# ACM 模板

Wajov

September 2, 2017

## 目录

# 1 字符串算法

## 1.1 Manacher

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, ans, p[N << 1];
char c, s[N << 1];
int main()
{
    s[0] = '$';
    while ((c = getchar()) != '\n')
    {
        s[++n] = '#';
        s[++n] = c;
    }
    s[++n] = '#';
    for (int i = 1, j = 0; i <= n; i++)
    {
        p[i] = i < j + p[j] ? min(p[(j << 1) - i], j + p[j] - i) : 1;
        while (s[i + p[i]] == s[i - p[i]])
            p[i]++;
        if (i + p[i] > j + p[j])
            j = i;
        ans = max(ans, p[i] - 1);
    }
    printf("%d\n", ans);
    return 0;
}
```

## 1.2 Knuth–Morris–Pratt

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, m, num, p[N], ans[N];
char a[N + 10], b[N + 10];
int main()
{
    scanf("%s%s", a + 1, b + 1);
    n = strlen(a + 1);
    m = strlen(b + 1);
    for (int i = 2, j = 0; i <= m; i++)
    {
        for (; j > 0 && b[j + 1] != b[i]; j = p[j]);
        if (b[j + 1] == b[i])
            j++;
        p[i] = j;
    }
    for (int i = 1, j = 0; i <= n; i++)
    {
        for (; j > 0 && b[j + 1] != a[i]; j = p[j]);
        if (b[j + 1] == a[i])
            j++;
        if (j == m)
        {
            ans[++num] = i - j + 1;
            j = p[j];
        }
    }
    for (int i = 1; i < num; i++)
        printf("%d␣", ans[i]);
    printf("%d\n", ans[num]);
    return 0;
}
```

## 1.3 扩展 Knuth–Morris–Pratt

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, m, p[N], ex[N];
char a[N + 10], b[N + 10];
int main()
{
    scanf("%s%s", a + 1, b + 1);
    n = strlen(a + 1);
    m = strlen(b + 1);
    for (int i = 2, j = 0; i <= m; i++)
        {
            p[i] = i < j + p[j] ? min(p[i - j + 1], j + p[j] - i) : 0;
            for (; i + p[i] <= m && b[i + p[i]] == b[p[i] + 1]; p[i]++);
            if (i + p[i] > j + p[j])
                j = i;
        }
        for (int i = 1, j = 0; i <= n; i++)
        {
            ex[i] = i <= j + ex[j] ? min(p[i - j + 1], j + ex[j] - i) : 0;
            for (; i + ex[i] <= n && ex[i] < m && a[i + ex[i]] == b[ex[i] + 1]; ex[i]++);
            if (i + ex[i] > j + ex[j])
                j = i;
        }
    for (int i = 1; i < n; i++)
        printf("%d␣", ex[i]);
    printf("%d\n", ex[n]);
    return 0;
}
```

## 1.4 Aho–Corasick

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, t, tmp, now, pos, ans, son[N][26], num[N], p[N];
char a[N + 10], b[N + 10];
queue<int> q;
void Insert(char s[])
{
    int t = 1, tmp;
    for (int i = 0; s[i]; i++)
    {
        tmp = s[i] - 97;
        if (!son[t][tmp])
            son[t][tmp] = ++pos;
        t = son[t][tmp];
    }
    num[t]++;
}
int main()
{
    pos = 1;
    scanf("%s%d", a, &n);
    for (int i = 1; i <= n; i++)
    {
        scanf("%s", b);
        Insert(b);
    }
    q.push(1);
    while (!q.empty())
    {
        now = q.front();
        q.pop();
        for (int i = 0; i < 26; i++)
            if (son[now][i])
            {
                for (t = p[now]; t > 0 && son[t][i] == 0; t = p[t]);
                p[son[now][i]] = t ? son[t][i] : 1;
                q.push(son[now][i]);
            }
    }
    t = 1;
    for (int i = 0; a[i]; i++)
    {
        tmp = a[i] - 97;
        for (; t > 0 && son[t][tmp] == 0; t = p[t]);
        t = t ? son[t][tmp] : 1;
        for (int j = t; j > 1 && num[j] > -1; j = p[j])
        {
            ans += num[j];
            num[j] = -1;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

### 1.5 后缀数组

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int n, a[N], b[N], sum[N], tmp[N], id[N], rk[N + 10];
char s[N + 10];
void Sort(int a[], int m)
{
    memset(sum, 0, sizeof(sum));
    for (int i = 1; i <= n; i++)
        sum[a[i]]++;
    for (int i = 1; i <= m; i++)
        sum[i] += sum[i - 1];
    for (int i = n; i; i--)
        tmp[id[i]] = sum[a[id[i]]]--;
    for (int i = 1; i <= n; i++)
        id[tmp[i]] = i;
}
int main()
{
    scanf("%s", s + 1);
    n = strlen(s + 1);
    for (int i = 1; i <= n; i++)
        a[id[i] = i] = s[i] - 97;
    Sort(a, 25);
    for (int i = 1; i <= n; i <<= 1)
    {
        for (int j = 1, t = 0; j <= n; j++)
            rk[id[j]] = a[id[j]] == a[id[j - 1]] && b[id[j]] == b[id[j - 1]] ? t : ++t;
        for (int j = 1; j <= n; j++)
        {
            a[j] = rk[j];
            b[j] = rk[min(i + j, n + 1)];
        }
        Sort(b, n);
        Sort(a, n);
    }
    for (int i = 1; i < n; i++)
        printf("%d␣", rk[i]);
    printf("%d\n", rk[n]);
    return 0;
}
```

# 2 图算法

## 2.1 拓扑排序

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001, M = 1000001;
int n, m, u, v, tot, num, Head[N], Next[M], Link[M], ans[N];
bool flag[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
void DFS(int x)
{
    flag[x] = true;
    for (int i = Head[x], j; i; i = Next[i])
        if (!flag[j = Link[i]])
            DFS(j);
    ans[++num] = x;
}
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
    }
    for (int i = 1; i <= n; i++)
        if (!flag[i])
            DFS(i);
    for (int i = n; i > 1; i--)
        printf("%d␣", ans[i]);
    printf("%d\n", ans[1]);
    return 0;
}
```

## 2.2 Floyd–Warshall

```
#include <bits/stdc++.h>
using namespace std;
const int N = 101;
int n, m, u, v, c, d[N][N];
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            d[i][j] = i == j ? 0 : INT_MAX >> 1;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &c);
        d[u][v] = d[v][u] = min(d[u][v], c);
    }
    for (int k = 1; k <= n; k++)
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j < n; j++)
            printf("%d␣", d[i][j] == INT_MAX >> 1 ? -1 : d[i][j]);
        printf("%d\n", d[i][n] == INT_MAX >> 1 ? -1 : d[i][n]);
    }
    return 0;
}
```

## 2.3 Floyd-Warshall（最小环）

```
#include <bits/stdc++.h>
using namespace std;
const int N = 101;
int n, m, u, v, c, t, num, Min, a[N][N], d[N][N], p[N][N], ans[N];
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
        {
            a[i][j] = i == j ? 0 : INT_MAX / 3;
            p[i][j] = i;
        }
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &c);
        a[u][v] = a[v][u] = min(a[u][v], c);
    }
    memcpy(d, a, sizeof(d));
    Min = INT_MAX / 3;
    for (int k = 1; k <= n; k++)
    {
        for (int i = 1; i < k; i++)
            for (int j = 1; j < i; j++)
                if (d[i][j] + a[i][k] + a[k][j] < Min)
                {
                    Min = d[i][j] + a[i][k] + a[k][j];
                    for (num = 0, t = j; t != i; t = p[i][t])
                        ans[++num] = t;
                    ans[++num] = i;
                    ans[++num] = k;
                }
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                if (d[i][k] + d[k][j] < d[i][j])
                {
                    d[i][j] = d[i][k] + d[k][j];
                    p[i][j] = p[k][j];
                }
    }
    printf("%d\n", Min);
    for (int i = 1; i < num; i++)
        printf("%d␣", ans[i]);
    printf("%d\n", ans[num]);
    return 0;
}
```

## 2.4 Bellman-Ford+ 队列

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001, M = 100001;
int n, m, s, u, v, c, now, tot, Head[N], Next[M << 1], Link[M << 1], Cost[M << 1], d[N];
bool flag[N];
queue<int> q;
inline void AddEdge(int u, int v, int c)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Cost[tot] = c;
    Head[u] = tot;
}
int main()
{
    scanf("%d%d%d", &n, &m, &s);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &c);
        AddEdge(u, v, c);
        AddEdge(v, u, c);
    }
    for (int i = 1; i <= n; i++)
        d[i] = INT_MAX;
    d[s] = 0;
    q.push(s);
    flag[s] = true;
    while (!q.empty())
    {
        now = q.front();
        q.pop();
        flag[now] = false;
        for (int i = Head[now], j; i; i = Next[i])
            if (d[now] + Cost[i] < d[j = Link[i]])
            {
                d[j] = d[now] + Cost[i];
                if (!flag[j])
                {
                    q.push(j);
                    flag[j] = true;
                }
            }
    }
    for (int i = 1; i < n; i++)
        printf("%d␣", d[i]);
    printf("%d\n", d[n]);
    return 0;
}
```

## 2.5 Dijkstra+ 堆

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001, M = 100001;
int n, m, s, u, v, c, now, tot, d[N], Head[N], Next[M << 1], Link[M << 1], Cost[M << 1];
bool flag[N];
priority_queue<pair<int, int> > q;
inline void AddEdge(int u, int v, int c)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Cost[tot] = c;
    Head[u] = tot;
}
int main()
{
    scanf("%d%d%d", &n, &m, &s);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &c);
        AddEdge(u, v, c);
        AddEdge(v, u, c);
    }
    for (int i = 1; i <= n; i++)
        d[i] = INT_MAX;
    q.push(make_pair(d[s] = 0, s));
    while (!q.empty())
    {
        now = q.top().second;
        q.pop();
        if (flag[now])
            continue;
        flag[now] = true;
        for (int i = Head[now], j; i; i = Next[i])
            if (d[now] + Cost[i] < d[j = Link[i]])
            {
                d[j] = d[now] + Cost[i];
                q.push(make_pair(-d[j], j));
            }
    }
    for (int i = 1; i < n; i++)
        printf("%d␣", d[i] == INT_MAX ? -1 : d[i]);
    printf("%d\n", d[n] == INT_MAX ? -1 : d[n]);
    return 0;
}
```

## 2.6 Kruskal

```
#include <bits/stdc++.h>
#define fi first
#define se second
using namespace std;
const int N = 100001;
int n, m, x, y, ans, a[N];
pair<int, pair<int, int> > e[N];
int Get(int x)
{
        if (a[x] != x)
                a[x] = Get(a[x]);
        return a[x];
}
int main()
{
        scanf("%d%d", &n, &m);
        for (int i = 1; i <= m; i++)
                scanf("%d%d%d", &e[i].se.fi, &e[i].se.se, &e[i].fi);
    for (int i = 1; i <= n; i++)
        a[i] = i;
        sort(e + 1, e + m + 1);
        for (int i = 1; i <= m; i++)
        {
                x = Get(e[i].se.fi);
                y = Get(e[i].se.se);
                if (x != y)
                {
                        a[x] = y;
                        ans += e[i].fi;
                }
        }
        printf("%d\n", ans);
        return 0;
}
```

## 2.7 Prim+ 堆

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001, M = 100001;
int n, m, s, u, v, c, now, ans, tot, Head[N], Next[M << 1], Link[M << 1], Cost[M << 1], d[N];
bool flag[N];
priority_queue<pair<int, int> > q;
inline void AddEdge(int u, int v, int c)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Cost[tot] = c;
    Head[u] = tot;
}
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &c);
        AddEdge(u, v, c);
        AddEdge(v, u, c);
    }
    for (int i = 1; i <= n; i++)
        d[i] = INT_MAX;
    q.push(make_pair(d[1] = 0, 1));
    while (!q.empty())
    {
        now = q.top().second;
        q.pop();
        if (flag[now])
            continue;
        ans += d[now];
        flag[now] = true;
        for (int i = Head[now], j; i; i = Next[i])
            if (Cost[i] < d[j = Link[i]])
            {
                d[j] = Cost[i];
                q.push(make_pair(-d[j], j));
            }
    }
    printf("%d\n", ans);
    return 0;
}
```

## 2.8 Tarjan（强连通分量）

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001, M = 1000001;
int n, m, u, v, tot, num, idx, Head[N], Next[M], Link[M], dfn[N], low[N];
bool flag[N];
stack<int> s;
vector<int> sub[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
void DFS(int x)
{
    s.push(x);
    flag[x] = true;
    low[x] = dfn[x] = ++idx;
    for (int i = Head[x], j; i; i = Next[i])
        if (!dfn[j = Link[i]])
        {
            DFS(j);
            low[x] = min(low[x], low[j]);
        }
        else if (flag[j])
            low[x] = min(low[x], dfn[j]);
    if (low[x] == dfn[x])
    {
        int t;
        num++;
        do
        {
            t = s.top();
            s.pop();
            flag[t] = false;
            sub[num].push_back(t);
        }
        while (t != x);
    }
}
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i])
```

```
            DFS(i);
    printf("%d\n", num);
    for (int i = 1; i <= num; i++)
    {
        for (int j = 0; j < sub[i].size() - 1; j++)
            printf("%d␣", sub[i][j]);
        printf("%d\n", sub[i][sub[i].size() - 1]);
    }
    return 0;
}
```

## 2.9 Tarjan（点双连通分量）

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001, M = 1000001;
int n ,m, u, v, tot, num, idx, Head[N], Next[M << 1], Link[M << 1], dfn[N], low[N];
bool flag[N];
stack<int> s;
vector<int> sub[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
void DFS(int x, int y)
{
    s.push(x);
    flag[x] = true;
    low[x] = dfn[x] = ++idx;
    for (int i = Head[x], j; i; i = Next[i])
    {
        if ((j = Link[i]) == y)
            continue;
        if (!dfn[j])
        {
            DFS(j, x);
            low[x] = min(low[x], low[j]);
        }
        else if (flag[j])
            low[x] = min(low[x], dfn[j]);
    }
    if (x != y && low[x] >= dfn[y])
    {
        int t;
        num++;
        do
        {
            t = s.top();
            s.pop();
            flag[t] = false;
            sub[num].push_back(t);
        }
        while (t != y);
        s.push(y);
        flag[y] = true;
    }
}
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
        AddEdge(v, u);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i])
        {
```

```cpp
            DFS(i, i);
            s.pop();
            flag[i] = false;
        }
    printf("%d\n", num);
    for (int i = 1; i <= num; i++)
    {
        for (int j = 0; j < sub[i].size() - 1; j++)
            printf("%d␣", sub[i][j]);
        printf("%d\n", sub[i][sub[i].size() - 1]);
    }
    return 0;
}
```

## 2.10 Tarjan（边双连通分量）

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001, M = 1000001;
int n, m, u, v, tot, num, idx, Head[N], Next[M << 1], Link[M << 1], dfn[M << 1], low[N];
bool flag[N];
stack<int> s;
vector<int> sub[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
void DFS(int x, int y)
{
    s.push(x);
    flag[x] = true;
    low[x] = dfn[x] = ++idx;
    for (int i = Head[x], j; i; i = Next[i])
    {
        if ((j = Link[i]) == y)
            continue;
        if (!dfn[j])
        {
            DFS(j, x);
            low[x] = min(low[x], low[j]);
        }
        else if (flag[j])
            low[x] = min(low[x], dfn[j]);
    }
    if (low[x] > dfn[y])
    {
        int t;
        num++;
        do
        {
            t = s.top();
            s.pop();
            flag[t] = false;
            sub[num].push_back(t);
        }
        while (t != x);
    }
}
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
        AddEdge(v, u);
    }
    for (int i = 1; i <= n; i++)
        if (!dfn[i])
        {
            DFS(i, i);
            num++;
            while (!s.empty())
            {
                flag[s.top()] = false;
                sub[num].push_back(s.top());
                s.pop();
            }
        }
    printf("%d\n", num);
    for (int i = 1; i <= num; i++)
    {
        for (int j = 0; j < sub[i].size() - 1; j++)
            printf("%d␣", sub[i][j]);
        printf("%d\n", sub[i][sub[i].size() - 1]);
    }
    return 0;
}
```

## 2.11 匈牙利

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1001, M = 10001;
int n, m, k, u, v, tot, ans, Head[N], Next[M], Link[M], p[N];
bool flag[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
bool DFS(int x)
{
    for (int i = Head[x], j; i; i = Next[i])
        if (!flag[j = Link[i]])
        {
            flag[j] = true;
            if (p[j] == 0 || DFS(p[j]))
            {
                p[j] = x;
                return true;
            }
        }
    return false;
}
int main()
{
    scanf("%d%d%d", &n, &m, &k);
    for (int i = 1; i <= k; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
    }
    for (int i = 1; i <= n; i++)
    {
        memset(flag, false, sizeof(flag));
        if (DFS(i))
            ans++;
    }
    printf("%d\n", ans);
    return 0;
}
```

## 2.12 Kuhn-Munkres

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

```
using namespace std;
const int N = 101;
int n, m, t, ans, a[N][N], lx[N], ly[N], slack[N], p[N];
bool fx[N], fy[N];
bool DFS(int x)
{
    fx[x] = true;
    for (int i = 1, t; i <= m; i++)
        if (!fy[i])
        {
            t = lx[x] + ly[i] - a[x][i];
            if (!t)
            {
                fy[i] = true;
                if (p[i] == 0 || DFS(p[i]))
                {
                    p[i] = x;
                    return true;
                }
            }
            else
                slack[i] = min(slack[i], lx[x] + ly[i] - a[x][i]);
        }
    return false;
}
bool Find(int x)
{
    memset(fx, false, sizeof(fx));
    memset(fy, false, sizeof(fy));
    return DFS(x);
}
int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &a[i][j]);
    for (int i = 1; i <= n; i++)
    {
        lx[i] =  INT_MIN;
        for (int j = 1; j <= m; j++)
            lx[i] = max(lx[i], a[i][j]);
    }
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
            slack[j] = INT_MAX;
        while (!Find(i))
        {
            t = INT_MAX;
            for (int j = 1; j <= m; j++)
                if (!fy[j])
                    t = min(t, slack[j]);
            for (int j = 1; j <= n; j++)
                if (fx[j])
                    lx[j] -= t;
            for (int j = 1; j <= m; j++)
                if (fy[j])
                    ly[j] += t;
                else
                    slack[j] -= t;
        }
    }
    for (int i = 1; i <= m; i++)
        if (p[i])
            ans += a[p[i]][i];
    printf("%d\n", ans);
    return 0;
}
```

## 2.13 Dinic

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1001, M = 10001;
int n, m, S, T, u, v, r, tot, ans
int Head[N], cur[N], Next[M << 1], Link[M << 1], Rest[M << 1], d[N], From[N], Edge[N];
queue<int> q;
inline void AddEdge(int u, int v, int r)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Rest[tot] = r;
    Head[u] = tot;
}
bool BFS()
{
    for (int i = 1; i <= n; i++)
        d[i] = INT_MAX;
    d[S] = 0;
    q.push(S);
    while (!q.empty())
    {
        int now = q.front();
        q.pop();
        for (int i = Head[now], j; i; i = Next[i])
            if (Rest[i] > 0 && d[now] + 1 < d[j = Link[i]])
            {
                d[j] = d[now] + 1;
                q.push(j);
            }
    }
    return d[T] < INT_MAX;
}
bool DFS(int x)
{
    if (x == T)
    {
        int tmp = INT_MAX;
        for (int i = T; i != S; i = From[i])
            tmp = min(tmp, Rest[Edge[i]]);
        for (int i = T; i != S; i = From[i])
        {
            Rest[Edge[i]] -= tmp;
            Rest[Edge[i] ^ 1] += tmp;
        }
        ans += tmp;
        return true;
    }
    for (int &i = cur[x], j; i; i = Next[i])
        if (Rest[i] > 0 && d[x] + 1 == d[j = Link[i]])
        {
            From[j] = x;
            Edge[j] = i;
            if (DFS(j))
                return true;
        }
    return false;
}
int main()
{
    scanf("%d%d%d%d", &n, &m, &S, &T);
    tot = 1;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d", &u, &v, &r);
        AddEdge(u, v, r);
        AddEdge(v, u, 0);
    }
    while (BFS())
    {
        memcpy(cur, Head, sizeof(cur));
        while (DFS(S));
    }
    printf("%d\n", ans);
    return 0;
}
```

## 2.14 Edmonds-Karp（最小费用最大流）

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1001, M = 10001;
int n, m, S, T, u, v, r, c, tmp, tot, sum, ans1, ans2;
int Head[N], Next[M << 1], Link[M << 1], Rest[M << 1], Cost[M << 1], d[N], From[N], Edge[N];
bool flag[N];
queue<int> q;
inline void AddEdge(int u, int v, int r, int c)
{
        Next[++tot] = Head[u];
        Link[tot] = v;
        Rest[tot] = r;
        Cost[tot] = c;
        Head[u] = tot;
}
bool BFS()
{
        for (int i = 1; i <= n; i++)
                d[i] = INT_MAX;
        d[S] = 0;
        q.push(S);
        flag[S] = true;
        while (!q.empty())
        {
                int now = q.front();
                q.pop();
                flag[now] = false;
                for (int i = Head[now], j; i; i = Next[i])
                        if (Rest[i] > 0 && d[now] + Cost[i] < d[j = Link[i]])
                        {
                                d[j] = d[now] + Cost[i];
                                From[j] = now;
                                Edge[j] = i;
                                if (!flag[j])
                                {
                                        q.push(j);
                                        flag[j] = true;
                                }
                        }
        }
        return d[T] < INT_MAX;
}
int main()
{
    scanf("%d%d%d%d", &n, &m, &S, &T);
    tot = 1;
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d%d%d", &u, &v, &r, &c);
        AddEdge(u, v, r, c);
        AddEdge(v, u, 0, -c);
    }
    while (BFS())
    {
        sum = 0;
        tmp = INT_MAX;
        for (int i = T; i != S; i = From[i])
        {
            sum += Cost[Edge[i]];
            tmp = min(tmp, Rest[Edge[i]]);
        }
        for (int i = T; i != S; i = From[i])
        {
            Rest[Edge[i]] -= tmp;
            Rest[Edge[i] ^ 1] += tmp;
        }
        ans1 += tmp;
        ans2 += tmp * sum;
    }
    printf("%d␣%d\n", ans1, ans2);
        return 0;
}
```

# 3 树算法

## 3.1 Tarjan（最近公共祖先）

```
#include <bits/stdc++.h>
#define fi first
#define se second
using namespace std;
const int N = 1000001, M = 1000001;
int n, m, u, v, tot, Head[N], Next[N << 1], Link[N << 1], a[N], ans[M];
bool flag[N];
vector<pair<int, int> > Q[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
int Get(int x)
{
    if (a[x] != x)
        a[x] = Get(a[x]);
    return a[x];
}
void DFS(int x)
{
    flag[x] = true;
    a[x] = x;
    for (int i = 0; i < Q[x].size(); i++)
        ans[Q[x][i].se] = Get(a[Q[x][i].fi]);
    for (int i = Head[x], j; i; i = Next[i])
        if (!flag[j = Link[i]])
        {
            DFS(j);
            a[j] = x;
        }
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i < n; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
        AddEdge(v, u);
    }
    scanf("%d", &m);
    for (int i = 1; i <= m; i++)
    {
        scanf("%d%d", &u, &v);
        Q[u].push_back({v, i});
        Q[v].push_back({u, i});
    }
    DFS(1);
    for (int i = 1; i <= m; i++)
        printf("%d\n", ans[i]);
    return 0;
}
```

## 3.2 树链剖分

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int n, u, v, num, tot
int Head[N], Next[N << 1], Link[N << 1], d[N], f[N], s[N], son[N], top[N], idx[N], key[N];
inline void AddEdge(int u, int v)
{
    Next[++tot] = Head[u];
    Link[tot] = v;
    Head[u] = tot;
}
void DFS1(int x)
{
    d[x] = d[f[x]] + 1;
    s[x] = 1;
    for (int i = Head[x], j; i; i = Next[i])
        if (!d[j = Link[i]])
        {
            f[j] = x;
            DFS1(j);
            s[x] += s[j];
            if (s[j] > s[son[x]])
                son[x] = j;
        }
}
void DFS2(int x)
{
    top[x] = x == son[f[x]] ? top[f[x]] : x;
    key[idx[x] = ++num] = x;
    if (son[x])
        DFS2(son[x]);
    for (int i = Head[x], j; i; i = Next[i])
    {
        j = Link[i];
        if (f[j] == x && j != son[x])
            DFS2(j);
    }
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i < n; i++)
    {
        scanf("%d%d", &u, &v);
        AddEdge(u, v);
        AddEdge(v, u);
    }
    DFS1(1);
    DFS2(1);
    for (int i = 1; i < n; i++)
        printf("%d␣", key[i]);
    printf("%d\n", key[n]);
    return 0;
}
```

# 4 数据结构

## 4.1 字母树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int pos, son[N][26], num[N];
void Insert(char s[])
{
    int t = 1, tmp;
    for (int i = 0; s[i]; i++)
    {
        tmp = s[i] - 97;
        if (!son[t][tmp])
            son[t][tmp] = ++pos;
        t = son[t][tmp];
    }
    num[t]++;
}
int Find(char s[])
{
    int t = 1, tmp;
    for (int i = 0; s[i]; i++)
    {
        tmp = s[i] - 97;
        if (!son[t][tmp])
            return 0;
        t = son[t][tmp];
    }
    return num[t];
}
int main()
{
    pos = 1;
    return 0;
}
```

## 4.2 并查集

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, a[N], b[N];
int Find(int x)
{
    if (a[x] != x)
        a[x] = Find(a[x]);
    return a[x];
}
void Merge(int x, int y)
{
    if ((x = Find(x)) == (y = Find(y)))
        return;
    b[x] < b[y] ? a[x] = y : a[y] = x;
    if (b[x] == b[y])
        b[x]++;
}
int main()
{
    for (int i = 1; i <= n; i++)
        a[i] = i;
    return 0;
}
```

## 4.3 树状数组

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int n, sum[N];
void Add(int x, int y)
{
    for (; x <= n; x += x & -x)
        sum[x] += y;
}
int Sum(int x)
{
    int ans = 0;
    for (; x; x -= x & -x)
        ans += sum[x];
    return ans;
}
int main()
{
    return 0;
}
```

## 4.4 张昆玮线段树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int n, SIZE, a[N], sum[N << 2];
void Build()
{
    for (SIZE = 1; SIZE < n + 2; SIZE <<= 1);
    for (int i = 1; i <= n; i++)
        sum[SIZE + i] = a[i];
    for (int i = SIZE - 1; i; i--)
        sum[i] = sum[i << 1] + sum[(i << 1) + 1];
}
void Add(int x, int y)
{
    for (x += SIZE; x; x >>= 1)
        sum[x] += y;
}
int Sum(int x, int y)
{
    int ans = 0;
    for (x += SIZE - 1, y += SIZE + 1; x ^ y ^ 1; x >>= 1, y >>= 1)
    {
        if ((x & 1) == 0)
            ans += sum[x ^ 1];
        if ((y & 1) == 1)
            ans += sum[y ^ 1];
    }
    return ans;
}
int main()
{
    return 0;
}
```

## 4.5 线段树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int num, a[N], l[N << 1], r[N << 1], sum[N << 1], lab[N << 1];
inline void Label(int p, int x, int y, int z)
{
    sum[p] += (y - x + 1) * z;
    lab[p] += z;
}
inline void Down(int p, int x, int y)
{
    if (x < y)
    {
        int z = x + y >> 1;
        Label(l[p], x, z, lab[p]);
        Label(r[p], z + 1, y, lab[p]);
    }
    lab[p] = 0;
}
inline void Up(int p)
{
    sum[p] = sum[l[p]] + sum[r[p]];
}
void Build(int p, int x, int y)
```

```
{
    if (x == y)
    {
        sum[p] = a[x];
        return;
    }
    int z = x + y >> 1;
    Build(l[p] = ++num, x, z);
    Build(r[p] = ++num, z + 1, y);
    Up(p);
}
void Add(int p, int x, int y, int a, int b, int c)
{
    Down(p, x, y);
    if (x == a && y == b)
    {
        Label(p, x, y, c);
        return;
    }
    int z = x + y >> 1;
    if (b <= z)
        Add(l[p], x, z, a, b, c);
    else if (a > z)
        Add(r[p], z + 1, y, a, b, c);
    else
    {
        Add(l[p], x, z, a, z, c);
        Add(r[p], z + 1, y, z + 1, b, c);
    }
    Up(p);
}
int Sum(int p, int x, int y, int a, int b)
{
    Down(p, x, y);
    if (x == a && y == b)
        return sum[p];
    int z = x + y >> 1;
    if (b <= z)
        return Sum(l[p], x, z, a, b);
    else if (a > z)
        return Sum(r[p], z + 1, y, a, b);
    else
        return Sum(l[p], x, z, a, z) + Sum(r[p], z + 1, y, z + 1, b);
}
int main()
{
    num = 1;
    return 0;
}
```

## 4.6 伸展树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int root, pos, l[N], r[N], f[N], key[N], s[N], num[N];
inline void L(int p)
{
    int t = f[p];
    if (r[t] = l[p])
        f[l[p]] = t;
    if (f[p] = f[t])
        t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
    f[t] = p;
    l[p] = t;
    s[p] = s[t];
    s[t] = s[l[t]] + s[r[t]] + num[t];
}
inline void R(int p)
{
    int t = f[p];
    if (l[t] = r[p])
        f[r[p]] = t;
    if (f[p] = f[t])
        t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
    f[t] = p;
    r[p] = t;
    s[p] = s[t];
    s[t] = s[l[t]] + s[r[t]] + num[t];
}
void Splay(int p)
{
    for (int t; t = f[p]; )
        if (!f[t])
            p == l[t] ? R(p) : L(p);
        else
            if (p == l[t])
                t == l[f[t]] ? (R(t), R(p)) : (R(p), L(p));
            else
                t == r[f[t]] ? (L(t), L(p)) : (L(p), R(p));
    root = p;
}
void Insert(int x)
{
    int p, t;
    bool flag = false;
    for (p = root; p; p = x < key[p] ? l[p] : r[p])
    {
        t = p;
        s[p]++;
        if (key[p] == x)
        {
            flag = true;
            break;
        }
    }
    if (flag)
        num[p]++;
    else
    {
        p = ++pos;
        key[p] = x;
        s[p] = num[p] = 1;
        if (root)
        {
            f[p] = t;
            x < key[t] ? l[t] = p : r[t] = p;
        }
    }
    Splay(p);
}
void Delete(int x)
{
    int p, q, t;
    for (p = root; key[p] != x; p = x < key[p] ? l[p] : r[p])
        s[p]--;
    s[p]--;
    if (!(--num[p]))
        if (!l[p] || ! r[p])
        {
            if (p == root)
                root = l[p] + r[p];
            else
                p == l[f[p]] ? l[f[p]] = l[p] + r[p] : r[f[p]] = l[p] + r[p];
            f[l[p] + r[p]] = f[p];
        }
        else
        {
            for (q = l[p]; r[q]; q = r[q]);
            for (t = l[p]; r[t]; t = r[t])
                s[t] -= num[q];
            q == l[f[q]] ? l[f[q]] = l[q] + r[q] : r[f[q]] = l[q] + r[q];
            f[l[q] + r[q]] = f[q];
            key[p] = key[q];
            num[p] = num[q];
        }
}
int Rank(int x)
{
    int p = root, t = s[l[root]];
    while (key[p] != x)
        if (x < key[p])
        {
            p = l[p];
            t -= s[r[p]] + num[p];
        }
        else
        {
            t += num[p];
            p = r[p];
            t += s[l[p]];
        }
    Splay(p);
    return t + 1;
}
int Select(int x)
{
    int p = root, t = s[l[root]];
    while (x < t + 1 || x > t + num[p])
        if (x < t + 1)
        {
            p = l[p];
            t -= s[r[p]] + num[p];
        }
        else
        {
            t += num[p];
            p = r[p];
            t += s[l[p]];
        }
    Splay(p);
    return key[p];
}
int Pred(int x)
{
    int p = root, t;
    while (p)
        if (x > key[p])
        {
            t = p;
            p = r[p];
        }
        else
            p = l[p];
    Splay(t);
    return key[t];
}
int Succ(int x)
{
    int p = root, t;
    while (p)
        if (x < key[p])
        {
            t = p;
            p = l[p];
        }
        else
            p = r[p];
    Splay(t);
    return key[t];
}
int main()
{
    return 0;
}
```

## 4.7 伸展树（区间）

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int root, pos, l[N], r[N], f[N], s[N], key[N], lab[N], sum[N];
bool flag[N];
inline void Down(int p)
{
    if (l[p])
    {
        key[l[p]] += lab[p];
        lab[l[p]] += lab[p];
        sum[l[p]] += s[l[p]] * lab[p];
        if (flag[p])
        {
            flag[l[p]] = !flag[l[p]];
            swap(l[l[p]], r[l[p]]);
        }
    }
    if (r[p])
    {
        key[r[p]] += lab[p];
        lab[r[p]] += lab[p];
        sum[r[p]] += s[r[p]] * lab[p];
        if (flag[p])
        {
            flag[r[p]] = !flag[r[p]];
            swap(l[r[p]], r[r[p]]);
        }
    }
    lab[p] = 0;
    flag[p] = false;
}
inline void Up(int p)
{
    s[p] = s[l[p]] + s[r[p]] + 1;
    sum[p] = sum[l[p]] + sum[r[p]] + key[p];
}
inline void L(int p)
{
    int t = f[p];
    if (r[t] = l[p])
        f[l[p]] = t;
    if (f[p] = f[t])
        t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
    f[t] = p;
    l[p] = t;
}
inline void R(int p)
{
    int t = f[p];
    if (l[t] = r[p])
        f[r[p]] = t;
    if (f[p] = f[t])
        t == l[f[t]] ? l[f[t]] = p : r[f[t]] = p;
    f[t] = p;
    r[p] = t;
}
void Splay(int p, int T)
{
    for (int q, t; (q = f[p]) != T; )
        if (f[q] == T)
        {
            p == l[q] ? R(p) : L(p);
            Up(q), Up(p);
        }
        else
        {
            t = f[q];
            if (p == l[q])
                q == l[t] ? (R(q), R(p)) : (R(p), L(p));
            else
```

```
                q == r[t] ? (L(q), L(p)) : (L(p), R(p));
            Up(t), Up(q), Up(p);
        }
    if (!T)
        root = p;
}
int Select(int x)
{
    int p = root, t = s[l[root]];
    Down(p);
    while (x != t + 1)
    {
        if (x < t + 1)
            t -= s[r[p = l[p]]] + 1;
        else
            t += s[l[p = r[p]]] + 1;
        Down(p);
    }
    return p;
}
void Insert(int x, int y)
{
    int p = Select(x + 1);
    Splay(p, 0);
    Down(p);
    for (p = r[p]; l[p]; p = l[p])
        Down(p);
    Down(p);
    l[p] = ++pos;
    f[pos] = p;
    sum[pos] = key[pos] = y;
    Splay(pos, 0);
}
void Delete(int x)
{
    int p = Select(x + 1);
    Splay(p, 0);
    Down(p);
    for (p = l[p]; r[p]; p = r[p])
        Down(p);
    Down(p);
    f[r[root]] = p;
    r[p] = r[root];
    f[l[root]] = 0;
    Splay(p, 0);
}
void Add(int x, int y, int z)
{
    Splay(Select(x), 0);
    Splay(Select(y + 2), root);
    key[l[r[root]]] += z;
    lab[l[r[root]]] += z;
    sum[l[r[root]]] += s[l[r[root]]] * z;
    Up(r[root]), Up(root);
}
void Reverse(int x, int y)
{
    Splay(Select(x), 0);
    Splay(Select(y + 2), root);
    flag[l[r[root]]] = !flag[l[r[root]]];
    swap(l[l[r[root]]], r[l[r[root]]]);
    Up(r[root]), Up(root);
}
int Sum(int x, int y)
{
    Splay(Select(x), 0);
    Splay(Select(y + 2), root);
    return sum[l[r[root]]];
}
int main()
{
    root = 1;
    pos = 2;
    r[1] = s[1] = 2;
    f[2] = s[2] = 1;
    return 0;
}
```

## 4.8 节点大小平衡树

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100001;
int root, pos, l[N], r[N], f[N],  s[N], num[N], key[N];
inline void L(int p)
{
    int t = r[p];
    if (r[p] = l[t])
        f[l[t]] = p;
    l[t] = p;
    if (f[t] = f[p])
        p == l[f[p]] ? l[f[p]] = t : r[f[p]] = t;
    f[p] = t;
    s[t] = s[p];
    s[p] = s[l[p]] + s[r[p]] + num[p];
    if (p == root)
        root = t;
}
inline void R(int p)
{
    int t = l[p];
    if (l[p] = r[t])
        f[r[t]] = p;
    r[t] = p;
    if (f[t] = f[p])
        p == l[f[p]] ? l[f[p]] = t : r[f[p]] = t;
    f[p] = t;
    s[t] = s[p];
    s[p] = s[l[p]] + s[r[p]] + num[p];
    if (p == root)
        root = t;
}
void Fix(int p, bool flag)
{
    if (flag)
        if (s[l[r[p]]] > s[l[p]])
            R(r[p]), L(p);
        else
            if (s[r[r[p]]] > s[l[p]])
                L(p);
            else
                return;
    else
        if (s[r[l[p]]] > s[r[p]])
            L(l[p]), R(p);
        else
            if (s[l[l[p]]] > s[r[p]])
                R(p);
            else
                return;
    Fix(l[p], 0);
    Fix(r[p], 1);
    Fix(p, 0);
    Fix(p, 1);
}
void Insert(int p, int q, int x)
{
    if (!p)
    {
        p = ++pos;
        if (q)
            x < key[q] ? l[q] = p : r[q] = p;
        else
            root = p;
        key[p] = x;
        f[p] = q;
        s[p] = num[p] = 1;
    }
    else
    {
        s[p]++;
        if (x == key[p])
            num[p]++;
        else
        {
            Insert(x < key[p] ? l[p] : r[p], p, x);
            Fix(p, x > key[p]);
        }
    }
}
void Delete(int x)
{
    int p, q, t;
    for (p = root; key[p] != x; p = x < key[p] ? l[p] : r[p])
        s[p]--;
    s[p]--;
    if (!(--num[p]))
        if (!l[p] || ! r[p])
        {
            if (p == root)
                root = l[p] + r[p];
            else
                p == l[f[p]] ? l[f[p]] = l[p] + r[p] : r[f[p]] = l[p] + r[p];
            f[l[p] + r[p]] = f[p];
        }
        else
        {
            for (q = l[p]; r[q]; q = r[q]);
            for (t = l[p]; r[t]; t = r[t])
                s[t] -= num[q];
            q == l[f[q]] ? l[f[q]] = l[q] + r[q] : r[f[q]] = l[q] + r[q];
            f[l[q] + r[q]] = f[q];
            key[p] = key[q];
            num[p] = num[q];
        }
}
int Rank(int x)
{
    int p = root, t = s[l[root]];
    while (key[p] != x)
        if (x < key[p])
        {
            p = l[p];
            t -= s[r[p]] + num[p];
        }
        else
        {
            t += num[p];
            p = r[p];
            t += s[l[p]];
        }
    return t + 1;
}
int Select(int x)
{
    int p = root, t = s[l[root]];
    while (x < t + 1 || x > t + num[p])
        if (x < t + 1)
        {
            p = l[p];
            t -= s[r[p]] + num[p];
        }
        else
        {
            t += num[p];
            p = r[p];
            t += s[l[p]];
        }
    return key[p];
}
int Pred(int x)
{
    int p = root, t;
    while (p)
        if (x > key[p])
        {
            t = p;
            p = r[p];
        }
        else
            p = l[p];
    return key[t];
}
int Succ(int x)
{
    int p = root, t;
    while (p)
        if (x < key[p])
        {
            t = p;
            p = l[p];
        }
        else
            p = r[p];
    return key[t];
}
int main()
{
    return 0;
}
```

# 5 数论

## 5.1 快速幂

```
#include <bits/stdc++.h>
using namespace std;
int a, b, ans;
int main()
{
    scanf("%d%d", &a, &b);
    ans = 1;
    while (b)
    {
        if (b & 1)
            ans = ans * a;
        a = a * a;
        b >>= 1;
    }
    printf("%d\n", ans);
    return 0;
}
```

## 5.2 Euclid

```
#include <bits/stdc++.h>
using namespace std;
int a, b;
int gcd(int a, int b)
{
    return b ? gcd(b, a % b) : a;
}
int main()
{
    scanf("%d%d", &a, &b);
    printf("%d\n", gcd(a, b));
    return 0;
}
```

## 5.3 扩展 Euclid

```
#include <bits/stdc++.h>
using namespace std;
int a, b, x, y, t;
int gcd(int a, int b, int &x, int &y)
{
    if (b)
    {
        int t, xt, yt;
        t = gcd(b, a % b, xt, yt);
        x = yt;
        y = xt - a / b * yt;
        return t;
    }
    else
    {
        x = 1;
        y = 0;
        return a;
    }
}
int main()
{
    scanf("%d%d", &a, &b);
    t = gcd(a, b, x, y);
    printf("%d␣%d␣%d\n", x, y, t);
    return 0;
}
```

## 5.4 Euler 筛

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000001;
int n, num, p[N], fai[N], miu[N];
bool flag[N];
int main()
{
    scanf("%d", &n);
    fai[1] = miu[1] = 1;
    for (int i = 2; i <= n; i++)
    {
        if (!flag[i])
        {
            p[++num] = i;
            fai[i] = i - 1;
            miu[i] = -1;
        }
        for (int j = 1; j <= num; j++)
        {
            if (i * p[j] > n)
                break;
            flag[i * p[j]] = true;
            if (i % p[j] == 0)
            {
                fai[i * p[j]] = fai[i] * p[j];
                miu[i * p[j]] = 0;
                break;
            }
            else
            {
                fai[i * p[j]] = fai[i] * (p[j] - 1);
                miu[i * p[j]] = -miu[i];
            }
        }
    }
    printf("%d\n", num);
    for (int i = 1; i < num; i++)
        printf("%d␣", p[i]);
    printf("%d\n", p[num]);
    for (int i = 1; i < n; i++)
        printf("%d␣", fai[i]);
    printf("%d\n", fai[n]);
    for (int i = 1; i < n; i++)
        printf("%d␣", miu[i]);
    printf("%d\n", miu[n]);
    return 0;
}
```

# 6 计算几何

## 6.1 线段相交

```
#include <bits/stdc++.h>
#define x first
#define y second
#define x1 first.first
#define y1 first.second
#define x2 second.first
#define y2 second.second
using namespace std;
typedef pair<double, double> Point;
typedef pair<Point, Point> Segment;
Segment a, b;
inline double Cross(Point a, Point b, Point c)
{
    return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
}
int main()
{
    scanf("%lf%lf%lf%lf%lf%lf%lf%lf", &a.x1, &a.y1, &a.x2, &a.y2, &b.x1, &b.y1, &b.x2, &b.y2);
    if (max(a.x1, a.x2) < min(b.x1, b.x2) || max(b.x1, b.x2) < min(a.x1, a.x2))
        puts("NO");
    else if (max(a.y1, a.y2) < min(b.y1, b.y2) || max(b.y1, b.y2) < min(a.y1, a.y2))
        puts("NO");
    else
        puts(Cross(a.x, a.y, b.x) * Cross(a.x, a.y, b.y) <= 0 &&
        Cross(b.x, b.y, a.x) * Cross(b.x, b.y, a.y) <= 0 ? "YES" : "NO");
    return 0;
}
```

## 6.2 多边形面积

```
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;
typedef pair<double, double> Point;
const int N = 1000001;
int n;
double ans;
Point p[N];
inline double Cross(Point a, Point b, Point c)
{
    return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
```

```
        scanf("%lf%lf", &p[i].x, &p[i].y);
    for (int i = 3; i <= n; i++)
        ans += Cross(p[1], p[i - 1], p[i]);
    printf("%.5f\n", ans / 2);
    return 0;
}
```

## 6.3 Graham 扫描

```
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;
typedef pair<double, double> Point;
const int N = 100001;
int n, top;
Point p[N], s[N];
inline double Sqr(double x)
{
    return x * x;
}
inline double Dist(Point a, Point b)
{
    return sqrt(Sqr(a.x - b.x) + Sqr(a.y - b.y));
}
inline double Cross(Point a, Point b, Point c)
{
    return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
}
inline bool cmp(Point a, Point b)
{
    return Cross(p[0], a, b) > 0 || Cross(p[0], a, b) == 0 && Dist(p[0], a) < Dist(p[0], b);
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%lf%lf", &p[i].x, &p[i].y);
        if (p[i].y < p[0].y || p[i].y == p[0].y && p[i].x < p[0].x)
            swap(p[0], p[i]);
    }
    sort(p + 1, p + n, cmp);
    s[top = 1] = p[0];
    for (int i = 1; i < n; i++)
    {
        for (; top > 1 && Cross(s[top - 1], s[top], p[i]) < 0; top--);
        s[++top] = p[i];
    }
    for (; top > 2 && Cross(s[top - 1], s[top], s[1]) < 0; top--);
    printf("%d\n", top);
    for (int i = 1; i <= top; i++)
        printf("%.5f␣%.5f\n", s[i].x, s[i].y);
    return 0;
}
```

## 6.4 最小圆覆盖

```
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;
typedef pair<double, double> Point;
const int N = 1000001;
int x, y, n;
double r;
Point O, p[N];
inline double Sqr(double x)
```

```
{
    return x * x;
}
inline double Dist(Point a, Point b)
{
    return sqrt(Sqr(a.x - b.x) + Sqr(a.y - b.y));
}
inline Point Calc(Point a, Point b, Point c)
{
    if (fabs((b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y)) < 1e-5)
        if (Dist(a, c) > Dist(b, c))
            return {(a.x + c.x) / 2, (a.y + c.y) / 2};
        else
            return {(b.x + c.x) / 2, (b.y + c.y) / 2};
    double k1, k2, b1, b2;
    k1 = (a.x - c.x) / (c.y - a.y);
    b1 = (a.y + c.y) / 2 - k1 * (a.x + c.x) / 2;
    k2 = (b.x - c.x) / (c.y - b.y);
    b2 = (b.y + c.y) / 2 - k2 * (b.x + c.x) / 2;
    return {(b2 - b1) / (k1 - k2), (k1 * b2 - k2 * b1) / (k1 - k2)};
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
        scanf("%lf%lf", &p[i].x, &p[i].y);
    random_shuffle(p + 1, p + n + 1);
    O = p[1];
    r = 0;
    for (int i = 2; i <= n; i++)
        if (Dist(O, p[i]) > r)
        {
            O = p[i];
            r = 0;
            for (int j = 1; j < i; j++)
                if (Dist(O, p[j]) > r)
                {
                    O = {(p[i].x + p[j].x) / 2, (p[i].y + p[j].y) / 2};
                    r = Dist(O, p[j]);
                    for (int k = 1; k < j; k++)
                        if (Dist(O, p[k]) > r)
                        {
                            O = Calc(p[i], p[j], p[k]);
                            r = Dist(O, p[k]);
                        }
                }
        }
    printf("%.5f␣%.5f\n%.5f\n", O.x, O.y, r);
    return 0;
}
```