Data Structures

Lab # 11

Data & Knowledge Engineering Laboratory
Kyung Hee University, Korea

Chapter 9 Programming Exercise

Lab 11

- 1. Exercise 3
- 2. Exercise 4
- 3. Exercise 5
- 4. Exercise 8

실습 준비

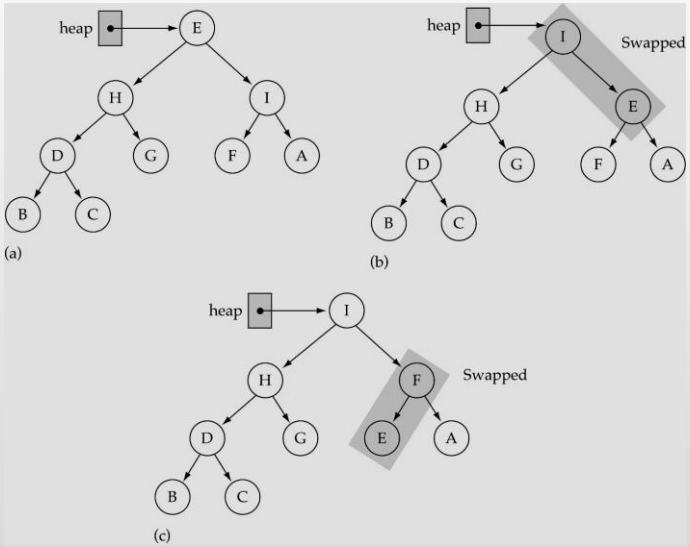
- **샘플 코드 중에서 Chapter9에 있는 소스를 사용함**
  - ❖ …₩labplus_CRLF₩labplus₩Lab, C++ 3rd₩Chapter9₩

- **사용 코드**
  - ❖ PQ : PQType.h, Heap.h를 사용 (1,2,3,4), SortedList(3), StackType(4)

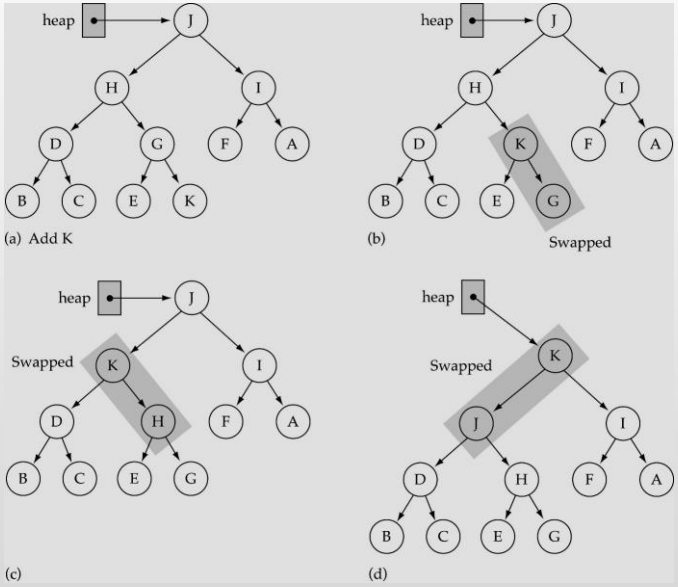
1. Exercise 3

■ 문 제

- ❖ A. ReheapDown을 nonrecursive 방식으로 작성하여라.
- ❖ B. ReheapUp을 nonrecursive 방식으로 작성하여라.
- ❖ C. 위 두 프로그램을 Big-O 개념으로 설명하여라.



ReheapDown



ReheaUp

1-help slides (1/2)

```
template<class ItemType>
void HeapType<ItemType>::ReheapUp(int root, int bottom)
// bottom은 새로 삽입된 노드로 제일 아래 레벨의 제일 오른쪽 노드를 가리킴
{
      int parent;
      bool reheaped = false; // bottom이 제 위치를 찾아가서 reheap이 되면 true

      while(bottom > root && !reheaped)
      {
            parent = ____________________; // bottom 값으로부터 부모 노드의 위치 계산

            if(elements[parent] < elements[bottom])
            {
                  Swap(elements[parent] , elements[bottom]);// bottom을 부모 노드와 값 교환
                  ________________;  // bottom이 이제 부모 노드의 위치를 가리킴
            } else
                  reheaped = true;
}
```

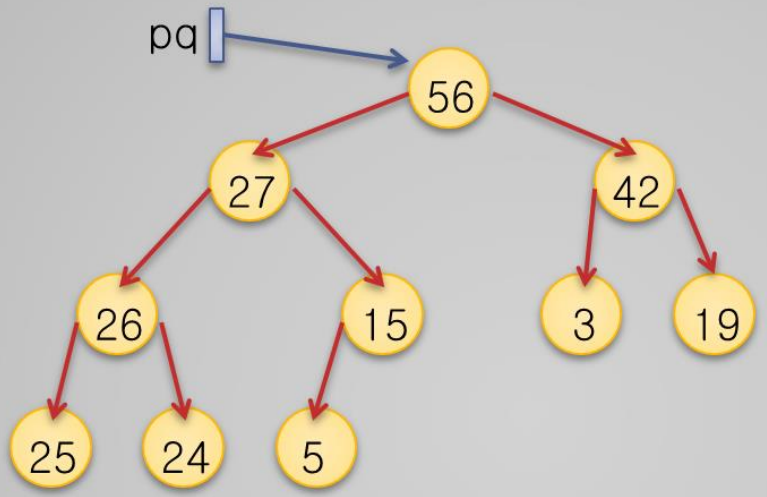
1-help slides (2/2)

```
template<class ItemType>
void HeapType<ItemType>::ReheapDown(int root, int bottom)
// root 노드를 제외하고 나머지 노드들은 heap의 조건을 만족하고 있음
{
    int maxChild, leftChild, rightChild;
    bool reheaped = false; // root가 제 위치를 찾아가 reheap이 되면 True

    leftChild = _____; // root 값으로부터 왼쪽 자식노드의 위치 계산

    while(leftChild <= bottom && !reheaped)
    {
        if(leftChild == bottom) // 왼쪽 자식 노드 하나만 있는 경우
            maxChild = _____;
        else{
            rightChild = _____;
            maxChild = (elements[leftChild] <= elements[rightChild]) ? _____ : _____;
        }

        if(elements[root] < elements[maxChild]) {
            Swap(elements[root], elements[maxChild]);
            root = _____; // maxChild 가 root의 새로운 위치가 됨
            leftChild = _____;
        } else
            reheaped = true;
    }
}
```

2. Exercise 4

■ 문 제

◈ 우선순위 큐가 다음 그림과 같이 힙으로 구현되었다.



a. 위의 힙에 다음과 같은 연산을 수행시킨 결과를 보여라.

```
pq.Enqueue(28);
pq.Enqueue(2);
pq.Enqueue(40);
pq,Dequeue(x);
pq.Dequeue(y);
pq.Dequeue(z);
```

b. (a)번 문제의 연산을 수행하고 난 이후에는 x, y, z가 어떤 값이 되는가?

2-help slide

- **결과를 예측해보고 (힙의 상태) 테스트 드라이버를 통하여 예측한 결과와 실제 결과를 비교해 봅시다.**

문제에 주어진 PQ세팅

```
#include "PQType.h"
#include <iostream>
using namespace std;

int main()
{
        PQType<int> pqueue(50);

        pqueue.Enqueue(56);
        pqueue.Enqueue(27);
        pqueue.Enqueue(42);
        pqueue.Enqueue(26);
        pqueue.Enqueue(15);
        pqueue.Enqueue(3);
        pqueue.Enqueue(19);
        pqueue.Enqueue(25);
        pqueue.Enqueue(24);
        pqueue.Enqueue(5);


        return 0;
}
```

a,b 문제에 수행될 코드

```
int x,y,z;
pqueue.Enqueue(28);
pqueue.Enqueue(2);
pqueue.Enqueue(40);

pqueue.Dequeue(x);
pqueue.Dequeue(z);
pqueue.Dequeue(y);

cout << x << " " << y << " " << z << endl;
```

오른쪽 코드를 삽입하세요.

3. Exercise 5

■ 문 제

- ❖ 우선순위 큐는 연결 리스트로 구현할 수 있다. 단, 저장되는 순서는 내림차순 (가장 큰 원소로부터 작은 원소)으로 저장된다.
  - A. PQType의 정의는 어떻게 변경되어야 하는가?
  - B. 연결 리스트로 구현하는 Enqueue연산을 작성하여라.
  - C. 연결 리스트로 구현하는 Dequeue 연산을 작성하여라.
  - D. 위의 두 Enqueue와 Dequeue 연산을 Heap의 연산과 비교하라 (Big-O개념)

3-help slides

- **기존의 Sorted List는 오름차순으로 구현되어 있다.**

- **그러나 우선순위 큐 같은 경우 값을 가져올 때 내림차순으로 가져온다.**

  ➔ 정렬리스트의 오름차순을 내림차순으로 변경해야한다.

- **A. PQType은 어떻게 변경되어야 하는가?**

```
template<class ItemType>
class PQLLType
{
public:
        PQLLType();
        ~PQLLType();
        void MakeEmpty();
        bool IsEmpty() const;
        bool IsFull() const;
        void Enqueue(ItemType newItem);
        void Dequeue(ItemType& item);
        void Print();
private:
        int length;
        SortedType<ItemType> linkedlist;
        //int maxItems;
};
```

수정된 PQ에서는 Sorted Linked List 객체를 하나 가지고 있습니다.
(PQ 객체가 소멸될 때, likedlist 객체는 자동으로 소멸자가 호출되어 노드들의 메모리를 반환하게 되므로 PQ의 소멸자에서 likedlist 객체의 노드들을 반환하지 않아야 함)

maxItems는 linked list에서 큰 의미가 없으므로 삭제하였습니다. 이에 따라 생성자, IsFull() 함수에서 maxItems 대신 linkedlist 객체의 메소드를 사용하여 개수를 알아내는 것으로 수정해야 합니다.

3-help slides

B. Enqueue 연산을 정의하세요.

```
//리스트에 아이템을 추가합니다.
template<class ItemType>
void PQLLType<ItemType>::Enqueue(ItemType newItem)
{
        if(linkedlist.IsFull())
                throw FullPQLL();
        else
        {
                length++;
                __________; // 아이템 추가
        }
}
```

※리스트 구현부 중 InsertItem()함수의 부등호 방향을 바꾸어 오름차순을 내림차순으로 변경하세요.

C. Dequeue 연산을 정의하세요.

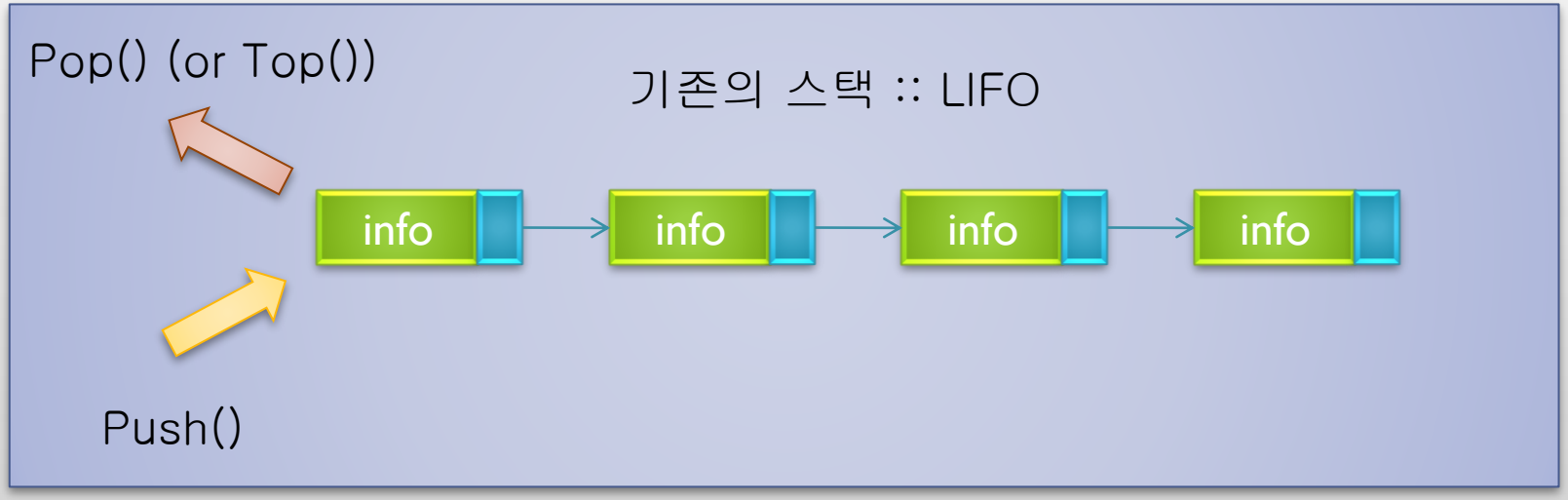
```
//리스트의 가장 앞에 있는 아이템을 가져오고,
//해당 아이템을 삭제합니다.
template<class ItemType>
void PQLLType<ItemType>::Dequeue(ItemType& item)
{
        if (length == 0)
                throw EmptyPQLL();
        else  {
                _________________; // iterator를 사용할 준비
                _________________; //리스트에서 아이템을 얻고
                _________________; //해당 아이템 삭제
        length--;
        }
}
```

※SortedList에서는 리스트의 제일 앞에 있는 아이템을 읽어 오기 위한 메소드를 제공하지 않습니다. 따라서. Iterator를 사용하여 가져와야 합니다.

4. Exercise 8

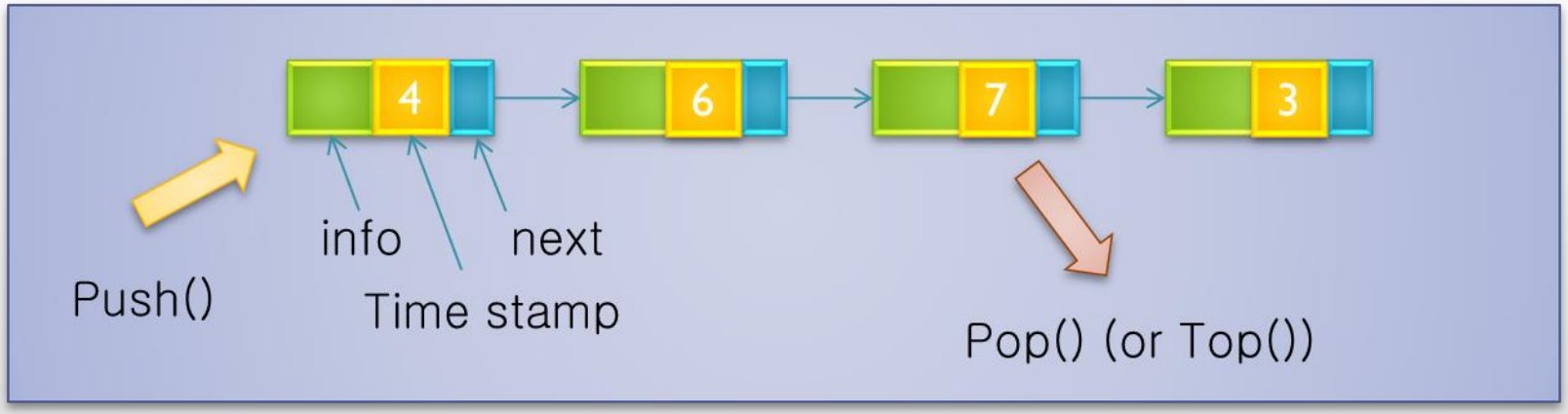
문 제

- 스택은 우선순위 큐로 구현된다. 각 원소가 스택에 쌓일 때 타임 스탬프(time stamp)가 찍힌다 (타임 스탬프는 0과 INT_MAX사이의 숫자이다. 한 원소가 스택에 쌓일 때마다 하나씩 증가된 값이 할당된다.)
- A. 최우선순위 원소는 무엇인가?
- B. 4장에 있는 명세서를 이용하여 Push 와 Pop 알고리즘을 작성하여라.
- C. 4장에서 구현된 Push,Pop 연산과 위의 두 연산을 Big-O 개념으로 비교하여라.



- **Time-Stamp를 기준으로 우선순위를 결정함**
  - Time-Stamp가 높을수록 높은 우선순위를 가짐

- **우선순위 큐를 통한 스택 :: 우선순위 순으로 빠져나간다.**



- **이와 같은 구조에서 Push와 Pop 함수에서 변경해야 할 부분에 대해서 생각해 보세요.**