二分图最大匹配

霍尔定理 Hall Theorem

Theorem：二分图 $G=<V_1,V_2,E>$ 存在从 V_1 到 V_2 的完全匹配的充要条件是对于 V_1 中所有子集 A，有 $|N(A)|\geq|A|$.

Proof：见《离散数学及其应用》P287.

Inference 1：若二分图 $G=<V_1,V_2,E>$ 满足：V_1 中每个顶点至少关联 t 条边，V_2 中每个顶点最多关联 t 条边，则存在从 V_1 到 V_2 的完全匹配.

Proof：任取 V_1 中 n 个顶点组成 V_1 的子集 A，它们至少关联 nt 条边，这 nt 条边至少关联 V_2 中 n 个顶点，于是 $|N(A)|\geq|A|$，由霍尔定理，存在从 V_1 到 V_2 的完全匹配.

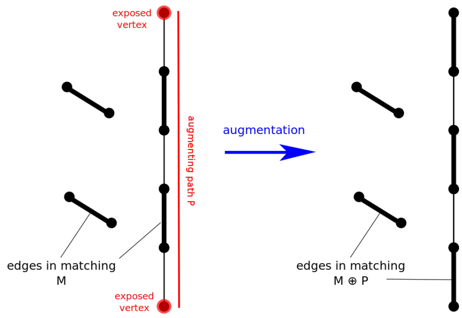
Inference 2：若二分图 $G=<V_1,V_2,E>$ 满足：V_1 中每个顶点恰关联 k 条边，V_2 中每个顶点恰关联 k 条边，则存在 k -正则完美匹配.

Proof：由 Inference 1 容易推出.

增广路定理 Berge's Lemma

Concept：

- 交错路：从非匹配点开始，匹配边和非匹配边交替向前
- 增广路：从非匹配点开始、到非匹配点结束的交错路
- 增广：



Berge's Lemma：找不到增广路时，得到最大匹配。

匈牙利算法 Hungarian Method

Idea：不断从左边的非匹配点寻找增广路，由交错路的定义知，从左到右都是非匹配边，从右到左都是匹配边，因此我们可以给二分图定向，dfs 寻找增广路。

Implementation：依次考虑左边所有点，如果从它出发可以找到一条增广路径，则顺路改变匹配。

Complexity：$O(VE)$

ATT：建图时只需要建从左往右的边，且不需要更改编号。

Code：

`match[x]` 存储右边 x 所匹配的左边的编号。

```
bool vis[N];
int match[N]; // match[x] is the one
bool dfs(int x){
    for(int i = head[x]; i; i = edge[i].nxt){
        if(vis[edge[i].to]) continue;
        vis[edge[i].to] = true;
        if(!match[edge[i].to] || dfs(match[edge[i].to])){
            match[edge[i].to] = x;
            return true;
        }
```

```
    }
    return false;
}

int Hungarian(){
    int cnt = 0;
    for(int i = 1; i <= n1; i++){
        for(int i = 1; i <= n2; i++)
            vis[i] = false;
        if(dfs(i))  cnt++;
    }
    return cnt;
}

int main(){
    scanf("%d%d%d", &n1, &n2, &m);
    for(int i = 1; i <= m; i++){
        int u, v; scanf("%d%d", &u, &v);
        addEdge(u, v);
    }
    printf("%d\n", Hungarian());
    return 0;
}
```

转换为网络最大流模型

Idea：左边所有点接源点，右边所有点接汇点，原来的边从左往右连边，所有边容量都为 1，则最大流即是最大匹配。

Complexity：使用 **Dinic** 算法求最大流，复杂度 $O(\sqrt{V}E)$

补充

- 二分图最大独立集：选最多的点，使得两两不相邻

  最大独立集大小 $=n-$ 最大匹配数

- 二分图最小点覆盖：选最少的点，使得每条边都有至少一个端点被选，其补集为独立集。

  最小点覆盖数 $=n-$ 最大独立集大小

  最小点覆盖数 $=$ 最大匹配数（**Konig** 定理）