

图

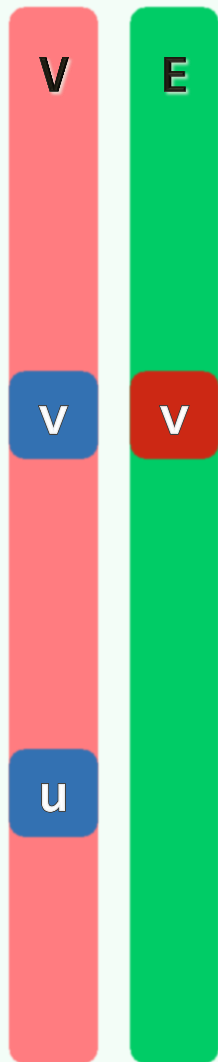
邻接矩阵：动态操作

10-B4

邓俊辉

deng@tsinghua.edu.cn

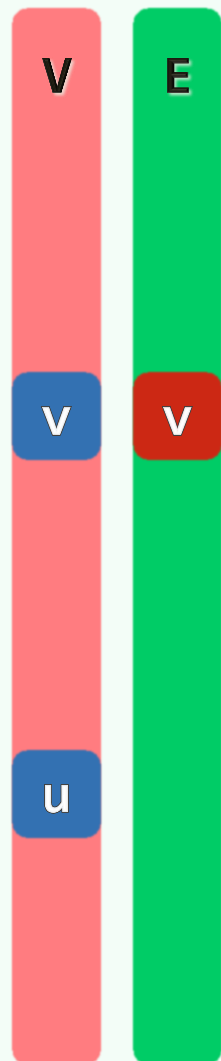
边的插入



```
void insert( Te const & edge, int w, Rank v, Rank u ) {  
    if ( exists(v, u) ) return; //忽略已有的边  
  
    E[v][u] = new Edge<Te>( edge, w ); //创建新边 (权重为w)  
  
    e++; //更新边计数  
  
    V[v].outDegree++; //更新顶点v的出度  
  
    V[u].inDegree++; //更新顶点u的入度  
}
```



边的删除



```
Te remove( Rank v, Rank u ) { //删除 (已确认存在的) 边(v, u)
```

```
    Te eBak = edge(v, u); //备份边(v, u)的信息
```

```
    delete E[v][u]; E[v][u] = NULL; //删除边(v, u)
```



```
    e--; //更新边计数
```

```
    V[v].outDegree--; //更新顶点v的出度
```

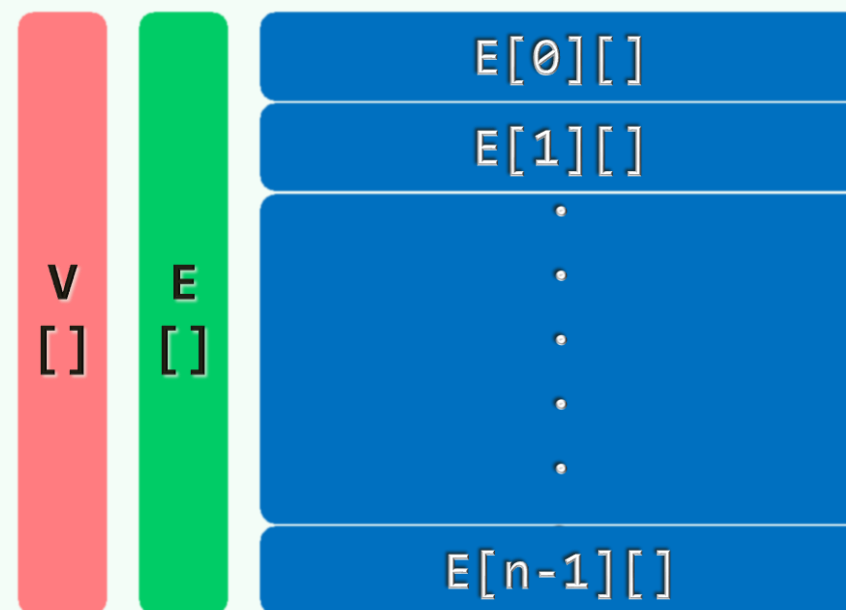
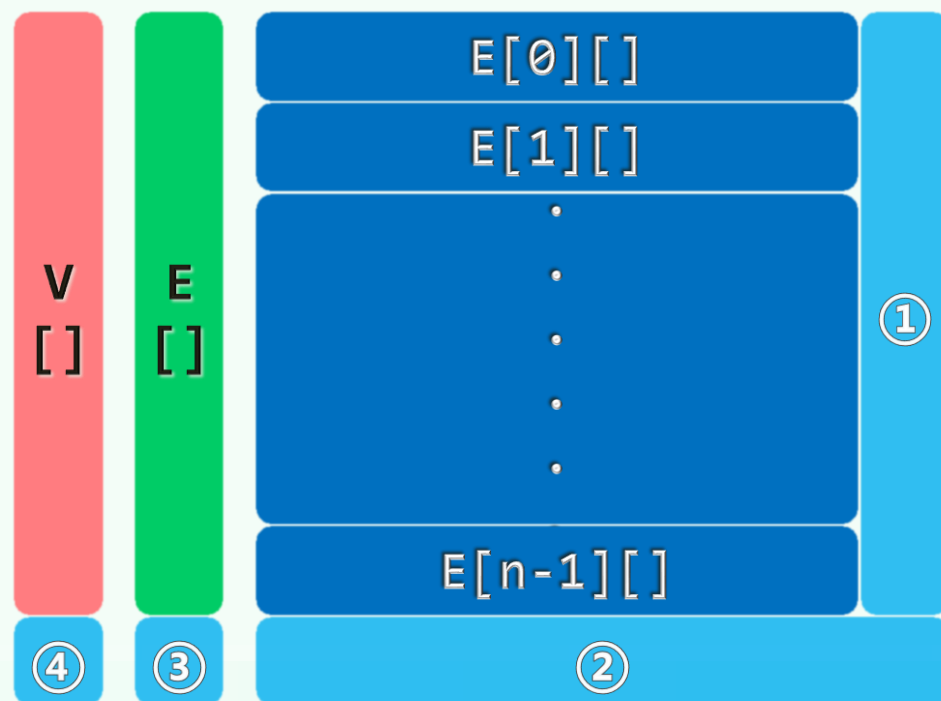
```
    V[u].inDegree--; //更新顶点u的入度
```

```
    return eBak; //返回被删除边的信息
```

```
}
```

顶点插入

```
Rank insert( Tv const & vertex ) { //插入顶点, 返回编号  
    for ( Rank u = 0; u < n; u++ ) E[u].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( Rank v ) { //删除顶点及其关联边, 返回该顶点信息

    for ( Rank u = 0; u < n; u++ ) //删除所有出边

        if ( exists( v, u ) ) { delete E[v][u]; V[u].inDegree--; e-- }

    E.remove(v); n--; //删除第v行

    Tv vBak = vertex( v ); V.remove( v ); //备份之后, 删除顶点v

    for ( Rank u = 0; u < n; u++ ) //删除所有入边及第v列

        if ( Edge<Te> * x = E[u].remove( v ) )

            { delete x; V[u].outDegree--; e--; }

    return vBak; //返回被删除顶点的信息

}
```