

## 12B1 完全二叉堆：结构

#数据结构邓神

完全二叉树：

采用 向量，List 并不足够

采用 BBST 虽然可以实现，但是太过高级

以向量为形，以Tree为神，实现二者的有机集合

完全二叉树 Complete Binary Tree

**完全二叉树**

Complete Binary Tree

平衡因子处处非负的AVL 而且...

若  $bf(v) = 0$ ，则  $lc(v)$  满

若  $bf(v) = 1$ ，则  $rc(v)$  满

结构性

**结构性**

逻辑上，等同于完全二叉树

物理上，直接借助向量实现

逻辑节点与物理元素

依层次遍历次序彼此对应

```
#define Parent( i ) ( ( i - 1 ) >> 1 )
#define LChild( i ) ( 1 + ( ( i ) << 1 ) ) //奇数
#define RChild( i ) ( ( 1 + ( i ) ) << 1 ) //偶数
```

只在最低层缺失若干个节点

在逻辑上，等同于完全数

物理上，直接借助向量实现

在物理上的所有元素依然构成一个向量，但是在逻辑结构上完全是一个二叉树

## Complete Binary Heap 完全二叉堆 形具神备

PQ\_ComplHeap = PQ + Vector

~~PQ\_ComplHeap~~ = ~~PQ~~ + ~~Vector~~

```
❖ template <typename T> class PQ_ComplHeap : public PQ<T>, public Vector<T>
{
protected:
    Rank percolateDown( Rank n, Rank i ); //下滤
    Rank percolateUp( Rank i ); //上滤
    void heapify( Rank n ); //Floyd建堆算法
public:
    PQ_ComplHeap( T* A, Rank n ) //批量构造
    { copyFrom( A, 0, n ); heapify( n ); }
    void insert( T ); //按照比较器确定的优先级次序，插入词条
    T getMax() { return _elem[0]; } //读取优先级最高的词条
    T delMax(); //删除优先级最高的词条
};
```

我没有使用它老师自己写的Vector数组，同时C++本身的vector与老师的并不一致，所有代码稍有缺别

```
#define Parent(i) ((i-1) >> 1)
#define LChild(i) (1+((i)<<1))
#define RChild(i) ((1+(i)) << 1)

template <typename T> class PQ_ComplHeap : public PQ<T> {
    vector<T> _elem;

    typedef int Rank;
    Rank percolateDown(Rank n, Rank i);
    Rank percolateUp(Rank i);
    void heapify(Rank n);
public:
    PQ_ComplHeap(T* A, Rank n){
        _elem.copy(A, 0, n);
    }
};
```

```

        heapify(n);
    }
    void insert(T);
    T getMax(){ // O(1)
        return _elem[0];
    }
    T delMax();
};

```

## 堆序性 灵魂!

数值: 只要  $0 < i$ , 必须要满足  
 $H[i] < H[\text{Parent}(i)]$

根据上述规则, 不难推断出, 最大元必须在根节点处

