

자료구조 실습08

Data Structures Lab08

Lab08 예제(1/2)

◎ 목표:

- ☞ Array를 이용한 Heap (Heap with array)
- ☞ 재귀(Recursive) 함수, 상속(Inheritance), Heap (Heap)
- ☞ Heap을 이용한 우선순위 큐(Priority Queue) 이해

◎ 내용:

- ☞ 과제
 - Array기반의 Min/Max Heap 구현

◎ 방법:

- ☞ Heap의 메커니즘을 분석하고 Max/Min Heap의 ADT를 바탕으로 구현
- ☞ 재귀함수를 이해하고 이를 바탕으로 각 기능을 구현

Lab08 예제(2/2)

◎ 내용

- ☞ 재귀 함수를 사용하는 Heap을 작성
- ☞ ItemType은 list에서 사용하던 ItemType을 사용
 - ID를 기준으로 정렬하는 Heap을 구성

◎ 고려사항

- ☞ Min/Max Heap을 특정 자료형과 무관(Generic)하게 정의
 - ItemType에서는 비교 연산자 (>, >=, <, <=, ==, != 등)와 출력연산자 재정의(overloading)를 사용
- ☞ Heap 출력은 Max Heap 출력
 - Application에 MaxHeap을 구현
- ☞ 삭제할 때 해당 노드의 레벨을 고려하여 Heap의 형태를 유지할 수 있도록 구현
- ☞ Root 노드가 삭제되는 priority queue 의 기능을 구현

예제: HeapBase Class ADT(1/2)

```
template <typename T>
class HeapBase
{
    public:
        HeapBase();                // default constructor
        virtual ~HeapBase();       // destructor

        // check the heap state whether empty or full.
        bool IsEmpty();             // check heap is empty
        bool IsFull();             // check heap is full

        int GetLength() const;      // get number of current node
        void MakeEmpty();           // make empty
        virtual int Add(T item);    // add new node
        virtual int Delete(T item); // delete node
        virtual T Pop();            // pop a root node
}
```

예제: HeapBase Class ADT(2/2)

```
virtual void RetrieveItem(T &item, bool &found);    // retrieve item
virtual void PrintHeap();                          // print nodes in Heap
```

// pure virtual functions for recursive process.

```
virtual void ReheapDown(int iparent, int ibottom) = 0;
virtual void ReheapUp(int iroot, int ibottom) = 0;
virtual void Delete(T item, bool &found, int iparent) = 0;
virtual void Retrieve(T &item, bool &found, int iparent) = 0;
```

protected:

```
T *m_pHeap;           // element array.
int m_iLastNode;      // number of nodes in heap.
int m_nMaxSize;       // maximum array size.
```

```
};
```

예제: MaxHeap Class ADT

```
template <typename T>
class MaxHeap : public HeapBase<T>
{
    public:
        MaxHeap();                // default constructor
        MaxHeap(int size);        // constructor

        // recursive functions.
        virtual void ReheapDown(int iparent, int ibottom);           // reheap down
        virtual void ReheapUp(int iroot, int ibottom);              // reheap up
        virtual void Delete(T item, bool &found, int iparent);       // delete node
        virtual void Retrieve(T &item, bool& found, int iparent);    // search node
};
```

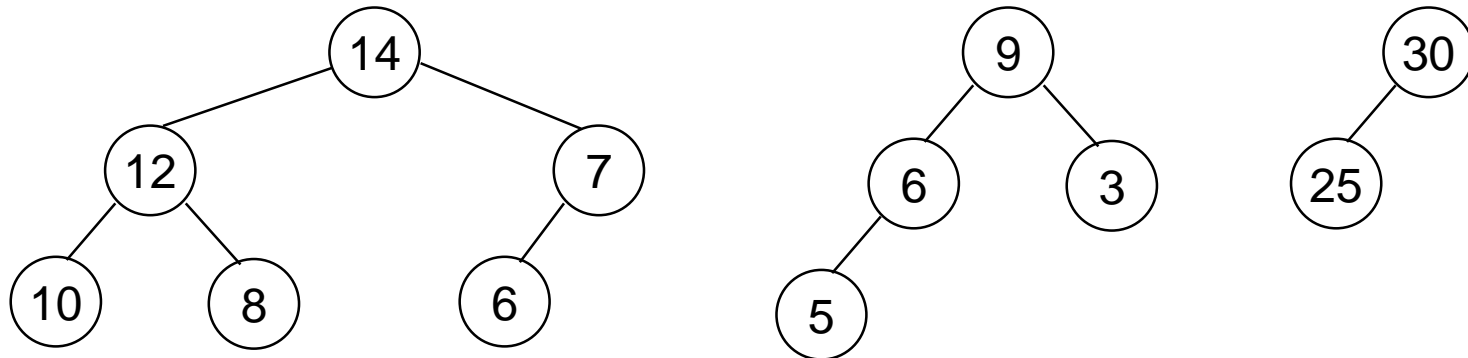
예제: MinHeap Class ADT

```
template <typename T>
class MinHeap : public HeapBase<T>
{
    public:
        MinHeap();                // default constructor
        MinHeap(int size);        // constructor

        // recursive functions.
        virtual void ReheapDown(int iparent, int ibottom);           // reheap down
        virtual void ReheapUp(int iroot, int ibottom);              // reheap up
        virtual void Delete(T item, bool &found, int iparent);       // delete node
        virtual void Retrieve(T &item, bool& found, int iparent);    // search node
};
```

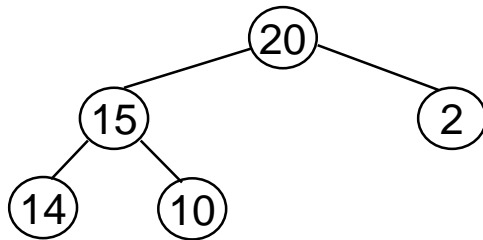
Lab08: Reference – MaxHeap(1/2)

- ◎ Max tree는 각 노드의 키 값이 그 자식의 키 값보다 작지 않은 tree
- ◎ Max heap은 max tree 이면서 complete binary tree

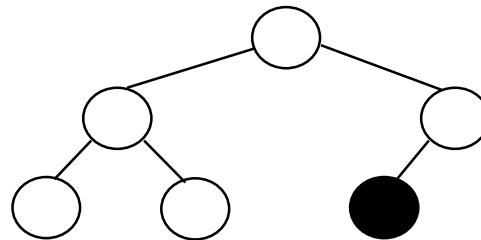


Lab08: Reference – MaxHeap(2/2)

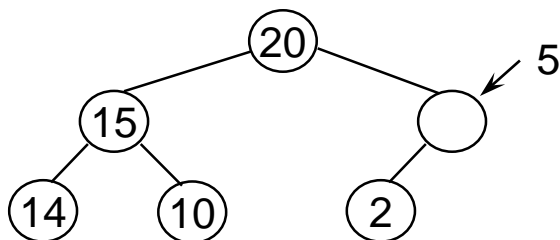
☞ 5, 21을 삽입할 때의 예



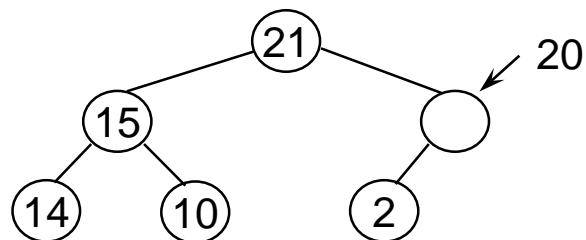
(a) 삽입 전 Heap
Before insert



(b) 새 노드의 초기 위치
Position of new node



(c) Heap (a)에 5를 삽입
insert 5-node into (a) heap



(d) Heap (a)에 21을 삽입
insert 21-node into (a) heap