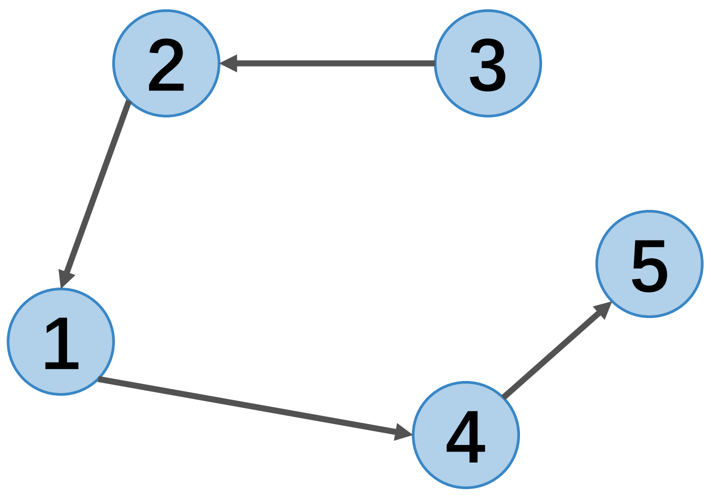
Warshall算法解释

梁润泽

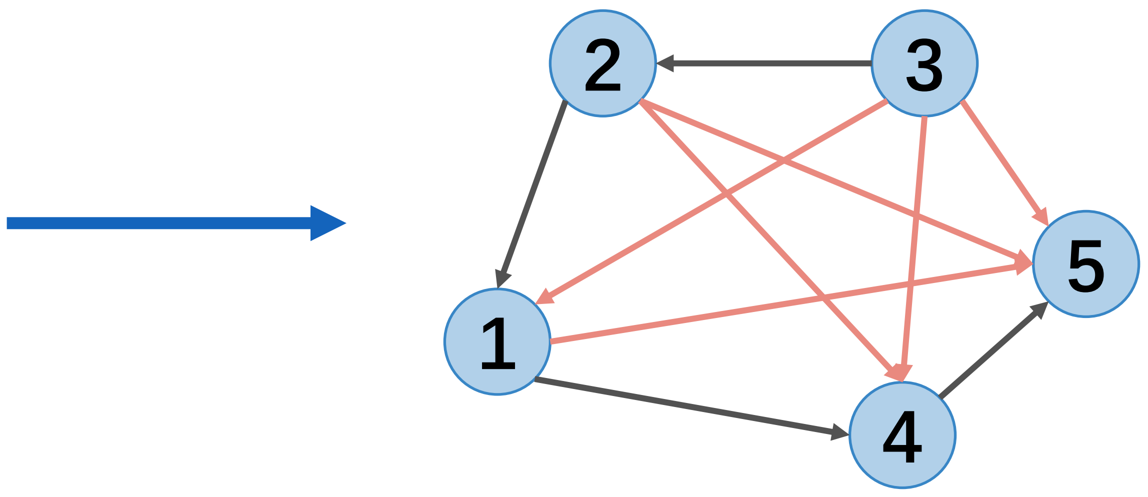
2021.3.26

Warshall算法

- 问题：给定一个关系R，求它的传递闭包$t(R)$

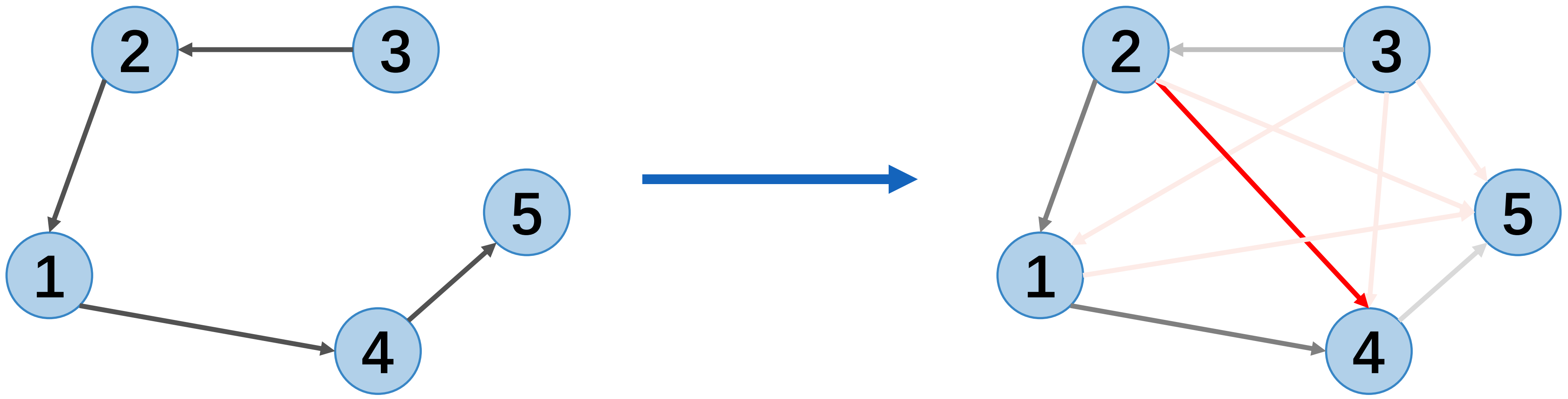


关系R，关系矩阵B



R的传递闭包$t(R)$

Warshall算法

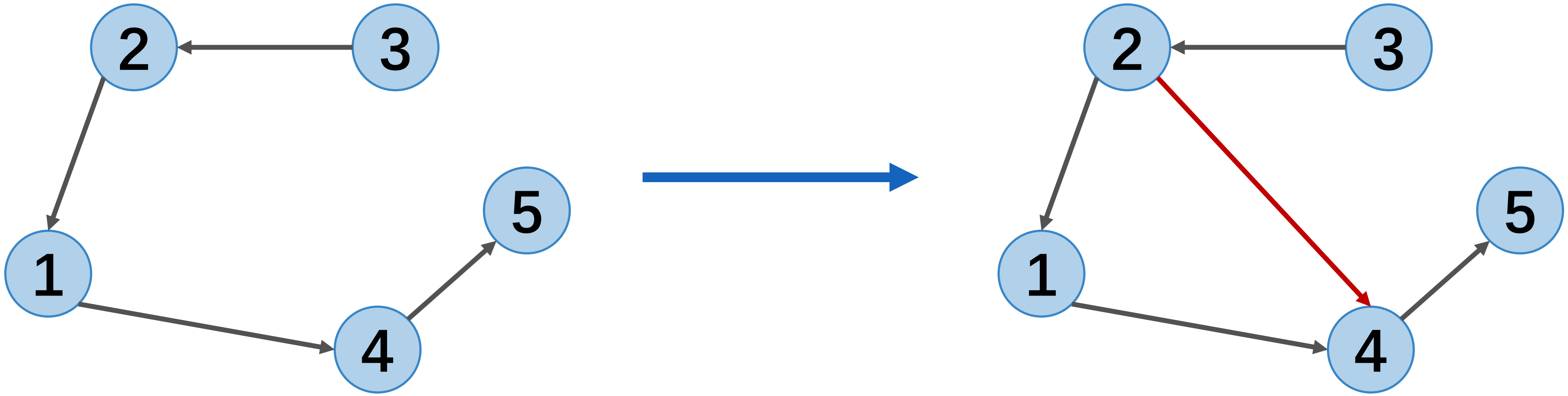


观察

- $<2,1>\in R$，因此$<2,1>\in t(R)$
- $<2,4>\notin R$，但2通过1作为桥梁到达了4，因此$<2,4>\in t(R)$

- Warshall算法：从R开始，依次考虑1、2、3、4、5的桥梁作用，逐步迭代到$t(R)$

Warshall算法: Step 1



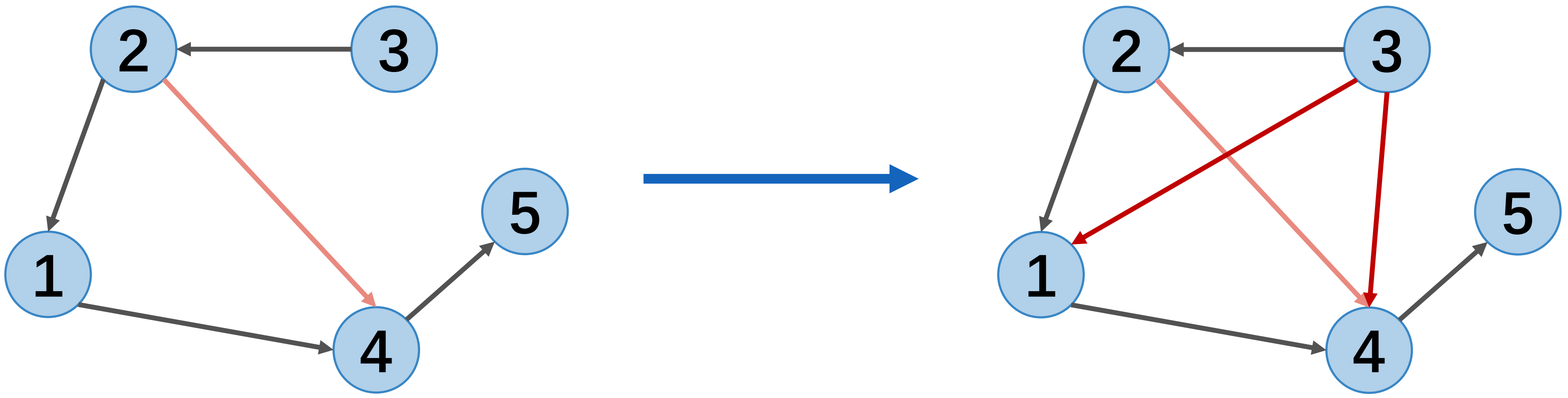
关系R，关系矩阵B_0

中间结果矩阵B_1

观察

- $B_0[4,5] = 1$，因此$B_1[4,5] = 1$
- $B_0[2, 1] = 1$， $B_0[1, 4] = 1$,因此$B_1[2,4] = 1$

Warshall算法: Step 2

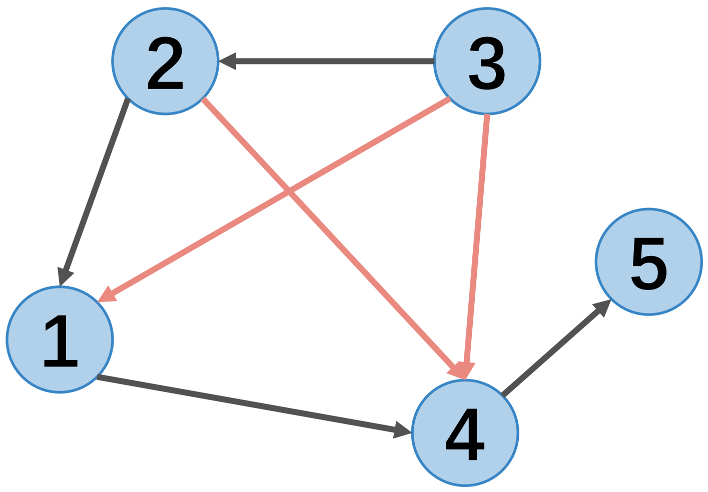


中间结果矩阵B_1 中间结果矩阵B_2

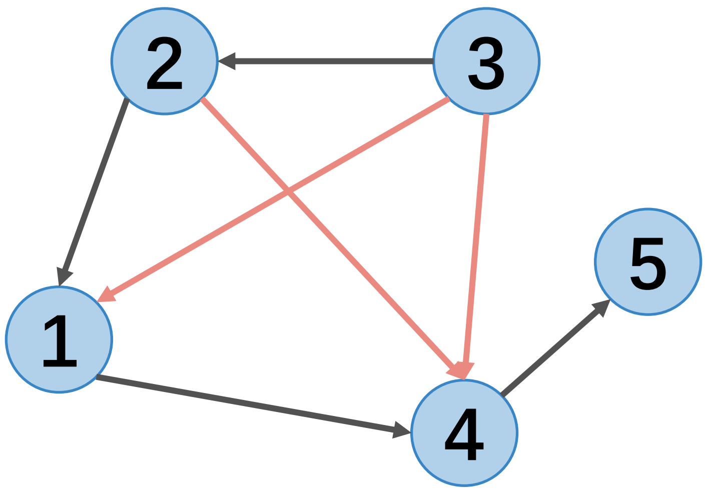
观察

- $B_1[3,2] = 1$， $B_1[2,1] = 1$,因此$B_2[3,1] = 1$
- $B_1[3,2] = 1$， $B_1[2,4] = 1$,因此$B_2[3,4] = 1$ (2→4为第一步扩充)

Warshall算法: Step 3



中间结果矩阵B_2

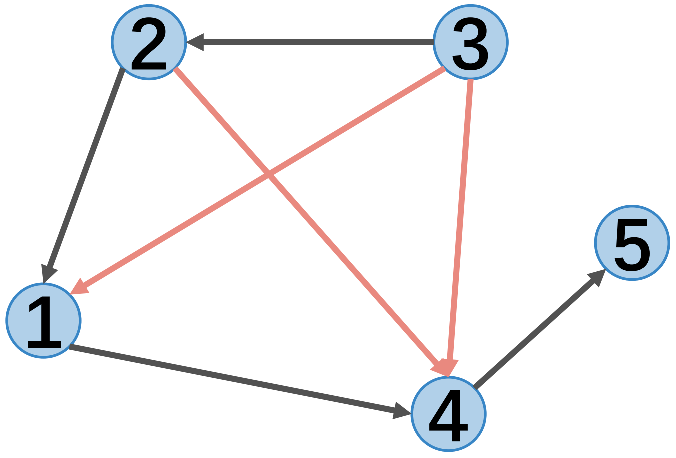


中间结果矩阵B_3

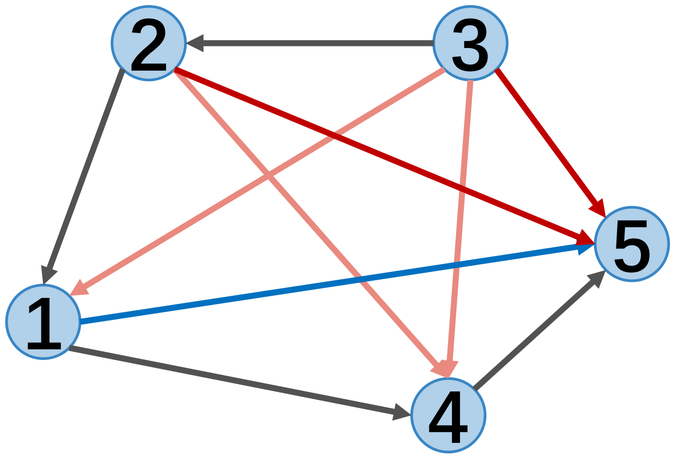
观察

- 节点3没有桥梁作用

Warshall算法: Step 4



中间结果矩阵B_3



中间结果矩阵B_4

Warshall算法伪代码

令B[j, i]表示矩阵B第j行第i列的元素，

（1） 令矩阵B = M(R)；

（2） 令 i =1， n = $|A|$ ； **外循环对列进行**

（3） *for* j =1 *to* n

  if B[j, i]=1 *then*

   for k=1 *to* n

    B[j, k]=B[j, k] ∨B[i, k]

将第i行的元素加到第j行上（逻辑加）

（4） i=i+1；

（5） *if* i≤n *then go to* (3)

  else stop 且 M (R^+) = B

代码运行示例

(2) i=4

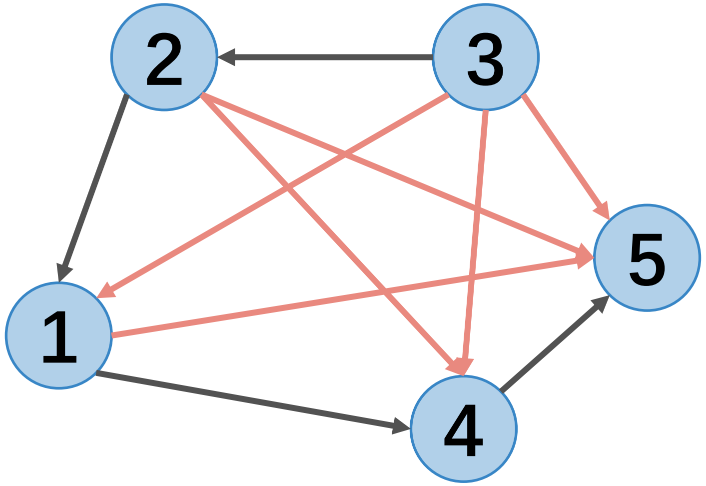
(3) j=1 to 5(以j=1为例）

  if B[1,4]=1 then

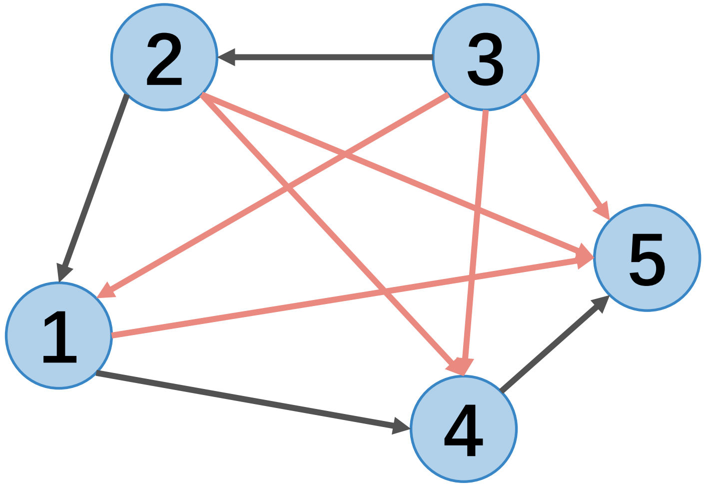
   for k=1 to 5（以k=5为例）

    B[1,5]=B[1,5]∨B[4,5]

Warshall算法: Step 5



中间结果矩阵B_4

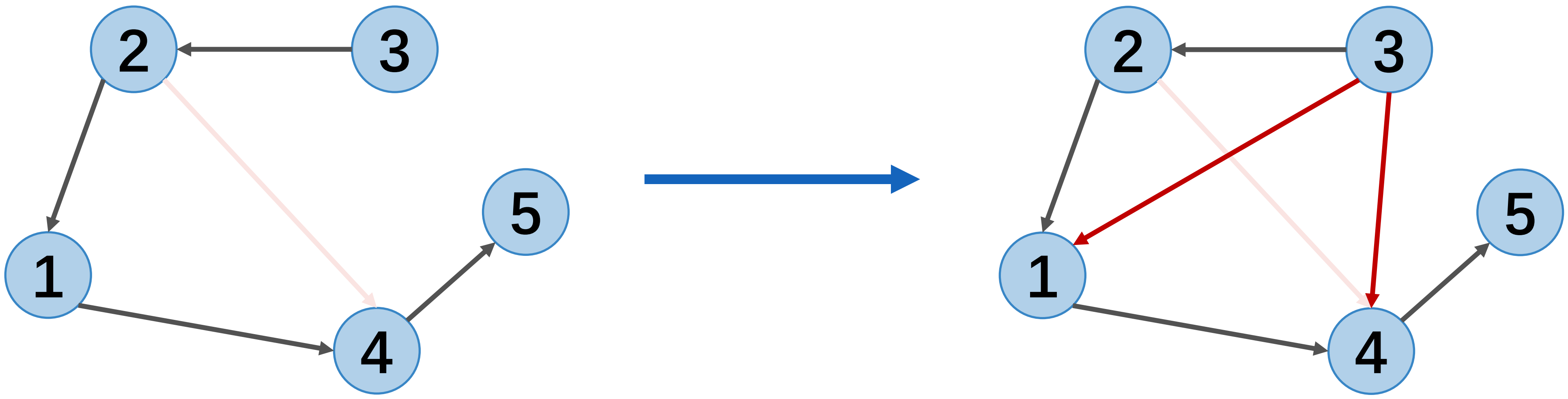


结果矩阵B_5=$t(R)$

观察

- Warshall算法给出了正确的结果
- Warshall算法为什么正确？
- B_1、 B_2、 B_3、 B_4、 B_5各代表什么含义？

Warshall算法: Step 2中B_2的含义



中间结果矩阵B_1

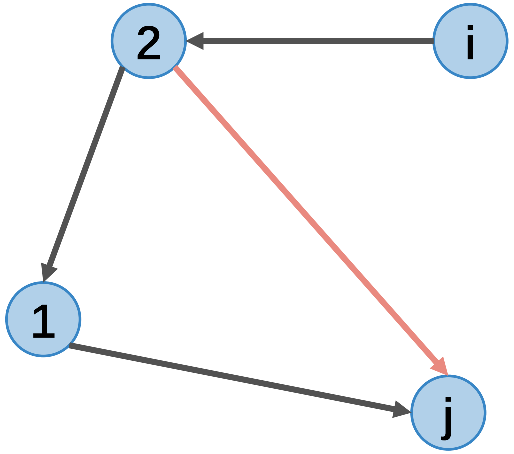
中间结果矩阵B_2

观察

- $B_2[3,4] = 1$，是因为3在B_0中能以{1，2}为桥梁到达4
- $B_2[3,5] = 0$，是因为3在B_0中不能以{1，2}为桥梁到达5
- 注：为了简便，我们也称3在B_0中能以{1}为桥梁到达2

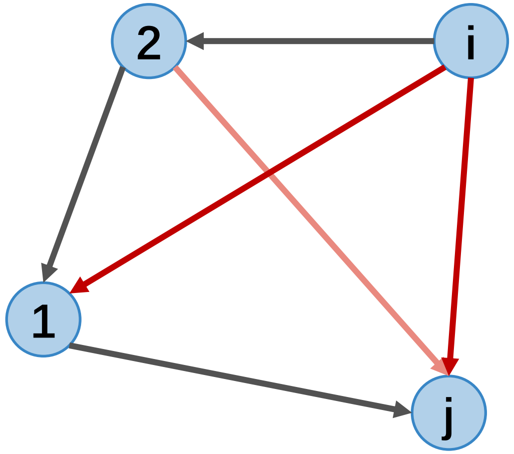
Warshall算法: Step 2中B_2的含义的猜想

Step1



B_1[i,j]=1⇔i在B_0中以 {1}为桥梁到达了j

Step2



B_2[i,j]=1⇔i在B_1中以 {2}为桥梁到达了j

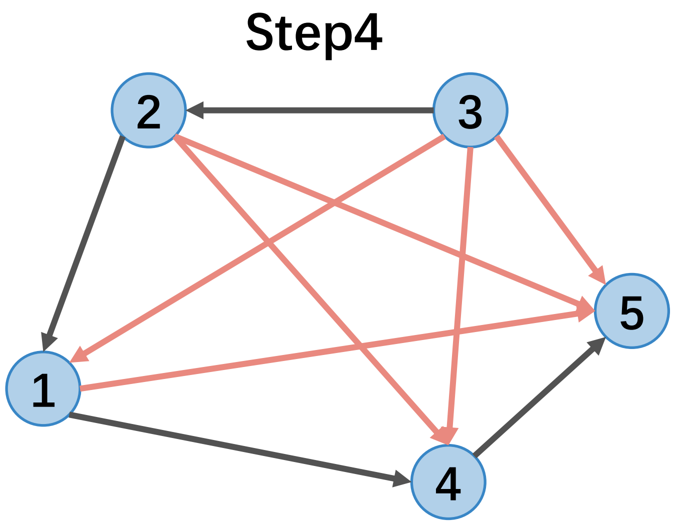
结论

- B_2[i, j]=1⇔i在B_0中以 {1, 2}为桥梁到达了j

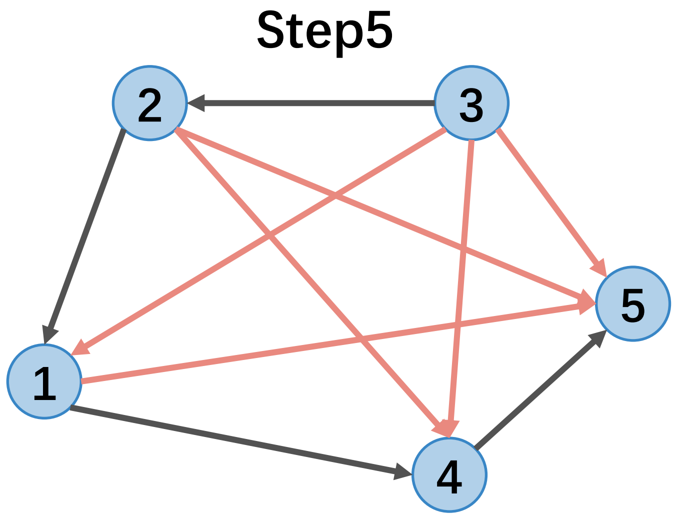
- i在B_0中以 {1}为桥梁到达了j

- B_0[i, 2]=1，且2在B_0中以 {1}为桥梁到达了j
- 或i在B_0中以 {1}为桥梁到达了2，且B_0[2,j]=1

Warshall算法: Step 5中B_5的含义



B_4[i, j]=1⇔i在B_0中以{1，2，3，4}为桥梁到达了j



B_5[i, j]=1⇔i在B_4中以{5}为桥梁到达了j

性质

B_5[i, j]=1⇔i在B_0中以{1，2，3，4，5}为桥梁到达了j

$\Leftrightarrow <\mathrm{i},\mathrm{j}> \in t(R)$

Warshall算法：文字版

- 设有限集合A中有n个元素，分别为$\{a_1, a_2, \ldots, a_n\}$，关系$R$
- 我们说，以下情况，a_j能通过$\{a_1\}$作为中间节点到达a_k：
  - （1） $< a_j, a_k > \in R$(不需要中间节点， a_j直接能到达a_k)；
  - （2） $< a_j, a_1 > \in R$,且$< a_1, a_k > \in R$，此时a_1作为联通a_j和a_k的中间节点
- 我们说，以下情况， a_j能通过$\{a_1, a_2\}$作为中间节点到达a_k:
  - （1） a_j能通过$\{a_1\}$作为中间节点到达a_k，即
    - （1.1） $< a_j, a_k > \in R$
    - （1.2） $< a_j, a_1 > \in R$,且$< a_1, a_k > \in R$

  （2） a_j能通过$\{a_1\}$作为中间节点到达a_2，并且a_2能通过$\{a_1\}$作为中间节点到达a_k
    - （2.1） $< a_j, a_2 > \in R$, $< a_2, a_k > \in R$
    - （2.2） $< a_j, a_2 > \in R$, $< a_2, a_1 > \in R$, $< a_1, a_k > \in R$
    - （2.3） $< a_j, a_1 > \in R$, $< a_1, a_2 > \in R$， $< a_2, a_k > \in R$

Warshall算法：文字版

- 一般地，我们说，以下情况， a_j能通过$\{a_1, a_2, \dots, a_{i-1}, a_i\}$作为中间节点到达$a_k$:
  - a_j能通过$\{a_1, a_2, \dots, a_{i-1}\}$作为中间节点到达$a_k$
  - a_j能通过$\{a_1, a_2, \dots, a_{i-1}\}$作为中间节点到达$a_i$,且$a_i$能通过$\{a_1, a_2, \dots, a_{i-1}\}$作为中间节点到达$a_k$
- 并且，$< a_j, a_k > \in t(R)$,当且仅当a_j能通过$\{a_1, a_2, \dots, a_{n-1}, a_n\}$作为中间节点到达$a_k$
- 于是，我们把上述过程设计为递归算法，这就是Warshall算法

Warshall算法：文字版

令B[j, i]表示矩阵B第j行第i列的元素，

（1） 令矩阵B = M(R)；

（2） 令 i =1， n = $|A|$ ； **外循环对列进行**

（3） *for* j =1 *to* n

if B[j, i]=1 *then*

for k=1 *to* n

B[j, k]=B[j, k] ∨B[i, k]

将第i行的元素加到第j行上（逻辑加）

（4） i=i+1；

（5） *if* i≤n *then go to* (3)

else stop 且 M (R^+) = B

第（1）步时，仅当 $< a_j, a_k > \in R$时B[j,k]=1

第（3）步开始时，如果 a_j能通过$\{a_1, a_2, \ldots, a_{i-1}\}$作为中间节点到达$a_k$,则B[j,k]=1

第（3）步结束时，如果 a_j能通过$\{a_1, a_2, \ldots, a_i\}$作为中间节点到达$a_k$,则B[j,k]=1