杂项

加速 `sync_with_stdio(0);cin.tie(0);cout.tie(0)`

随机数：`random_device` 或 `chrono::system_clock::now().time_since_epoch().count()`

最短路

dijkstra

$O(mlogn)$

```
struct Node {
    int y, v;
    Node(int _y, int _v) { y = _y, v = _v; }
};

int n, m, s, t, dist[100005];
vector<Node> edge[100005];
set<pair<int, int>> q;

void dijkstra(int s, int t) {
    q.clear();
    memset(dist, 127, sizeof(dist));
    dist[s] = 0;
    for (int i = 1; i <= n; i++) q.insert({dist[i], i});
    for (; !q.empty();) {
        int x = q.begin()->second;
        q.erase(q.begin());
        if (x == t || dist[x] > 1 << 30) break;
        for (auto i : edge[x]) {
            if (dist[x] + i.v < dist[i.y]) {
                q.erase({dist[i.y], i.y});
                dist[i.y] = dist[x] + i.v;
                q.insert({dist[i.y], i.y});
            }
        }
    }
    cout << dist[t] << endl;
}
```

SPFA

$O(km)$ k 为常数，最差为 $O(nm)$

```
struct Node{
    int v,w;
};

vector<Node>edge[N];
int dis[N],cnt[N],vis[N];
queue<int>q;

bool spfa(int n,int s){
    memset(dis,63,sizeof(dis));
    memset(cnt,0,sizeof(cnt));
    memset(vis,0,sizeof(vis));
    dis[s]=0,vis[s]=1;
    q.push(s);
    while(!q.empty()){
        int u=q.front();
        q.pop();
        vis[u]=0;
        for(auto x:edge[u]){
            int v=x.v,w=x.w;
            if(dis[v]>dis[u]+w){
                dis[v]=dis[u]+w;
                cnt[v]=cnt[u]+1;
                if(cnt[v]>=n)return false;
                if(!vis[v])q.push(v),vis[v]=1;
            }
        }
    }
    return true;
}
```

最小生成树

Kruskal

$O(mlogm)$ 加边

```
struct Node{
    int x,y,v;
    bool operator < (const Node &a){
        return v<a.v;
    }
}a[500001];

int n,m,fa[500001];

int find(int i){
    if(fa[i]==i)return i;
    return fa[i]=find(fa[i]);
}

int Kruskal(){
    for(int i=1;i<=n;i++)fa[i]=i;
    sort(a+1,a+1+m);
    int cnt=n,ans=0;
    for(int i=1;i<=m;i++){
        int x=find(a[i].x),y=find(a[i].y);
        if(x!=y){
            fa[x]=y;
            ans+=a[i].v;
            cnt--;
        }
        if(cnt==1)break;
    }
    return cnt;
}
```

Prim

$O(n^2)$ 加点

```
struct Node{
    int y,v;
    Node(int _y,int _v){y=_y;v=_v;}
};

vector<Node>edge[5001];
int n,m,dist[5001];
bool b[5001];

void Prim(){
    memset(b,0,sizeof(b));
    memset(dist,127,sizeof(dist));
    dist[1]=0;
    int ans=0,tot=0;
    for(;;){
        int x=-1;
        for(int j=1;j<=n;j++){
            if(!b[j]&&dist[j]<1<<30)
                if(x==-1||dist[j]<dist[x])x=j;
        }
        if(x==-1)break;
        b[x]=1;
        ++tot;
        ans+=dist[x];
        for(auto j:edge[x]){
            dist[j.y]=min(dist[j.y],j.v);
        }
    }
    if(tot==n)
        cout<<ans<<endl;
    else
    cout<<"-1\n";
}
```

LCA

最近公共祖先

倍增预处理 $O(nlogn)$ 查询 $O(logn)$

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 6e5+7;

int n,m,s;
vector<int>v[N];
int d[N],f[N][32];

void dfs(int x,int y){
  d[x]=d[y]+1;
  f[x][0]=y;
  for(int i=1;i<=31;i++){
    f[x][i]=f[f[x][i-1]][i-1];
  }
  for(int i=0;i<v[x].size();i++){
    if(d[v[x][i]]==0)dfs(v[x][i],x);
  }
}

int lca(int x,int y){
  if(d[x]<d[y])swap(x,y);
  for(int i=31;i>=0;i--){
    if(f[x][i]!=0&&d[f[x][i]]>=d[y])
      x=f[x][i];
  }
  if(x==y)return x;
  for(int i=31;i>=0;i--){
    if(f[x][i]!=0&&f[y][i]!=0&&f[x][i]!=f[y][i]){
      x=f[x][i];
      y=f[y][i];
    }
  }
  return f[x][0];
}

int main(){
  cin>>n>>m>>s;
  for(int i=1;i<n;i++){
    int x,y;
    cin>>x>>y;
    v[x].push_back(y);
    v[y].push_back(x);
  }
```

```
  dfs(s,0);
  for(int i=1;i<=m;i++){
    int x,y;
    cin>>x>>y;
    cout<<lca(x,y)<<endl;
  }
}
```

SG 板

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

int sg[1010];

int main() {
    int n, p;
    cin >> n >> p;
    for (int i = 1; i <= n; i++) {
        set<int> s;
        for (int j = 1; j <= i; j *= p) {
            s.insert(sg[i - j]);
        }
        while (s.count(sg[i])) sg[i]++;
        printf("%d %d\n", i, sg[i]);
    }
    return 0;
}
```

字符串

KMP

```
void kmp(){
    n=s.size()+1,m=p.size()+1;//字符串下标从 1 开始
    int j=0;
    nxt[1]=0;
    for(int i=2;i<=m;i++){
        while(j>0&&p[j+1]!=p[i])
            j=nxt[j];
        if(p[j+1]==p[i])
            j++;
        nxt[i]=j;
    }
    j=0;
    for(int i=1;i<=n;i++){
        while((j==m)||(j>0&&p[j+1]!=s[i]))
            j=nxt[j];
        if(p[j+1]==s[i])
            j++;
        f[i]=j;
    }
}
```

EXKMP(z-algorithm)

求 s 和他的后缀的最长公共前缀

```
void exkmp(){
    int L=1,R=0;
    z[1]=0;
    for(int i=2;i<=2;i++){
        if(i>R)
            z[i]=0;
        else{
            int k=i-L+1;
            z[i]=min(z[k],R-i+1);
        }
        while(i+z[i]<=n&&s[z[i]+1]==s[i+z[i]])
            ++z[i];
        if(i+z[i]-1>R)
            L=i,R=i+z[i]=1;
    }
}
```

Manacher

求出字符串中最长回文片段

```
void manacher(){
    n=s.size();
    t.resize(2*n+10);
    int m=0;
       t[0]='$';
    for(int i=0;i<n;i++){
        t[++m]=s[i],t[++m]='$';
    }
    int M=0,R=0;
    for(int i=0;i<m;i++){
        if(i>R)
            p[i]=1;
        else
            p[i]=min(p[2*M-i],R-i+1);
        while(i-p[i]>=0&&i+p[i]<=m&&t[i-p[i]]==t[i+p[i]])
            ++p[i];
        if(i+p[i]-1>R)
            M=i,R=i+p[i]-1;
    }
    int ans=0;
    for(int i=0;i<=m;i++){
        ans=max(ans,p[i]);
    }
    cout<<ans-1<<endl;
}
```

最小表示法

```
void getmin(string s){
    int n=s.size();
    s=s+s;
    int i=0,j=1;
    while(j<n){
        int k=0;
        while(k<n&&s[i+k]==s[j+k])
            ++k;
        if(s[i+k]>s[j+k])
            i+=k+1;
        else
            j+=k+1;
        if(i==j)j++;
        if(i>j)swap(i,j);
    }
    for(int k=i;k<=i+n;k++)cout<<s[k];
}
```

数论

快速幂

```
long long qpow(long long a, long long b) {
  long long res = 1;
  while (b > 0) {
    if (b & 1) res = res * a;
    a = a * a;
    b >>= 1;
  }
  return res;
}
```

exgcd

```cpp
i64 exgcd(i64 a, i64 b, i64 &x, i64 &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    i64 g = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return g;
}

// ax + b = 0 (mod m)
std::pair<i64, i64> sol(i64 a, i64 b, i64 m) {
    assert(m > 0);
    b *= -1;
    i64 x, y;
    i64 g = exgcd(a, m, x, y);
    if (g < 0) {
        g *= -1;
        x *= -1;
        y *= -1;
    }
    if (b % g != 0) {
        return {-1, -1};
    }
    x = x * (b / g) % (m / g);
    if (x < 0) {
        x += m / g;
    }
    return {x, m / g};
}
```

欧拉筛

```
std::vector<int> minp, primes;

void sieve(int n) {
    minp.assign(n + 1, 0);
    primes.clear();

    for (int i = 2; i <= n; i++) {
        if (minp[i] == 0) {
            minp[i] = i;
            primes.push_back(i);
        }

        for (auto p : primes) {
            if (i * p > n) {
                break;
            }
            minp[i * p] = p;
            if (p == minp[i]) {
                break;
            }
        }
    }
}

bool isprime(int n) {
    return minp[n] == n;
}
```

基姆拉尔森公式

年月日推导出当天的星期

```
const int d[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

bool isLeap(int y) {
    return y % 400 == 0 || (y % 4 == 0 && y % 100 != 0);
}

int daysInMonth(int y, int m) {
    return d[m - 1] + (isLeap(y) && m == 2);
}

int getDay(int y, int m, int d) {
    int ans = 0;
    for (int i = 1970; i < y; i++) {
        ans += 365 + isLeap(i);
    }
    for (int i = 1; i < m; i++) {
        ans += daysInMonth(y, i);
    }
    ans += d;
    return (ans + 2) % 7 + 1;
}
```

线段树

```
struct Tag {
    int mul, add;
};

struct Node {
    int len, sum;
    Tag tag;
} tr[N<<2];

Tag operator + (const Tag &a, const Tag &b) {
    return {a.mul * b.mul % mod, (a.add * b.mul + b.add) % mod};
}

Node operator + (const Node &l, const Node &r) {
    Node a;
    a.sum = (l.sum + r.sum) % mod;
    a.tag = {1, 0};
    a.len = l.len + r.len;
    return a;
}

void update(int now) {
    tr[now] = tr[ls] + tr[rs];
}

void settag(int now, Tag t) {
    tr[now].tag = tr[now].tag + t;
    tr[now].sum = (tr[now].sum * t.mul + tr[now].len * t.add) % mod;
}

void pushdown(int now) {
    if (tr[now].tag.mul != 1 || tr[now].tag.add) {
        settag(ls, tr[now].tag);
        settag(rs, tr[now].tag);
        tr[now].tag = {1, 0};
    }
}

void build(int now, int l, int r) {
    if (l == r) {
        tr[now] = {1, a[l], {1, 0}};
```

```
        return;
    }
    int mid = (l + r) >> 1;
    build(ls, l, mid);
    build(rs, mid + 1, r);
    update(now);
}

void modify(int now, int l, int r, int s, int t, Tag val) {
    if (l <= s && r >= t) {
        settag(now, val);
        return;
    }
    pushdown(now);
    int mid = (s + t) >> 1;
    if (l <= mid) modify(ls, l, r, s, mid, val);
    if (r > mid) modify(rs, l, r, mid + 1, t, val);
    update(now);
}

int query(int now, int l, int r, int s, int t) {
    if (l <= s && r >= t) return tr[now].sum;
    pushdown(now);
    int mid = (s + t) >> 1;
    int ans = 0;
    if (l <= mid) ans = (ans + query(ls, l, r, s, mid)) % mod;
    if (r > mid) ans = (ans + query(rs, l, r, mid + 1, t)) % mod;
    return ans;
}
```