图论算法:

(1) 最短路径:

1) 三角不等式: $\delta(u,v) \le \delta(u,x) + \delta(x,v)$

2) 松弛: relax(u, v, w(u,v))

为什么叫松弛

变为了更松的边(因为多了一个中转)

```
if (v.d > u.d + w(u,v)) then
    v.d = u.d + w(u,v)
    v.π = u
```

(2) bell man-ford

init

n-1轮 对所有边松弛

检验是否有

```
BELLMANFORD(G, W, S)
for each v ∈ G.V do:
    v.d ← +∞
s.d ← 0
for i ← 1 to |G.V|-1 do
    for each edge (u,v) ∈ G.E Do
        Relax(u, v, w(u,v))
for each edge (u,v) ∈ G.E do:
    if (v.d + u.d + w(u,v)) Then
```

[illegible] 1

```
            return false
    return true
```

(2)' spfa

```
SPFA(G, w, s):
    for each vertex v ≠ s in V(G):
        d(v) ← +∞
    d(s) ← 0
    push s into Q
    while Q is not empty do
        u ← pop Q
        for each edge (u, v) in E(G) do:
            if d(v) > d(u) + w(u, v) then
                d(v) ← d(u) + w(u, v)
                π(v) ← u
                if v is not in Q then
                    push v in Q
```

(3) dijkstra:

```
DIJKSTRA(G, w, s)
    for each vertex v in G:
        d(v) ← +∞
    d(s) ← 0
    Q ← V 所有的节点,
    init visited
    while Q is not empty do:
        u ← pop Q
        visited[u] = 1
        for each v ∈ u.nei do:
            if d(v) > d(u) + w(u,v) then
                d(v) ← d(u) + w(u,v)
                π(v) ← u
                update Q resign v
```

(4) floyd-warshell 算法:

FLOYD-WARSHELL:

- $D^0 \leftarrow W$
- $P \leftarrow 0$ path
- for $k \leftarrow 1$ to N do
  - for $j \leftarrow 1$ to N do
    - if $D^{k-1}[i,j] > D^{k-1}[i,k] + D^{k-1}[k,j]$ Then
      - $D^k[i,j] \leftarrow D^{k-1}[i,k] + D^{k-1}[k,j]$
      - $P[i,j] \leftarrow k$
    - ELSE:
      - $D^k[i,j] \leftarrow D^{k-1}[i,j]$
- return D^n, P

状态压缩(不重要)

(5) 网络流问题：

①流：

1) 流入 = 流出，非 s, t

2) $out(s) = out(t)$

3) 有限容量 $0 \le f(e) \le C_e$

②余图：

1) $C_e > f(e)$ 时，$C_f((u,v)) \leftarrow C_e - f(e)$

2) 逆向边：$C_f((v,u)) = f(e)$

③增广路径：

路径上流量已经满的（等于路径上最小容量）

④增广：

撤销部分流量。

```
class ford-fulkerson:
    def __init__(self, n, graph, s, t):
        self.s = s
        self.t = t
    def bfs(self, s, t, pre:[List]) -> visited:
```

```
    init visited
    q = Queue
    push s in q
    visited[s] = 1

    while q is not empty:
        u = pop q
        for v in neighbour(u):
            if visited[v] = 0:
                push v in q
                visited[v] = 0
                if v==t:
                    return True
    return False

def handle(self):
    new pre
    max_flow = 0
    while self.bfs(self.s, self.t, pre):
        path_flow = float("inf")
        s = self.t
```

```
        while s != self.s:
            min(path_flow,        找到瓶颈
                self.graph[pre[s]][s])
            s = pre[s]
        max_flow += path_flow
        while v != self.s:
            u = pre[v]            剩余容量
            self.graph[u][v] -= path_flow
            self.graph[v][u] += path_flow
                                  加反向边
    return max_flow
```

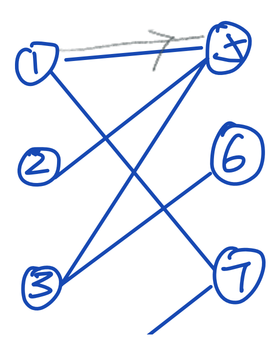
(6) 匹配问题：

① 交错路：非匹配边，匹配边……

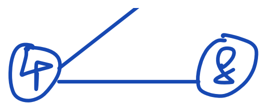
② 增广路径：

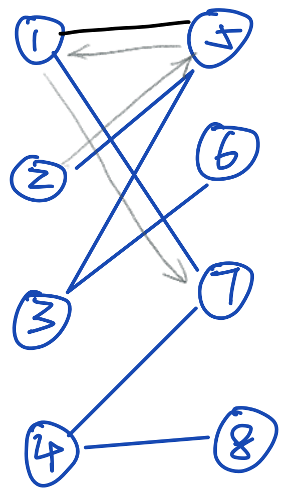
从未匹配点，走交替路，

如果终点为另一个未匹配点，称为增广路径。



1 → 5 一条匹配边



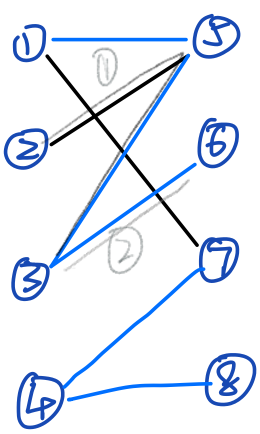


2 → 5 → 1 → 7

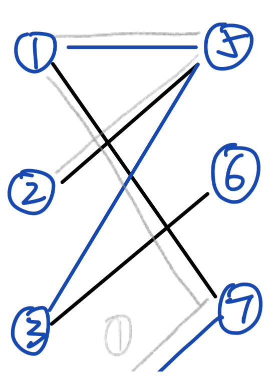
↓ 反过来

2 → 5 → 1 → 7

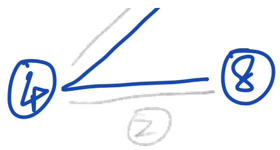
增广路取反，可以得到一个更大的匹配



3 → 6



4 → 8



M为G的最大匹配 $\Longleftrightarrow$ 不存在相对于M的增广路径

(7) 作业题：

1) 素数伴侣（一边素数，另一边非素数，相加素数）

2) 网络流问题

3) DIJKSTRA 有节点权重（节点滞留）

4) 有负权边