b] = false;
57
58         map[a].push_back(b);
59     }
60
61     memset(state, 0, sizeof(state));
62     valid = true;
63
64     topology_sort();
65
66     return 0;
67 }

```

6 Number

6.1 Catalan

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \frac{2(2n+1)}{n+2} C_n,$$

6.2 Extend Euclidean.cpp

```

1 int extgcd(int a,int b,int &x,int &y){
2     int d=a;
3     if(b){d=extgcd(b,a%b,y,x),y-=(a/b)*x;}
4     else x=1,y=0;
5     return d;
6 }//ax+by=1 ax同餘 1 mod b

```

6.3 GaussElimination

```

1 const int MAXN = 300;
2 const double EPS = 1e-8;
3 int n;
4 double A[MAXN][MAXN];
5 void Gauss() {
6     for(int i = 0; i < n; i++) {
7         bool ok = 0;
8         for(int j = i; j < n; j++) {
9             if(fabs(A[j][i]) > EPS) {
10                 swap(A[j], A[i]);
11                 ok = 1;
12                 break;
13             }
14         }
15         if(!ok) continue;
16         double fs = A[i][i];
17         for(int j = i+1; j < n; j++) {
18             double r = A[j][i] / fs;
19             for(int k = i; k < n; k++) {
20                 A[j][k] -= A[i][k] * r;
21             }
22         }
23     }
24 }

```

6.4 Matrix

```

1 template<typename T,int N=2>
2 struct Mat { //Matrix
3     unsigned long long v[N][N];
4     Mat operator*(Mat b) const {
5         Mat val;
6         for (int i = 0; i < N; i++) {
7             for (int j = 0; j < N; j++) {
8                 val.v[i][j] = 0;
9                 for (int k = 0; k < N; k++) {
10                     val.v[i][j] += v[i][k] * b.v[k][j];
11                 }
12             }
13         }
14         return val;
15     }
16 };

```

6.5 Prime table

```

1 void PrimeTable(){
2     is_notp.reset();
3     is_notp[0] = is_notp[1] = 1;
4     for (int i = 2; i < N; i++){
5         if (is_notp[i]) continue;
6         p.push_back(i);

```



```

7   for (int j=0;i*p[j]<N&&j<p.size();j++){
8       is_notp[i*p[j]] = 1;
9       if(i%p[j]==0)break;
10  }
11  }
12 }

```

```

19  for(int j=i;j<s.size();j++){
20      v=s[j]-'a';
21      if(!trie[u][v])return;
22      u=trie[u][v];
23      if(val[u]dp[i]=(dp[i]+dp[j+1])%MOD;
24  }
25  return;
26 }

```

7 String

7.1 KMP

```

1 void bulid_fail_funtion(string B, int *fail){
2     int len = B.length(), current_pos;
3     current_pos = fail[0] = -1;
4     for (int i = 1; i<len; i++){
5         while (current_pos != -1 && B[current_pos + 1] != B
6             [i]){
7             current_pos = fail[current_pos];
8         }
9         if (B[current_pos + 1] == B[i])current_pos++;
10        fail[i] = current_pos;
11    }
12 void match(string A, string B, int *fail){
13     int lenA = A.length(), lenB = B.length();
14     int current_pos = -1;
15     for (int i = 0; i<lenA; i++){
16         while (current_pos != -1 && B[current_pos + 1] != A
17             [i]){
18             current_pos = fail[current_pos];
19         }
20         if (B[current_pos + 1] == A[i])current_pos++;
21         if (current_pos == lenB - 1){//match! A[i-LenB+1,i
22             j=B
23             current_pos = fail[current_pos];
24         }
25     }
26 }
27 int main(){
28     int t, i;
29     string s;
30     for (i = 0, cin >> t; i<t; i++){
31         cin >> s;
32         int fail[N];
33         bulid_fail_funtion(s, fail);
34         int p = s.length() - 1;
35         if (fail[p] != -1 && (p + 1) % (p - fail[p]) == 0)
36             printf("%d\n", p - fail[p]);
37         else printf("%d\n", p + 1);
38     }
39 }

```

7.3 Zvalue

```

1 void z_value(){
2     int lens = s.size(), l = 0, r = 0;
3     z[0] = 0;
4     for (int i = 1; i < lens; i++){
5         if (i>r)z[i] = 0;
6         else{
7             int ip = i - 1;
8             if (ip + z[ip] < z[l])z[i] = z[ip];
9             else z[i] = r - l + 1;
10        }
11        while (i + z[i] < lens&&s[i + z[i]] == s[z[i]])z[i]
12            ++;
13        if (i + z[i] - 1 > r){
14            l = i;
15            r = l + z[i] - 1;
16        }
17    }
18 }

```

7.2 Trie

```

1 //init sz=1 trie[0]=0
2 void insert(string s){
3     int u=0,v;
4     for(int i=0;i<r.size();i++){
5         v=r[i]-'a';
6         if(!trie[u][v]){
7             memset(trie[sz],0,sizeof(trie[sz]));
8             val[sz]=0;
9             trie[u][v]=sz++;
10        }
11        u=trie[u][v];
12    }
13    val[u]=1;
14    return;
15 }
16 void search(string s,int i){
17     int u=0,v;
18     dp[i]=0;

```