

# Homework Review

2021.12.06

选用你认为最合理的方法证明  $f(n) \neq \Omega(n^3)$  , 其中  $f(n) = 2n^2 + 6n + 5$

方法一：正面证明

对于  $\forall c > 0, \exists N = \frac{13}{c}$ , 当  $n > N$  时, 有

$$f(n) = 2n^2 + 6n + 5 \leq 13n^2 < cn^3$$

因此也就不存在  $c_0$ , 使得满足  $\Omega(n^3)$  的定义。

方法二：反证法

假设有  $f(n) = \Omega(n^3)$ , 则根据定义有如下关系

$$\exists c_0 > 0, n_0 > 0, \forall n > n_0, s.t. f(n) = 2n^2 + 6n + 5 \geq c_0 n^3$$

而我们知道, 对于  $n > 1$ , 有  $2n^2 + 6n + 5 < 13n^2$ , 而当  $n > N = \frac{13}{c_0}$  时, 有如下推导

$$n > N = \frac{13}{c_0}$$

$$c_0 n > 13$$

$$c_0 n^3 > 13n^2$$

——by刘安珉

因此, 有  $f(n) = 2n^2 + 6n + 5 < 13n^2 < c_0 n^3$  成立。这与假设矛盾, 因此  $f(n) \neq \Omega(n^3)$

方法三：问题二的应用

非常显然, 因为  $f(n) \leq (2 + 6 + 5)n^2 = 13n^2$ , 因此, 根据定义

$$\exists c_0 = 13, n_0 = 2, \forall n > n_0, s.t. f(n) = 2n^2 + 6n + 5 \leq c_0 n^2$$

有  $f(n) = O(n^2)$ , 那么根据问题二的证明, 我们有  $f(n) \neq \Omega(n^3)$

选用你认为最合理的方法证明 $f(n) \neq \Omega(n^3)$  , 其中 $f(n) = 2n^2 + 6n + 5$

假设  $f(n) = O(n^m), f(n) = \Omega(n^{m+1})$  同时成立, 即:

$$\exists c > 0, n_0 > 0, \forall n \geq n_0, f(n) \leq c \cdot n^m$$

$$\exists c' > 0, n'_0 > 0, \forall n \geq n'_0, f(n) \geq c' \cdot n^{m+1}$$

取  $n > \max\{n_0, n'_0, \frac{c}{c'}\}$ , 则  $c \cdot n^m \geq f(n) \geq c' \cdot n^{m+1}$ , 即  $n \leq \frac{c}{c'}$ , 矛盾!

所以  $f(n) = O(n^m), f(n) = \Omega(n^{m+1})$  不能同时成立, 即:

$$f(n) = O(n^m) \implies f(n) \neq \Omega(n^{m+1})$$

- a. 使用分组策略对 GS 算法进行优化。给出你认为合理的 GS 分配优化算法描述，并分析结果的正确性（是否完美匹配？是否稳定匹配？是否会停机？）。
- b. 若  $A, B$  集合中人数不相等，同时改变完美匹配的定义为  $|M| = \min(|A|, |B|)$ ，问此时 GS 算法应该如何调整。给出调整后的算法描述，并分析其性质。

大家给出的分组策略：

- (1) 将  $A$  分为  $n$  个元素数目相等的分组, 分别标记为  $A_1, A_2, \dots, A_n$ , 同样对  $B$  也分为  $B_1, B_2, \dots, B_n$
- (2) 首先, 对分组  $A_1$  中的各元素进行原 GS 算法的匹配。直至分组  $A_1$  中各元素都在集合  $B$  中找到了匹配。
- (3) 其次, 对分组  $B_1$  中所有未被加入匹配的元素进行原 GS 算法的匹配。直至分组  $B_1$  中各元素都在集合  $A$  中找到了匹配。
- (4) 按照 (2)(3) 的次序与规则循环交替对剩余的分组分别进行原 GS 算法的匹配。
- (5) 重复 (2)(3)(4), 直至所有元素都得到配对。

- 1、可停机
- 2、可得到稳定匹配
- 3、可得到完美匹配

---

**Algorithm 3** Gale-Shapley Algorithm Modified for Unbalanced Match

---

```
1: start with  $R = \{\}$ 
2: Let  $A$  be the one in  $M$  and  $W$  with the smaller size
3: Let  $B$  be the one in  $M$  and  $W$  with the larger size
4: for each  $a$  in  $A$  who is unmatched do
5:   for each  $b$  in  $B$  with the highest rank( $a, b$ ) who was not proposed to by  $a$  do
6:     if  $b$  is unmatched then
7:        $R += (a, b)$ 
8:     else if  $b$  prefers  $a$  than her current couple  $a'$  then
9:        $R -= (a', b)$ 
10:       $R += (a, b)$ 
11:     else
12:        $b$  rejects  $a$ 
13:     end if
14:   end for
15: end for
16: return  $R$ 
```

---

可停机  
可得到稳定匹配  
可得到完美匹配

人数较少的一方发出请求即可！

给定 $n$ 个集合，每个集合中均具有 $m$ 个结点. 任何一个集合中的节点均与不在该集合中的所有节点之间存在一条带权无向边. 现要求设计算法从 $n$ 个集合中各挑出一个结点，组成 $m$ 个大小为 $n$ 的新集合，使得任何一个新集合内的节点之间的边的权值之和均超过某一给定的阈值 $K$ ，同时给出该问题的严格数学表示以及可能的实际意义.



## 1. 算法设计:

设  $n$  个集合分别为  $S_1, S_2, \dots, S_n$ , 设  $m$  个新集合分别为  $T_1, T_2, \dots, T_m$ ,  $w(u, v)$  表示两点  $u, v$  间的边权。定义若  $u, v$  同属于某个  $S_i$  则  $w(u, v) = -\infty$ 。

考虑搜索算法, 考虑当前已经组成了  $m$  个大小为  $i$  的集合, 考虑第  $i + 1$  个集合的结点的分配方案一共有  $m!$  种, 枚举这所有  $m!$  种合并方案, 记录图的状态, 然后进行下一层搜索。

在搜索过程中可以计算出当前分配方案下第  $T_j$  内的结点之间的权值  $val_j$ , 预处理出  $sum_i$  表示  $\sum_{j=1}^{i-1} \max_{u \in S_i, v \in S_j} w(u, v)$ , 如果  $val_j + \sum_{t=j+1}^n sum_t < K$ , 即后续选取了剩余集合中最大的边也不能使得  $T_j$  最终满足要求, 则说明当前分配方案不合理, 做可行性剪枝, 回退到上一层。

由于需要进行  $n - 1$  次合并, 且第  $i$  层搜索计算  $val_1, val_2, \dots, val_m$  需要花费  $O(mi)$  的时间, 即对于给定的方案, 可以在  $O(n^2m)$  的时间内验证是否满足条件, 所以总时间复杂度为  $O(n^2m(m!)^{n-1})$ 。



## 2. 数学表示:

沿用算法设计中的定义, 即求出  $m$  个集合满足

$$\min_{i=1}^m \sum_{u,v \in T_i} w(u,v) > K$$

1. 考虑一个有  $n$  个节点的图，假设每个点内有  $m$  个小点，颜色分别为  $1, 2, \dots, m$ 。设  $G$  表示该图的邻接矩阵， $u_i$  表示点  $u$  内颜色为  $i$  的点，设  $w(u_i, v_j)$  小点  $u_i, v_j$  间的边权。定义

$$w(u_i, v_j) = \begin{cases} -\infty & G_{u,v} = 1 \wedge i = j \\ 0 & \text{otherwise} \end{cases}$$

归约到本题并取  $K = 0$ ，则相当于判断一个图是不是  $m$  - 分图。即辨别一个图是否为  $m$  - 分图的问题可以归约到本题，而当  $m > 2$  时，该问题是 NPC 问题。所以当  $m > 2$  时，本题也是 NPC 问题。

2. 当  $n = 2$  时，将图中所有边权小于  $K$  的边删除，对剩下的边作二分图匹配，那么是否存在完美匹配等价于本题是否有解，复杂度与二分图匹配的复杂度相同。

高考问题：描述基于平行志愿的高考考生与高校之间匹配算法，并分析其稳定匹配的唯一性。

```
DISTRIBUTE(rank, wishes, colleges)
1   for student in rank with order
2       for wish in wishes[student] with order
3           if isNotFull(colleges[wish])
4               enroll(student, colleges[wish])
5               break for- wishes
```

稳定且唯一

算法描述

1 行表示对每个考生进行遍历，遍历顺序为高考顺位（事先已按照总成绩，单科成绩等计算排序得到）。

2 行表示对一个考生的志愿进行遍历，遍历顺序为志愿填报顺序。

3 行表示高校对考生的志愿进行检查，如果高校未录取满，则可以执行下面的语句。

4 行表示高校录取了考生。

5 行表示高校录取考生后，跳出考生志愿选择的循环，进行下一考生的录取。

该算法为简单的匹配算法，令学生人数为  $n$ ，每人志愿个数为  $m$ ，则最多匹配  $nm$  次，算法复杂

度为  $O(nm)$ 。

——by蔡冰逸

任务调度问题：优化目标为寻找一种调度策略使超时任务数最少。设计一种你认为最合适的算法，试图寻找算法的反例；若无反例，试图证明算法正确性。

要使得超时任务数最少，即使得在超时前完成的任务数量最多。考虑一种贪心算法：

1. 以当前已经完成的任务集合为  $S \leftarrow \emptyset$ ，总时间  $T \leftarrow 0$ ；
2. 对所有任务按照  $d$  升序排序；
3. 依次枚举所有任务  $t_i, d_i$ ，令  $S \leftarrow S \cup \{t_i\}$ ,  $T \leftarrow T + t_i$ ；
4. 如果  $T > d_i$ ，则将所完成的任务中耗时最长的任务  $t_{\max}$  删去，即  
 $T \leftarrow T - t_{\max}$ ,  $S \leftarrow S \setminus \{t_{\max}\}$ ；——by 何佳林
5. 重复执行 3, 4，最终所能在超时前所能完成最多的任务数量即为  $|S|$ ；
6. 用优先队列来加速找到耗时最长的任务的过程，时间复杂度  $O(n \log n)$ 。



考虑证明其正确性：

1. 证明：对于任意的一个合法的任务序列，都可以转化为一个  $d$  不降序的合法的任务序列。

考虑对于一个合法的任务序列  $\{(t_1, d_1), (t_2, d_2), \dots, (t_m, d_m)\}$ ，若存在  $i$  满足  $d_i > d_{i+1}$ ，那么有

$$\sum_{k=1}^i t_k < \sum_{k=1}^{i+1} t_k \leq d_{i+1} < d_i$$

所以交换任务  $i, i+1$  后的任务序列，依然是一个合法序列。不断重复这个操作，最终一定会得到一个  $d$  不降序的合法任务序列。

2. 证明：贪心算法求出的一定是一个合法序列。

考虑按  $d$  排序后的任务序列，假设贪心算法根据前  $i-1$  个任务求出来的序列是合法序列且总用时为  $T$ 。

1. 如果  $T + t_i \leq d_i$ ，则直接加入第  $i$  个任务得到的序列也是合法序列。
2. 如果  $T + t_i > d_i$ ，取出  $t_{\max} \geq t_i$  删去，有  $T + t_i - t_{\max} \leq T \leq d_i$ ，也是合法序列。

——by 何佳林



3. 证明：贪心算法求出的是最长的合法序列。

假设贪心的结果是  $A$ ，假设  $A$  不是最优的，由于任意一个合法序列都可以转化为  $d$  不降序的序列，那么就一定存在一个  $d$  不降序的结果  $O$ ， $O$  是和  $A$  最相近的一个最优解，即  $\exists k$ ， $O$  和  $A$  的前  $k - 1$  个选择都相同，第  $k$  个选择不同。

假设对于  $A$ ， $A$  第  $k$  个选择的是  $(t_i, d_i)$ ，而  $O$  第  $k$  个选择的是  $(t_j, d_j)$ ， $i < j$ ，于是有  $d_i \leq d_j$ 。

设  $O' = O - \{(t_j, d_j)\} + \{(t_i, d_i)\}$ ，讨论  $O'$  的合法性。

1. 如果  $t_j \geq t_i$ ，因为  $A$  的前  $k$  个选择合法，并且由于  $t_i \leq t_j$ ， $O'$  中在  $i$  之后的任务的完成时间不会变长，所以  $O'$  也是合法的最优解。
2. 如果  $t_j < t_i$ ，根据贪心的流程，由于  $i$  没有被删除，那么  $j$  也一定在  $A$  中，考虑在  $O$  中  $j$  后选的任务  $(t_m, d_m)$ ：
  1. 如果均满足  $t_m < t_i$ ，那么这些任务一定也在  $A$  中，并且  $A$  比  $O$  多一个任务  $(t_i, d_i)$ ，这与  $O$  是最优解矛盾。
  2. 如果  $O'$  中没有不合法任务，则  $O'$  也是合法的最优解；
  3. 如果  $O'$  中存在不合法任务，且存在  $t_m \geq t_i$ ，考虑一个最小的  $m$ ，令  $O'' = O - \{(t_m, d_m)\} + \{(t_i, d_i)\}$ ，由于  $m$  前的任务都在  $A$  中都合法， $m$  后的任务的任务的完成时间不会变长，所以  $O''$  也是合法的最优解。

——by 何佳林

综上所述，总能构造出一个最优解与  $A$  的前  $k$  个选择都相同，这与假设矛盾。所以贪心算法求出的结果就是最优解。

平面中第一象限内有  $n$  个矩形。要求选出尽可能多的矩形，使得选出的矩形间两两不重叠，问最多能选出多少矩形。设计算法并分析算法正确性。

大家给出并举出反例的策略：右上角顶点到原点的距离、对角线交叉点到原点的距离、X或Y坐标

本问题为NPC问题，不存在多项式复杂度的算法

## 使用交换论证方法证明Kruskal算法的正确性

设有最优解  $T : e_1, e_2, e_3, e_4 \dots e_{n-1}$  总权重为  $W$

Kruskal 算法得到解  $T_0 : a_1, a_2, a_3, a_4 \dots a_{n-1}$  总权重为  $W_0$

对  $T$  中任意一条边  $e_k \notin T_0$ , 删掉  $e_k$ , 得到两棵小树  $T_1, T_2$

从  $T_0$  中选择一条边  $e_r \notin T_0 - (T_1 + T_2)$ , 加入到  $T_1 + T_2$  中, 且可以使之连通, 则会生成一棵新树  $T_3$ .

——朱悦婷

下面证明  $T_3$  优于  $T$ , 即  $|e_r| \leq |e_k|$ :

因为加入  $e_r$  使联通, 所以  $e_r$  能构造连接  $e_k$  起点和终点的路径, 若  $|e_r| > |e_k|$ , 而按照 Kruskal 算法, 会按照从小到大的顺序选择边, 则会选择  $|e_k|$ , 那么  $e_k \in T_0$ , 矛盾.

所以,  $|e_r| \leq |e_k|$ , 将  $T$  中的边替换为  $T_0$  中的边生成的树是更优的, 所以  $T_0$  优于任何一个包含  $e_k \notin T_0$  的  $T$ , 所以是贪心最优解.

## 使用归纳法证明Dijkstra算法的正确性

定义 $S=\{1,2,\dots,x\}$ 为已求出最短路径的顶点， $dist[u]$ 为从start点开始经过S中的点到u点的最短距离， $short[u]$ 从start开始到u的全局最短路径（路径中可能有部分的点不在S中）。

可知  $short[u] \leq dist[u]$ 。

归纳基础：

$|R| = 1, R = \{s\}, dist[s] = short[s] = 0.$

归纳假设：

·令u为最后一个加入R的节点，让R'为 $R \cup \{u\}$ 。我们假设对任意的 $x \in R, dist[x]=short[x]$ 。



归纳证明：

我们只需证明在R'中,  $\text{dist}[u] = \text{short}[u]$ , 即可完成证明。

假设存在一个矛盾的例子，最短的路径从s到u是Q，那么  $l(Q) < d(u)$

Q从R'中的某个点开始，其中包含了一些点不在R'中，我们假设y是第一个不在R'中的点，x是与之相连的R'中的点。

设 $Q_x$ 是Q的s到x的子路径，那么显然  $l(Q_x) + l(xy) \leq l(Q)$

由假设我们可以知道 $d(x)$ 是从s到x最短的路径长度，那么  $d(x) \leq l(Q_x)$

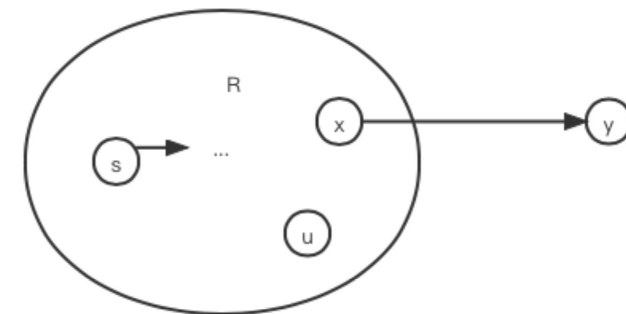
那么可以得到  $d(x) + l(xy) \leq l(Q)$  ——(1)

既然y与x相连接，那么要使y也被Dijkstra算法更新，得到  $d(y) \leq d(x) + l(xy)$  ——(2)

最后我们观察到，既然在y之前，u已经被Dijkstra算法所挑选，那么  $d(u) \leq d(y)$  ——(3)

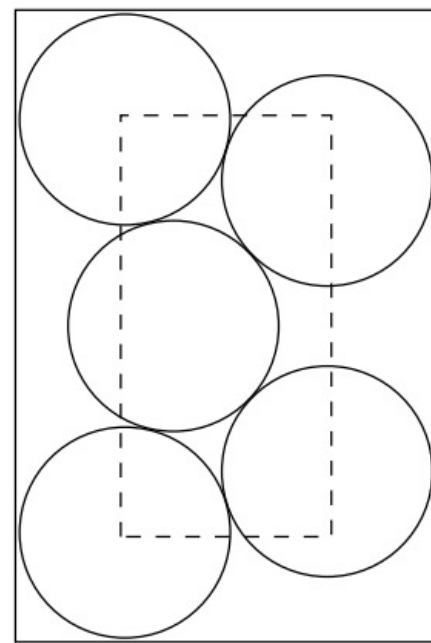
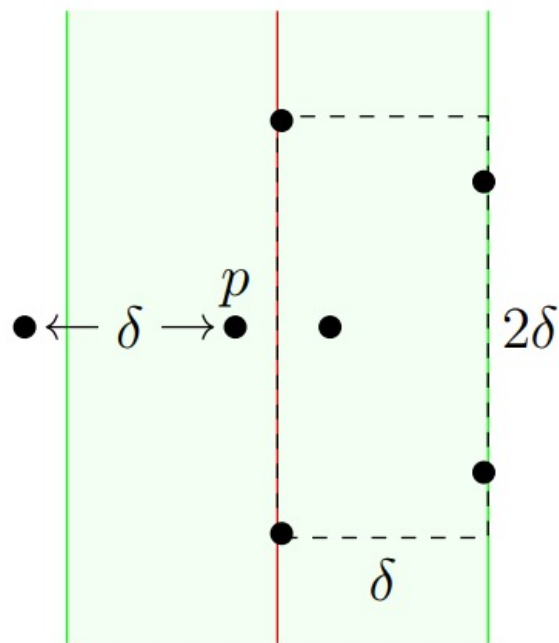
根据(1)(2)(3)我们可以得出  $d(u) \leq l(Q)$ ，这与一开始的  $l(Q) < d(u)$  矛盾。所以不存在这样的路径Q。

那么我们就证明了  $\text{dist}[u] = \text{short}[u]$ 。也即得到了Dijkstra算法的正确性。



习题2：分治算法之最近点对问题，结合示意图分别对何老师和郑欣同学描述的合并算法进行阐述，并尝试编程验证。

遍历 $L'$ 中的每个点 $p$ , 对每个 $p$ 在 $R'$ 中找到所有满足 $|y_p - y_q| < \delta$ 的点 $q$ , 并将 $\delta$ 更新为 $\min(\delta, \|p - q\|)$

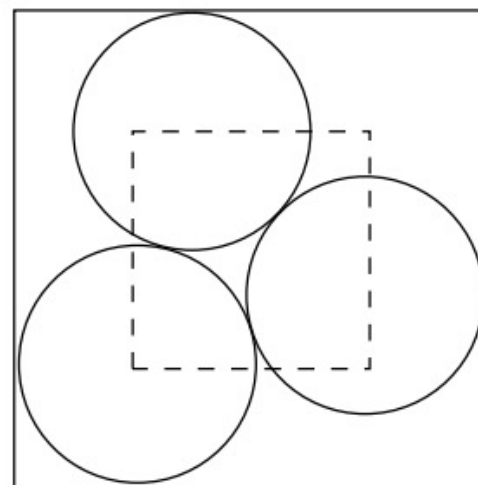
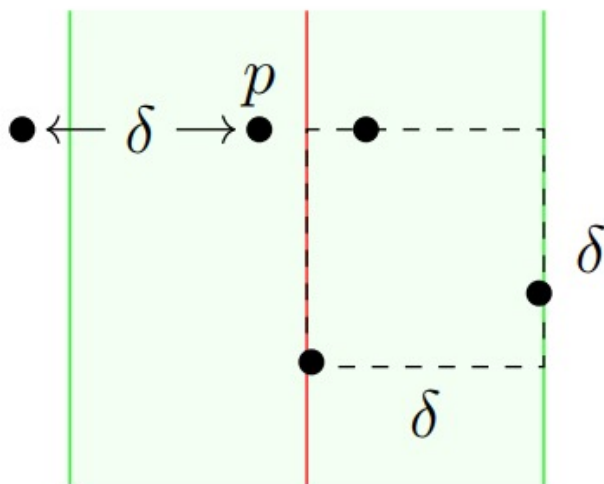


——郑欣

$p$  所匹配的区域是一个  $\delta \times 2\delta$  的矩形 (不包含边界). 由于矩形中的点两两之间距离不小于  $\delta$ , 因此矩形中至多只有 5 个点. 这等价于在  $2\delta \times 3\delta$  的矩形中放尽可能多的半径为  $\delta/2$  的圆, 使得圆互不重叠且不与矩形边界相交



分别遍历 $L'$ 和 $R'$ 中的每个点 $p$ , 对每个 $p$ 找到对面所有满足 $y_p - \delta < y_q \leq y_p$ 的点 $q$ , 并将 $\delta$ 更新为 $\min(\delta, \|p - q\|)$



——郑欣

该算法每个点至多只需要匹配 3 个对面的点

问题1：利用假设代入法计算 $T(n) = T\left(\frac{n}{7}\right) + T\left(\frac{5n}{14}\right) + cn$ 的复杂度

假设  $T(n) = O(n)$ ，那么假设有  $T(n) \leq dn$ ，则有如下推导

$$\begin{aligned} T(n) &= T\left(\frac{n}{7}\right) + T\left(\frac{5n}{14}\right) + cn \\ &\leq d\frac{n}{7} + d\frac{5n}{14} + cn \\ &= d\frac{n}{2} + cn \end{aligned}$$

——by刘安珉

那么根据假设代入法的定义，有

$$\begin{aligned} d\frac{n}{2} + cn &\leq dn \\ d &\geq 2c \end{aligned}$$

所以，是存在  $d$  满足条件的，因此  $T(n) = O(n)$ 。

问题2：直接递归计算 $T(n) = 9T\left(\frac{n}{3}\right) + cn^2$ 的复杂度

假设 $n = 3^k$ ，可得

$$\begin{aligned}
 T(n) &= 9 T\left(\frac{n}{3}\right) + cn^2 \\
 &= 9 \left( 9 T\left(\frac{n}{3^2}\right) + c\left(\frac{n}{3}\right)^2 \right) + cn^2 \\
 &= 9^2 T\left(\frac{n}{3^2}\right) + 2cn^2 \\
 &\dots \\
 &= 9^k T\left(\frac{n}{3^k}\right) + kcn^2 \\
 &= c' 9^{\log_3 n} + cn^2 \log_3 n \\
 &= (c + c')n^2 \log_3 n \\
 &= O(n^2 \log_3 n)
 \end{aligned}$$

——by肖子扬

问题3：对 $T(n) \leq T(C_1n) + T(C_2n) + \dots + T(C_kn) + cn$ ，其中 $C_1 + C_2 + \dots + C_k < 1$ ，证明 $T(n) \leq dn$ ，其中 $d \geq \frac{c}{1-(C_1+C_2+\dots+C_k)}$

假设 $T(n) \in O(n)$ 即 $T(n) \leq dn$ .

我们有 $d \geq \frac{c}{1-(C_1+C_2+\dots+C_k)}$ ，当  $n=1$  时， $T(n) \leq dn$ 成立.

假设 $n \leq \lambda$ 时， $T(n) \leq dn$ 成立.

——by李茗畦

当 $n = \lambda + 1$ 时，我们假设 $\lambda$ 是足够大的,使得 $\forall i = 1, 2 \dots k, \lambda \geq \frac{c_i}{1-c_i}$ ,

所以 $c_i(\lambda + 1) \leq \lambda, i=1, 2, 3 \dots k$ .

所以 $T(C_i n) \leq d(C_i n), i = 1, 2, 3 \dots k$ .

因为 $T(n) \leq T(C_1n) + T(C_2n) + \dots T(C_kn) + cn$ ,

所以 $T(n) \leq (C_1 + C_1 + \dots C_k)dn + cn \leq dn$ .

数据集  $D = \{(x_i, y_i)\}$  , 求  $k$  段线性回归函数  $S = \{S_1, S_2, \dots, S_k\}$  的分段点 , 使  $\sum_{i=1}^k \text{Error}(S_i, a_i, b_i) + ck$  最小。其中  $S_i$  表示第  $i$  段分段函数 , 假设惩罚系数  $c$  是已知的给定值。要求 :

1. 写出递归表达式。
2. 给出时间复杂度分析。

**递归表达式** 考虑子序列  $A_{l..r} = (X, Y)$ , 可计算线性回归的平方损失

$$L(A_{l..r}) = n \sum y_i^2 - \left( \sum y_i \right)^2 - \frac{\left( n \sum x_i y_i - \sum x_i \sum y_i \right)^2}{n \sum x_i^2 - \left( \sum x_i \right)^2}$$

用  $f(i)$  表示原序列的前  $i$  项分成若干段得到的最小损失和, 则有以下转移方程

$$f(i) = \min_{0 \leq j < i} f(j) + L(A_{j+1..i}) + c \quad (1 \leq i \leq n)$$

其中初始状态为  $f(0) = 0$ .

**复杂度分析** 计算每个  $f(i)$  都需要知道所有  $f(j)$  ( $j < i$ ), 故时间复杂度为  $\sum_i i = O(n^2)$ .



动态规划——上下文无关算法：

1. 分析例子：


文法：

$$\begin{cases} S \rightarrow AB|BC \\ A \rightarrow BA|a \\ B \rightarrow CC|b \\ C \rightarrow AB|a \end{cases}$$

派生串为aabbab,使用填表法分析派生过程。

2. 给出动态规划表达式 $opt(i, l)$ , 其中 $i$ 是目标串中字符的序号,  $l$ 是第 $i$ 个字符及其之后子串的长度。
3. 给出算法过程。

派生过程：



|             |             |             |            |            |          |
|-------------|-------------|-------------|------------|------------|----------|
| $\{S, C\}$  |             |             |            |            |          |
| $\{A\}$     | $\{B\}$     |             |            |            |          |
| $\emptyset$ | $\emptyset$ | $\{S, C\}$  |            |            |          |
| $\{B\}$     | $\emptyset$ | $\{A\}$     | $\{S, C\}$ |            |          |
| $\{B\}$     | $\{S, C\}$  | $\emptyset$ | $\{S, A\}$ | $\{S, C\}$ |          |
| $\{A, C\}$  | $\{A, C\}$  | $\{B\}$     | $\{B\}$    | $\{A, C\}$ | $\{B\}$  |
| <b>a</b>    | <b>a</b>    | <b>b</b>    | <b>b</b>   | <b>a</b>   | <b>b</b> |

给出函数 $source(s)$  ,  $s$  为一个字符或字符的集合 , 函数返回 $s$ 在单一规则下派生源头的集合。则动态规划表达式为：

$$opt(i, l) = \begin{cases} \bigcup_{k=1}^{k=l} source(opt(i, k) \times opt(i + k, l - k)) & l > 1 \\ source(a[i]) & l = 1 \end{cases}$$

Ford-Fulkerson 算法在边容量为无理数时可能不停机。以图1为例， $r = \frac{1}{2}(\sqrt{5} - 1)$ ，其他边容量无穷大，则 FF 算法可能不停机。

1. 给出算法不停机的执行过程，要求写出前 4 步并画出每一步的残差图
2. Ford-Fulkerson 算法是否存在极限不等于最大流的情况？若存在，在图 1 的基础上进行改动，并加以说明；若不存在，说明理由。

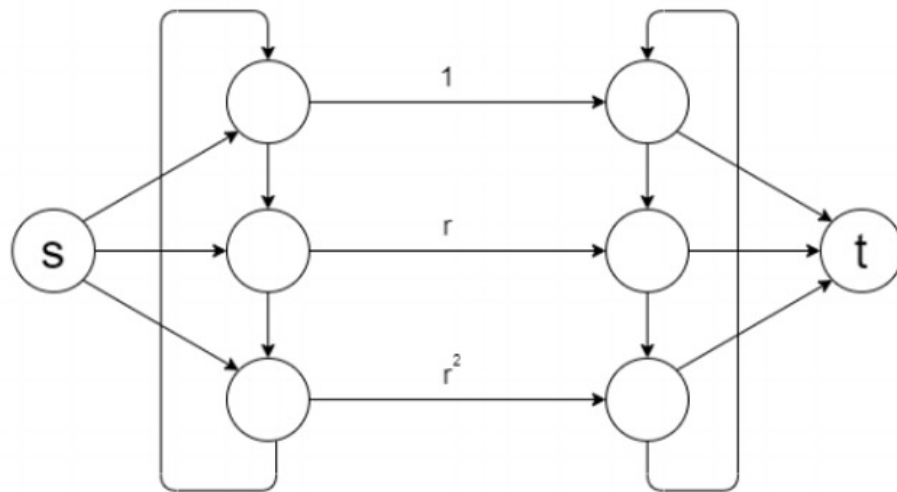


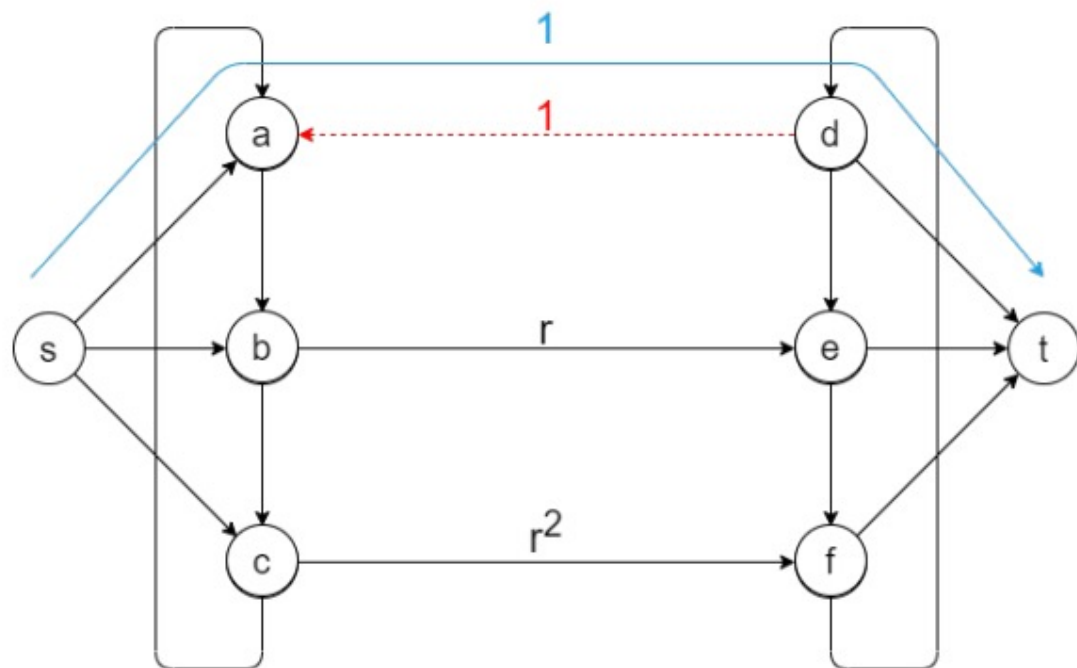
图 1: FF 算法不停机示例

## 第一步

增广路径 :  $s \rightarrow a \rightarrow d \rightarrow t$

$$f = 1$$

$$F = 1$$



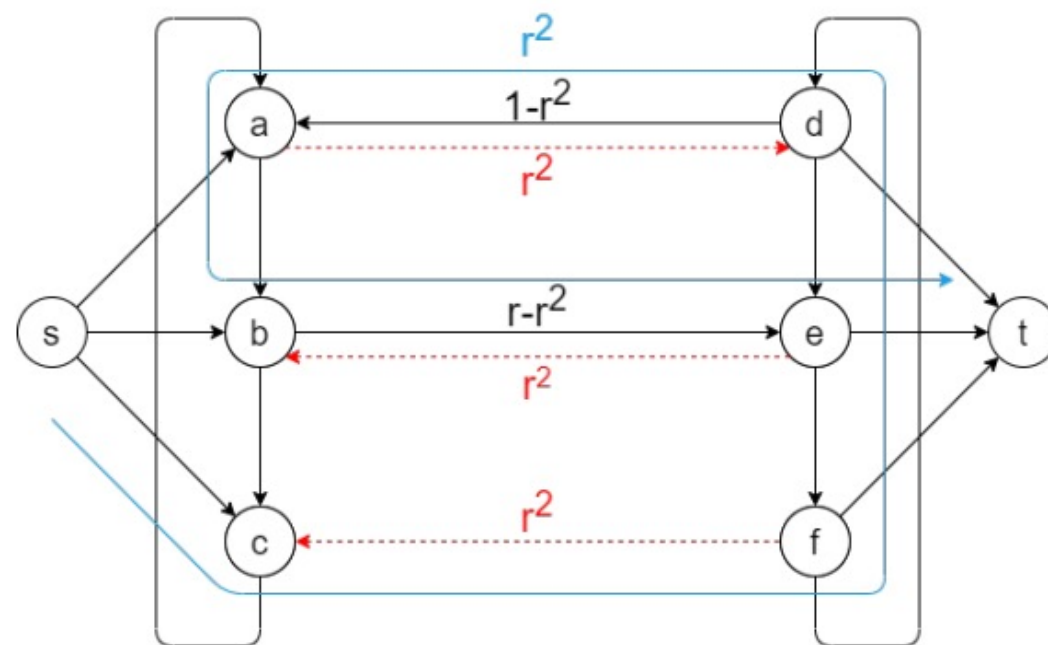
第一步残差网络

## 第二步

增广路径 :  $s \rightarrow c \rightarrow f \rightarrow d \rightarrow a \rightarrow b \rightarrow e \rightarrow t$

$$f = r^2$$

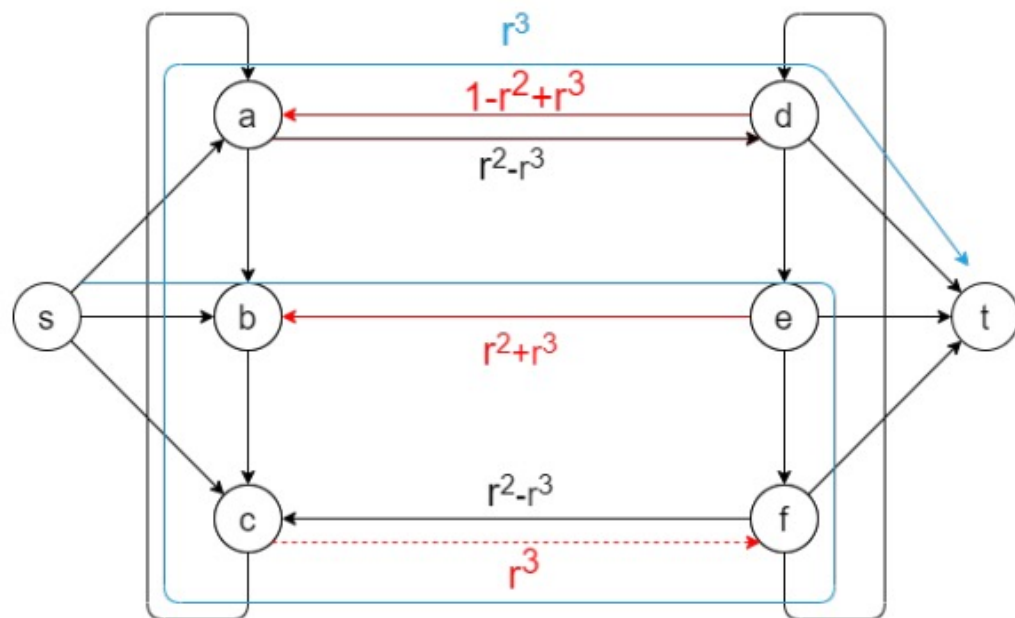
$$F = 1 + r^2$$



第二步残差网络

## 第三步

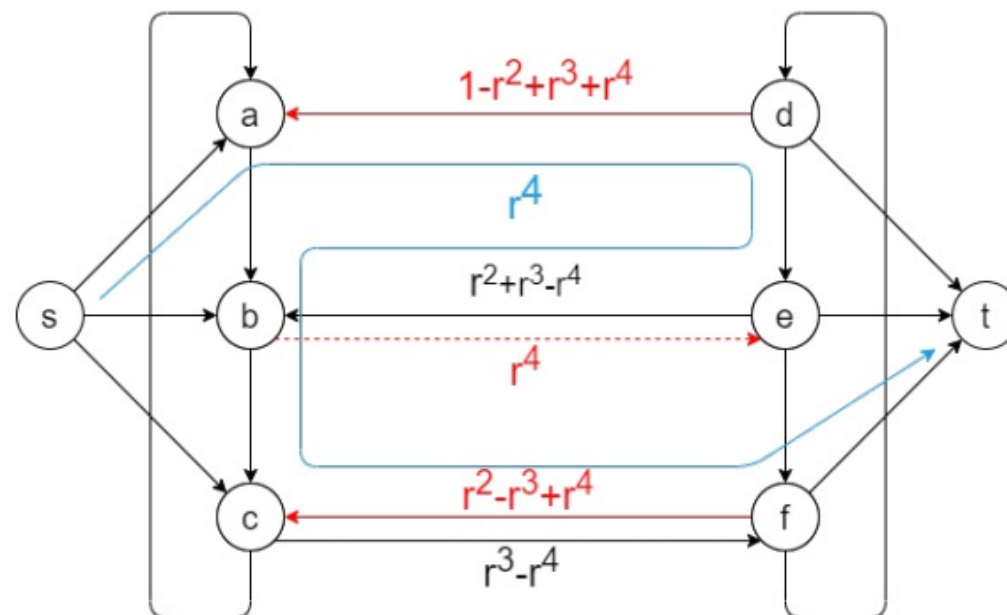
增广路径 :  $s \rightarrow b \rightarrow e \rightarrow f \rightarrow c \rightarrow a \rightarrow d \rightarrow t$   
 $f = r - r^2 = r^3$   
 $F = 1 + r^2 + r^3$



第三步残差网络

## 第四步

增广路径 :  $s \rightarrow a \rightarrow d \rightarrow e \rightarrow b \rightarrow c \rightarrow f \rightarrow t$   
 $f = r^2 - r^3 = r^4$   
 $F = 1 + r^2 + r^3 + r^4$

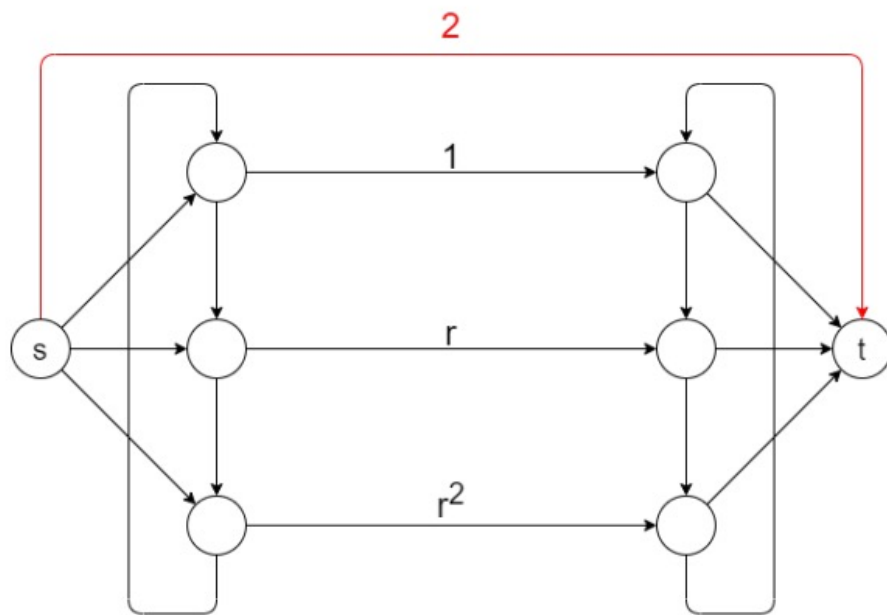


第四步残差网络



算法执行第二步之后， 每一步都可以构造出流量为 $r^i$ 的流， 其中  $i \in [2, +\infty)$

$$F = 1 + r^2 + r^3 + r^4 + \dots = 1 + r^2 \frac{1}{1 - r} = 2 \quad \longrightarrow \quad \text{最终收敛至最大流， 但不会停机}$$



FF 算法不停机且极限不为最大流的例子

THANK YOU !