最大流

最大流图初始化

```
int cntn = m + n, cnte = m * n + m + n;
vector<int> to(cnte * 2 + 10), p(cnte * 2 + 10), nxt(cnte * 2 + 10, -1);
vector<int> head(cntn + 10, -1); int cnt = 0, s = cntn + 1, t = cntn + 2;
auto add = [&](int u, int v, int w) -> void {
    to[cnt] = v, p[cnt] = w, nxt[cnt] = head[u], head[u] = cnt ++;
    return ;
};
```

最大流建图数组，修改$cntn$以及$cnte$适应不同的题目要求。

EK算法 $O(nm^2)$

```
auto EK = [&]() -> int {
    vector<int> c(n + 10), pre(n + 10);
    auto bfs = [&]() -> bool {
        vector<int> vis(n + 10);
        queue<int> q;
        vis[s] = 1, c[s] = inf;
        q.push(s);
        while(!q.empty()){
            int x = q.front(); q.pop();
            for(int i = head[x];~ i;i = nxt[i]){
                int y = to[i];
                if(vis[y] == 0 && f[i] > 0){
                    vis[y] = 1;
                    c[y] = min(c[x], f[i]);
                    pre[y] = i;
                    if(y == t) return true;
                    q.push(y);
                }
            }
        }
        return false;
    };
    int ans = 0;
    while(bfs()){
        ans += c[t];
        for(int i = t;i != s;i = to[pre[i] ^ 1]){
            f[pre[i]] -= c[t], f[pre[i] ^ 1] += c[t];
        }
    }
    return ans;
};
```

前向星建图，bfs遍历，可以应对$1e3 \sim 1e4$边数量的图

$Dinic$算法 $O(n^2m)$

```
auto dinic = [&]() -> int {
    vector<int> cur(cntn + 10), d(cntn + 10);
    auto bfs = [&]() -> bool {
```

```
            queue<int> q;
            d.clear(), d.resize(cntn + 10, -1);
            d[s] = 0, cur[s] = head[s];
            q.push(s);
            while(!q.empty()){
                auto x = q.front(); q.pop();
                for(int i = head[x];~ i;i = nxt[i]){
                    int y = to[i];
                    if(d[y] == -1 && p[i]){
                        d[y] = d[x] + 1;
                        cur[y] = head[y];
                        if(y == t) return true;
                        q.push(y);
                    }
                }
            }
            return false;
        };
        auto find = [&](auto self, int x, int limit) -> int {
            if(x == t) return limit;
            int flow = 0;
            for(int i = cur[x];~ i && flow < limit;i = nxt[i]){
                cur[x] = i;
                int y = to[i];
                if(d[y] == d[x] + 1 && p[i]){
                    int f = self(self, y, min(p[i], limit - flow));
                    if(!f) d[y] = -1;
                    p[i] -= f, p[i ^ 1] += f, flow += f;
                }
            }
            return flow;
        };
        int ans = 0, flow;
        while(bfs()) while((flow = find(find, s, inf))) ans += flow;
        return ans;
    };
```

同上前向星建图，暴力优化，可以应对$1e4 \sim 1e5$边数量的图。

```
#define inf 0x3f3f3f3f3f3f3f3f
struct MaxiFlow{
    int to[CNTM << 1], p[CNTM << 1], nxt[CNTM << 1];
    int head[CNTN], cur[CNTN], d[CNTN];
    int cnt, s, t;
    int n, m;
    void ini(int S, int T, int cntn, int cntm){
        s = S, t = T;
        n = cntn, m = cntm;
        memset(head, -1, sizeof(int) * (n + 1));
        return ;
    }
    void addEdge(int u, int v, int w){
        to[cnt] = v, p[cnt] = w, nxt[cnt] = head[u], head[u] = cnt++;
        to[cnt] = u, p[cnt] = 0, nxt[cnt] = head[v], head[v] = cnt++;
        return ;
    }
    int dinic(){
        auto bfs = [&]() -> bool {
            queue<int> q;
            memset(d, -1, sizeof(int) * (n + 1));
            d[s] = 0, cur[s] = head[s];
            q.push(s);
            while(!q.empty()){
                auto x = q.front(); q.pop();
                for(int i = head[x];~ i;i = nxt[i]){
```

```
                    int y = to[i];
                    if(d[y] == -1 && p[i]){
                        d[y] = d[x] + 1;
                        cur[y] = head[y];
                        if(y == t) return true;
                        q.push(y);
                    }
                }
            }
            return false;
        };
        auto find = [&](auto self, int x, int limit) -> int {
            if(x == t) return limit;
            int flow = 0;
            for(int i = cur[x];~ i && flow < limit;i = nxt[i]){
                cur[x] = i;
                int y = to[i];
                if(d[y] == d[x] + 1 && p[i]){
                    int f = self(self, y, min(p[i], limit - flow));
                    if(!f) d[y] = -1;
                    p[i] -= f, p[i ^ 1] += f, flow += f;
                }
            }
            return flow;
        };
        int ans = 0, flow;
        while(bfs()) while((flow = find(find, s, inf))) ans += flow;
        return ans;
    }
};
```

使用数组封装，需要提前计算图中节点数量。