

图

邻接矩阵：静态操作

10-B3

邓俊辉

deng@tsinghua.edu.cn

# 顶点的读写

V

E

V

Tv & vertex(Rank v) { return V[v].data; } //数据

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

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

VStatus & status(Rank v) { return V[v].status; } //状态

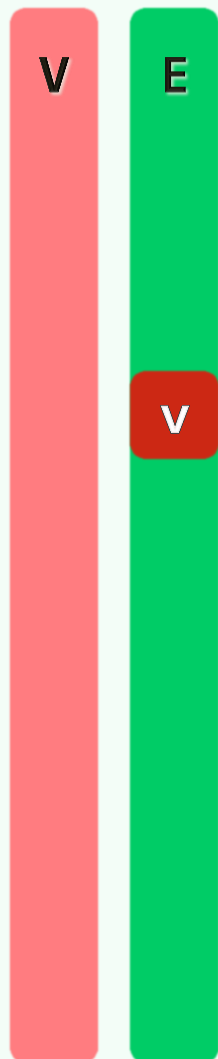
int & dTime(Rank v) { return V[v].dTime; } //时间标签dTime

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

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

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

# 边的读写



```
bool exists( Rank v, Rank u ) { //判断边(v, u)是否存在 (短路求值)  
    return (v < n) && (u < n) && E[v][u] != NULL;  
} //以下假定exists(v, u) = true
```



```
Te & edge( Rank v, Rank u ) { return E[v][u]->data; } //数值  
EType & type( Rank v, Rank u ) { return E[v][u]->type; } //类型  
int & weight( Rank v, Rank u ) { return E[v][u]->weight; } //权重
```

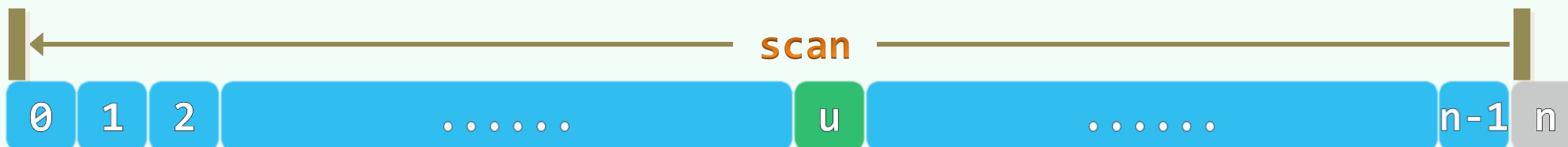
# 邻点的枚举

V

E

对于任意顶点 $v$ ，如何枚举其所有的邻接顶点 (**neighbor**) ?

```
Rank firstNbr( Rank v ) { return nextNbr( v, n ); } //假想哨兵
```



```
Rank nextNbr( Rank v, Rank u ) { //若已枚举至邻居 $u$ ，则转向下一邻居
```

```
    while (  $-1 < u$  ) && ! exists( v,  $--u$  ); //逆向顺序查找
```

```
    return u;
```

```
} //  $O(n)$ ——改用邻接表，可提高至  $O(1 + \text{outDegree}(v))$ 
```