11.2 - The Ford Fulkerson Algorithm

最大流问题

最大流问题的定义是：给定一个流网络 G，找到一个流，使其流量值达到最大

最大流算法

我们从一个可行解开始

对所有的边 e 设置 $f(e) = 0$

- 这意味着一开始所有的边上都没有流量
- 这个初始解满足所有的约束条件（容量约束和流量守恒），因此是一个可行的流量解
- 但是，这个解显然不是最优的，因为总流量为零
- 接下来的步骤是尝试在满足所有约束条件的前提下逐步增加流量，以找到最大流量解

增加流

流量从源节点 s 开始，流向汇节点 t。所以我们需要找到一条从 s 到 t 的路径 (称为 $s-t$ 路径)，并通过这条路径传输流量

我们可以传输多少流量?

- 路径上能够传输的最大流量等于路径上所有边中最小的容量值 c_e

因此我们可以使用残差图中的路径:

- 向前推动流量 (Push Flow Forward)：在有剩余容量的边上，可以增加流量。这意味着在当前流量小于边的容量的情况下，可以将更多的流量沿着边的方向传输
- 向后推动流量 (Push Flow Backward)：在已经携带流量的边上，可以减少流量。这意味着可以通过减少边上的流量来调整网络中的流量分布，从而在其他路径上增加流量

残差图（Residual Graph）

残差图 G_f 是基于原始图 G 构造的，其节点集 V_f 与原始图 G 的节点集 V 相同

对于下面的步骤，我们可能会有疑问，流是怎么确定的
对于下面给出的图示，有三个流，我们应该如何选择初始流?

- 答案虽然是随机，但是我们选择那一条流量最大的流最好

于是在选择了一条流的情况下，我们便开始了前向边与后向边的筛选与残差图的绘制

残差图可以被如下定义:

前向边（Forward Edge）

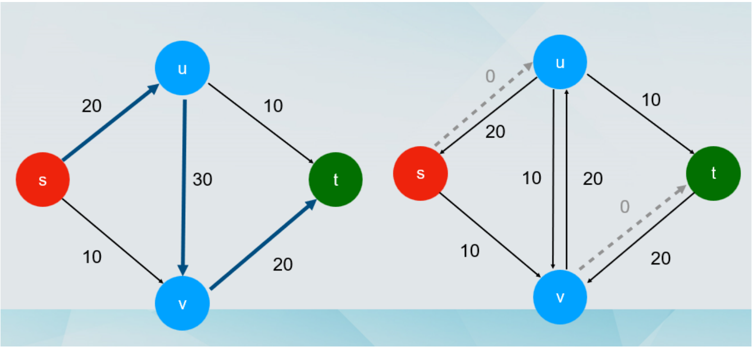
对于在 E 中的每条边 $e = (u, v)$，如果 $f(e) < c_e$，则

- 这条边有 $c_e - f(e)$ 个剩余容量单位
- 这个值称为边 e 的残差容量（Residual Capacity）
- 这样的边称为前向边（Forward Edge）

后向边（Backward Edge）

对于在 E 中的每条边 $e = (u, v)$，如果 $f(e) > 0$，则

- 在残差图 G_f 中存在一条边 $e' = (v, u)$，其容量为 $f(e)$
- 这样的边称为后向边（Backward Edge）



在残差图中增加流

- 找到残差图中的一条 $s - t$ 路径 P，我们叫这个路径为增广路径（augmenting path）
- 定义 P 的瓶颈（bottleneck）为 $\text{bottleneck}(P, f)$，这是指 P 上任意边的最小残差容量（minimum residual capacity）
- 定义流量 f 到 f' 的增广为 augment(f,P)
  - 这是通过增广路径更新流量 f 到 f' 来完成的

增广流量算法

```
augment(f, P):
    b = bottleneck(P, f)

    For each edge e = (u, v) in P:
        If e is a forward edge then
            f(e) = f(e) + b
        Else
            e' = (v, u)
            f(e') = f(e') - b
        End If
    End For

    Return f
```

可行性（容量约束）

假设 $f' = \text{augment}(f, P)$，那么 f' 是一个流吗？

考虑 P 中的一个任意边 $e = (u, v)$

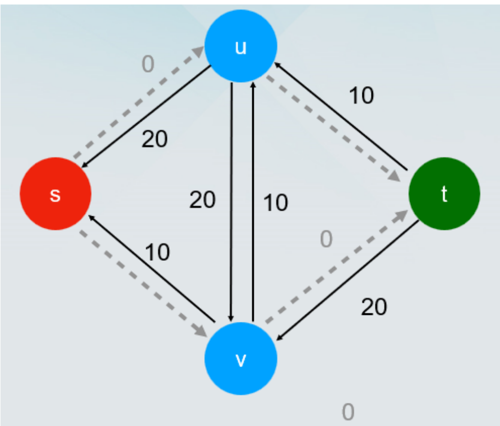
- 假设 e 是一条前向边
  - $0 \le f(e) \le f'(e) = f(e) + b \le f(e) + (c_e - f(e)) = c_e$
- 假设 e 是反向边
  - $c_e \ge f(e) \ge f'(e) - b \ge f(e) - f(e) = 0$

上述条件成立，表明容量约束条件（capacity condition）是满足的

The Ford-Fulkerson Algorithm

```
Max-Flow

        Initially set f(e) = 0 for all e in E.

        While there exists an s-t path in the residual graph G_f:
            Choose such a path P
            f' = augment(f, P)
            Update f to be f'
            Update the residual graph to be G_{f'}

        Endwhile

        Return (f)
```

1. 所有边的初始流量设置为 $f(e) = 0$
2. 在残差图 G_f 中找到一条 $(s-t)$ 路径 P，并将其作为增广路径
   1. 在路径 P 上增广流量 f，得到新的流量 f'
   2. 使用增广函数 $f' = \text{augment}(f, P)$
   3. 更新残差图 G_f 为 $G_{f'}$
   4. 对新的流量 f' 和残差图 $G_{f'}$ 重复相同的过程，直到残差图中不存在 $(s-t)$ 路径为止



算法分析

- 可行性
  - 如果算法终止，它是否能产生一个流？
- 终止性
  - 算法是否总能终止？
- 运行时间
- 正确性/最优性

- 算法是否能产生最大流?

可行性

考虑 0 flow 时，显然可行

在每次增广调用时，假设当前流量 f 是可行的，通过增广路径 P，得到新的流量 f' 也是可行的（算法已保证正确性）
而在最开始我们的 f 显然是可行的，因为我们从一个可行的流开始

因此在算法的任何一部中，流都是可行的，这说明了可行性

终止性

- 在算法中，我们只考虑了整数情况
- 在每次增广中，流量的值严格增加
  - 取增广路径 P 上的第一条边 e
  - 这条边 e 必须与源点相连，并且必须是正向边
  - 我们通过瓶颈容量增加当前流量 f
  - 新的流量 f' 比 f 增加了瓶颈容量
- 算法不会在值之间循环，因此算法最终会终止
- 但是这并不总是意味着终止，因为最大流必须是有界限的
  - 我们需要找到一个可以用作最大流的界限
  - 我们可以使用从源点出发的总容量 C 作为界限

$$\sum_{e \text{ out of } s} c_e$$

  - 因此，算法将在最多 C 步内终止
    - 由于容量是整数，在每次增广步骤中，流量至少增加 1
    - 因此对于整数情况，这个算法的终止条件成立
    - 显然对于实数情况就不成立了

运行时间

不妨假设图中每个节点都至少与一条边相连

- 这意味着 $O(m+n)=O(m)$，n 是节点的数量，m 是边的数量
- 这个假设说明 m 和 n 在同一数量级，$m+n=cn=cm$，c 是一个常数
- 因此 Ford-Fulkerson 算法的运行时间为 $O(mF)$，F 是最大流的值
  - 这显然不是一个多项式时间的算法，这是伪多项式，F 和 n 之间如果有比例关系，运行时间相当恐怖