#ADS05 Binomial Queue

二项队列

左式堆将合并、删除最小元的操作的时间控制在**O(logN)**,二项队列进一步减少了这个时间。

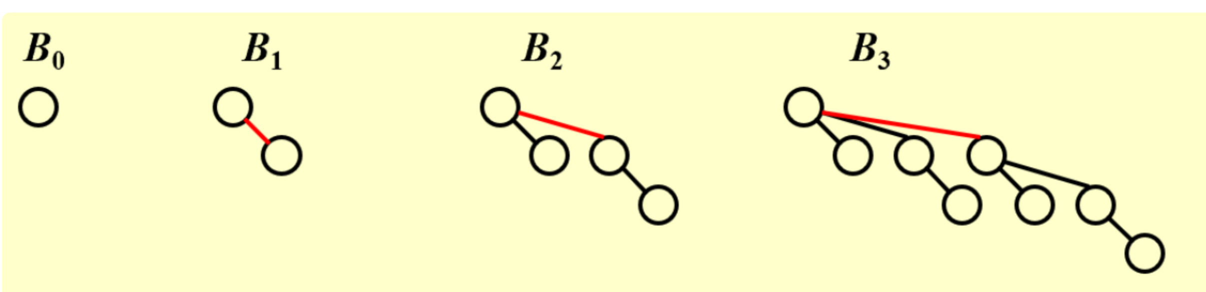
Binomial以最坏时间O(logN)支持上述操作，**插入操作平均花费常数时间**

Structure：

A binomial queue is not a heap-ordered tree, but rather **a collection of heap-ordered trees**, known as a forest. Each heap-ordered tree is a binomial tree.

A binomial tree of height 0 is a one-node tree.

A binomial tree, B_k, of height k is formed by attaching a binomial tree, B_{k-1}, to the root of another binomial tree, B_{k-1}.



B_k consists of a root with k children, which are B_0 to B_{k-1}.

B_k has exactly 2^k nodes. The number of nodes at depth d is Ckd

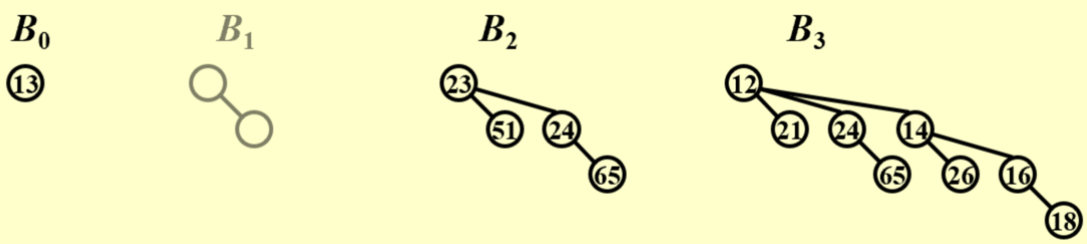
B_k structure + heap order + one binomial tree for each height

–> A priority queue of any size can be uniquely represented by a collection of binomial trees.

如果我们把堆序添加在二项树上，并允许任意高度上最多有一棵二项树，那么我们能**用二项树的集合唯一标识任意大小的优先队列**。

【Example】Represent a priority queue of size 13 by a collection of binomial trees.

Solution: $13 = 2^0 + 0\times2^1 + 2^2 + 2^3 = 1101_2$



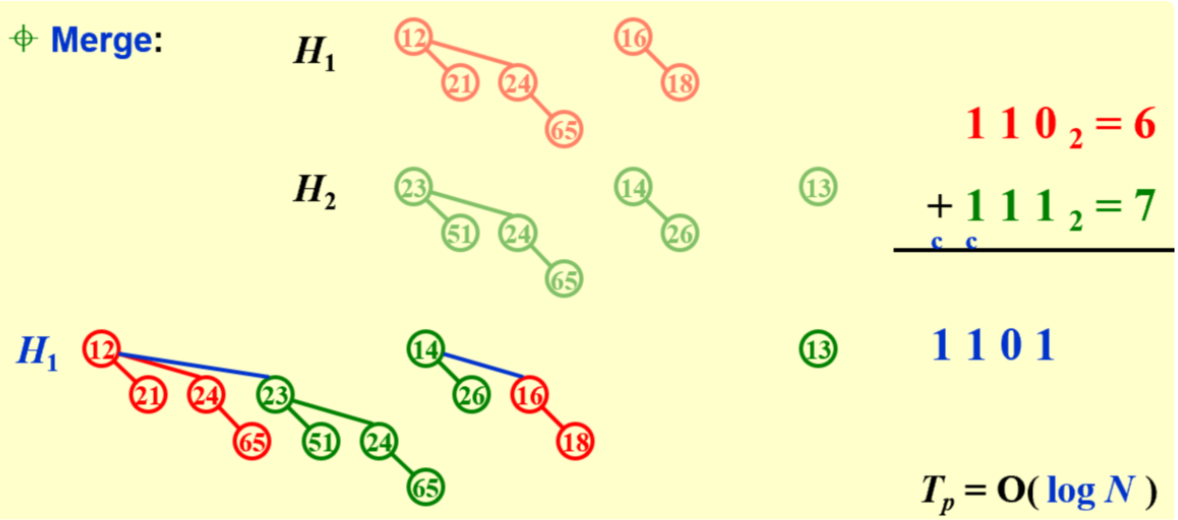
FindMin

The minimum is in one of the roots.

There are at most $\lceil \log N \rceil$ roots, hence $T_p = O(\log N)$.

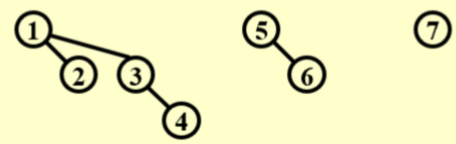
We can remember the minimum and update whenever it is changed. Then this operation will take O(1)

Merge:



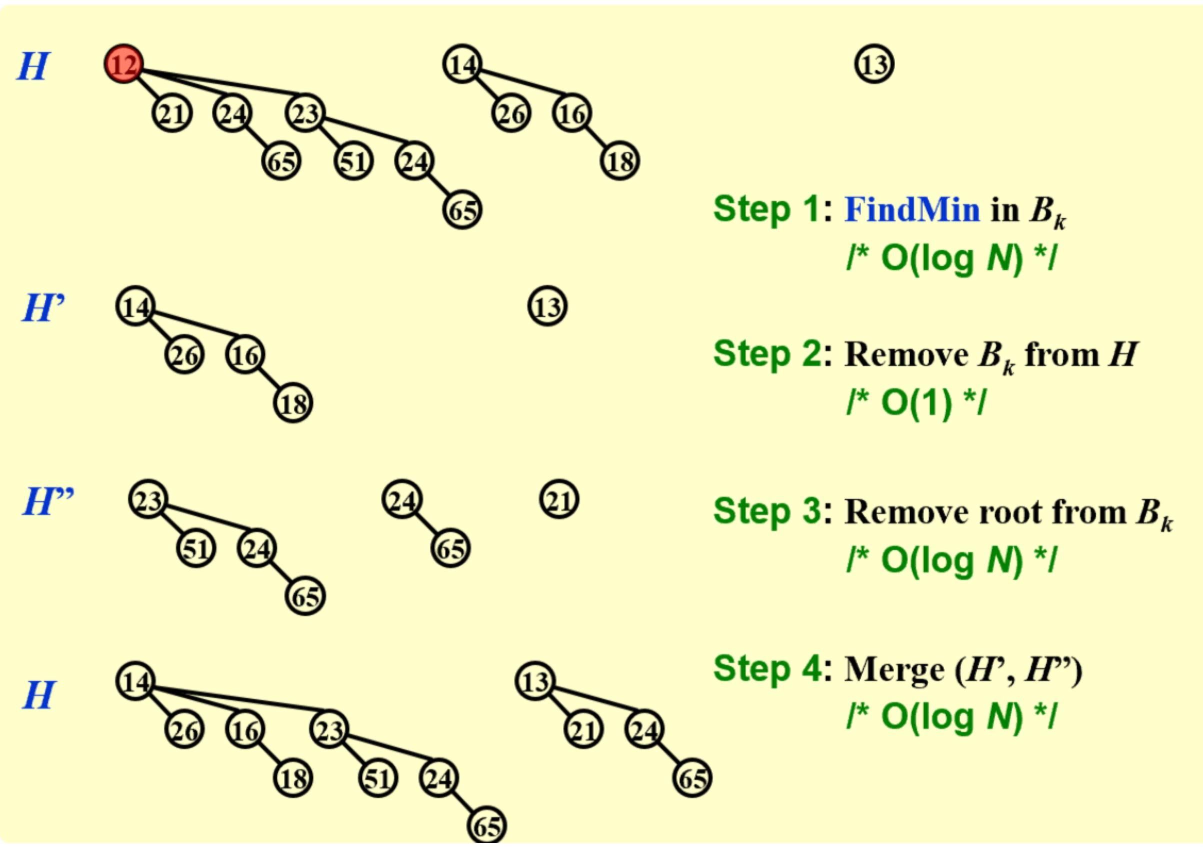
Insert is a special case for merging

【Example】Insert 1, 2, 3, 4, 5, 6, 7 into an initially empty queue.



Performing N inserts on a initially empty binomial queue will take O(N) worst–case time. **Hence the average time is constant.**

DeleteMIn(H)



Implementation

Binomial queue = array of binomial trees

Operation	Property	Solution
DeleteMin	Find all the subtrees quickly	
Merge	The children are ordered by their sizes	

Decreasing.

Child – Sibling

Structure:

Declaration C 复制代码

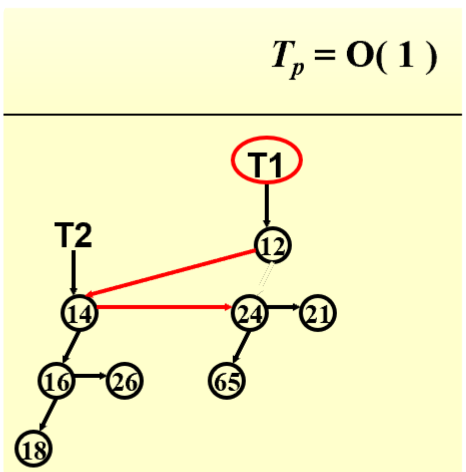
```
typedef struct BInNode *Position
typedef struct Collection *BinQueue
typedef struct BinNode *BinTree

struct BinNode
{
    ElementType element;
    Position LeftChild;
    Position NextSibling;
}

struct Collection
{
    int CurrentSize;
    BinTree TheTrees[MaxTrees];
}
```

Conbine C 复制代码

```c
BinTree ConbineTrees(BinTree T1,BinTree T2)
{
    //merge equal-sized T1 and T2
    if(T1->Element > T2->Element)
        //attach the larger one to the smaller one
        return ConbineTrees(T2,T1)
    //insert T2 to the front of the children list of T1
    T2->NextSibling = T1->LeftChild;
    T1->LeftChild = T2;
    return T1;
}
```



C 复制代码

```
BinQueue Merge(BinQueue H1, BInQueue H2)
{
    BinTree T1,T2,Carry = NULL;
    int i,j;
    if(H1->CurrentSize + H2->CurrentSize > Capacity)
        ErrorMessage();

    H1->CurrectSize += H2->CurrentSize;
    for(i=0,j=1;j<=H1->CurrentSize;i++,j+=2)
    {
        //current trees
        T1 = H1->TheTrees[i];
        T2 = H2->TheTrees[i];
        switch(4*!!carry + 2*!!T2+!!T1)
        {
            case 0: /*000*/
            case 1: /*001*/ break;
            case 2: /*010*/
                H1->TheTree[i] = T2;
                H2->TheTree[i] = NULL;
                break;
            case 4: /*100*/
                H1->TheTree[i] = Carry;
                Carry = NULL;
                break;
            case 3: /*011*/
                Carry = BombineTree(T1,T2);
                H1->TheTree[i] = H2->TheTree[i] = NULL;
                break;
            case 5: /*101*/
                Carry = BombineTree(T1,Carry);
                H1->TheTree[i] = NULL;
                break;
            case 6: /*110*/
                Carry = BombineTree(T2,Carry);
                H2->TheTree[i] = NULL;
                break;
            case 7: /*111*/
                H1->TheTrees[i] = Carry;
                Carry = BombineTree(T1,T2);
                H2->TheTree[i] = NULL;
                break;
        }
    }
    return H1;
```

```
}
```

DeleteMin

复制代码

```
ElementType DeleteMin(BinQueue H)
{
    BinQueue DeleteQueue;
    Position DeleteTree,OldRoot;
    Elementtype MinItem = Infinity;
    int i,j,MinTree;

    if(IsEmpty(H)) {PrintErrorMessage();return -Infinity;}

    for(i=0;i<MaxTrees;i++) // Find the minimum item
    {
        if(H->TheTrees[i] && H->TheTrees[i] ->Element < MinItem)
        {
            MinItem = H->TheTrees[i] ->Element;
            MinTree = i;
        }
    }

    DeleteTree = H->TheTrees[MinTree];
    H->TheTree[MinTree] = NULL; //remove the MinTree
    OldRoot = DeleteTree; // remove the root;
    DeletedTree = DeletedThee->LeftChild;
    Free(OldRoot);
    DeleteQueue = Initialize();
    DeleteQueue->CurrentSize = (1<<MinTree) - 1; //2^MinTree - 1
    for(j = MInTree - 1;j>=0;j--)
    {
        DeleteQueue->TheTrees[j] = DeletedTree;
        DeletedTree = DeletedTree->Nextsibling;
        DeletedQueue->TheTrees[j]->NextSibling = NULL;
    }
    H->CurrentSize -= DeleteQueue->CurrentSize + 1;
    H = Merge(H,DeleteQueue);
    return MinItem;
}
```

【Chaim】

A binomial queue of N elements can be built by N successive insertions in O(N) time.

2-1 分数 1 作者 陈越 单位 浙江大学

Which of the following binomial trees can represent a binomial queue of size 42?

○ A. $B_0\ B_1\ B_2\ B_3\ B_4\ B_5$

◉ B. $B_1\ B_3\ B_5$

○ C. $B_1\ B_5$

○ D. $B_2\ B_4$

答案正确：1 分 创建提问

2-2 分数 1 作者 陈越 单位 浙江大学

For a binomial queue, __ takes a constant time on average.

○ A. merging

○ B. find-max

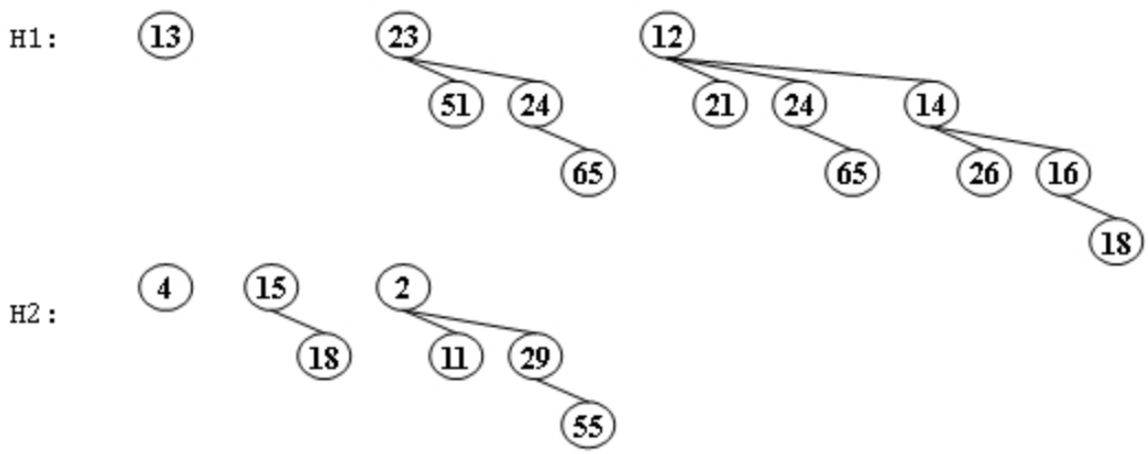
○ C. delete-min

◉ D. insertion

答案正确：1 分 创建提问

2-3 分数 2 作者 陈越 单位 浙江大学

Merge the two binomial queues in the following figure. Which one of the following statements must be FALSE?

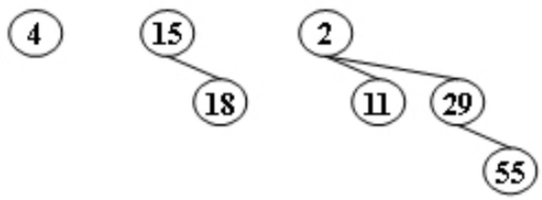


- ○ A. there are two binomial trees after merging, which are B_2 and B_4
- ○ B. 13 and 15 are the children of 4
- ○ C. if 23 is a child of 2, then 12 must be another child of 2
- ◉ D. if 4 is a child of 2, then 23 must be another child of 2

答案正确：2 分 创建提问

2-4 分数 2 作者 陈越 单位 浙江大学

Delete the minimum number from the given binomial queues in the following figure. Which one of the following statements must be FALSE?



- ○ A. there are two binomial trees after deletion, which are B_1 and B_2
- ○ B. 11 and 15 can be the children of 4
- ◉ C. 29 can never be the root of any resulting binomial tree
- ○ D. if 29 is a child of 4, then 15 must be the root of B_1

答案正确：2 分 创建提问