Contents

1 Setting

1.1 /.vimrc

```
syntax on
color torte
set nu ts=4 sw=4 ai mouse=a bs=2 ci hls  ru nocp
    showmatch ar fencs=utf-8
set guifont=Consolas:h10
filetype plugin indent on
so $VIMRUNTIME/mswin.vim
behave mswin

autocmd CursorMoved  * exe printf('match VisualNOS /\V
    \<%s\>/', escape(expand('<cword>'), '/\'))
autocmd CursorMovedi * exe printf('match VisualNOS /\V
    \<%s\>/', escape(expand('<cword>'), '/\'))

map <F5> :r ~/sample.cpp<CR>
map <F9> :call Compile()<CR>
map! <F9> <ESC>:call Compile()<CR>
map <F10> :call Run()<CR>
map! <F10> <ESC>:call Run()<CR>

func! Compile()
    exec "w"
    exec "!g++ -Wall -Wshadow -std=gnu++0x % -o %< 2>
        log.txt"
    exe "cg log.txt"
    cw 5
endfunc

func! Run()
    exec "!./%<"  # "!%<" if windows
endfunc

cd ~/Desktop # C:\Users\???\Desktop
```

2 Basic

2.1 /buglist

```
/*
cmp 不能 return true
變數宣告在迴圈費時，要小心使用
<<運算小心溢位，good way: (1LL << x)
prime_table小心i,j溢位
 */
```

2.2 Builtin

```
— Built-in Function: int __builtin_ffs (T x)

Returns one plus the index of the least significant 1-
    bit of x, or if x is zero, returns zero.
返回右起第一个 '1' 的位置。

— Built-in Function: int __builtin_clz (T x)

Returns the number of leading 0-bits in x, starting at
    the most significant bit position. If x is 0, the
    result is undefined.
返回左起第一个 '1' 之前0的个数。

— Built-in Function: int __builtin_ctz (T x)

Returns the number of trailing 0-bits in x, starting at
    the least significant bit position. If x is 0, the
    result is undefined.
返回右起第一个 '1' 之后的0的个数。

— Built-in Function: int __builtin_popcount (T x)

Returns the number of 1-bits in x.
返回 '1' 的个数。

— Built-in Function: int __builtin_parity (T x)

Returns the parity of x, i.e. the number of 1-bits in x
    modulo 2.
返回 '1' 的个数的奇偶性。

T is unsigned, unsigned long, unsigned long long
```

2.3 BinarySearch

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

2.4 int128

```cpp
istream &operator >> (istream &is, __int128 &x) {
    char buf[30];
    is >> buf;
    bool minus = false;
    int len = strlen(buf);
    x = 0;
    for (int i=0; i<len; i++) {
        if (i==0 && buf[i]=='-') minus = true;
        else x = x*10 + buf[i] - 48;
    }
    if (minus) x*=-1;
    return is;
}
ostream &operator << (ostream &os, __int128 &x) {
    vector<int> v;
    __int128 tmp = x;
    bool minus = tmp < 0;
    if (minus) tmp *= -1;

    while(tmp > 0) {
        v.push_back(tmp%10);
        tmp/=10;
    }
    if (minus) os << "-";
    for (int i=(int)v.size()-1; i>=0; i--) os << v[i];
    return os;
}
```

2.5 Mergesort

```cpp
long long sol(int L, int R) {
  if (R - L <= 1)return 0;
  int M = (R + L) / 2;
  long long ans = sol(L, M) + sol(M, R);
  int i = L, j = M, k = L;
  while (i < M || j < R) {
    if (i >= M)
      buf[k] = arr[j++];
    else if (j >= R)
      buf[k] = arr[i++];
    else {
      if (arr[i]<=arr[j])
        buf[k] = arr[i++];
      else {
        buf[k] = arr[j++];
        ans += M - i;
      }
    }
    k++;
  }
  for (int k = L; k < R; k++) arr[k] = buf[k];
  return ans;
}
```

2.6 ThreeSearch

```cpp
#include <bits/stdc++.h>
using namespace std;
#define N 20
int t,n,i,j;
struct happy{
  double a,b,c;
}h[N];
double f2(double x,double a,double b,double c){return a*(x-b)*(x-b)+c;}
double f(double x){
  double ans=0;
  for(int i=0;i<n;i++){
    ans=max(ans,f2(x,h[i].a,h[i].b,h[i].c));
//    cout<<ans<<'\n';
  }
```

```cpp
  return ans;
}
int main(){
  cin.tie(NULL);
  for(cin>>t;i<t;i++){
    for(cin>>n,j=0;j<n;j++)
      cin>>h[j].a>>h[j].b>>h[j].c;
    double L=0,R=300,M,MM;
    while(R-L>1e-9){
      M=L+(R-L)/3;
      MM=(M+R)/2;
//      cout<<L<<' '<<M<<' '<<MM<<' '<<R<<'\n';
      if(f(M)>f(MM))L=M;
      else R=MM;
    }
    cout<<fixed<<setprecision(5)<<f(L)<<'\n';
  }
}
```

3 Data and Structure

3.1 Disjoint Set

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

3.2 Segment Tree

```cpp
int bulit(int L,int R,int x) {
    if(L==R)return heap[x - 1]=arr[L];
    int M=(L+R)>>1;
    return heap[x-1]=bulit(L, M, (x << 1))+bulit(M + 1, R
        , (x << 1) + 1);
}
void modify(int L,int R,int x,int a,int b,int mo) {
    if(b<L||R<a)return;
    if(L==R){heap[x-1]+=mo; return;}
    int M=(L+R)>>1;
    modify(L,M,(x<<1),a,b,mo);
    modify(M+1,R,(x<<1)+1,a,b,mo);
    heap[x - 1] += mo;
    return;
}
int quest(int L,int R,int x,int a,int b) {
    if(b<L||R<a)return 0;
    if(a<=L&&R<=b)return heap[x - 1];
    int M=(L+R)>>1;
    return quest(L,M,(x<<1),a,b)+quest(M+1,R,(x<<1)+1,a,b
        );
}
```

3.3 Treap

```cpp
struct Treap{
  Treap *l, *r;
  int val, key, pri;
  Treap(int _val, int _key) :
    val(_val), key(_key), l(NULL), r(NULL), pri(rand())
        {}
  Treap(){};
};
Treap* merge(Treap* a, Treap* b){
  if (!a || !b)return a ? a : b;
  if (a->pri > b->pri){
    a->r = merge(a->r, b);
    return a;
  }else{
    b->l = merge(a, b->l);
```

```
15      return b;
16    }
17  }
18  void split(Treap* t, int k, Treap *&a, Treap *&b){
19    if (!t)a = b = NULL;
20    else if (t->key <= k){
21      a = t;
22      split(t->r, k, a->r, b);
23    }else {
24      b = t;
25      split(t->l, k, a, b->l);
26    }
27    return;
28  }
29  Treap* insert(Treap* t, int k){
30    Treap *tl, *tr;
31    split(t, k, tl, tr);
32    return merge(tl, merge(new Treap(k, ti++), tr));
33  }
34  Treap* remove(Treap* t, int k){
35    Treap *tl, *tr;
36    split(t, k - 1, tl, t);
37    split(t, k, t, tr);
38    return merge(tl, tr);
39  }
```

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)
                            ^(1<<m)^1);
34                      if(j&&!(k&1))update(k,(k<<1)^3);
35                  }
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              }
33          }else if(low[s]>dep[t]){
34              st.push(e);
35              low[s]=dep[t];
36          }
37      }
38      if(fa<0&&child==1)is_AP[s]=0;
39      return low[s];
40  }
```

5.2 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  }
```

5.3 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 ?
          x < b.x : y < b.y;}
 6    bool operator ==(const loc& b)const {return x == b.x
          && y == b.y;}
 7    loc operator -(const loc& b)const {return loc(x - b.x
          , y - b.y);}
 8    int cross(const loc& b)const {return x * b.y - y * b.
          x;}
 9    int dis(loc a, loc b) {return (x - b.x) * (x - b.x) +
          (y - b.y) * (y - b.y);}
10  };
11  vector<loc>p, p1;
12  int n;
13  void convexhull() {
14    sort(p.begin(), p.end());
15    p.erase(unique(p.begin(), p.end()), p.end());
16    p1.clear();
17    p1.resize(p.size());
18    int m = 0;
19    for (int i = 0; i < p.size(); i++) {
20      while (m > 1 && (p1[m - 1] - p1[m - 2]).cross(p[i]
            - p1[m - 2]) <= 0)m--;
21      p1[m++] = p[i];
22    }
23    int k = m;
24    for (int i = p.size() - 2; i >= 0; i--) {
```

```
25    while (m > k && (p1[m - 1] - p1[m - 2]).cross(p[i]
          - p1[m - 2]) <= 0)m--;
26    p1[m++] = p[i];
27  }
28  if (n > 1)m--;
29  p1.resize(m);
30 }
```

5.4 Dinic

```
1 struct dinic{
2   static const int M = 10000;
3   static const int INF = 1e9;
4   struct Edge{
5     int v;
6     int f; //residual flow
7     int re;
8   };
9   int n, s, t, level[M], now[M];
10  vector<Edge> e[M];
11  void init(int _n, int _s, int _t){
12    n = _n; s = _s; t = _t;
13    for (int i = 0; i <= n; i++)e[i].clear();
14  }
15  void add_edge(int u, int v, int f){
16    e[u].push_back({ v, f, (int)e[v].size() });
17    e[v].push_back({ u, f, (int)e[u].size() - 1 });
18  }
19  bool bfs(){
20    fill(level, level + n + 1, -1);
21    queue<int> q;
22    q.push(s); level[s] = 0;
23    while (!q.empty()){
24      int u = q.front(); q.pop();
25      for (auto it : e[u]){
26        if (it.f > 0 && level[it.v] == -1){
27          level[it.v] = level[u] + 1;
28          q.push(it.v);
29        }
30      }
31    }
32    return level[t] != -1;
33  }
34  int dfs(int u, int nf){
35    if (u == t)return nf;
36    int res = 0;
37    while (now[u] < e[u].size()){
38      Edge &it = e[u][now[u]];
39      if (it.f>0 && level[it.v] == level[u] + 1){
40        int tf = dfs(it.v, min(nf, it.f));
41        res += tf; nf -= tf; it.f -= tf;
42        e[it.v][it.re].f += tf;
43        if (nf == 0)return res;
44      }
45      else now[u]++;
46    }
47    if (!res)level[u] = -1;
48    return res;
49  }
50  int flow(int res = 0){
51    while (bfs()){
52      int temp;
53      memset(now, 0, sizeof(now));
54      while (temp = (dfs(s, INF))){
55        res += temp;
56      }
57    }
58    return res;
59  }
60 }d;
```

5.5 KM

```
1 int n;
2 int Left[N];
3 double w[N][N], Lx[N], Ly[N];
4 bitset<N> vx, vy;
5
6 bool match(int i) {
7   vx[i] = true;
8   for (int j = 1; j <= n; j++) {
9     if ((fabs(Lx[i] + Ly[j] - w[i][j]) < 1e-9) && !vy[j
           ]) {
10      vy[j] = true;
11      if (!Left[j] || match(Left[j])) {
12        Left[j] = i;
13        return true;
14      }
15    }
16  }
17  return false;
18 }
19
20 void update() {
21   double a = 1e30;
22   for (int i = 1; i <= n; i++) {
23     if (vx[i])for (int j = 1; j <= n; j++) {
24       if (!vy[j])a = min(a, Lx[i] + Ly[j] - w[i][j]);
25     }
26   }
27   for (int i = 1; i <= n; i++) {
28     if (vx[i])Lx[i] -= a;
29     if (vy[i])Ly[i] += a;
30   }
31 }
32
33 void KM() {
34   for (int i = 1; i <= n; i++) {
35     Left[i] = Lx[i] = Ly[i] = 0;
36     for (int j = 1; j <= n; j++) {
37       Lx[i] = max(Lx[i], w[i][j]);
38     }
39   }
40   for (int i = 1; i <= n; i++) {
41     while (true) {
42       vx.reset(); vy.reset();
43       if (match(i))break;
44       update();
45     }
46   }
47 }
```

5.6 Longest Common Ancestor

```
1 const int LOG = 20;
2 int par[N][LOG];
3 int tin[N], tout[N];
4 int timer = 0;
5
6 void dfs(int v, int p){
7   tin[v] = ++timer;
8   par[v][0] = p;
9   for(int it: G[v]){
10    if(it != p)dfs(it, v);
11  }
12  tout[v] = ++timer;
13 }
14
15 void Doubling(){
16   for(int i = 1; i < N; ++i){
17     for(int j = 1; j < LOG; ++j){
18       par[i][j] = par[par[i][j - 1]][j - 1];
19     }
20   }
21 }
22
23 bool anc(int v, int u){
24   return tin[v] <= tin[u] && tout[u] <= tout[v];
25 }
```

```
26
27  int LCA(int v, int u){
28      if(anc(v, u))return v;
29      for(int j = LOG - 1; j >= 0; --j){
30          if(!anc(par[v][j], u))v = par[v][j];
31      }
32      return par[v][0];
33  }
34
35  int main(){
36      dfs(root, root);
37      Doubling();
38  }
```

5.7 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 }
```

5.8 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 }
```

5.9 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 Graph Shortest Path

6.1 BellmanFord

```
1  struct Edge{
2      int t, w;
3  };
4  int v, e;
5  int d[N], cnt[N];
6  bitset<N> inq;
7  queue<int>Q;
8  vector<Edge>G[N];
9
10 void addEdge(int from, int to, int w){
11     G[from].push_back({to,w});
12 }
13
14 bool hasnegativeCycle(){
15     while(!Q.empty())Q.pop();
16     for(int i = 1; i <= v;i++){
17         inq[i] = true;
18         cnt[i] = d[i] = 0;
19         Q.push(i);
20     }
21     while(!Q.empty()){
22         int s = Q.front(); Q.pop();
```

```
23        inq[s] = false;
24        for(Edge it: G[s]){
25            if(d[it.t] > d[s] + it.w){
26                d[it.t] = d[s] + it.w;
27                if(inq[it.t])continue;
28                Q.push(it.t);
29                inq[it.t] = true;
30                if(++cnt[it.t] > v)return true;
31            }
32        }
33    }
34    return false;
35 }
```

6.2 dijkstra

```
1 struct Edge{
2     int from,to,w;
3 };
4 vector<Edge>E;
5 vector<int>v[N];
6 bitset<N> vis;
7 void init(){
8     E.clear();
9     for(int i=0;i<N;i++){
10         v[i].clear();
11     }
12 }
13
14 void addEdge(int from,int to,int w){
15     v[from].push_back(E.size());
16     E.push_back(Edge{from,to,w});
17 }
18
19 void dijkstra(int s,int d[],int p[]){// set d[] INF &&
        set p[] -1
20     d[s]=0;
21     priority_queue<PII,vector<PII>,greater<PII>>pq;
22     vis.reset();
23     pq.push(MP(d[s],s));
24     while(!pq.empty()){
25         PII k=pq.top(); pq.pop();
26         if(vis[k.second])continue;
27         vis[k.second]=true;
28         for(auto it:v[k.second]){
29             Edge e=E[it];
30             if(d[e.to]>d[e.from]+e.w){
31                 d[e.to]=d[e.from]+e.w;
32                 p[e.to]=e.from;
33                 pq.push(MP(d[e.to],e.to));
34             }
35         }
36     }
37 }
```

6.3 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
                        ] + dis[k][j]);
27
28                    ans = min(ans, dis[i][j] + dis[k][i
                        ] + dis[j][k]);
29                }
30            }
31        }
32
33        if(ans == INF) cout << -1 << endl;
34        else cout << ans << endl;
35    }
36
37    return 0;
38 }
```

6.4 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
20 LL lon[S];
21 int cnt[S];
22 int n, m;
23 bool cycle;
24 bool inqueue[S];
25
26 void dfs(int start) {
27     stack<int> st;
28     st.push(start);
29
30     bool book[S];
31     memset(book, false, sizeof(book));
32
33     while(!st.empty()) {
34         int cur = st.top();
35         // cout << cur << endl;
36         st.pop();
37         lon[cur] = -MAX;
38         book[cur] = true;
39
40         for(int i = 0; i < map[cur].size(); i++) {
41             int next = map[cur][i].d;
42             if(!book[next]) st.push(next);
43         }
44     }
45 }
46
47 void spfa(int start) {
48     memset(inqueue, false, sizeof(inqueue));
49     for(int i = 0; i < S; i++) lon[i] = MAX;
```

```
50      cycle = false;
51
52      queue<int> q;
53      q.push(start);
54      lon[start] = 0;
55      inqueue[start] = true;
56
57      while(!q.empty()) {
58          int cur = q.front();
59          q.pop();
60          inqueue[cur] = false;
61          // cout << "AT: " << cur << " " << cnt[cur] <<
                  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
83  int main() {
84      cin >> n >> m;
85
86      for(int i = 0; i < m; i++) {
87          int a, b;
88          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"
            << endl;
97      else cout << lon[n] << endl;
98
99      return 0;
100 }
```

```
12      LL ret=1,tmp=x%m;
13      while(y){
14          if(y&1)ret=ret*tmp%m;
15          tmp=tmp*tmp%m;
16          y>>=1;
17      }
18      return ret;
19  }
20
21  void init(){
22      facs.clear();
23      LL x=m,sq=(LL)sqrt(m);
24      phi=1;
25      for(LL i=2;i<=sq;i++){
26          if(x%i)continue;
27          phi*=i-1; x/=i;
28          facs.push_back(i);
29          while(x%i==0){
30              phi*=i;
31              x/=i;
32          }
33      }
34      if(x>1){
35          phi*=x-1;
36          facs.push_back((int)x);
37      }
38      k=facs.size();
39      dp[0]=1;
40      memset(dp2,0,sizeof(dp2));
41      for(int i=1;i<M;i++){
42          LL tmp=i;
43          for(int j=0;j<k;j++){
44              dp2[i][j]=dp2[i-1][j];
45              while(tmp%facs[j]==0){
46                  tmp/=facs[j];
47                  dp2[i][j]++;
48              }
49          }
50          dp[i]=dp[i-1]*tmp%m;
51      }
52      return;
53  }
54
55  int main(){
56      while(cin>>n>>m){
57          init();
58          while(n--){
59              LL ans=1;
60              int x,y;
61              cin>>x>>y;
62              for(int i=0;i<k;i++){
63                  ans=ans*pw(facs[i],dp2[x][i]-dp2[x-y][i
                      ]-dp2[y][i])%m;
64              }
65              ans=ans*dp[x]%m;
66              ans=ans*pw(dp[y],phi-1)%m;
67              ans=ans*pw(dp[x-y],phi-1)%m;
68              cout<<ans<<'\n';
69          }
70      }
71  }
```

7   Number

7.1   Catalan

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

7.2   Combination

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long LL;
4  const int M=1000005;
5  int n,k;
6  LL m,phi;
7  vector <int> facs;
8  LL dp[M],dp2[M][32];
9
10 LL pw(LL x,LL y){
11     // cout<<x<<' '<<y<<'\n';
```

7.3   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
```

7.4   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 }
```

7.5 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 };
```

7.6 Phi

```
1  void phi_table(int n){
2      phi[1] = 1;
3      for(int i = 2; i <= n; i++){
4          if(phi[i])continue;
5          for(int j = i; j < n; j += i){
6              if(!phi[j])phi[j] = j;
7              phi[j] = phi[j] / i * (i - 1);
8          }
9      }
10 }
```

7.7 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 }
```

8 RMQ

8.1 Mo

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N=100005;
4  int a[N];
5  int curmax;
6  int app[N], cnt[N];
7
8  struct Query{
9      int L, R, qid, bid;
10     bool operator < (const Query&rhs) const {
11         if(bid != rhs.bid)return bid < rhs.bid;
12         return R < rhs.R;
13     }
14 }q[N];
15
16 bool cmp(Query a,Query b){
17     return a.L < b.L;
18 }
19
20 void add(int x){
21     int now = ++app[x];
22     cnt[now - 1]--;
23     cnt[now]++;
24     curmax=max(curmax, now);
25 }
26
27 void sub(int x){
28     int now = --app[x];
29     cnt[now + 1]--;
30     cnt[now]++;
31     if(!cnt[curmax])curmax--;
32 }
33
34 int main(){
35     int n, Q;
36     int ans[N];
37     cin >> n >> Q;
38     for(int i = 1;i <= n;i++){
39         cin >> a[i];
40     }
41     int k=floor(sqrt(n/1.0));
42     for(int i = 0;i < Q;i++){
43         cin >> q[i].L >> q[i].R;
44         q[i].qid = i;
45     }
46     sort(q, q + Q,cmp);
47     for(int i = 0;i < Q;i++){
48         q[i].bid = i / k;
49     }
50     sort(q, q + Q);
51     for(int i=0, curL = 1, curR = 0;i < Q;i++){
52         // cout<<i<<' '<<q[i].L<<' '<<q[i].R<<'\n';
53         while(curR < q[i].R){
54             curR++;
55             add(a[curR]);
56         }
57         while(q[i].R < curR){
58             sub(a[curR]);
59             curR--;
60         }
61         while(curL < q[i].L){
62             sub(a[curL]);
63             curL++;
64         }
65         while(q[i].L < curL){
66             curL--;
67             add(a[curL]);
68         }
69         ans[q[i].qid] = curmax;
70     }
71     for(int i=0;i<Q;i++){
72         cout<<ans[i]<<'\n';
```

```
73        }
74 }
```

```
23          if(val[u])dp[i]=(dp[i]+dp[j+1])%MOD;
24        }
25      return;
26 }
```

9 String

9.1 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
            [i]){
6       current_pos = fail[current_pos];
7     }
8     if (B[current_pos + 1] == B[i])current_pos++;
9     fail[i] = current_pos;
10  }
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
            [i]){
17       current_pos = fail[current_pos];
18     }
19     if (B[current_pos + 1] == A[i])current_pos++;
20     if (current_pos == lenB - 1){//match! A[i-lenB+1,i
            ]=B
21       current_pos = fail[current_pos];
22     }
23   }
24 }
25 int main(){
26   int t, i;
27   string s;
28   for (i = 0, cin >> t; i<t; i++){
29     cin >> s;
30     int fail[N];
31     bulid_fail_funtion(s, fail);
32     int p = s.length() - 1;
33     if (fail[p] != -1 && (p + 1) % (p - fail[p]) == 0)
            printf("%d\n", p - fail[p]);
34     else printf("%d\n", p + 1);
35   }
36 }
```

9.2 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;
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];
```

9.3 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 - l;
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    if (i + z[i] - 1 > r){
13      l = i;
14      r = l + z[i] - 1;
15    }
16  }
17 }
```