6.图

邻接矩阵

复杂接口

06-B4

邓俊辉

deng@tsinghua.edu.cn

边的插入

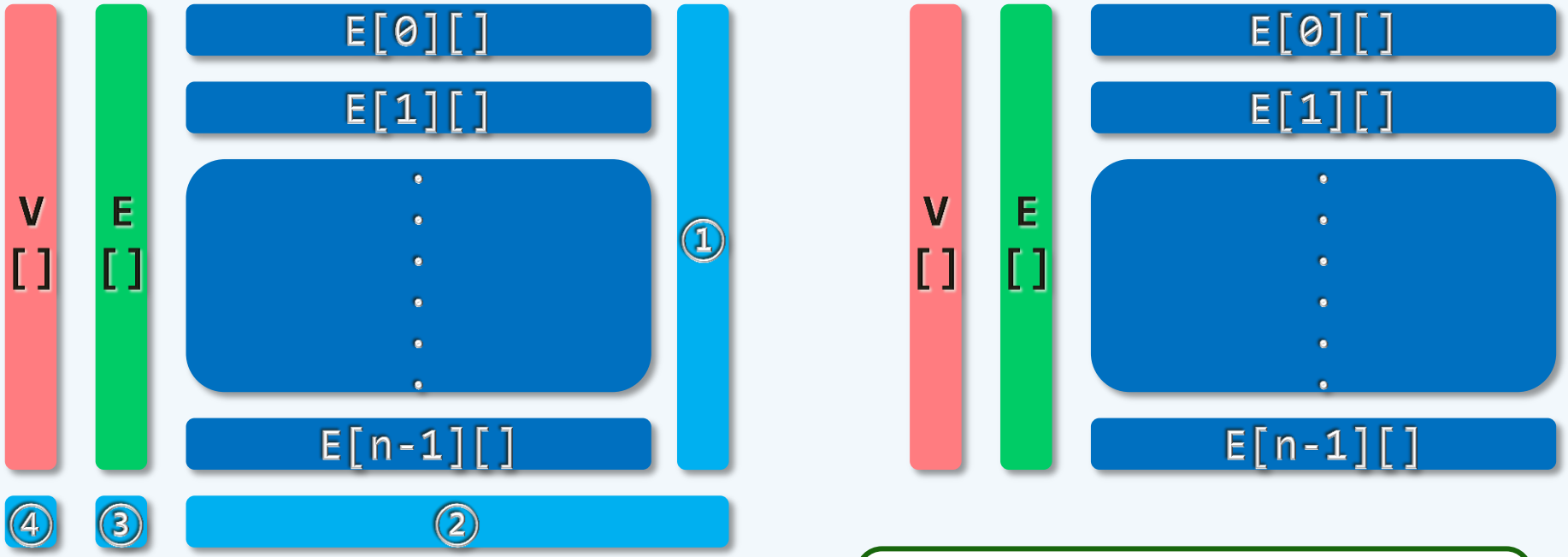
```
❖void insert( Te const & edge, int w, int i, int j ) { //插入(i, j, w)
    if ( exists(i, j) ) return; //忽略已有的边
    E[i][j] = new Edge<Te>( edge, w ); //创建新边
    e++; //更新边计数
    V[i].outDegree++; //更新关联顶点i的出度
    V[j].inDegree++; //更新关联顶点j的入度
  }
```

边的删除

```
❖Te remove( int i, int j ) { //删除顶点i和j之间的联边（exists(i, j)）
    Te eBak = edge(i, j); //备份边(i, j)的信息
    delete E[i][j]; E[i][j] = NULL; //删除边(i, j)
    e--; //更新边计数
    V[i].outDegree--; //更新关联顶点i的出度
    V[j].inDegree--; //更新关联顶点j的入度
    return eBak; //返回被删除边的信息
  }
```

顶点插入

```
❖int insert( Tv const & vertex ) { //插入顶点，返回编号
    for ( int j = 0; j < n; j++ ) E[j].insert( NULL ); n++; //①
    E.insert( Vector< Edge<Te>* >( n, n, NULL ) ); //②③
    return V.insert( Vertex<Tv>( vertex ) ); //④
}
```



顶点删除

```
❖ Tv remove( int i ) { //删除顶点及其关联边，返回该顶点信息
    for ( int j = 0; j < n; j++ ) //删除所有出边
       if ( exists( i, j ) ) { delete E[i][j]; V[j].inDegree--; }
    E.remove(i); n--; //删除第i行
    Tv vBak = vertex( i ); V.remove( i );  //备份之后，删除顶点i
    for ( int j = 0; j < n; j++ ) //删除所有入边及第i列
       if ( Edge<Te> * e = E[j].remove( i ) ) { delete e; V[j].outDegree--; }
    return vBak; //返回被删除顶点的信息
  }
```