Contents

1 Basic

1.1 .vimrc

```
imap jj <Esc>
```

1.2 Increase Stack Size

```
//stack resize
asm( "mov %0,%%esp\n" ::"g"(mem+10000000) );
//change esp to rsp if 64-bit system

//stack resize (linux)
#include <sys/resource.h>
void increase_stack_size() {
  const rlim_t ks = 64*1024*1024;
  struct rlimit rl;
  int res=getrlimit(RLIMIT_STACK, &rl);
  if(res==0){
    if(rl.rlim_cur<ks){
      rl.rlim_cur=ks;
      res=setrlimit(RLIMIT_STACK, &rl);
    }
  }
}
```

2 Graph

2.1 HLD

```
struct segment_tree{
    #define MAXN 100100
    #define right(x) x << 1 | 1
    #define left(x) x << 1
    int* arr;
    LL sum[4*MAXN];
    const int inf = 1e9;

    void pull(int ind) {
        sum[ind] = sum[right(ind)]+sum[left(ind)];
    };
    /// root => 1
    void build(int ind, int l, int r) {

        if( r - l == 1) {
            sum[ind] = 0;
            return;
        }
        int mid = (l+r)>>1;
        build( left(ind), l, mid );
        build( right(ind), mid, r );
        pull(ind);
    }
    LL query_sum(int ind, int L, int R, int ql, int qr)
         {
        if( L >= qr || R <= ql ) return 0;
        if( R <= qr && L >= ql ) {
            return sum[ind];
        }
        int mid = (L+R)>>1;
        return query_sum(left(ind), L, mid, ql, qr) +
            query_sum(right(ind), mid, R, ql, qr);
    }
    void modify(int ind, int L, int R, int ql, int qr,
        int x) {
        if( L >= qr || R <= ql ) return;
        if( R <= qr && L >= ql ) {
            sum[ind] = x;
            return;
        }
        int mid = (L+R)>>1;
        modify(left(ind), L, mid, ql, qr, x);
        modify(right(ind), mid, R, ql, qr, x);
        pull(ind);
    }
};
```

```
struct Tree{
    segment_tree seg;
    #define MAXN 100010
    #define maxm (maxn<<1)
    int n;
    struct edge { int u, v; };
    vector<edge> e;
    void addedge(int x, int y) {
        G[x].pb( SZ(e) );
        G[y].pb( SZ(e) );
        e.pb( edge{x, y}  );
    }
    int siz[MAXN],max_son[MAXN],pa[MAXN],dep[MAXN];
    /*size of subtree、index of max_son, parent index、
        depth*/
    int link_top[MAXN],link[MAXN],Time;
    /*chain top、index in segtree、time stamp*/
    std::vector<int >G[MAXN];

    void init(int N) {
        n = N;
        e.clear();
        for(int i = 1; i <= n; i++) G[i].clear();
    }
    void find_max_son(int x){
        siz[x]=1;
        max_son[x]=-1;
        for(int e_ind : G[x]) {
            int v = e[e_ind].u == x ? e[e_ind].v : e[
                e_ind].u ;
            if( v == pa[x] )continue;
            pa[v] = x; dep[v] = dep[x] + 1;
            find_max_son(v);
            if(max_son[x] == -1 || siz[v] > siz[max_son
                [x]])
                max_son[x] = v;
            siz[x] += siz[v];
        }
    }
    void build_link(int x,int top){
        link[x] = ++Time;/*記錄x點的時間戳*/
        link_top[x] = top;
        if(max_son[x] == -1)return;
        build_link( max_son[x], top);/*優先走訪最大孩子
            */
        for(int e_ind : G[x]) {
            int v = e[e_ind].u == x ? e[e_ind].v : e[
                e_ind].u ;
            if( v == max_son[x] || v == pa[x] )continue
                ;
            build_link(v, v);
        }
    }
    inline int lca(int a,int b){
        /*求LCA，可以在過程中對區間進行處理*/
        int ta=link_top[a],tb=link_top[b];
        while(ta != tb){
            if(dep[ta]<dep[tb]){
                std::swap(ta,tb);
                std::swap(a,b);
            }
            //interval [ link[ta], link[a] ]
            a = pa[ta];
            ta = link_top[a];
        }
        return dep[a] < dep[b] ? a:b;
    }
    int query(int a,int b){
        int ret = 0;
        int ta=link_top[a],tb=link_top[b];
        while(ta != tb){
            if(dep[ta]<dep[tb]){
                std::swap(ta,tb);
                std::swap(a,b);
            }
            //interval [ link[ta],link[a] ]
            a = pa[ta];
            ta = link_top[a];
        }
        if( a == b ) return ret;
        else {
            if(dep[a]>dep[b])
                swap(a,b);
            //interval [ link[a],link[b] ]
            // if operate on edges ==> [ link[ max_son[
                ta] ], link[b] ]
        }
    }
    /// Heavy Light Decomposition
    void HLD() {
        // root is indexed 1 here !
        find_max_son(1);
        build_link(1, 1);
    }
     void modify(int a, int b, int x) {
        // modify the path from a -> b to x
        //( which is [ link[a] .. link[b] ] on the
            segment tree)
        seg.modify(1, 1, n+1, link[a], link[b]+1, x);
        // this segment tree uses [ 1 ..n+1 )
    }
}tree;
```

2.2 Hungarian

```
// edge and node index starting from 0
// dfs  version below
/* to do
#define ___maxNodes
num_left = ?
*/
struct Edge {
    int from;
    int to;
    int weight;
    Edge(int f, int t, int w):from(f), to(t), weight(w)
        {}
};
vector<int> G[___maxNodes]; /* G[i] 存储顶点 i 出发的边
    的编号 */
vector<Edge> edges;
int num_nodes;
int num_left;
int num_right;
int num_edges;
int matching[___maxNodes]; /* matching result */
int check[___maxNodes];

bool dfs(int u) {
    for (auto i = G[u].begin(); i != G[u].end(); ++i) {
        // 对 u 的每个邻接点
        int v = edges[*i].to;
        if (!check[v]) {     // 要求不在交替路中
            check[v] = true; // 放入交替路
            if (matching[v] == -1 || dfs(matching[v]))
                {
                // 如果是未盖点，说明交替路为增广路，则
                    交换路径，并返回成功
                matching[v] = u;
                matching[u] = v;
                return true;
            }
        }
    }
    return false; // 不存在增广路，返回失败
}
int hungarian() {
    int ans = 0;
    memset(matching, -1, sizeof(matching));
    for (int u=0; u < num_left; ++u) {
        if (matching[u] == -1) {
            memset(check, 0, sizeof(check));
            if (dfs(u)) ++ans;
        }
    }
    return ans;
}
```

2.3 KM

```
// 最小帶權匹配~ km算法
//http://acm.csie.org/ntujudge/contest_view.php?id=836&
    contest_id=449
#include <bits/stdc++.h>
using namespace std;

struct bipartite {
    #define maxn 602
    #define INF 0xfffffff
    int sx[maxn], sy[maxn], mat[maxn][maxn];
    int x[maxn], y[maxn], link[maxn];
    int N, M, slack;

    int DFS(int t) {
        int tmp;
        sx[t] = 1;
        for (int i = 0; i < M; i++) {
            if (!sy[i]) {
                tmp = x[t] + y[i] - mat[t][i];
                if (tmp == 0) {
                    sy[i] = 1;
                    if (link[i] == -1 || DFS(link[i]))
                        {
                        link[i] = t;
                        return 1;
                    }
                }
                else if (tmp < slack) slack = tmp;
            }
        }
        return 0;
    }
    int KM() {
        for (int i = 0; i < N; i++) {
            x[i] = 0;
            for (int j = 0; j < M; j++) {
                if (mat[i][j] > x[i]) x[i] = mat[i][j];
            }
        }
        for (int j = 0; j < M; j++) { y[j] = 0; }
        memset(link, -1, sizeof(link));
        for (int i = 0; i < N; i++) {
            while (1) {
                memset(sx, 0, sizeof(sx));
                memset(sy, 0, sizeof(sy));
                slack = INF;
                if (DFS(i)) break;
                for (int j = 0; j < N; j++) {
                    if (sx[j]) x[j] -= slack;
                }
                for (int j = 0; j < M; j++) {
                    if (sy[j]) y[j] += slack;
                }
            }
        }

        int ans = 0;
        int cnt = 0;
        int t;
        for (int i = 0; i < M; i++)
        {
            t = link[i];
            if (t >= 0 && mat[t][i] != -INF)
            {
                cnt ++;
                ans += mat[t][i];
            }
        }
        // 最大權 : 沒有負號
        return -ans;
    }
    void init(int n,int m) {
        N = n, M = m;
        for (int i = 0; i < N; i++)
            for (int j = 0; j < M; j++)
                mat[i][j] = -INF;
    }
    void input() {
        for(int i = 0; i < N; i++)
            for(int j =0;j<M;j++) {
                // fill in mat[i][j]
                // stands for the weighting , but
                    negative sign !
                // if 最大權 : 沒有負號
            }

    }
}km;

int main(){
    int n,E;
    while (scanf("%d", &n) != EOF)
    {
        km.init(n, n);
        km.input();
        cout<< km.KM() <<endl;
    }
    return 0;
}
```

2.4 Bi-vertex-connected Subgraph

```
#include <bits/stdc++.h>
using namespace std;
#ifdef DEBUG
    #define debug(...) printf(__VA_ARGS__)
#else
    #define debug(...) (void)0
#endif
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
const int inf = 0x7fffffff; //beware overflow
const LL INF = 0x7fffffffffffffff; //beware overflow
#define mem(x, y) memset(x, (y), sizeof(x) );
#define IOS ios_base::sync_with_stdio(0); cin.tie(0)
template<typename A, typename B>
ostream& operator <<(ostream &s, const pair<A,B> &p) {
     return s<<"("<<p.first<<","<<p.second<<")";
}
template<typename T>
ostream& operator <<(ostream &s, const vector<T> &c) {
    s << "[ ";
    for (auto it : c) s << it << " ";
    s << "]";
    return s;
}
template<typename T>
ostream& operator << (ostream &o, const set<T> &st) {
    o << "{";
    for (auto it=st.begin(); it!=st.end(); it++) o << (
        it==st.begin() ? "" : ", ") << *it;
    return o << "}";
}
template<typename T1, typename T2>
ostream& operator << (ostream &o, const map<T1, T2> &mp
    ) {
    o << "{";
    for (auto it=mp.begin(); it!=mp.end(); it++) {
        o << (it==mp.begin()?"":", ") << it->fi << ":"
            << it->se;
    }
    o << "}";
    return o;
}

//    regard every vbcc as a set of edges
/** needed for tarjan **/
#define maxn 100005
#define maxm 100005
int n, m;
struct Edge{int s, t;};
vector<Edge> edge;
```

```
int dfn[maxn], low[maxn];
stack<int> st;
bool vis[maxn];
int Time;
bool vis_e[maxm];
int bcnt, vbb[maxm];
vector<int> vb[maxm];
vector<int> G[maxn];
/** **/

void tarjan(int s){
    dfn[s] = low[s] = ++Time;
    vis[s] = true;
    for(int e_ind : G[s]){
        if(!vis_e[e_ind]){
            vis_e[e_ind] = true; st.push(e_ind);
            int to = edge[e_ind].s + edge[e_ind].t - s;
            if(!vis[to]){
                tarjan(to);
                low[s] = min(low[s], low[to]);
                if(low[to] >= dfn[s]){
                    vb[bcnt].clear();
                    while(1){
                        int t = st.top();st.pop();
                        vbb[t] = bcnt;
                        vb[bcnt].push_back(t);
                        if(t == e_ind) break;
                    }
                    bcnt++;
                }
            }else
                low[s] = min(low[s], dfn[to]);
        }
    }
}
void init_tarjan() {

    mem(vis, false); mem(vis_e, false);
    Time = bcnt = 0; edge.clear();
    for(int i = 1; i <= n; i++) G[i].clear();
}

int main() {

    cin >> n >> m;
    init_tarjan();
    for(int i = 0; i < m; i++) {
        int a, b; scanf("%d %d", &a, &b);
        edge.push_back(Edge{a,b});
        G[a].push_back((int)edge.size()-1);
        G[b].push_back((int)edge.size()-1);
    }
    tarjan(1);

}
```

2.5 Bi-edge-connected Subgraph

```


/** needed for tarjan **/
#define maxn 100005
#define maxm 100005
int n, m;
int dfn[maxn], low[maxn];
stack<int> st;
int Time;
int bcnt;
vector<int> G[maxn];
bool in_cyc[maxn];
/** **/

void tarjan(int s, int p){
    dfn[s] = low[s] = ++Time;
    st.push(s);
    for(int to : G[s]) if( to != p ){
        if(!dfn[to]) {
            tarjan(to, s);
            low[s] = min(low[s], low[to]);
            if( low[to] > dfn[s]) {
                // is cut_edge
                // pop stack 的過程也可以寫在這
                // 但最後(after tarjan)還要多判stack
                    not empty的情況
                /*
                if( low[to] > dfn[s]) {

                in_cyc[bcnt] = st.top()!=to;
                while(1){
                    int t = st.top();st.pop();
                    id[t] = bcnt;
                    if(t == to) break;
                }
                bcnt++;
                */
            }

            }
        }else
            low[s] = min(low[s], dfn[to]);
    }

    if(low[s] == dfn[s]){
        in_cyc[bcnt] = st.top()!=s;
        while(1){
            int t = st.top();st.pop();
            id[t] = bcnt;
            if(t == s) break;
        }
        bcnt++;
    }
}
void init_tarjan() {
    Time = bcnt = 0;
}
int main() {
  cin >> n >> m;
  init_tarjan();
  for(int i = 0; i < m; i++) {
        int a, b; scanf("%d %d", &a, &b);
        G[a].pb(b), G[b].pb(a);
  }
  mem( in_cyc , false);
  tarjan(1, 1);

}
```

2.6 SCC

```
#include <bits/stdc++.h>

using namespace std;
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second

const int inf = 1e9;
const LL INF = 1e18;
const int mod = 1e9+7;
#define maxn 100050

int n, m;
vector<int> g[maxn];
stack<int> Stack;
int scnt, Time;
int belong[maxn], dfn[maxn], low[maxn], indegree[maxn];
bool instack[maxn];
void input(){
  cin >> n >> m;
  for(int i = 0; i < m; i++){
    int a, b; scanf("%d%d", &a, &b);
    g[a].pb(b);
  }
}
void init() {
  scnt = Time = 0;
```

```
  for(int i = 1; i <= n; i++)
    g[i].clear();
  while(!Stack.empty()) Stack.pop();
  memset(indegree, 0, sizeof(indegree));
  memset(dfn, 0, sizeof(dfn));
  memset(instack, false, sizeof(instack));
}
void dfs(int u) {
  dfn[u] = low[u] = ++Time;
  Stack.push(u); instack[u] = true;
  for(int v : g[u]) {
    if ( !dfn[v] ) {
      dfs(v);
      low[u] = min(low[u], low[v]);
    }
    else if(instack[v])
      low[u] = min(low[u], dfn[v]);
  }
  if(low[u] == dfn[u]) {
    scnt++;
    int tp;
    do{
      tp = Stack.top(); Stack.pop();
      instack[tp] = false;
      belong[tp] = scnt;
    } while(tp != u);
  }
}
void tarjan() {
  for(int i = 1; i <= n; i++)
    if(!dfn[i])
      dfs(i);
}
int main(){
  int T;cin >> T;
  while(T--){
    init();
    input();
    tarjan();
    for(int i = 1; i <= n; i++) {
      for(int v : g[i]) {
        if(belong[v] != belong[i])
          indegree[belong[v]]++;
      }
    }
    LL ans = 0;
    for(int i = 1; i <= scnt; i++)
      if(!indegree[i]) ans++;
    cout << ans << endl;
  }
  return 0;
}
```

2.7 Steiner Tree(PECaveros)

```
// Minimum Steiner Tree
// O(V 3^T + V^2 2^T)
struct SteinerTree{
#define V 33
#define T 8
#define INF 1023456789
  int n, dst[ V ][ V ], dp[ 1 << T ][ V ], tdst[ V
      ];
  void init( int _n ){
    n = _n;
    for( int i = 0 ; i < n ; i ++ ){
      for( int j = 0 ; j < n ; j ++ )
        dst[ i ][ j ] = INF;
      dst[ i ][ i ] = 0;
    }
  }
  void add_edge( int ui , int vi , int wi ){
    dst[ ui ][ vi ] = min( dst[ ui ][ vi ] , wi );
    dst[ vi ][ ui ] = min( dst[ vi ][ ui ] , wi );
  }
  void shortest_path(){
    for( int k = 0 ; k < n ; k ++ )
      for( int i = 0 ; i < n ; i ++ )
        for( int j = 0 ; j < n ; j ++ )
          dst[ i ][ j ] = min( dst[ i ][ j ],
                              dst[ i ][ k ] + dst[ k
                                  ][ j ] );
  }
  int solve( const vector<int>& ter ){
    int t = (int)ter.size();
    for( int i = 0 ; i < ( 1 << t ) ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        dp[ i ][ j ] = INF;
    for( int i = 0 ; i < n ; i ++ )
      dp[ 0 ][ i ] = 0;
    for( int msk = 1 ; msk < ( 1 << t ) ; msk ++ ){
      if( msk == ( msk & (-msk) ) ){
        int who = __lg( msk );
        for( int i = 0 ; i < n ; i ++ )
          dp[ msk ][ i ] = dst[ ter[ who ] ][ i ];
        continue;
      }
      for( int i = 0 ; i < n ; i ++ )
        for( int submsk = ( msk - 1 ) & msk ; submsk ;
                submsk = ( submsk - 1 ) & msk )
          dp[ msk ][ i ] = min( dp[ msk ][ i ],
                               dp[ submsk ][ i ] +
                                   dp[ msk ^ submsk
                                   ][ i ] );
      for( int i = 0 ; i < n ; i ++ ){
        tdst[ i ] = INF;
        for( int j = 0 ; j < n ; j ++ )
          tdst[ i ] = min( tdst[ i ],
                           dp[ msk ][ j ] + dst[ j ][ i
                               ] );
      }
      for( int i = 0 ; i < n ; i ++ )
        dp[ msk ][ i ] = tdst[ i ];
    }
    int ans = INF;
    for( int i = 0 ; i < n ; i ++ )
      ans = min( ans , dp[ ( 1 << t ) - 1 ][ i ] );
    return ans;
  }
} solver;
```

2.8 Edmond's Matching Algorithm

```
//带花树,Edmonds's matching algorithm,一般图最大匹配
//Problem:http://acm.timus.ru/problem.aspx?space=1&num
    =1099
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;
const int N=250;
int n;
int head;
int tail;
int Start;
int Finish;
int link[N];      //表示哪个点匹配了哪个点
int Father[N];    //这个就是增广路的Father……但是用起来
    太精髓了
int Base[N];      //该点属于哪朵花
int Q[N];
bool mark[N];
bool map[N][N];
bool InBlossom[N];
bool in_Queue[N];

void CreateGraph(){
    int x,y;
    scanf("%d",&n);
    while (scanf("%d%d",&x,&y)!=EOF)
      map[x][y]=map[y][x]=1;
}

void BlossomContract(int x,int y){
    fill(mark,mark+n+1,false);
    fill(InBlossom,InBlossom+n+1,false);
    #define pre Father[link[i]]
    int lca,i;
```

```
    for (i=x;i;i=pre) {i=Base[i]; mark[i]=true; }
    for (i=y;i;i=pre) {i=Base[i]; if (mark[i]) {lca=i;
        break;} }  //寻找lca之旅……一定要注意i=Base[i]
    for (i=x;Base[i]!=lca;i=pre){
        if (Base[pre]!=lca) Father[pre]=link[i]; //对于
            BFS树中的父边是匹配边的点，Father向后跳
        InBlossom[Base[i]]=true;
        InBlossom[Base[link[i]]]=true;
    }
    for (i=y;Base[i]!=lca;i=pre){
        if (Base[pre]!=lca) Father[pre]=link[i]; //同理
        InBlossom[Base[i]]=true;
        InBlossom[Base[link[i]]]=true;
    }
    #undef pre
    if (Base[x]!=lca) Father[x]=y;      //注意不能从lca
        这个奇环的关键点跳回来
    if (Base[y]!=lca) Father[y]=x;
    for (i=1;i<=n;i++)
      if (InBlossom[Base[i]]){
          Base[i]=lca;
          if (!in_Queue[i]){
              Q[++tail]=i;
              in_Queue[i]=true;      //要注意如果本来连
                  向BFS树中父结点的边是非匹配边的点，可
                  能是没有入队的
          }
      }
}

void Change(){
    int x,y,z;
    z=Finish;
    while (z){
        y=Father[z];
        x=link[y];
        link[y]=z;
        link[z]=y;
        z=x;
    }
}

void FindAugmentPath(){
    fill(Father,Father+n+1,0);
    fill(in_Queue,in_Queue+n+1,false);
    for (int i=1;i<=n;i++) Base[i]=i;
    head=0; tail=1;
    Q[1]=Start;
    in_Queue[Start]=1;
    while (head!=tail){
        int x=Q[++head];
        for (int y=1;y<=n;y++)
          if (map[x][y] && Base[x]!=Base[y] && link[x
              ]!=y)   //无意义的边
            if ( Start==y || link[y] && Father[link[y]]
                )    //精髓地用Father表示该点是否
                BlossomContract(x,y);
            else if (!Father[y]){
                Father[y]=x;
                if (link[y]){
                    Q[++tail]=link[y];
                    in_Queue[link[y]]=true;
                }
                else{
                    Finish=y;
                    Change();
                    return;
                }
            }
    }
}

void Edmonds(){
    memset(link,0,sizeof(link));
    for (Start=1;Start<=n;Start++)
      if (link[Start]==0)
        FindAugmentPath();
}

void output(){
    fill(mark,mark+n+1,false);
    int cnt=0;
    for (int i=1;i<=n;i++)
      if (link[i]) cnt++;
    printf("%d\n",cnt);
    for (int i=1;i<=n;i++)
      if (!mark[i] && link[i]){
          mark[i]=true;
          mark[link[i]]=true;
          printf("%d %d\n",i,link[i]);
      }
}

int main(){
//    freopen("input.txt","r",stdin);
    CreateGraph();
    Edmonds();
    output();
    return 0;
}
```

2.9 Tree Decomposition

```
//codeforces Digit Tree
//http://codeforces.com/problemset/problem/715/C
#include <bits/stdc++.h>
using namespace std;
#ifdef DEBUG
    #define debug(...) printf(__VA_ARGS__)
#else
    #define debug(...) (void)0
#endif
#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second
#define all(x) (x).begin(),(x).end()
#define SZ(x) ((int)(x).size())
const int inf = 0x7fffffff; //beware overflow
const LL INF = 0x7fffffffffffffff; //beware overflow
#define mem(x, y) memset(x, (y), sizeof(x) );
#define IOS ios_base::sync_with_stdio(0); cin.tie(0)
template<typename A, typename B>
ostream& operator <<(ostream &s, const pair<A,B> &p) {
    return s<<"("<<p.first<<","<<p.second<<")";
}
template<typename T>
ostream& operator <<(ostream &s, const vector<T> &c) {
    s << "[ ";
    for (auto it : c) s << it << " ";
    s << "]";
    return s;
}
template<typename T>
ostream& operator << (ostream &o, const set<T> &st) {
    o << "{";
    for (auto it=st.begin(); it!=st.end(); it++) o << (
        it==st.begin() ? "" : ", ") << *it;
    return o << "}";
}
template<typename T1, typename T2>
ostream& operator << (ostream &o, const map<T1, T2> &mp
    ) {
    o << "{";
    for (auto it=mp.begin(); it!=mp.end(); it++) {
        o << (it==mp.begin()?"":", ") << it->fi << ":"
            << it->se;
    }
    o << "}";
    return o;
}

typedef long long ll;

bool isprime[100005];
vector<LL> primes;
LL M, PHI;
```

```
#define MOD M
ll modpow(ll a, ll b) {
  ll r = 1;
  while(b) {
    if(b&1) r=(r*a)%MOD;
    a=(a*a)%MOD;
    b >>= 1;
  }
  return r;
}
void Sieve(int n) {
  memset(isprime, 1, sizeof(isprime));
  isprime[1] = false;
  for(int i = 2; i <= n; i++) {
    if(isprime[i]) {
      primes.pb(i);
      for(int j = 2*i; j <= n; j += i)
        isprime[j] = false;
    }
  }
}

LL phi(LL n) {
  ll num = 1; ll num2 = n;
  for(ll i = 0; primes[i]*primes[i] <= n; i++) {
    if(n%primes[i]==0) {
      num2/=primes[i];
      num*=(primes[i]-1);
    }
    while(n%primes[i]==0) {
      n/=primes[i];
    }
  }
  if(n>1) {
    num2/=n; num*=(n-1);
  }
  n = 1;
  num *= num2;
  return num;
}
ll inv(ll a) {
  return modpow(a, PHI-1);
}
#define maxn 100005
struct edge{
    int u, v, dig;
    int no(int x) {
        return x == u ? v : u;
    }
};
vector<edge> e;
vector<int> G[maxn];
LL n,  ans;
bool vis[maxn];
int sz[maxn], dep[maxn];
LL tenPow[maxn];
int dfs(int u, int p, int d) {
    sz[u] = 1;
    dep[u] = d;
    for(int eind : G[u] ) {
        int v = e[eind].no(u);
        if(  v == p || vis[v] ) continue;
        sz[u] += dfs(v, u, d+1);
    }
    return sz[u];
}
int findCenter(int u, int p, int treesize) {
    for(int eind : G[u] ) {
        int v = e[eind].no(u);
        if( v == p || vis[v] ) continue;
        if( sz[v]*2 > treesize)
            return findCenter( v, u, treesize);
    }
    return u;
}

LL up[maxn], down[maxn];
int belong[maxn];
map<LL, LL> tot;
vector< map<LL, LL> > vec;
vector<int> pt;

void calc(int u, int p, int b, int d) {
    pt.pb( u );
    belong[u] = b;
    dep[u] = d;

    int id =  find_if( all(G[u]) ,[u,p](int x) { return
         e[x].no(u) == p; }) - G[u].begin();
    down[u] = ( down[p]*10 + e[ G[u][id] ].dig )%M;
    up[u] =  (tenPow[ d-1 ]*e[ G[u][id] ].dig + up[p] )
        %M;

    for(int eind : G[u]) {
        int v = e[eind].no(u);
        if( vis[v] || v == p ) continue;
        calc( v, u, b, d+1);
    }

    vec[b][ up[u] ]++;
    tot[ up[u] ]++;
}

LL solve(int cent) {
    //cent is the root now
    vector<int> L;
    for(int eind : G[cent]) {
        int v = e[eind].no(cent);
        if( !vis[v] ) {
            L.pb( v );
        }
    }
    vec.clear();
    vec.resize( SZ(L), {} );
    tot.clear();
    up[cent] = down[cent] = 0;
    dep[cent] = 0;
    pt.clear();
    for(int i = 0; i < SZ(L); i++)
        calc( L[i], cent, i, 1);

    LL ret = 0;
    for(int u : pt) {
        LL tmp = (-down[u]+M)%M;
        tmp = ( tmp*inv( tenPow[ dep[u] ] ))%M;
        ret += tot[ tmp ] - vec[ belong[u] ][ tmp ];
    }
    assert( (LL)count_if(all(pt), [] (int x) { return
        up[x] == 0; } ) == tot[0]);
    LL tmp = tot[0] + (LL)count_if(all(pt), [] (int x)
        { return down[x] == 0; } );
    debug("%lld\n", tmp);
    return ret+tmp;
}
void solveAll(int node) {
    dfs(node, -1, 0);
    int cent = findCenter(node, -1, sz[node]);
    ans += solve( cent );
    debug("%d %lld\n", cent, ans);
    vis[cent] = true;
    for(int eind : G[cent] ) {
        int v = e[eind].no(cent);
        if( vis[v] ) continue;
        solveAll(v);
    }
}
int main() {
    cin>>n>>M;
  Sieve( 100000 );
    PHI = phi(M);
    for(int i = 0; i < n-1; i++) {
        int a, b, c; scanf("%d %d %d", &a, &b, &c);
        G[a].pb( SZ(e) ); G[b].pb( SZ(e) );
        e.pb( edge{a, b, c} );
    }
    //init
    tenPow[0] = 1;
    for(int i = 1; i < maxn; i++) tenPow[i] = (tenPow[i
        -1]*10)%M;
    ans = 0;
    mem( vis, false);
    solveAll(0);
    cout<<ans<<endl;
}
```

2.10 Tree Longest Path

graph/Longest$_p ath_t ree(center).cpp$

3 Flow

3.1 Dinic Maxflow

```
/** Uva 820 **/
#include <bits/stdc++.h>

using namespace std;

#define mp make_pair
#define pb push_back
#define LL long long
#define pii pair<int,int>
#define PII pair<long long, long long>
#define fi first
#define se second

const int inf = 0x7fffffff;
const LL  INF = 0x7fffffffffffffff;
const int mod = 1e9+7;
#define maxn 105
int n;
struct edge{ int to, cap, rev; };
vector<edge> g[maxn];
int dis[maxn], iter[maxn];
void addedge(int from, int to, int cap) {
  g[to].pb(edge{from, cap, (int)g[from].size()});
  g[from].pb(edge{to, cap, (int)g[to].size()-1});
}
bool bfs(int s, int t) {
  memset(dis, -1, sizeof(dis));
  queue<int> que;
  que.push(s); dis[s] = 0;
  while(!que.empty()) {
    int tp = que.front(); que.pop();
    for(edge e : g[tp]) {
      if(e.cap > 0 && dis[e.to] == -1)
        dis[e.to] = dis[tp] + 1, que.push(e.to);
    }
  }
  return dis[t] != -1;
}
int dfs(int v, int t, int f) {
  if( v == t ) return f;
  for(int &i = iter[v]; i < g[v].size(); i++) {
    edge &e = g[v][i];
    if(e.cap > 0 && dis[v] < dis[e.to]) {
      int d = dfs(e.to, t, min(f, e.cap));
      if(d > 0) {
        e.cap -= d;
        g[e.to][e.rev].cap += d;
        f += d;
        return d;
      }
    }
  }
  return 0;
}
int dinic(int s, int t) {
  int ret = 0;
  while( bfs(s, t) ) {
    memset(iter, 0, sizeof(iter));
    int f;
    while(( f = dfs(s, t, inf) )> 0 )
      ret += f;
  }
  return ret;
}

void init() {
  for(int i = 1; i <= n; i++)
    g[i].clear();
}

int main(){
  int cnt = 1;
  while(scanf("%d", &n) == 1 && n != 0) {
    init();
    int s, t, c; scanf("%d%d%d", &s, &t, &c);
    while(c--) {
      int from, to, cap; scanf("%d%d%d", &from, &to, &
          cap);
      addedge(from ,to, cap);
    }
    printf("Network %d\n", cnt++);
    printf("The bandwidth is %d.\n\n", dinic(s, t));
  }
  return 0;
}
```

4 Data Structure

4.1 Disjoint Set

5 Math

5.1 Prime Table

```
#include <bits/stdc++.h>
using namespace std;
struct Prime_table {

    int prime[1000000]={2,3,5,7};
    int sz=4;
    // biggest prime < ub
    int ub=(1<<20);

    int check(int num){
        int k = 0;
        for(k = 0; k < sz && prime[k]*prime[k] <= num;
            k++){
            if( num % prime[k]==0)  return 0;
        }
        return 1;
    }
    void buildprime(){
        int currentPrime=7;
        int j=4;
        for(sz=4,j=4; currentPrime<ub; sz++, j=6-j){
            currentPrime=currentPrime+j;
            if (check(currentPrime)) {
               prime[sz] = currentPrime;
            }
            else{
               sz--;
            }
        }
     }
}ptable;
```

5.2 Miller Rabin Prime Test

```
#include <cstdio>
#include <vector>
#include <map>
#include <algorithm>
using namespace std;


long long mul(unsigned long long a, unsigned long long
    b, unsigned long long mod) {
    long long ret = 0;
    for (a %= mod, b %= mod; b != 0; b >>= 1, a <<= 1,
        a = a >= mod ? a - mod : a) {
        if (b&1) {
            ret += a;
            if (ret >= mod) ret -= mod;
        }
    }
    return ret;
}
```

```
long long mpow2(long long x, long long y, long long mod
    ) {
    long long ret = 1;
    while (y) {
        if (y&1)
            ret = mul(ret, x, mod);
        y >>= 1, x = mul(x, x, mod);
    }
    return ret % mod;
}
int isPrime(long long p, int it) { // implements by
    miller-babin
    if (p < 2)  return 0;
    if (p == 2) return 1;
    if (!(p&1)) return 0;
    long long q = p-1, a, t;
    int k = 0, b = 0;
    while (!(q&1))  q >>= 1, k++;

    while(it--) {
        a = rand()%(p-4) + 2;
        t = mpow2(a, q, p);
        b = (t == 1) || (t == p-1);
        for (int i = 1; i < k && !b; i++) {
            t = mul(t, t, p);
            if (t == p-1)
                b = 1;
        }
        if (b == 0)
            return 0;
    }

    return 1;
}


int main() {

    int testcase;
    scanf("%d", &testcase);
    while (testcase--) {
        long long n;
        scanf("%lld", &n);
        puts(isPrime(n, 1000)?"YES":"NO");
    }
    return 0;
}
```

5.3 Extended Euclidean Algorithm

```
/** normal gcd function using recursion **/
int gcd(int a, int b){
    if(b == 0) return a;
    return gcd(b, a%b);
}
// Find solution of ax + by = gcd(a, b)
// ps : x, y may be negative
int extgcd(int a, int b, int& x, int& y){
    int d = a;
    if(b != 0) {
        d = extgcd(b, a%b, y, x);
        y -= (a/b) * x;
    }else {
        x = 1, y = 0;
    }
    return d;
}
```

5.4 Gauss Elimination

```
// solving linear equations with gauss elimination
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

void print(vector< vector<double> > A) {
    int n = A.size();
    for (int i=0; i<n; i++) {
        for (int j=0; j<n+1; j++) {
            cout << A[i][j] << "\t";
            if (j == n-1) {
                cout << "| ";
            }
        }
        cout << "\n";
    }
    cout << endl;
}

vector<double> gauss(vector< vector<double> > A) {
    int n = A.size();

    for (int i=0; i<n; i++) {
        // Search for maximum in this column
        double maxEl = abs(A[i][i]);
        int maxRow = i;
        for (int k=i+1; k<n; k++) {
            if (abs(A[k][i]) > maxEl) {
                maxEl = abs(A[k][i]);
                maxRow = k;
            }
        }

        // Swap maximum row with current row (column by
            column)
        for (int k=i; k<n+1;k++) {
            double tmp = A[maxRow][k];
            A[maxRow][k] = A[i][k];
            A[i][k] = tmp;
        }

        // Make all rows below this one 0 in current
            column
        for (int k=i+1; k<n; k++) {
            double c = -A[k][i]/A[i][i];
            for (int j=i; j<n+1; j++) {
                if (i==j) {
                    A[k][j] = 0;
                } else {
                    A[k][j] += c * A[i][j];
                }
            }
        }
    }

    // Solve equation Ax=b for an upper triangular
        matrix A
    vector<double> x(n);
    for (int i=n-1; i>=0; i--) {
        x[i] = A[i][n]/A[i][i];
        for (int k=i-1;k>=0; k--) {
            A[k][n] -= A[k][i] * x[i];
        }
    }
    return x;
}

int main() {
    int n;
    cin >> n;

    vector<double> line(n+1,0);
    vector< vector<double> > A(n,line);

    // Read input data
    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
            cin >> A[i][j];
        }
    }

    for (int i=0; i<n; i++) {
        cin >> A[i][n];
    }

    // Print input
    print(A);
```

```
    // Calculate solution
    vector<double> x(n);
    x = gauss(A);

    // Print result
    cout << "Result:\t";
    for (int i=0; i<n; i++) {
        cout << x[i] << " ";
    }
    cout << endl;
}
```

5.5 FFT

```
typedef long double ld;
/* N must be 2^k and greater than array.size()
 * FFT( a );
 * FFT( b );
 * for(int i = 0; i<N; ++i) c[i] = conj(a[i] * b[i]);
 * FFT( c );
 * for(int i = 0; i<N; ++i) c[i] = conj(c[i]);
 * for(int i = 0; i<N; ++i) c[i] /= N;
 */
void FFT(vector< complex<ld> >& v) {
    int N = v.size();
    for(int i = 1, j = 0; i<N; ++i) {
        for(int k = N>>1; !((j^=k)&k); k>>=1);
        if(i>j) swap(v[i],v[j]);
    }
    for(int k = 2; k<=N; k<<=1) {
        ld w = -2.0*pi/k;
        complex<ld> deg(cos(w),sin(w));
        for(int j = 0; j<N; j+=k) {
            complex<ld> theta(1,0);
            for(int i = j; i<j+k/2; ++i) {
                complex<ld> a = v[i];
                complex<ld> b = v[i+k/2]*theta;
                v[i] = a+b;
                v[i+k/2] = (a-b);
                theta *= deg;
            }
        }
    }
}
```

5.6 NNT

```
/*
NTT( a );
NTT( b );
for(int i = 0; i<N; ++i)
    c[i] = (long long) a[i] * b[i] % mod;
NTT( c, true );
for(int i = 0; i<N; ++i)
    c[i] = (786433LL-12) * c[i] % mod;
*/

constexpr int mod = 786433;
constexpr int N = 65536;

void NTT(vector< int >& v, bool flag = false)
{
    for(int i = 1, j = 0; i<N; ++i)
    {
        for(int k = N>>1; !((j^=k)&k); k>>=1);
        if(i>j) swap(v[i],v[j]);
    }
    for(int k = 2; k<=N; k<<=1)
    {
        int deg = mypow(flag ? 524289 : 3, N / k);
        for(int j = 0; j<N; j+=k)
        {
            int theta = 1;
            for(int i = j; i<j+k/2; ++i)
            {
                int a = v[i];
                int b = (long long) v[i+k/2]*theta%mod;
                v[i] = (a+b) % mod;
                v[i+k/2] = (a-b+mod)%mod;
                theta = (long long) theta * deg % mod;
            }
        }
    }
}
```

6 string

6.1 Palindromic Tree

```
/**
    template

    len 表示所代表的回文子字串長度
    next[c] 表示所代表的回文子字串在頭尾各增加一個字符c
        後的回文字串其節點編號
    fail( sufflink ) 表示所代表的回文子字串不包括本身的
        最長後綴回文子串的節點編號
    cnt(非必要) 表示所代表的回文子字串在整體字串出現的
        次數(在建構完成後呼叫count()才能計算)
    //num(非必要) 表示所代表的回文子字串其後綴為回文字
        串的個數 <== not included
**/
struct palindromic_tree{
  struct node{
    int next[26],fail,len; /*這些是必要的元素*/
    int cnt ;              /*這些是額外維護的元素*/
    node(int l=0):fail(0),len(l),cnt(0),num(0){
      for(int i=0;i<26;++i)next[i]=0;
    }
  };
  std::vector<node >St;
  std::vector<char >s;
  int last,n;
  palindromic_tree():St(2),last(1),n(0){
    St[0].fail=1;
    St[1].len=-1;
    s.push_back(-1);
  }
  inline void clear(){
    St.clear();
    s.clear();
    last=1;
    n=0;
    St.push_back(0);
    St.push_back(-1);
    St[0].fail=1;
    s.push_back(-1);
  }
  inline int get_fail(int x){
    while(s[n-St[x].len-1]!=s[n])x=St[x].fail;
    return x;
  }
  inline void add(int c){
    s.push_back(c-='a');
    ++n;
    int cur=get_fail(last);
    if(!St[cur].next[c]){
      int now=St.size();
      St.push_back(St[cur].len+2);
      St[now].fail=St[get_fail(St[cur].fail)].next[c];
      /*不用擔心會找到空節點，由證明的過程可知*/
      St[cur].next[c]=now;
      St[now].num=St[St[now].fail].num+1;
    }
    last=St[cur].next[c];
    ++St[last].cnt;
  }
  inline void count(){/*cnt必須要在構造完後呼叫count()
      去計算*/
    std::vector<node>::reverse_iterator i=St.rbegin();
    for(;i!=St.rend();++i){
      St[i->fail].cnt+=i->cnt;
    }
  }
```

```
  inline int size(){/*傳回其不同的回文子串個數*/
    return St.size()-2;
  }
};
```

6.2 Suffix Array

```
/** Suffix Array **/
struct SuffixArray{

    /**
     DA(倍增)算法求 SA[N] 与 Rank[N] (时间O(NlogN),空
         间O(N))
    sa[i] : 表示 排在第i位的后缀 起始下标 , sa[ 0 .. n
        ] ==> sa[0] = n 是空字串
                                                consider
                                                    sa[ 1
                                                    .. n ]

                                                    stores
                                                     [ 0
                                                    .. n )
    Rank[i] : 表示后缀 suffix(i)排在第几, Rank[ 0 .. n
        ] ==> Rank[n] = 0 空字串
                                                consider
                                                    ==>
                                                    Rank[
                                                    0 .. n
                                                    )
                                                    stores
                                                     [ 1
                                                    .. n ]

    lcp[i] : 表示 suffix ( sa[i-1] ) 与 suffix ( sa[i]
        ) 的LCP 值, lcp[ 1 .. n ], lcp[1] = 0 (與空字串
        比較)
    h[i]: 表示 suffix(i)与其排名前一位的 LCP值, h[ 0 ..
         n )
    LCP : longest common prefix

    **/
    #define N maxn
    int cmp(int *r,int a,int b,int l){
        return (r[a]==r[b]) && (r[a+l]==r[b+l]);
    }
    // 用于比较第一关键字与第二关键字,
    // 比较特殊的地方是,预处理的时候,r[n]=0(小于前面出
        现过的字符)
    int wa[N],wb[N],ws[N],wv[N];
    int r[N], sa[N];
    int Rank[N], lcp[N];
    void DA(int n,int m){ //此处n比输入的n要多1, 为人工
        添加的一个字符, 用于避免CMP时越界
        int i,j,p,*x=wa,*y=wb;
        for(i=0;i<m;i++) ws[i]=0;
        for(i=0;i<n;i++) ws[x[i]=r[i]]++;
        for(i=1;i<m;i++) ws[i]+=ws[i-1];
        for(i=n-1;i>=0;i--) sa[--ws[x[i]]] = i;
        for(j=1,p=1;p<n;j*=2,m=p)
        {
            for(p=0,i=n-j;i<n;i++) y[p++]=i;
            for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-
                j;
            for(i=0;i<m;i++) ws[i]=0;
            for(i=0;i<n;i++) wv[i]=x[y[i]];
            for(i=0;i<n;i++) ws[wv[i]]++;
            for(i=1;i<m;i++) ws[i]+=ws[i-1];
            for(i=n-1;i>=0;i--) sa[--ws[wv[i]]]=y[i];
            for(swap(x,y),p=1,x[sa[0]]=0,i=1;i<n;i++)
                x[sa[i]] = cmp(y,sa[i-1],sa[i],j)?p-1:p
                    ++;
        }
    }
    void calLcp(int n){ // 此处N为实际长度
        int i,j,k=0;          // height[]的合法范围为 1-N
            , 其中0是结尾加入的字符
        for(i=1;i<=n;i++) Rank[sa[i]]=i;  // 根据SA求
            Rank
        for(i=0;i<n; lcp[Rank[i++]] = k ) // 定义: h[i]
            = height[ Rank[i] ]
        for(k?k--:0,j=sa[Rank[i]-1]; r[i+k]==r[j+k]; k
            ++); //根据 h[i] >= h[i-1]-1 来优化计算
            height过程
    }
    void init(char *s, int len) {
        for(int i = 0; i < len; i++) r[i] = (int)s[i];
        r[len] = 0;
    }
}SA;

char str[maxn];
int main() {
    scanf("%s",str);
    int n = strlen(str);

    SA.init(str, n);
    SA.DA(r,sa,n+1,128);  //注意区分此处为n+1,因为添加
        了一个结尾字符用于区别比较
    calLcp(n);


//    /** demonstrate
    assert(sa[0] == n);
    for(int i = 0; i <= n; i++) printf("%d ", sa[i]);
    printf("\n");
    assert(Rank[n] == 0);
    for(int i = 0; i <= n; i++) printf("%d ", Rank[i]);
    printf("\n");
    //height[0] 沒有意義
    assert(height[1] == 0); //since sa[0] is 空字串
    printf("  ");
    for(int i = 1; i <= n; i++) printf("%d ", height[i
        ]);
    printf("\n");
//    **/
}
```

6.3 Longest Palindromic Substring

palindromic substring.cpp