

图

邻接矩阵：模板实现

10-B2

邓俊辉

deng@tsinghua.edu.cn

Vertex

```
using VStatus = enum { UNDISCOVERED, DISCOVERED, VISITED };

template <typename Tv> struct Vertex { //不再严格封装

    Tv data; int inDegree, outDegree;

    VStatus status; // (如上三种) 状态

    int dTime, fTime; //时间标签

    Rank parent; //在遍历树中的父节点

    int priority; //在遍历树中的优先级 (最短通路、极短跨边等)

    Vertex( Tv const & d ) : //构造新顶点

        data( d ), inDegree( 0 ), outDegree( 0 ), status( UNDISCOVERED ),

        dTime( -1 ), fTime( -1 ), parent( -1 ), priority( INT_MAX ) {}

};
```

Edge

```
using EType = enum { UNDETERMINED, TREE, CROSS, FORWARD, BACKWARD };

template <typename Te> struct Edge { //不再严格封装

    Te data; //数据

    int weight; //权重

    EType type; //在遍历树中所属的类型

    Edge( Te const & d, int w ) : //构造新边

        data(d), weight(w), type(UNDETERMINED) {}

};
```

GraphMatrix

```
template <typename Tv, typename Te> class GraphMatrix : public Graph<Tv, Te> {  
private:
```

```
    Vector< Vertex<Tv> > V; //顶点集
```

```
    Vector< Vector< Edge<Te>* > > E; //边集
```

```
public: // 操作接口：顶点相关、边相关、...
```

```
    GraphMatrix() { n = e = 0; } //构造
```

```
    ~GraphMatrix() { //析构
```

```
        for ( Rank v = 0; v < n; v++ )
```

```
            for ( Rank u = 0; u < n; u++ )
```

```
                delete E[v][u]; //清除所有边记录
```

```
    }
```

```
};
```

