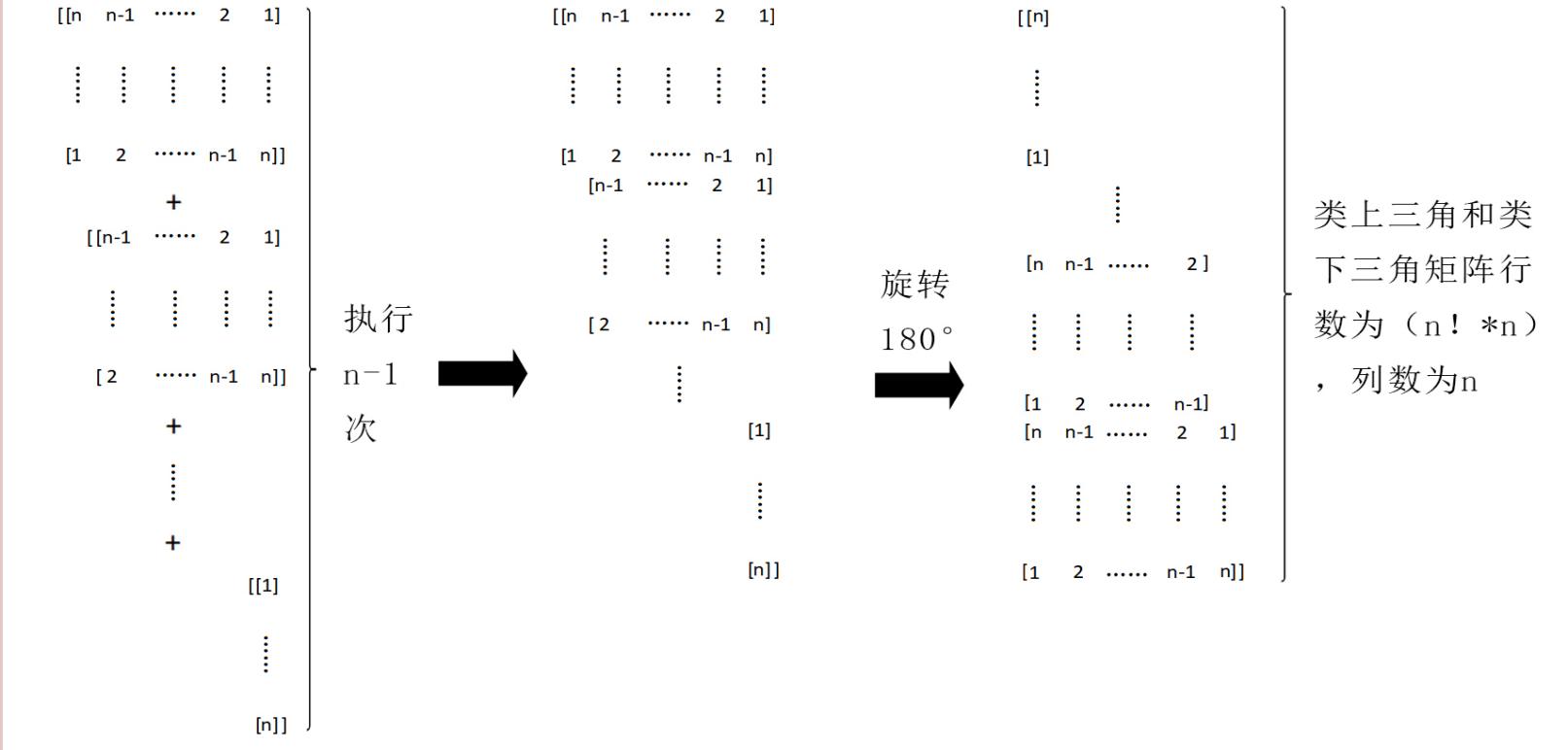
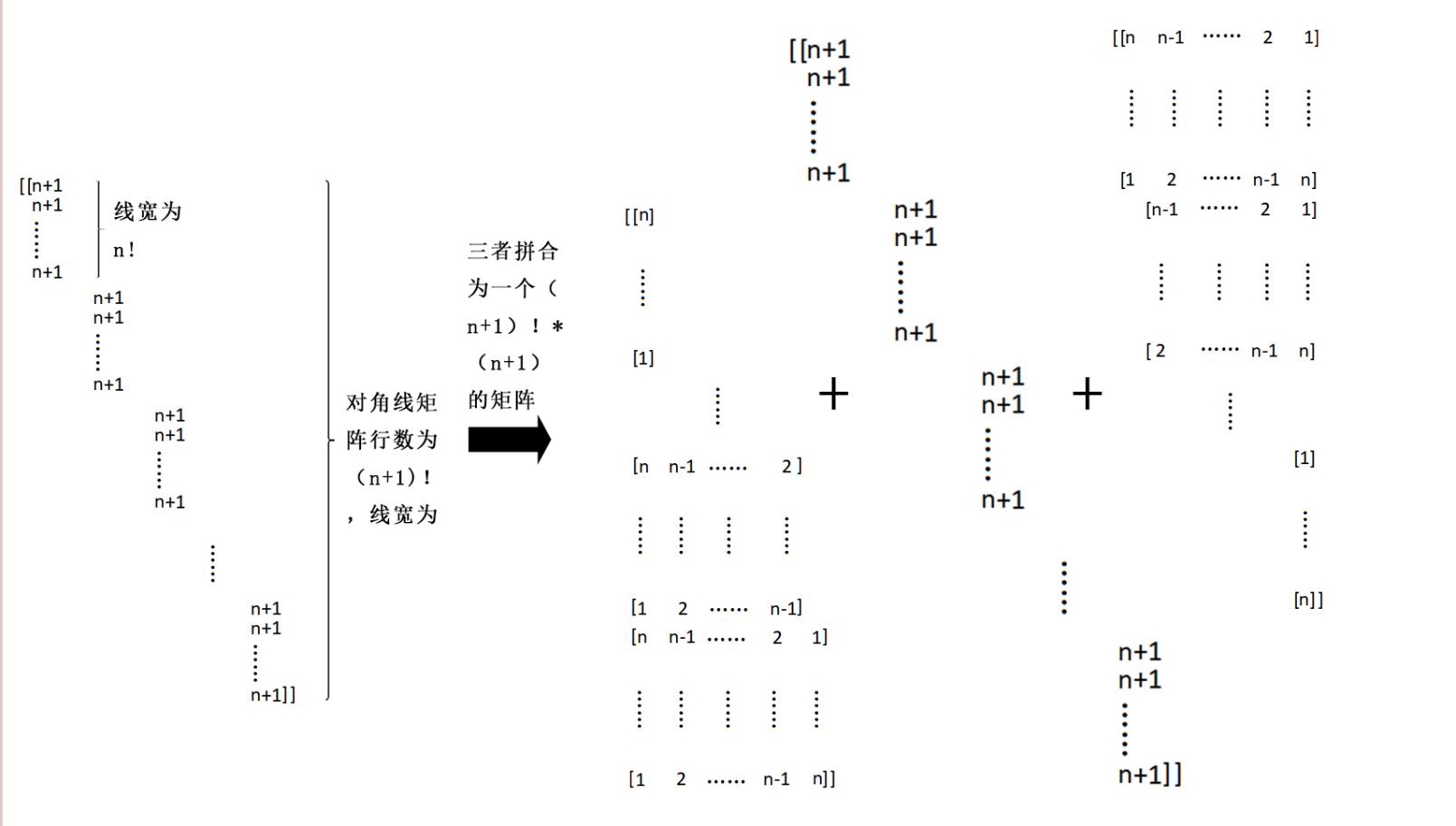
## 全排列的矩阵递推算法

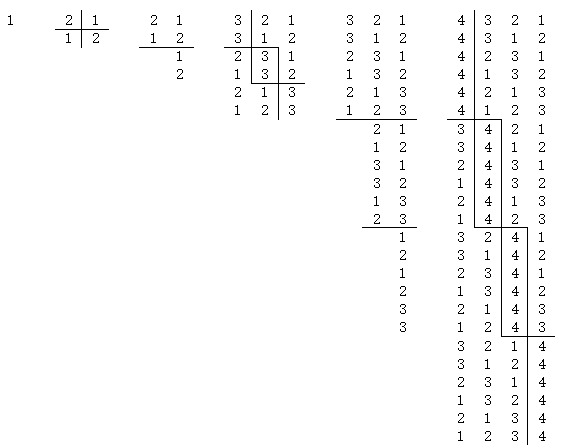
**2.1 全排列递推算法简介**

由于1位数的全排列结果为1，于是需要先构建一个大小1\*1值为1的矩阵初始矩阵。对于n个元素的算法如图1所示，先将n个元素全排列的结果矩阵进行切片，每次切去最左侧一列的数据，然后向下插入，直到矩阵仅剩一列数据为止，得到一个类上三角矩阵。然后将上一步得到的类上三角矩阵旋转180°得到一个类下三角矩阵。然后构建一个线宽为n!的对角线矩阵，将三者合并即可得到N+1个元素全排列的结果，如图2所示。

**图1**

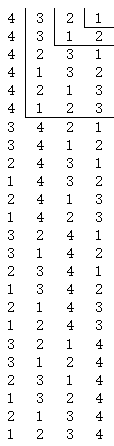


**图2**

现以求解4位数全排列为例，如图3所示。

**图3**

**2.2 全排列递推算法特性**

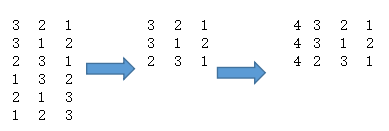
该算法存在两大特性，第一个特性是由该算法算出的矩阵旋转180°，矩阵不变。第二个特性是在你有了一个由该算法算得的n位数全排列矩阵，你可以直接反推得到(1~(n-1))位数全排列结果矩阵。例如反推(n-1)位数全排列结果矩阵，直接读取n位数全排列矩阵右上角或左下角(n-1)!行、(n-1)列的数据，即为(n-1)位数全排列结果矩阵。该特性如图4所示。

**图4**

该算法正推时间复杂度为O(1.5n2+3n),为O(n2)级时间复杂度，反推时间复杂度为O(1)。

## 哈密顿回路算法

**3.1 哈密顿回路与全排列的关联性**

因为无向图不需要考虑路径的方向性，且根据全排列递推矩阵第一个特性，可得上半矩阵与下半矩阵成反方向镜像。由于哈密顿回路可以选择任意一顶点为起始点，所以n个顶点的完全无向图的所有哈密顿回路为(n-1)位数全排列矩阵的上半矩阵加上n构成的新矩阵F，即n个顶点的无向完全图的哈密顿回路共有(n-1)!/2条。4个顶点的完全无向图的哈密顿回路如图5所示。

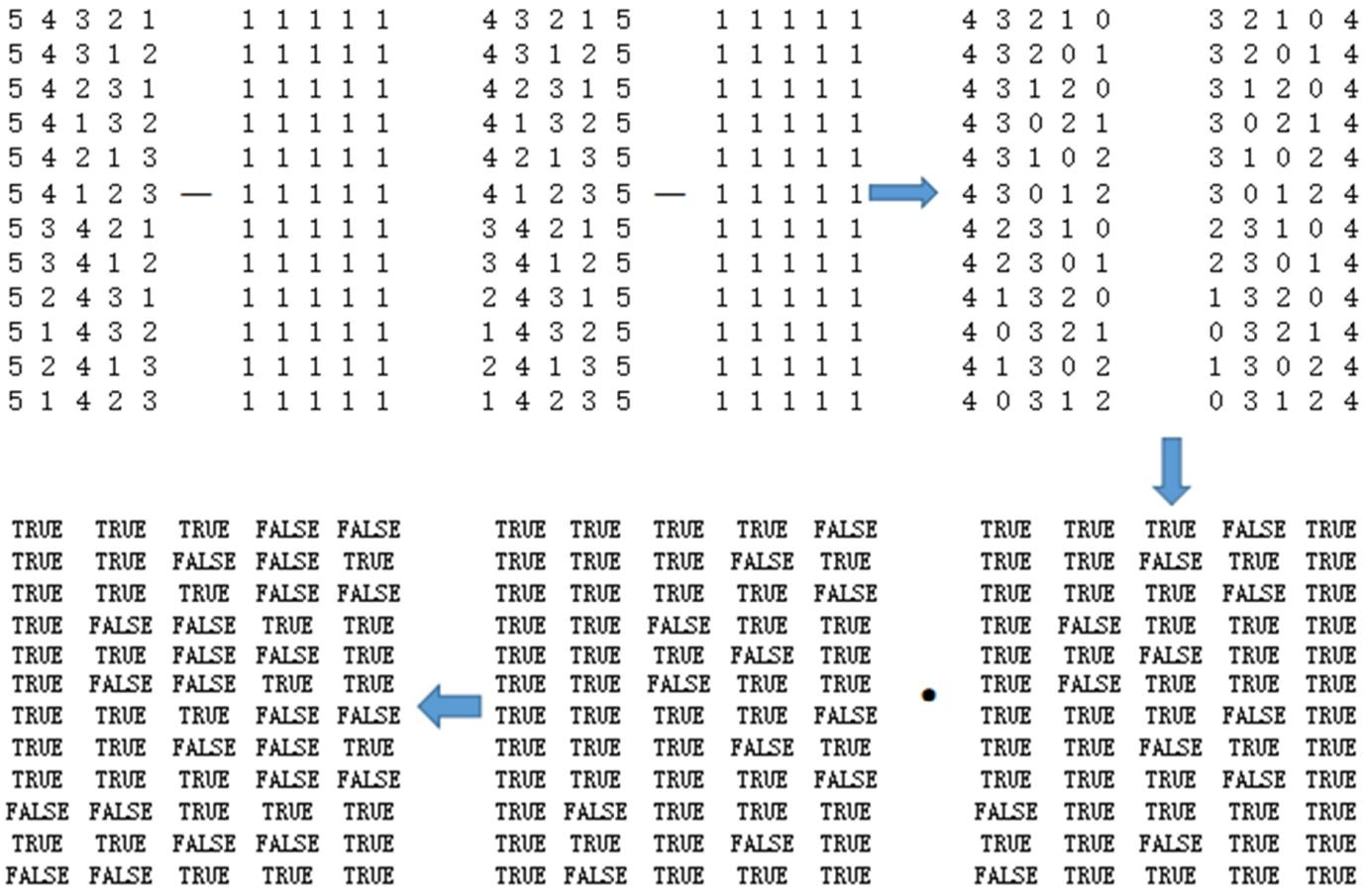
**图5**

**3.1 哈密顿回路算法简介**

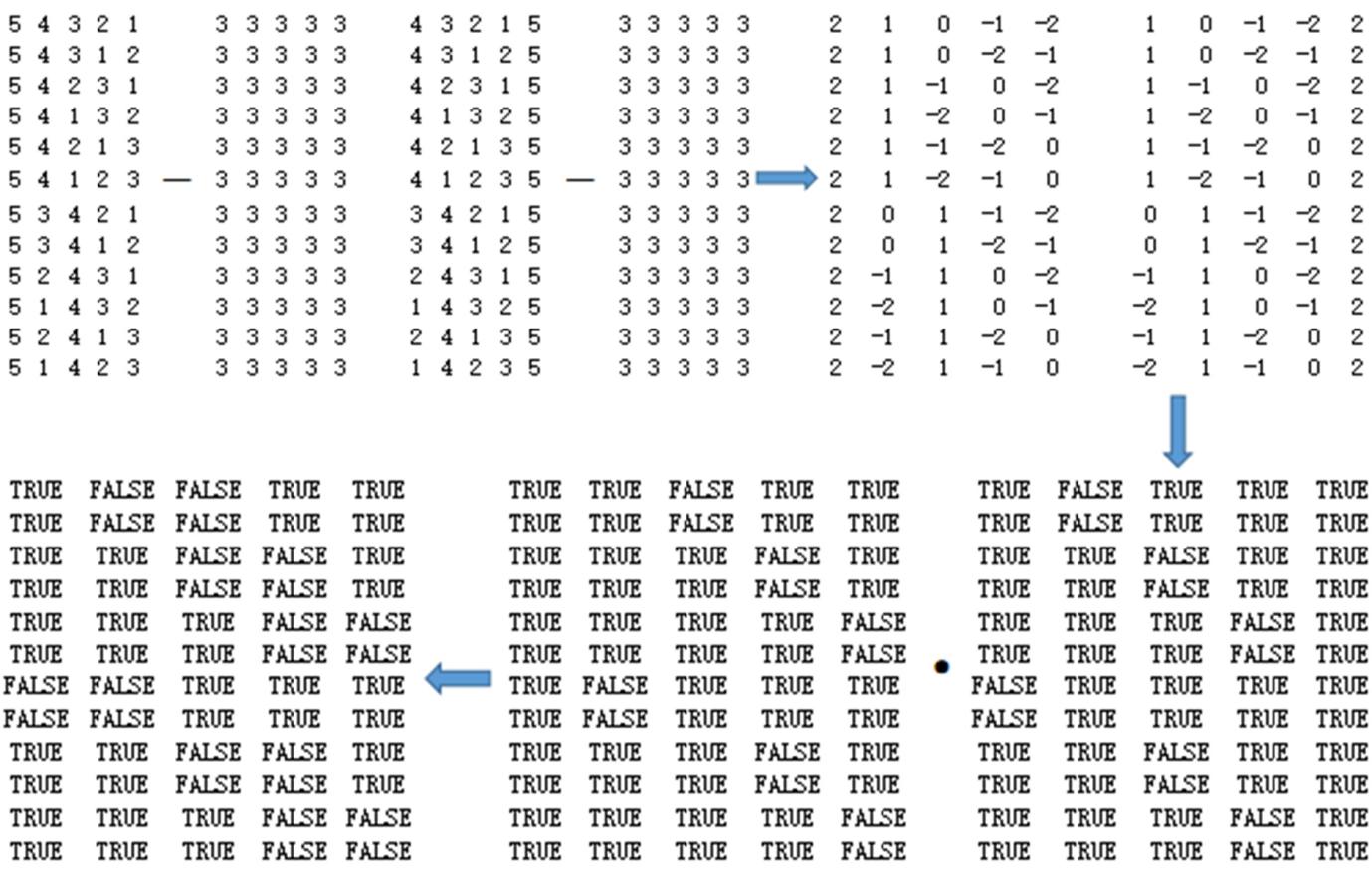
对于任意无向图G=(N,V)，N为顶点集，V为路径集。将其对比n个顶点的完全无向图，将缺少的路径构成一个缺失路径集V1，然后取出缺失路径集V1中所有路径对应的顶点，构成一个缺失顶点集N1,,然后计算出缺失顶点集N1中所有顶点的过渡矩阵。

设总顶点数为n的任意无向图，顶点s∈N1，s的过渡矩阵算法为(n-1)!/2\*n的(n-1)位数全排列矩阵的上半矩阵减去(n-1)!/2\*n值为s的矩阵，然后将其布尔化，得到矩阵T1。再将(n-1)!/2\*n的(n-1)位数全排列矩阵的上半矩阵的第一列以致最后一列，同样减去(n-1)!/2\*n值为s的矩阵，然后将其布尔化，得到矩阵T2。最后s的过渡矩阵T=T1·T2。

现以5个顶点无向图举例，计算顶点1和顶点3的过渡矩阵，如图6和图7所示。



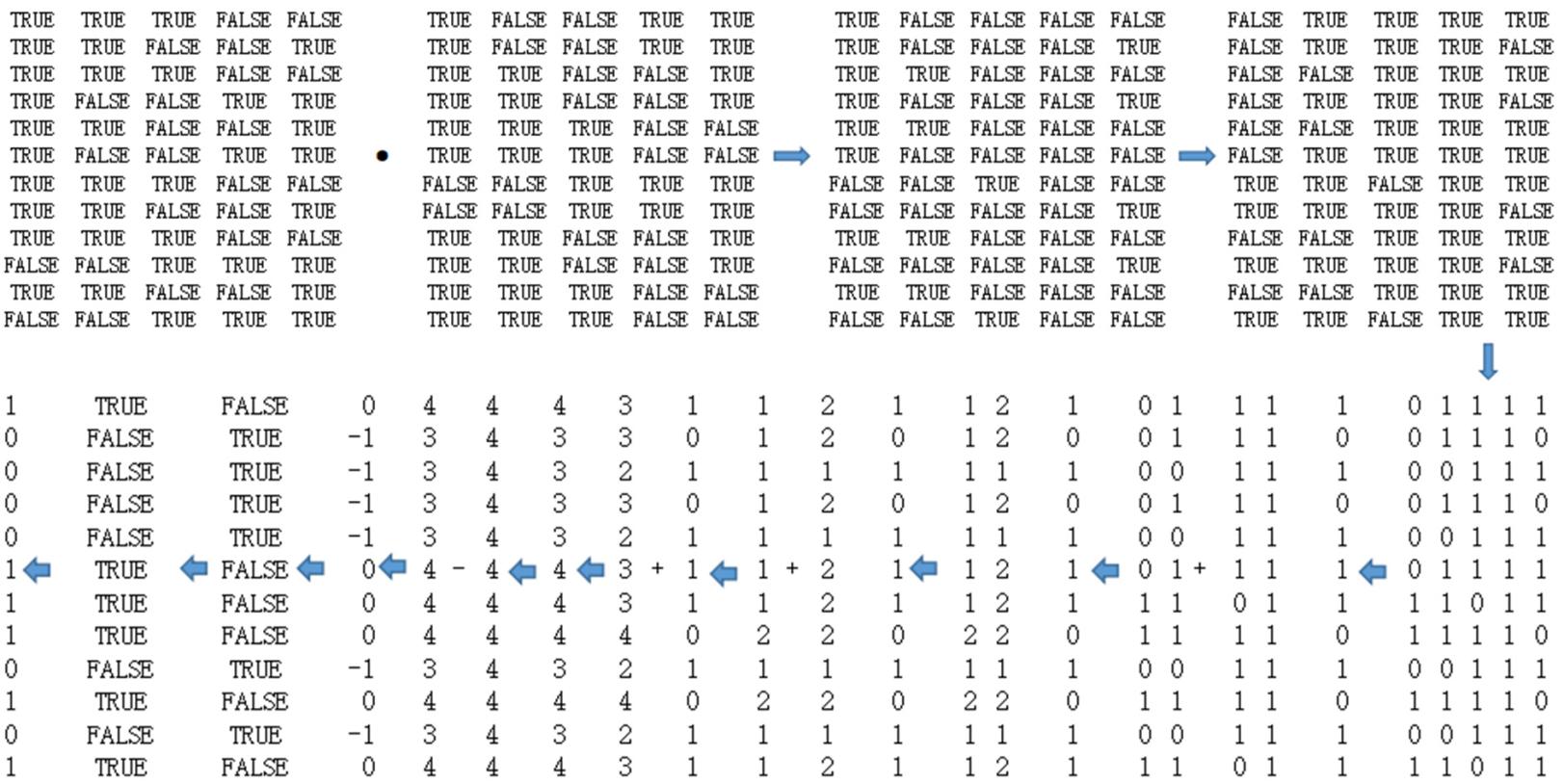
**图6**

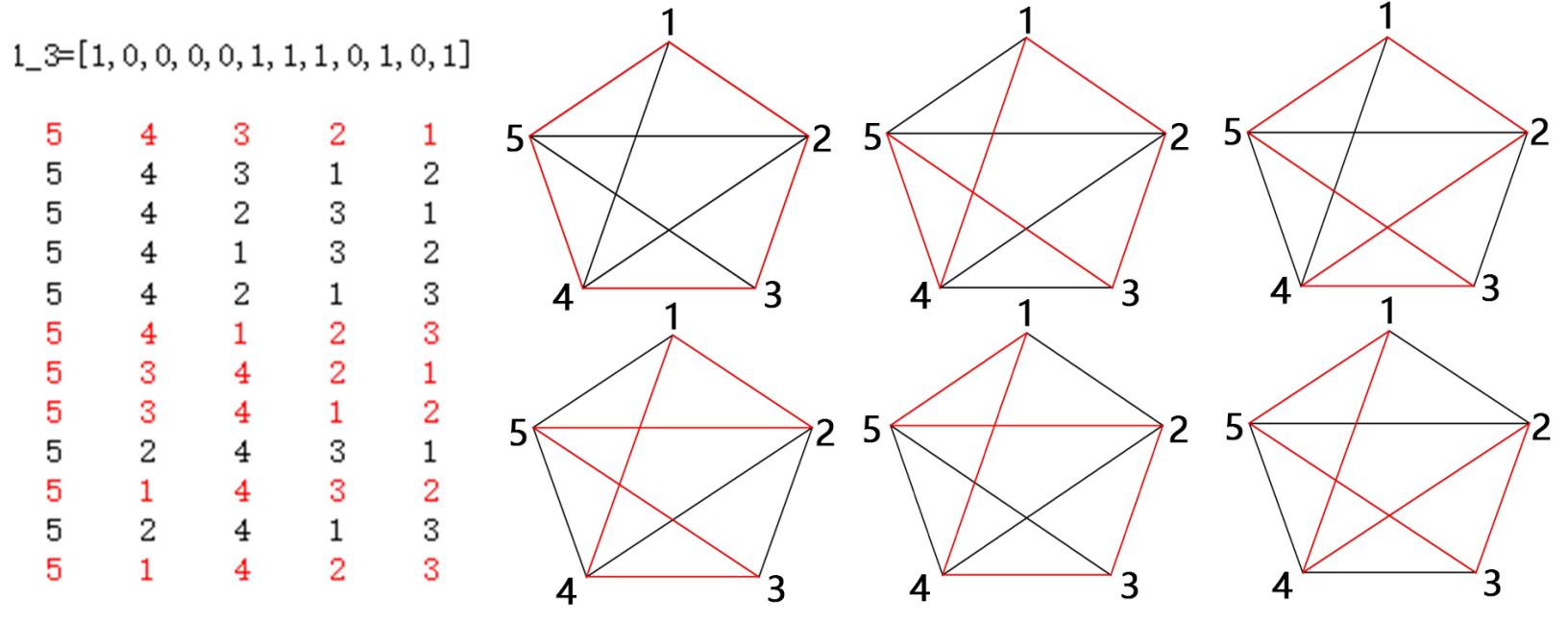


**图7**

设总顶点数为n的任意无向图，路径f∈V1，设构成路径的两个顶点为s和d，将s和d的过渡矩阵相乘，再反向，然后把数据类型转化为整型，将列重复二分相加，最后得到一个(n-1)!/2\*1的列向量A。生成一个(n-1)!/2\*1值为4的列向量B，令A-B=C，将列向量C转置得到CT,将CT布尔化然后反向，再将数据类型转化为整型，就得到了路径f的行向量D。行向量D的特点为按向量中值为1的序号取出矩阵F对应行的值，这些值为对比n个顶点的完全无向图缺少路径f的所有哈密顿回路。

现以5个顶点无向图举例，计算路径1\_3的行向量，以及缺少路径1\_3的所有哈密顿回路，如图8和图9所示。

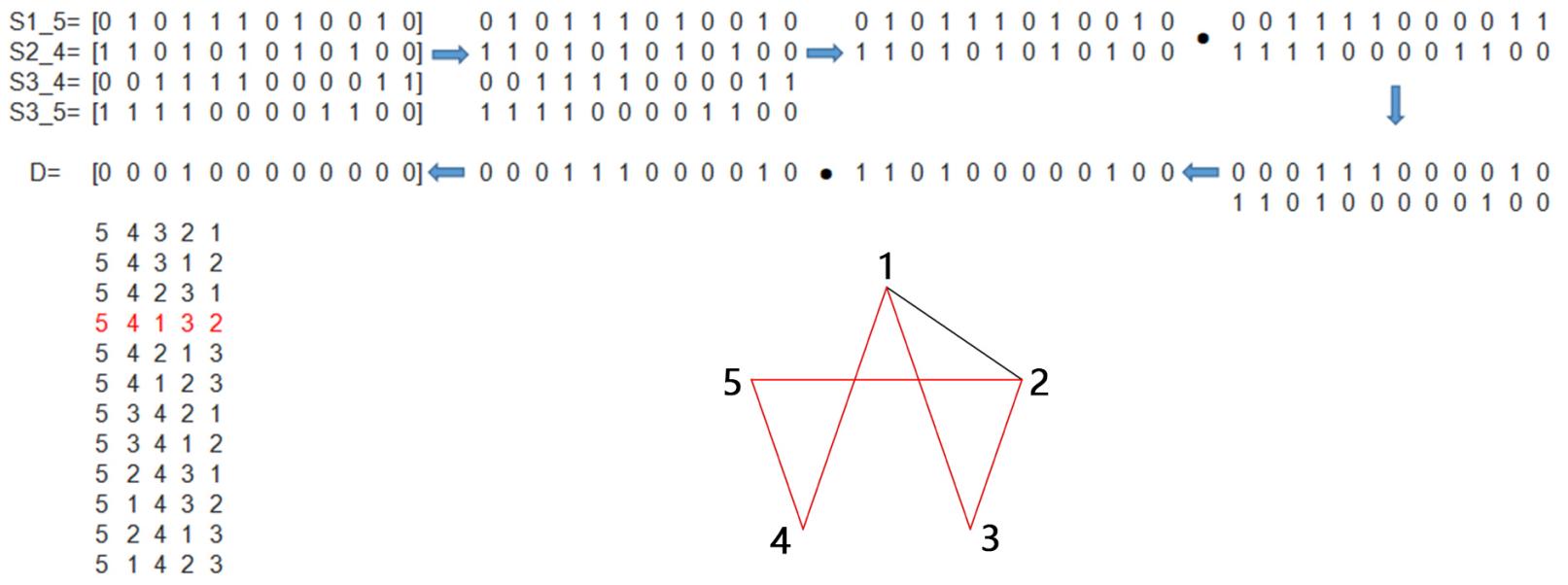


**图8**

**图9**

对于总顶点数为n的任意无向图，V1大小大于1的情况，需要算得V1中所有路径的行向量。然后将所有行向量组成一个矩阵，然后将矩阵按行二分相乘，最后得到行向量D，这里计算得到行向量D的特性与单一路径的行向量一致。

现以5个顶点无向图举例，设['1\_5','2\_4','3\_4', '3\_5']∈V1,行向量D以及该图所有哈密顿回路如图10。



**图10**