

图应用

双连通分量：算法

11-A2

她停了片刻：“不管怎么说，我觉得今晚比以前任何时候跟祖父都靠得近了。我想他老人家肯定会很高兴的。”

邓俊辉

deng@tsinghua.edu.cn

Graph::BCC()

```
#define hca(x) ( fTime(x) ) //利用此处闲置的fTime

template <typename Tv, typename Te>

void Graph<Tv, Te>::BCC( Rank v, int & clock, Stack<Rank> & S ) {

    hca(v) = dTime(v) = ++clock; status(v) = DISCOVERED; S.push(v);

    for ( Rank u = firstNbr(v); -1 < u; u = nextNbr(v, u) )
        switch ( status(u) )
        { /* ... 视u的状态分别处理 ... */ }

    status(v) = VISITED; //对v的访问结束

}

#undef hca
```

```
switch ( status(u) )
```

```
case UNDISCOVERED:
```

```
    parent(u) = v; type(v, u) = TREE; //拓展树边
```

```
    BCC( u, clock, S ); //从u开始遍历, 返回后...
```

```
    if ( hca(u) < dTime(v) ) //若u经后向边指向v的真祖先
```

```
        hca(v) = min( hca(v), hca(u) ); //则v亦必如此
```

```
    else //否则, 以v为关节点 (u以下即是一个BCC, 且其中顶点此时正集中于栈S的顶部)
```

```
        while ( u != S.pop() ); //弹出当前BCC中 (除v外) 的所有节点
```

```
    break;
```

```
switch ( status(u) )
```

```
case DISCOVERED:
```

```
    type(v, u) = BACKWARD;
```

```
    if ( u != parent(v) )
```

```
        hca(v) = min( hca(v), dTime(u) ); //更新hca(v), 越小越高
```

```
    break;
```

```
default: //VISITED (digraphs only)
```

```
    type(v, u) = dTime(v) < dTime(u) ? FORWARD : CROSS;
```

```
    break;
```

复杂度

❖ 运行时间与常规的DFS相同，也是 $O(n + e)$

自行验证：栈操作的复杂度也不过如此

❖ 除原图本身，还需一个容量为 $O(e)$ 的栈存放已访问的边

为支持递归，另需一个容量为 $O(n)$ 的运行栈

❖ 如何推广至有向图的**强连通分量**

(Strongly-connected component)

- Kosaraju's algorithm
- Tarjan's algorithm

