湖南大学
HUNAN UNIVERSITY

第十三章
所有点对间的最短路径问题

湖南大学信息科学与工程学院

课程回顾

无权图上的最短路径算法——BFS，复杂度$O(|V|+|E|)$

无负权的图上的最短路径算法——Dijkstra，复杂度$O(|E|+|V|\log|V|)$

一般图上的最短路径求解——Bellman-Ford，复杂度$O(|V||E|)$

目录

定义

所有点对间最短路径的问题是指以图G=(V,E)为对象，求G中每个点之间的最短路径问题。

基本算法

无权图上的全点对最短路径算法——|V|次BFS，复杂度O(|V||E|)

无负权的全点对最短路径算法——|V|次Dijkstra，复杂度O(|V||E|+|V|^{2}log|V|)

一般图上的最短路径求解——|V|次Bellman-Ford，复杂度O(|V|^2|E|)

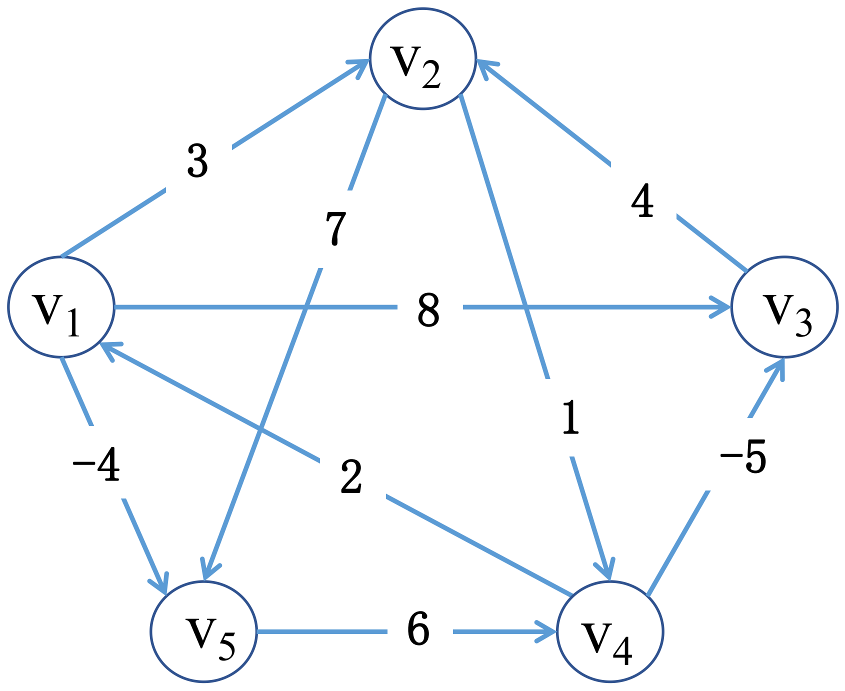
目标算法就是比上述算法复杂度要低

图的矩阵表示

给定图G = (V, E, W)，设|V| = n且V 中的点编号为$v_1, v_2, ..., v_n$，G 的邻接矩阵A = a[i][j] 是一个$n \times n$的矩阵，它满足这样的性质：

$$a[i][j] = \begin{cases} w_{ij}, & if\ (v_i, v_j) \in E \\ \infty, & if\ (v_i, v_j) \notin E \end{cases}$$

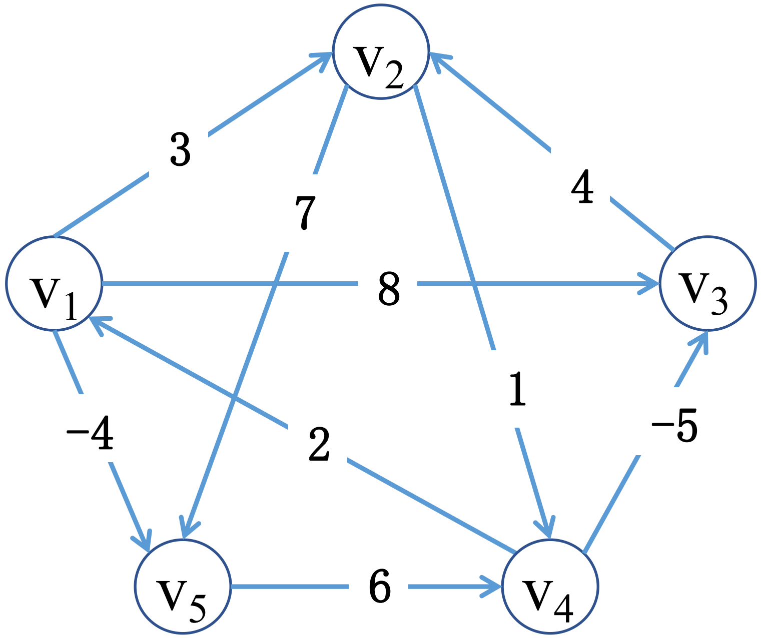
全点对最短路径问题



$$a[i][j]=\begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

全点对最短路径问题

给定一个带权图G=(V, E, W)，计算V中任意两点间距离，进而得到一个$n \times n$矩阵M，其中m[i][j] = $\delta(v_i, v_j)$



$$m[i][j] = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

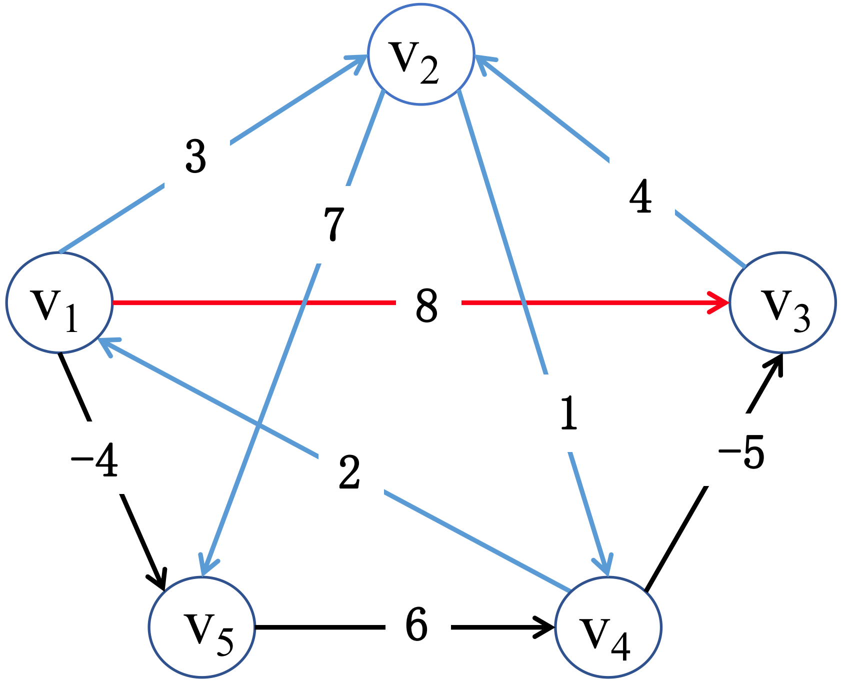
扩展定义

$l_{ij}^{(m)}$表示从v_i到v_j的所有边数小于等于m的路径中的最短路径长度

其中，当m=0时候，我们有

$$l_{ij}^{(0)} = \begin{cases} 0, & if \ \ i = j \\ \infty, & if \ \ i \neq j \end{cases}$$

扩展定义



$$l_{13}^{(1)}=8$$

$$l_{13}^{(3)}=-3$$

扩展定义

于是，我们有如下性质：

$l_{\mathrm{ij}}^{(n-1)}=\delta\left(v_i, v_j\right)$

如果$l_{\mathrm{ij}}^{(m)}=l_{\mathrm{ij}}^{(n-1)}$且图上没有负权环路，那么$l_{\mathrm{ij}}^{(m)}=l_{\mathrm{ij}}^{(\mathrm{m}+1)}=\ldots=l_{\mathrm{ij}}^{(n-1)}=\delta\left(v_i, v_j\right)$,

$$l_{\mathrm{ij}}^{(m)}=\min_k\{l_{\mathrm{ik}}^{(m-1)}+a[k][j], l_{\mathrm{ij}}^{(m-1)}\}$$

基本算法

```
Basic(A)
  for m=1 to n:
    for i=1 to n:
      for j=1 to n:
        for k=1 to n:
          if l_ik + a[k][j] ≤ l_ij
            then l_ij = l_ik + a[k][j]
```

复杂度：O(n^4)

基于矩阵乘法的扩展

上述计算类似于矩阵乘法，只是将其中的加法改成了求最小值，乘法改成了加法

于是，我们可以定义如下矩阵运算：给定矩阵A和矩阵B， $C = A \otimes B$定义如下

$$c[i][j]=\min_{k}\{a[i][k]+b[k][j]\}$$

⊗运算符的性质

交换律： $A \otimes B = B \otimes A$

结合律： $(A \otimes B) \otimes C = A \otimes (B \otimes C)$

所以， ⊗运算符对应的是一个闭合的半环

基于⊗运算符的改进

将$l_{ij}^{(m)}$都表示成矩阵$L^{(m)}$，于是$L^{(1)} = A$，$L^{(m)} = L^{(m-1)} \otimes A$

为了完全性，可以其中$L^{(0)}$定义为单位对角矩阵，对角线全是0

所以$L^{(m)}$等于A基于⊗求m+1次幂，记为$L^{(m)} = A^{(m+1)}$

基于分治算法，实现一个$O(n^3 \log n)$的算法。

扩展思考

如何利用上述算法的结果判断负权环路？

如果最终算出来的$L^{(m+1)}$的对角线上是有小于0的值，那么图上就有负权环路

目录

算法回顾

Basic(A)

for m=1 to n:

    for i=1 to n:

        for j=1 to n:

            for k=1 to n:

                if $l_{ik} + a[k][j] \le l_{ij}$

                    then $l_{ij} = l_{ik} + a[k][j]$

合并

算法回顾

1962年Robert Floyd和Stephen Warshall分别发明

相对于之前的算法，定义了更好的计算公式，减少一重循环，使得算法复杂度变成了$O(n^3)$

Floyd-Warshall算法中子问题定义

$d_{ij}^{(k)}$表示从v_i到v_j的所有仅使用集合$\{v_1, v_2, \ldots, v_k\}$中点的路径中的最短路径长度

其中，当k=0时候，我们有

$$d_{ij}^{(0)} = \begin{cases} a[i][j], & if\ (v_i, v_j)\text{存在} \\ \infty, & \text{否则} \end{cases}$$

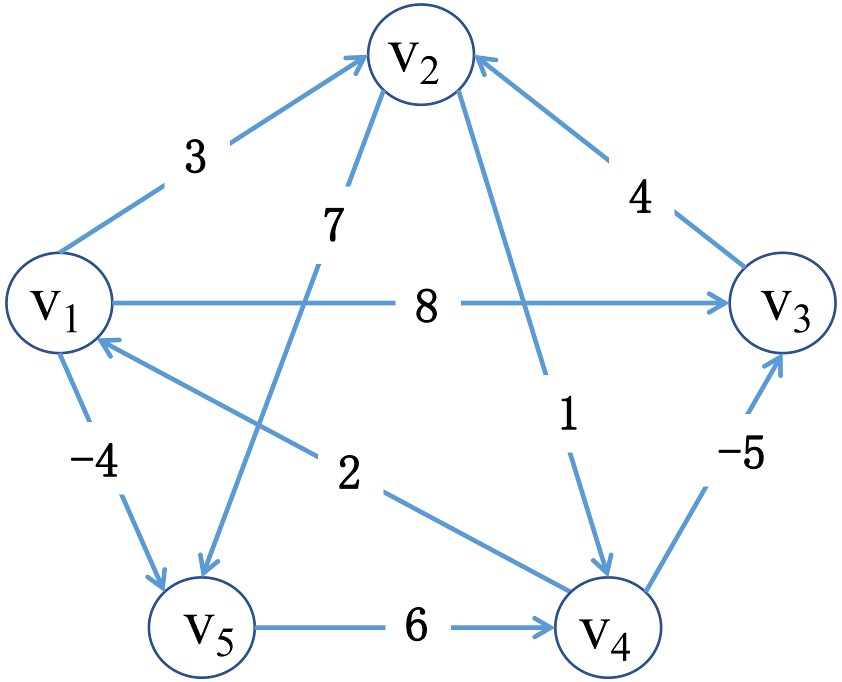
Floyd-Warshall算法中子问题定义

针对上述定义，我们有如下两个性质

如果图上没有负权环路，那么$d_{\mathrm{ij}}^{(n)}$=$\delta\left(v_i, v_j\right)$;

$$d_{\mathrm{ij}}^{(k)} = \min_k \{d_{\mathrm{i}k}^{(k-1)} + d_{kj}^{(k-1)}, d_{ij}^{(k-1)}\}$$

Floyd-Warshall算法中子问题定义



$$d_{14}^{(2)}=4$$

$$d_{14}^{(3)}=4$$

$$d_{14}^{(4)}=4$$

$$d_{14}^{(5)}=2$$

算法伪代码

Floyd-Warshall(A)

D=A;

for i=1 to n:

  for j=1 to n:

    for k=1 to n:

      if $d_{ik} + d_{kj} \leq d_{ij}$

        then $d_{ij} = d_{ik} + d_{kj}$

复杂度：O(n^3)

目录

Johnson算法

1977 年由Donald B. Johnson提出来的算法

|V|次Dijkstra的复杂度$O(|V||E|+|V|^2\log|V|)$，这个实际中的复杂度很可能低于Floyd-Warshall

但上述方法不能处理负权边，为此提出来Johnson算法

边的重赋权

Johnson算法核心在于将负权边转化成非负数

具体而言， Johnson算法找出一个函数h，对于任意边(u,v)，我们有

$$w_h(u,v) = w(u,v) + h(u) - h(v)$$

重赋权性质

性质：上述重赋权的优点在于两点u和v之间的不同路径，重赋权之前权重和重赋权之后权重的差值一样

证明： 设u和v之间一条路径为$ue_1v_1e_2 \ldots e_nv_n\, e_{n+1}v$，那么缩进法

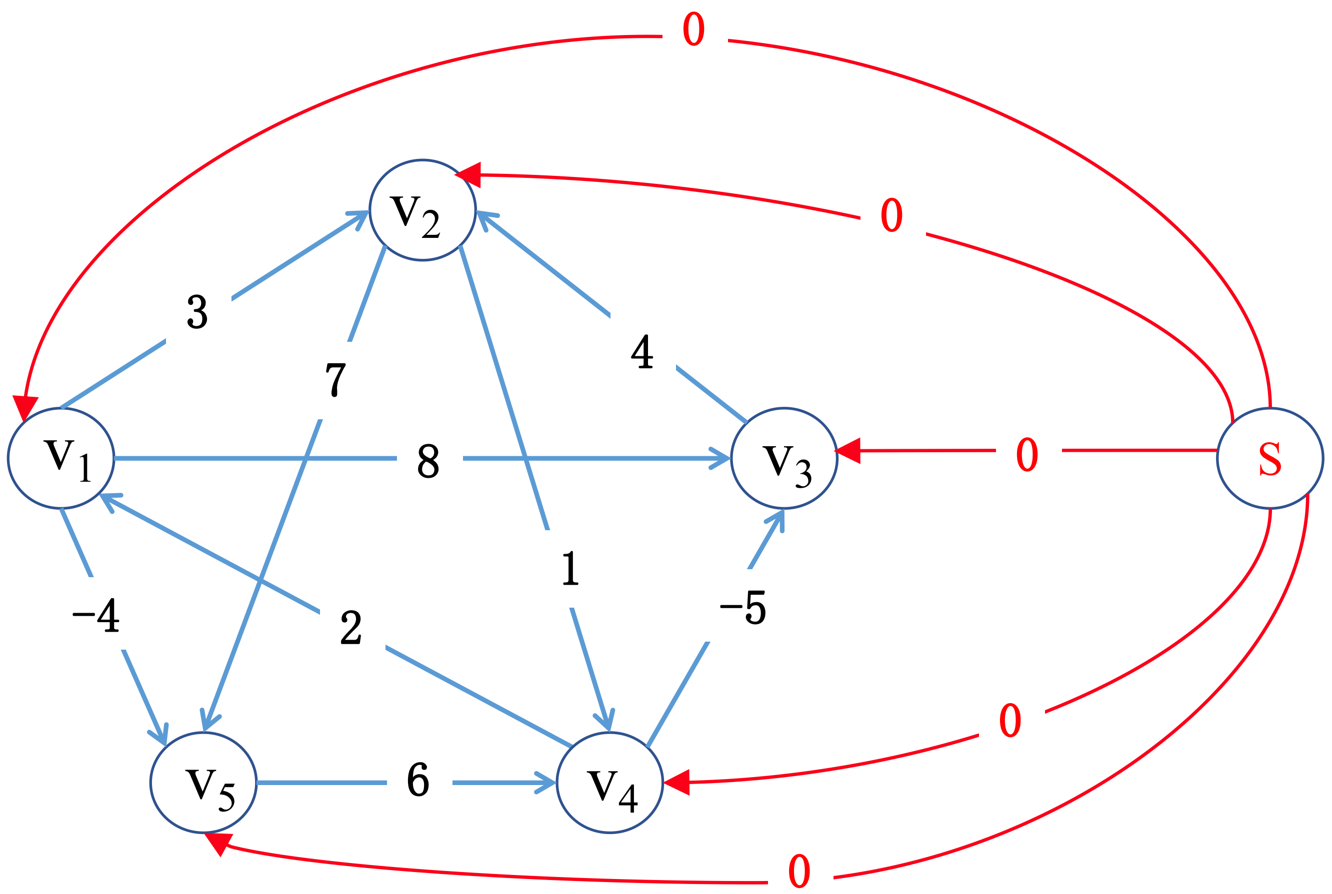
重赋权性质

于是，为了对图进行整体地重赋权以消除负权边，那么我们对于任意一条边(u, v)有$w_h(u,v) = w(u,v) + h(u) - h(v) \geq 0$

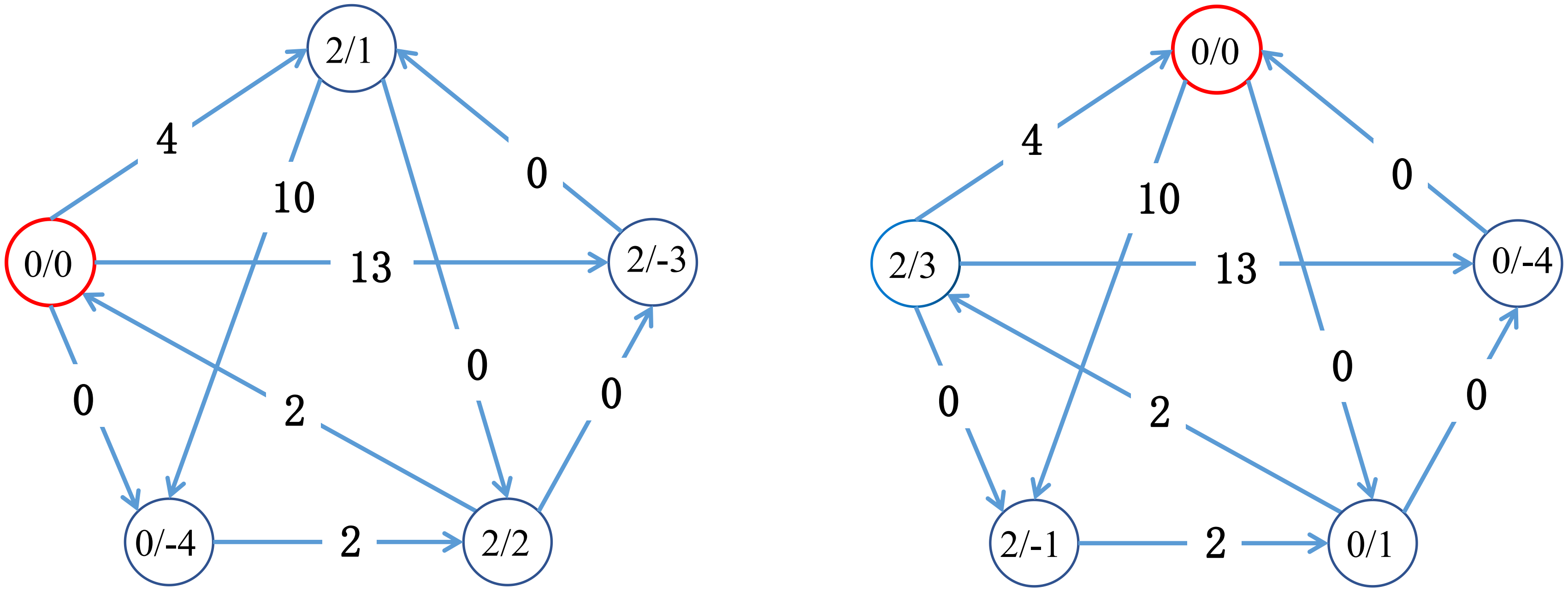
所有边的重赋权限制合起来就形成了一个差分约束系统

求解这个差分约束系统，就可得到重赋权函数h

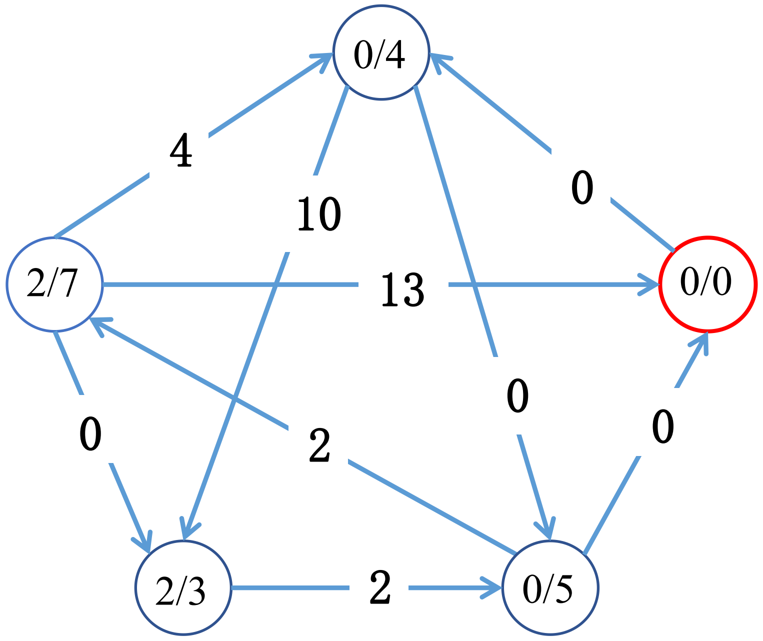
针对重赋权的差分约束系统

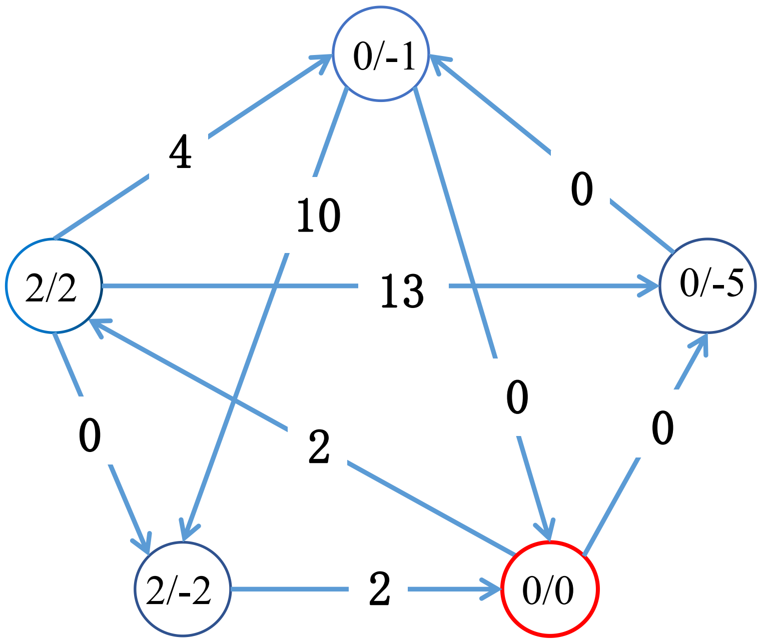


针对重赋权的差分约束系统

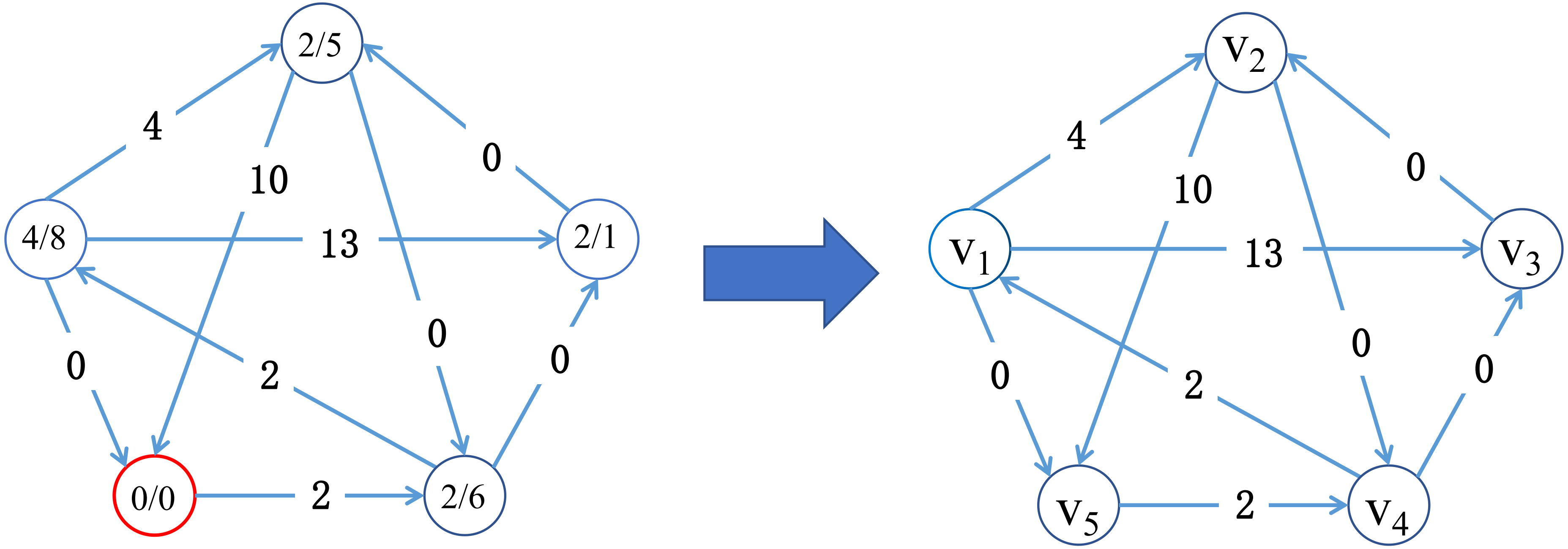


针对重赋权的差分约束系统





重赋权后的图



算法思路

第一步，先使用Bellman-Ford进行差分约束系统求解，得到重赋权函数h，如果h得不到说明有负权环路；

第二步，进行重赋权；

第三步，使用|V|次Dijkstra算法得到全点对间距离，进而得到原图上的最短路径；

复杂度： $O(|V||E|+|V|^2\log|V|)$

湖南大學
HUNAN UNIVERSITY

谢谢观赏

实事求是　敢为人先