1. Luồng trên đồ thị biểu diễn bằng ma trận kề

```
#include <bits/stdc++.h>
#define maxn 201
#define INF 2000000000

using namespace std;

int n, c[maxn][maxn], f[maxn][maxn], s, t;
int FlowVal;

int cl[maxn], q[maxn], Prev[maxn];
int FindPath() {
    int L=1, R=0;
    for(int i=1;i<=n;++i) cl[i]=0;
    q[++R]=s, cl[s]=1, Prev[s]=0;
    while (L<=R) {
        int u=q[L++];
        for(int v=1;v<=n;++v) if (cl[v]==0 && f[u][v]<c[u][v]) {
            cl[v]=1; Prev[v]=u;
            q[++R]=v;
            if(v==t) return 1;
        }
        for(int v=1;v<=n;++v) if(cl[v]==0 && f[v][u]>0) {
            cl[v]=1; Prev[v]=-u;
            q[++R]=v;
        }
    }
    return 0;
}

void  IncFlow() {
    int delta=INF;
    int v=t;
    while (Prev[v]!=0) {
        int u=Prev[v];
        if (u>0) delta=min(delta,c[u][v]-f[u][v]);
        else {
            u=-u;
            delta=min(delta,f[v][u]);
        }
        v=u;
    }

    FlowVal += delta;
    v=t;
    while (Prev[v]!=0) {
        int u=Prev[v];
        if (u>0) f[u][v] += delta; else {
            u=-u;
            f[v][u] -=delta;
        }
        v=u;
    }
```

```
}

int main() {
    freopen("inp.txt","r",stdin);
    freopen("out.txt","w",stdout);
    int m;
    scanf("%d %d %d %d", &n, &m, &s, &t);
    for(int i=1;i<=n;++i) for(int j=1;j<=n;++j) c[i][j]=0;
    for(int i=1;i<=m;++i) {
        int u, v, w;
        scanf("%d %d %d", &u, &v, &w);
        c[u][v] += w;
        c[v][u] += w;   // Do thi vo huong
    }
    FlowVal=0;
    for(int i=1;i<=n;++i) for(int j=1;j<=n;++j) f[i][j]=0;
    while (FindPath()) IncFlow();
    printf("%d",FlowVal);
}
```

2. Luồng trên đồ thị có hướng (biểu diễn bằng danh sách cạnh và mảng vector)

```
#include <bits/stdc++.h>
#define maxn 5001
#define maxm 10001
#define tr(i,c) for(typeof((c).begin()) i=(c).begin();i!=(c).end();i++)
#define INF 1000000

using namespace std;
struct edge{
    int x, y;
    int c, f;
};

int n, m;
edge e[2*maxm];
vector<int> g[maxn];
long long FlowVal;

deque<int> q;
int cl[maxn], prev[maxn];
void InitFlow() {
    for(int i=1;i<=2*m;i++) e[i].f=0;
    FlowVal=0;
}

bool FindPath() {
    q.clear();
    memset(cl,0,sizeof(cl));
    cl[n]=1; prev[n]=0;
    q.push_back(n);
    while (!q.empty()) {
        int u=q.front(); q.pop_front();
        tr(i,g[u]) {
            int id=*i;
            int v=e[id].y;
            if (cl[v]==0 && e[id].f<e[id].c) {
                cl[v]=1; prev[v]=id;
```

```
                q.push_back(v);
                if (v==1) return true;
            }
        }
    }
    return false;
}

void IncFlow() {
    int delta=INF;
    int v=1;
    while (prev[v]!=0) {
        int id=prev[v];
        delta=min(delta,e[id].c-e[id].f);
        v=e[id].x;
    }
    v=1;
    while (prev[v]!=0) {
        int id=prev[v];
        e[id].f+=delta;
        e[2*m+1-id].f-=delta;
        v=e[id].x;
    }
    FlowVal+=delta;
}

int main() {
    #ifndef ONLINE_JUDGE
    freopen("inp.txt","r",stdin);
    freopen("out.txt","w",stdout);
    #endif // ONLINE_JUDGE
    scanf("%d",&n);
    int u, v, w, id=0;
    while (scanf("%d%d%d",&u,&v,&w)==3) {
        ++id;
        e[id].x=u, e[id].y=v, e[id].c=w;
    }
    m=id;
    for(int i=1;i<=m;i++)
    e[2*m+1-i].x=e[i].y, e[2*m+1-i].y=e[i].x, e[2*m+1-i].c=0;
    for(int i=1;i<=2*m;i++) g[e[i].x].push_back(i);
    InitFlow();
    while (FindPath()) IncFlow();
    printf("%I64d",FlowVal);
}
```

3. Luồng cực đại trên đồ thị vô hướng biểu diễn bằng danh sách cạnh và mảng vector

```
#include <bits/stdc++.h>
#define maxn 100001
#define maxm 200001
#define INF 2000000000
#define tr(i,c) for(__typeof((c).begin()) i=(c).begin();i!=(c).end();++i)

using namespace std;
struct edge{int x,y,      // Hai đỉnh
                c,        // Độ thông qua
                fxy,      // Luồng từ x đến y
                fyx;      // Luồng từ y đến x
```

```
};

int n, m, s, t;
edge e[maxm];
vector<int> g[maxn];
int FlowVal;

int cl[maxn], q[maxn], Prev[maxn];
int FindPath() {
    for(int i=1;i<=n;++i) cl[i]=0;
    int L=1, R=0;
    q[++R]=s; cl[s]=1; Prev[s]=0;
    while (L<=R) {
        int u=q[L++];
        tr(i,g[u]) {
            int id=*i;
            if (id>0) {// fxy - cung xuoi, fyx - cung nguoc
                int v=e[id].y;
                if (cl[v]==0 && e[id].fxy<e[id].c) {
                    cl[v]=1; Prev[v]=id;
                    q[++R]=v;

                    if (v==t) return 1;
                } else if (cl[v]==0 && e[id].fyx>0) {
                    cl[v]=1; Prev[v]=-id;
                    q[++R]=v;
                }
            } else { // fyx - cung xuoi, fxy - cung nguoc
                int v=e[-id].x;
                if (cl[v]==0 && e[-id].fyx<e[-id].c) {
                    cl[v]=1; Prev[v]=-id;
                    q[++R]=v;
                    if (v==t) return 1;
                } else if (cl[v]==0 && e[-id].fxy>0) {
                    cl[v]=1; Prev[v]=id;
                    q[++R]=v;
                }
            }
        }
    }
    return 0;
}

void IncFlow() {
    // Tim luong tang luong delta
    int delta=INF;
    int v=t;
    while (Prev[v]!=0) {
        int id=Prev[v];
        if (e[abs(id)].y==v) {// v la dinh y
            if (id>0) delta=min(delta,e[id].c-e[id].fxy);
            else delta=min(delta,e[-id].fyx);
            v=e[abs(id)].x;
        } else {// v la dinh x
```

```
            if (id>0) delta=min(delta,e[id].c-e[id].fyx);
            else delta=min(delta,e[-id].fxy);
            v=e[abs(id)].y;
        }
    }
    // Tang luong
    FlowVal += delta;
    v=t;
    while (Prev[v]!=0) {
        int id=Prev[v];
        if (e[abs(id)].y==v) {
            if (id>0) e[id].fxy += delta;
            else e[-id].fyx -= delta;
            v=e[abs(id)].x;
        } else {
            if (id>0) e[id].fyx += delta;
            else e[-id].fxy -= delta;
            v=e[abs(id)].y;
        }
    }
}

int main() {
    freopen("inp.txt","r",stdin);
    freopen("out.txt","w",stdout);
    scanf("%d %d %d %d", &n, &m, &s, &t);
    for(int i=1;i<=m;++i) {
        int u, v, w;
        scanf("%d %d %d", &u, &v, &w);
        e[i].x=u, e[i].y=v, e[i].c=w;
        e[i].fxy=e[i].fyx=0;
        g[u].push_back(i);
        g[v].push_back(-i);
    }
    FlowVal=0;
    while (FindPath()) {
        IncFlow();
    }
    printf("%d",FlowVal);
}
```

4. Bài toán cặp ghép cực đại

Bài toán: Cho đồ thị hai phía G(X, Y) trong đó các cạnh được nối từ tập X sang tập Y. Hãy tìm một bộ cặp ghép cực đại.

Có thể nói đây là dạng bài tập phổ biến và hay sử dụng nhất của việc áp dụng luồng cực đại. Giả sử $X = (x_1, x_2, \dots, x_m)$, $Y = (y_1, y_2, \dots, y_n)$. Xây dựng một mạng với đỉnh phát s và đỉnh thu t với các cung:

- Cùng từ s nối đến các đỉnh thuộc tập X với độ thông qua bằng 1
- Cung nối từ tập X đến tập Y (là cạnh của đồ thị hai phía) với độ thông qua bằng $+\infty$
- Cung nối từ các đỉnh của tập Y đến t có độ thông qua bằng 1

Có thể thấy một bộ ghép cực đại trên đồ thị hai phía tương ứng với một luồng cực đại trên mạng này và bài toán tìm cặp ghép có thể được giải quyết bằng luồng cực đại.
Tuy nhiên do đặc điểm của mạng ta có thể cải tiến để giảm độ phức tạp của bài toán luồng cực đại trong trường hợp này. Cụ thể như sau:
Việc mô tả luồng có thể được thay thế bằng hai mảng
int Fx[maxm];// Fx[i]=k>0 có nghĩa là đỉnh x_i được ghép với đỉnh y_k
int Fy[maxn];// Fy[k]=i>0 có nghĩa là đỉnh y_k được ghép với x_i
Với mô tả luồng như trên thì một đường tăng luồng có thể được mô tả như là một dãy:

$$x_{i_1}, y_{i_1}, x_{i_2} \equiv Fy[y_{i_1}], y_{i_2}, x_{i_3} \equiv Fy[y_{i_2}], y_{i_3}, x_{i_4} \equiv F[y_{i_3}], \dots, x_{i_k} \equiv F[y_{i_{k-1}}], y_{i_k}, t$$

Trong đó $Fx[x_{i_1}] \neq y_{i_1}, Fx[x_{i_2}] \neq y_{i_2}, \dots, Fx[x_{i_k}] \neq y_{i_k}, Fy[y_k] = 0$
Bởi vì "cung xuôi" ứng với cạnh không nằm trong cặp ghép và cung ngược ứng với cạnh đã ghép rồi.
Do vậy một đường tăng luồng có thể mô tả bằng một số "nhịp":

$$(x_{i_1}, y_{i_1}, x_{i_2}), (x_{i_2}, y_{i_2}, x_{i_3}), \dots, (x_{i_k}, y_{i_k}, t)$$

Dễ thấy không cần nhớ $y_{i_1}(= Fx[x_{i_2}]), y_{i_2}(= Fx(x_{i_3}), \dots, y_{i_k}(\text{Qui ước} = Fx[0])$
Do đó "độ dài" của đường tăng luồng có thể được đo bằng số lượng "nhịp":

$$d[x_{i_1}] = 0, d[x_{i_2}] = 1, \dots, d[0] = d[x_{i_k}] + 1 = k + 1$$

(Chú ý ta coi đỉnh thu t là đỉnh số hiệu 0 ở bên trái).

Thuật toán tìm đường tăng luồng có thể viết:

```
bool FindPath() {
    int L=1, R=0;                          // Khởi động hàng đợi
    for(int u=1;u<=n;u++)
    if (Fx[u]==0) {                        // Fx[u]=0 có nghĩa u chưa ghép
        d[u]=0;                            // và được dùng làm xuất phát của
        q[++R]=u;                          // nhịp đầu tiên
    } else d[u]=INF;                       // d[u]=+∞ là đỉnh u chưa thăm
    d[0]=INF;                              // Đỉnh 0 chưa thăm
    while (L<=R) {
        int u=q[L++];
        FOR v ∈ Ke(u) {                    // (u,v) là cạnh của đồ thị
            if (d[Fy[v]]==INF) {
                d[Fy[v]]=d[u]+1;
                if (Fy[v]) q[++R]=y[v];
            }
        }
    }
    return (d[0]!=INF);
}
```

Một cải tiến quan trọng là trong một lần duyệt đồ thị theo chiều rộng (BFS) để tìm đường tăng luồng ta có thể tìm được nhiều đường tăng luồng khác nhau. Chính điều này cho ta một cải tiến quan trọng là bằng việc sử dụng một lần BFS ta có thể tìm được nhiều đường tăng luồng khác nhau (tương đương với việc ta có thể tăng cặp ghép thêm không chỉ một mà nhiều cạnh). Có thể thực hiện điều này bằng một hàm đệ qui từ một đỉnh u thuộc bên trái (X):

```
bool Ghep(int u) {
    if (u==0) return true;          // Do qui ước t=0
    FOR v∈Ke(u) {                   // (u,v) là cạnh của đồ thị hai phía
        if (d[Fy[v]]==d[u]+1)       // (u,v,Fy[v]) là một "nhịp"
        if (Ghep(Fy[v])) {          // Gọi đệ qui ghép Fy[v]∈X
```

```
            Fx[u]=v; Fy[v]=u;       // Nếu ghép được đổi lại Fx[u]=v, Fy[v]=u
            return true;            // Trả về true vì đã ghép được Fy[v]
        }
    }
    d[u]=INF;                       // d[u]=+∞ để lần sau không sử dụng u nữa
    return false;
}
```

Như vậy thuật toán luông cực đại sửa thành:

```
int MaxFlow() {
    int mf=0;                               // Khởi đầu luồng 0
    memset(Fx,0,sizeof(Fx));
    memset(Fy,0,sizeof(Fy));
    while (FindPath()) {                    // Nếu BFS đi đến được đỉnh 0
        for(int u=1;u<=n;u++)               // Thử qua các đỉnh Fx[u]=0 để tìm
            if (Fx[u]==0)                   // một đường tăng luồng từ u
                if (Ghep(u)) mf++;
    }
    return mf;
}
```

Dưới đây là toàn văn chương trình code bằng C++ 4.9.2 giải bài http://vn.spoj.com/problems/FMATCH/:

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

#define MAXN 50005
#define INF 1000000007
#define tr(i,c) for(typeof((c).begin()) i=(c).begin();i!=(c).end();i++)

using namespace std;

int m,n,res=0;
vector<int> g[MAXN];
int x[MAXN],y[MAXN],d[MAXN], q[MAXN];

bool FindPath() {
    int L=1, R=0;
    for(int u=1;u<=n;u++)
    if (x[u]==0) {
        d[u]=0;
        q[++R]=u;
    } else d[u]=INF;
    d[0]=INF;
    while (L<=R) {
        int u=q[L++];
        tr(i,g[u]) {
            int v=*i;
            if (d[y[v]]==INF) {
                d[y[v]]=d[u]+1;
                if (y[v]) q[++R]=y[v];
            }
        }
    }
    return (d[0]!=INF);
}

bool dfs(int u) {
    if (u==0) return(true);
    tr(i,g[u]) {
        int v=*i;
        if (d[y[v]]==d[u]+1)
```

```
        if (dfs(y[v])) {
            x[u]=v; y[v]=u;
            return true;
        }
    }
    d[u]=INF;
    return false;
}

void GhepMax() {
    while (FindPath()) {
        for(int u=1;u<=n;u++)
            if (x[u]==0)
                if (dfs(u)) res++;
    }
    printf("%d",res);
}

int main() {
    int p;
    scanf("%d%d%d",&n,&m,&p);
    for(int i=1;i<=p;i++) {
        int u,v; scanf("%d%d",&u,&v);
        g[u].push_back(v);
    }
    GhepMax();
}
```

Thuật toán trên còn gọi là thuật toán **`Hopkroft Karp`** với cách đặt vấn đề có khác một chút nhưng về mặt bản chất là tương tự.

5. Luồng Mincost

Với mỗi cung của mạng, ngoài độ thông qua, còn có thêm giá. Bài toán đặt ra là tìm luồng có giá trị bằng Val và có tổng chi phí là bé nhất (chi phí được tính bằng tích của luồng với giá)

Trong trường hợp này đường tăng luồng được tìm bằng Thuật toán Floy-Bellman (tìm đường đi ngắn nhất)

```
int cl[maxn], Prev[maxn];
queue<int> q;
bool FindPath() {
    q.clear();
    for(int i=1;i<=n;++i) cl[i]=0, kc[i]=INF;
    q.push(s); cl[s]=1, Prev[s]=0;
    while (!q.empty()) {
        int u=q.top; q.pop(); cl[u]=0;
        for(int v=1;v<=n;++v) if (f[u][v]<c[u][v] && kc[v]>kc[u]+a[u][v]) {
            kc[v]=kc[u]+a[u][v]; Prev[v]=u;
            if (cl[v]==0) {cl[v]=1; q.push(v);}
        }
        for(int v=1;v<=n;++v) if(f[v][u]>0 && kc[v]>kc[u]-a[v][u]) {
            kc[v]=kc[u]-a[v][u]; Prev[v]=-u;
            if (cl[v]==0) {cl[v]=1; q.push(v);}
        }
    }
    return (kc[t]<INF);
}
```

```
void  IncFlow() {
    int delta=Val-FlowVal;
    int v=t;
    while (Prev[v]!=0) {
        int u=Prev[v];
        if (u>0) delta=min(delta,c[u][v]-f[u][v]);
        else {
            u=-u;
            delta=min(delta,f[v][u]);
        }
        v=u;
    }

    FlowVal += delta;
    v=t;
    while (Prev[v]!=0) {
        int u=Prev[v];
        if (u>0) f[u][v] += delta; else {
            u=-u;
            f[v][u] -=delta;
        }
        v=u;
    }
}
```