

6. 图

邻接矩阵

简单接口

邓俊辉

deng@tsinghua.edu.cn

顶点的读写

❖ Tv & vertex(int i) { return V[i].data; } //数据

int inDegree(int i) { return V[i].inDegree; } //入度

int outDegree(int i) { return V[i].outDegree; } //出度

VStatus & status(int i) { return V[i].status; } //状态

int & dTime(int i) { return V[i].dTime; } //时间标签dTime

int & fTime(int i) { return V[i].fTime; } //时间标签fTime

int & parent(int i) { return V[i].parent; } //在遍历树中的父亲

int & priority(int i) { return V[i].priority; } //优先级数

邻点的枚举

❖ 对于任意顶点 i ，如何枚举其所有的邻接顶点 $neighbor$ ？

```
❖ int nextNbr( int i, int j ) { //若已枚举至邻居j，则转向下一邻居  
    while ( ( -1 < j ) && ! exists( i, --j ) ); //逆向顺序查找， $O(n)$   
    return j;  
} //改用邻接表可提高至 $O(1 + outDegree(i))$   
  
❖ int firstNbr( int i ) {  
    return nextNbr( i, n ); //假想哨兵  
} //首个邻居
```

边的读写

- ❖ `bool exists(int i, int j) { //判断边(i, j)是否存在 (短路求值)`
 `return (0 <= i) && (i < n) && (0 <= j) && (j < n) && E[i][j] != NULL;`
 `} //以下假定exists(i, j)...`
- ❖ `Te & edge(int i, int j) //边(i, j)的数据`
 `{ return E[i][j]->data; } //O(1)`
- ❖ `EType & type(int i, int j) //边(i, j)的类型`
 `{ return E[i][j]->type; } //O(1)`
- ❖ `int & weight(int i, int j) //边(i, j)的权重`
 `{ return E[i][j]->weight; } //O(1)`