[제 11 주 실습]

Deap으로 구현된 양끝 우선순위 큐

강 지 훈
jhkang@cnu.ac.kr

충남대학교 컴퓨터공학과

[문제 11]
Deap으로 구현된 양끝우선순위큐

문제 개요

- Deap으로 구현된 양끝 우선순위 큐를 구현한다.

Double-Ended Priority Queue

- It supports the following operations:
  1. Insert an element with an arbitrary key.
  2. Delete an element with the largest key.
  3. Delete an element with the smallest key.

Deaps

- Double-ended heap
  - supports the double ended priority queue operations.
    - insert
    - delete min
    - delete max

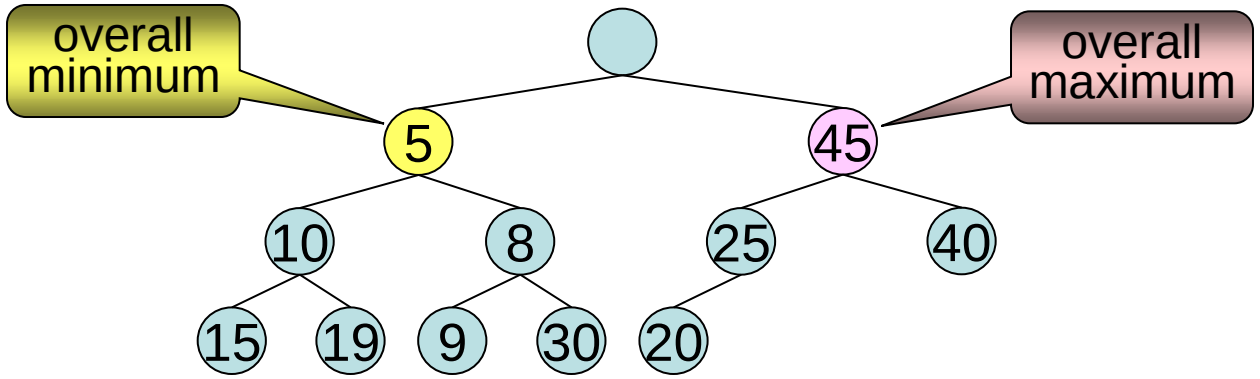
- $O(\log n)$ for each operation
  - n is the size of a deap.

- A deap is a complete binary tree that is either empty or satisfies the following properties:
  1. The root contains no element.
  2. The left subtree is a min-heap.
  3. The right subtree is a max-heap.
  4. If the right subtree is not empty, then let i be any node in the left subtree.

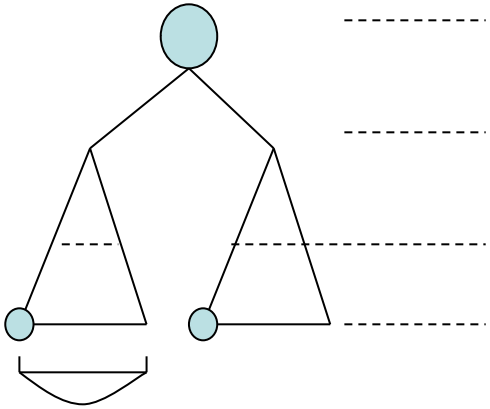
     Let j be the corresponding node in the right subtree.

     If such a j does not exist, then let j be the node in the right subtree that corresponds to the parent of i.

     The key in node i is less than or equal to the key in j.



How to compute the value of j:



2^{k-2} nodes

Level	leftmost node #	# of nodes at the level
1	2^0	2^0
2	2^1	2^1
k-1	2^{k-2}	2^{k-2}
k	2^{k-1}	2^{k-1}

$$j = i + 2^{k-2}$$

$$2^{k-1} \le i < 2^k$$

$$k - 1 \le \log_2 i < k$$

$$k - 1 = \lfloor \log_2 i \rfloor$$

$$k = \lfloor \log_2 i \rfloor + 1$$

$$j = i + 2^{(\lfloor \log_2 i \rfloor + 1) - 2}$$

$$= i + 2^{\lfloor \log_2 i \rfloor - 1}$$

Consequently,

$$j = i + 2^{\lfloor \log_2 i \rfloor - 1};$$

$$\text{if } (j > n)\ j /= 2;$$

출력화면

```
<<프로그램을 시작합니다>>
초기 DEPQ를 생성합니다.
[Debug] insert : On a Max side
[Debug] insert : On a MIN side
[Debug] insert : On a MIN side
[Debug] insert : On a Max side
[Debug] insert : On a Max side
[Debug] insert : On a MIN side
[Debug] PrintMin --
5
10 8
15
[Debug] PrintMax --
45
25 40

삽입할 값을 입력하세요 : 50
[Debug] insert : On a MIN side
[Debug] PrintMin --
5
10 8
15 25
[Debug] PrintMax --
50
45 40

삽입할 값을 입력하세요 : 3
[Debug] insert : On a MIN side
[Debug] PrintMin --
3
10 5
15 25 8
[Debug] PrintMax --
50
45 40

삽입할 값을 입력하세요 : 30
[Debug] insert : On a MIN side
[Debug] PrintMin --
3
10 5
15 25 8 30
[Debug] PrintMax --
50
45 40

삽입할 값을 입력하세요 : 20
[Debug] insert : On a Max side
[Debug] PrintMin --
3
10 5
15 25 8 30
[Debug] PrintMax --
50
45 40
20
삽입할 값을 입력하세요 : 2
[Debug] insert : On a Max side
[Debug] PrintMin --
2
3 5
15 10 8 30
[Debug] PrintMax --
50
45 40
20 25
삽입할 값을 입력하세요 : -1


DEPQ를 삭제합니다.

[Debug] Delete Min : 2
[Debug] PrintMin --
3
10 5
15 25 8 30
[Debug] PrintMax --
50
45 40
20
[Debug] Delete Min : 3
[Debug] PrintMin --
5
10 8
15 25 20 30
[Debug] PrintMax --
50
45 40

[Debug] Delete Min : 5
[Debug] PrintMin --
8
10 20
15 25 30
[Debug] PrintMax --
50
45 40

[Debug] Delete Min : 8
[Debug] PrintMin --
10
15 20
30 25
[Debug] PrintMax --
50
45 40

[Debug] Delete Min : 10
[Debug] PrintMin --
15
25 20
30
[Debug] PrintMax --
50
45 40

[Debug] Delete Min : 15
[Debug] PrintMin --
20
25 30
[Debug] PrintMax --
50
45 40
[Debug] Delete Min : 20
[Debug] PrintMin --
25
40 30
[Debug] PrintMax --
50
45

[Debug] Delete Min : 25
[Debug] PrintMin --
30
40 45
[Debug] PrintMax --
50

[Debug] Delete Min : 30
[Debug] PrintMin --
40
45
[Debug] PrintMax --
50

[Debug] Delete Min : 40
[Debug] PrintMin --
45
[Debug] PrintMax --
50
[Debug] Delete Min : 45
[Debug] PrintMin --
50
[Debug] PrintMax --

[Debug] Delete Min : 50
[Debug] PrintMin --
[Debug] PrintMax --

<<프로그램을 종료합니다>>
```

이 과제에서 필요한 객체는?

- AppView
- AppController
  - DoubleEndedPriorityQ
- DoubleEndedPriorityQ

AppControllor의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public void run()

AppView의 공개 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public AppView()
  - public void showInputSelectMenu()
  - public void showResult()
  - public int inputInt()

DoubleEndedPriorityQ의 멤버 함수는?

- 사용자에게 필요한 함수 (Public Functions)
  - public DoubleEndedPriorityQ ()
  - public DoubleEndedPriorityQ (int givenMaxSize)

  - public boolean isEmpty ()
  - public boolean isFull ()
  - public int size ()

  - public boolean add (int anElement)
  - public int min ()
  - public int max ()

Class “AppControllor“

AppControllor– 비공개 인스턴스 변수

```
public class AppControllor {
    private AppView _appView;
    private DoubleEndedPriorityQ _deap;
```

AppControllor 의 공개 함수 run()의 구현

```
public void run() {
    _deap = new DoubleEndedPriorityQ();
    Random random = new Random();
    _appView = new AppView();

    _appView.outputMsg(_appView.MSG_StartProgram);

    _appView.outputMsg(_appView.MSG_StartInitDepq);
    for (int i = 0; i < 9; i++)
    {
        int temp = random.nextInt(100);
        _deap.add(temp);
    }

    this.showPrintMinMax();

    int aKey = 0;
    while (aKey != -1) {
        _appView.outputMsg(_appView.MSG_RequestInput);
        aKey = _appView.inputInt();
        _deap.add(aKey);

        this.showPrintMinMax();
    }

    System.out.println(_appView.MSG_StartDeleteDepq);
    int count = _deap.size();
    for (int i = 0; i < count; i++) {
        int delMin = _deap.min();
        if (delMin == -1)
            _appView.outputMsg(_appView.MSG_Error_Delete);
        else
            _appView.outputMsg("[Debug] Delete Min : " + delMin + "\n");
        this.showPrintMinMax();
    }
    _appView.outputMsg(_appView.MSG_EndProgram);
}
```

AppControllor의 private Method

- AppControllor의 private Member function의 사용법과 구현
  - private void showPrintMinMax()
    - 디버깅용의 Min과 Max Tree를 출력하는 함수
    - DoubleEndedPriorityQ의 ShowMinHeap()과 ShowMaxHeap()를 차례로 호출

Class “AppView“

AppView – 비공개 인스턴스 변수

```
import java.util.Scanner;

public class AppView {
    private Scanner _scanner;
```

AppView의 Public Method

- AppView 의 Public Member의 선언
  - public String MSG_StartProgram = "<<프로그램을 시작합니다>>₩n";
  - public String MSG_EndProgram = "<<프로그램을 종료합니다>>₩n";
  - public String MSG_StartInitDepq = "초기 DEPQ를 생성합니다.₩n";
  - public String MSG_StartInputDepq = "DEPQ에 값을 삽입합니다₩n";
  - public String MSG_RequestInput = "삽입할 값을 입력하세요 : ";
  - public String MSG_PrintDepq = "DEPQ를 출력합니다.₩n";
  - public String MSG_StartDeleteDepq = "₩n₩nDEPQ를 삭제합니다.₩n";
  - public String MSG_Error_Delete = "Error: Qepq가 비어 있습니다.₩n";

AppView의 Public Method

- AppView 의 Public Member function의 사용법과 구현
  - public AppView()
    - 생성자
  - public void outputMsg(String aString)
    - aString을 출력
  - public String inputString()
    - String을 하나 입력 받아 반환
  - public int inputInt()
    - Int를 하나 입력 받아 반환

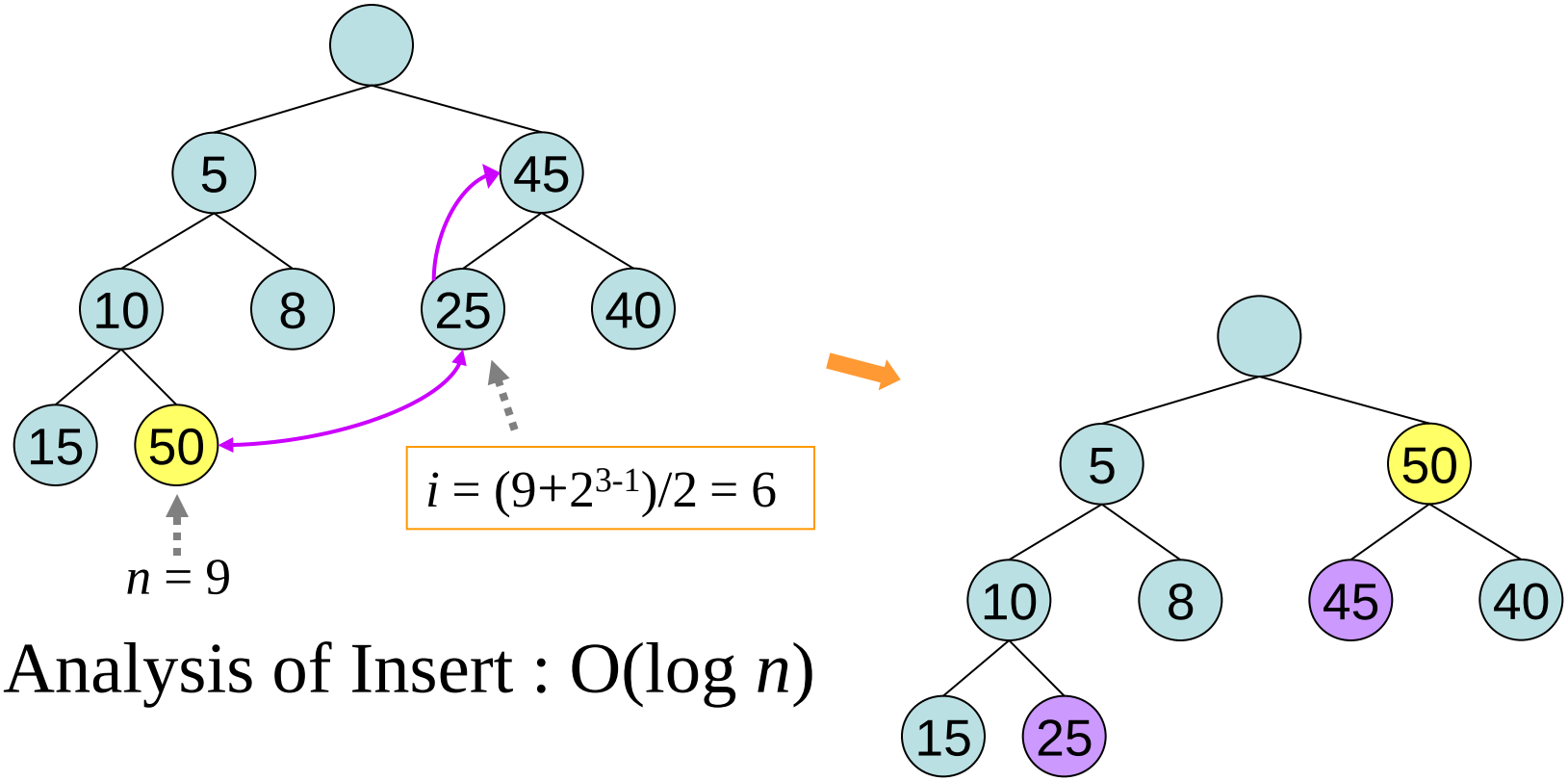
Class “DoubleEndedPriorityQ”

비공개 인스턴스 변수

```
public class DoubleEndedPriorityQ {
    private static final int MAX_SIZE = 100;
    private int[] _heap;
    private int _size;
    private int _maxSize;
```

Insertion to a Deap [1]

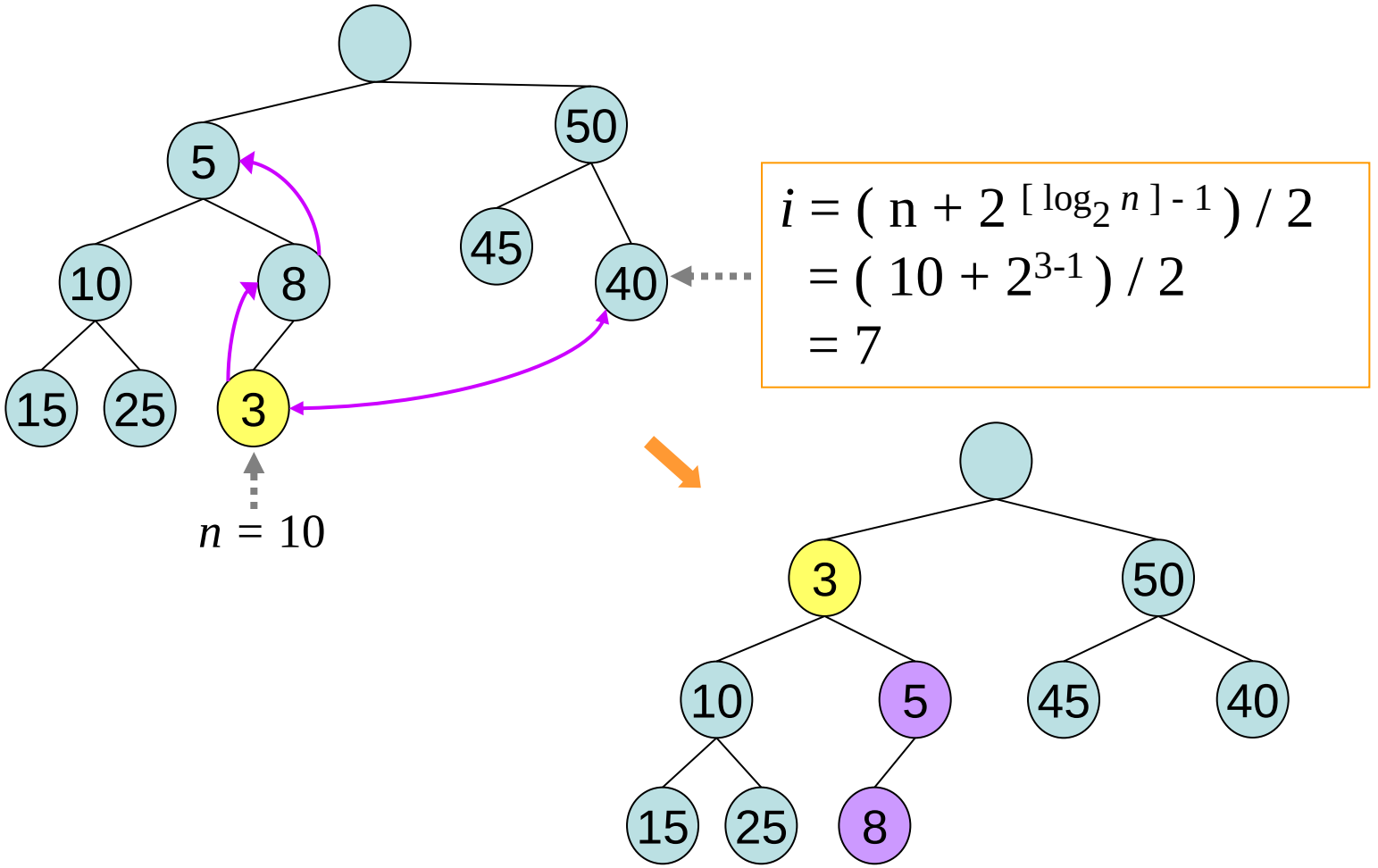
- On a MIN side:



- Analysis of Insert : O(log n)

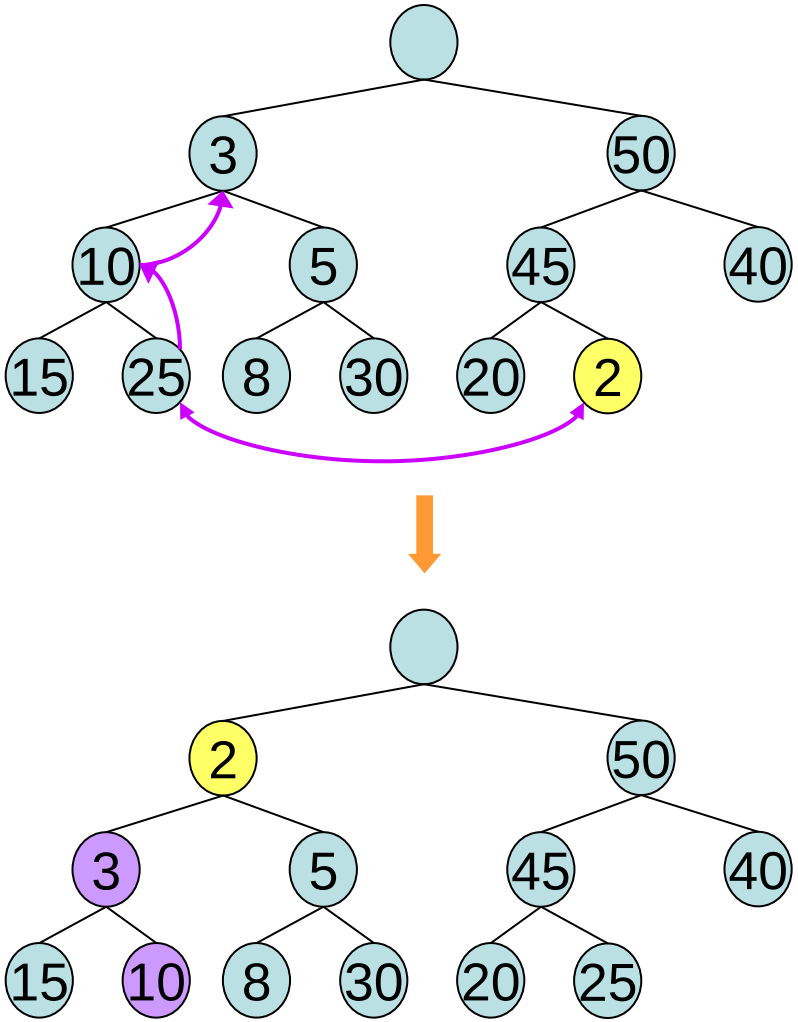
Insertion to a Deap [2]

- On a MIN side:



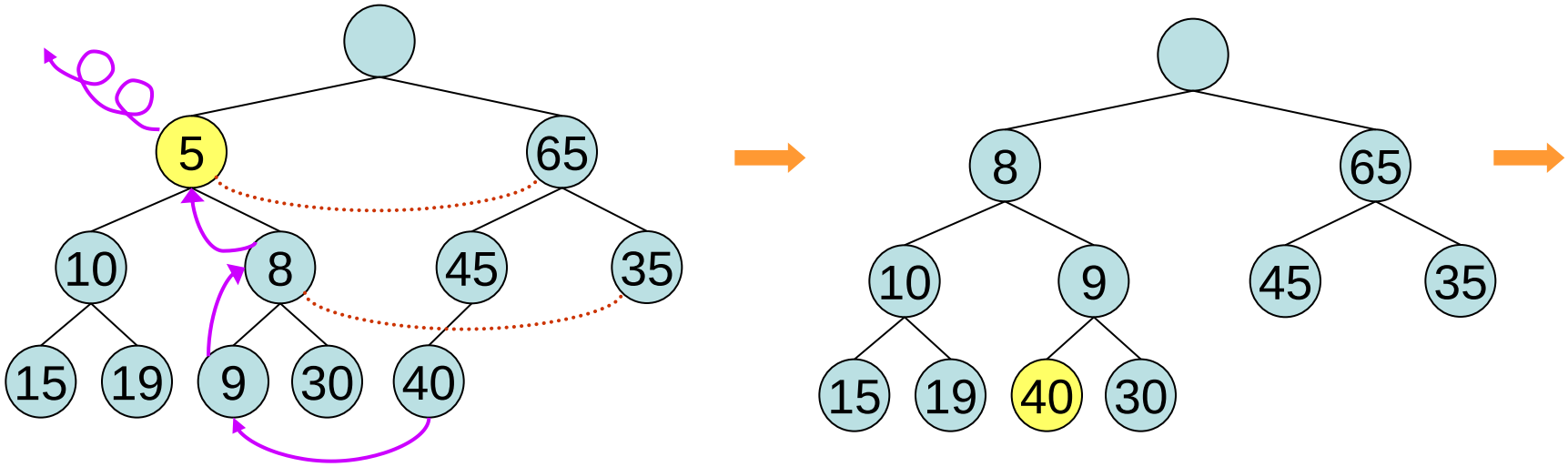
Insertion to a Deap [3]

- On a MAX side:



Deletion of Min/Max [1]

Deletion of Min



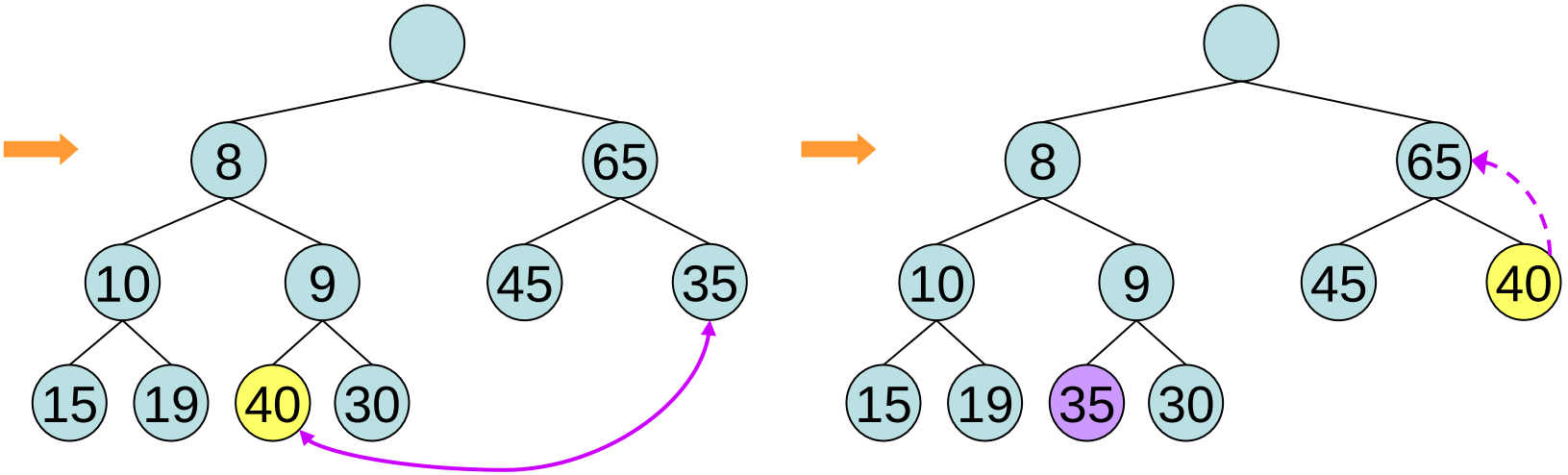
- Node 8 can be moved up since 8 ≤ 35 ≤ 65.
- Node 9 can be moved up since 9 ≤ 35.

At this time, we should insert 40 on the MIN side.

Deletion of Min/Max [2]

Deletion of Min (Cont'd)

- Modified Insertion



- Analysis of Delete : O(log *n*)

Delete of Max

- It is performed in a similar manner

Public Method

- DoubleEndedPriorityQ의 Public Member function의 구현
  - public DoubleEndedPriorityQ ()
  - public DoubleEndedPriorityQ (int givenMaxSize)
    - 생성자
    - 생성시 size를 무엇으로 선언 하여야 할까?
  - public boolean isEmpty ()
    - _heap이 비어 있는 경우 true를 return
    - 생성자 size를 초기화 한 것을 감안하여 작성
  - public boolean isFull ()
    - _heap이 가득차 있는 경우 true를 return
    - 생성자 size를 초기화 한 것을 감안하여 작성
  - public int size ()
    - 현재 삽입 되어 있는 size를 retrun
    - 생성자 size를 초기화 한 것을 감안하여 작성

Public Method

- DoubleEndedPriorityQ의 Public Member function의 구현
  - public boolean add (int anElement)
    - Heap이 가득 차 있는지 확인
    - Heap이 비어 있는지 확인
    - _size를 증가
    - 현재 heap의 삽입 위치가 min heap인지 max heap인지 확인 (calCurrentHeapLocation)
    - 최소 heap에 있는 경우 Max heap에 있는 partner를 확인 (findMaxPartner)
      - 삽입 되는 anElement가 maxheap에 있는 partner보다 크면
        - _heap의 마지막에 partner값을 삽입
        - Max에 partner의 위치와 anElement를 보내 삽입(addMax)
      - Partner보다 작으면
        - min에 마지막 위치와 anElement를 보내 삽입(addMin)
    - 최대 heap에 있는 경우
      - 최소 heap에 있는 경우를 참고하여 작성

Public Method

- DoubleEndedPriorityQ의 Public Member function의 구현
  - public int min ()
    - Heap이 비어 있는지 확인
    - 현재 min heap의 가장 작은 node를 변수 min에 저장
    - 총 heap의 가장 마지막에 있는 값을 _heap의 1번째로 이동
    - _size를 한 개 감소
    - 최 상위에 있는 min/max값과 _heap의 1번째 값 그리고 자식을 비교해가며 min heap을 재구성 (슬라이드 참고)
      - 힌트 : 1번째에 저장된 값과 자식의 값을 비교하여야 함
    - 마지막으로 찾아진 min heap의 작은 숫자가 있던 위치에 마지막에 위치한 값을 저장
    - 마지막에 위치한 값을 저장한 위치와 partner를 비교하여 deap의 조건에 맞으면 교환
    - min을 반환

Public Method

- DoubleEndedPriorityQ의 Public Member function의 구현
  - public int max ()
    - Min의 구현을 감안하여 구현한다

Public Method

- DoubleEndedPriorityQ의 Public Member function의 구현
  - public void ShowMaxHeap()
  - public void ShowMinHeap()
    - Min/ max heap을 모두 출력

```
public void ShowMinHeap() {
    System.out.print("[Debug] PrintMin --");
    for (int i = 2; i <= _size; i++) {
        if ((int) calBaseLog(i, 2) != (int) calBaseLog(i - 1, 2))
            System.out.println();
        if (!calCurrentHeapLocation(i))
            System.out.print(_heap[i] + " ");
    }
    System.out.println();
}
```

Heap 구조를 출력하기 위한 디버깅용으로 사용!

private Method

- DoubleEndedPriorityQ의 Public Member function의 구현
  - ```
    private double calBaseLog(double x, double base) {
        return Math.log(x) / Math.log(base);
    }
    ```
  - private boolean CheckRule()
    - Min heap을 확인하여 파트너의 대소 관계 비교를 하여
    - 만약 max Partner보다 값이 큰 element가 있는 경우 false를 반환
  - private boolean calCurrentHeapLocation(int aSize)
    - aSize를 기준으로 Node가 min에 있는지 max에 있는지 확인
    - Min에 있는 경우 return false
    ```

private Method

- DoubleEndedPriorityQ의 Public Member function의 구현
  - private int findMinPartner(int n)
  - private int findMaxPartner(int n)
    - 주어진 위치의 partner 위치를 확인
    - calBaseLog와 Math.pow를 이용하여 위치를 계산
  - private void addMin(int n, int item)
  - private void addMax(int n, int item)
    - 전달 받은 위치 n과 item을 적절한 위치에 삽입
    - n의 위치부터 parent로 node를 올라가며 확인

[문제 11] 요약

- Double-ended heap
  - Double-ended heap의 이해
    - 알고리즘의 수행 과정을 그림으로 그려 이해한 내용을 바탕으로 보고서를 작성
  - 기존의 Heap과 다른 점은 무엇인가?

- 각 요약에 대한 내용을 보고서에 작성하여 제출하세요.

과제 제출

과제 제출

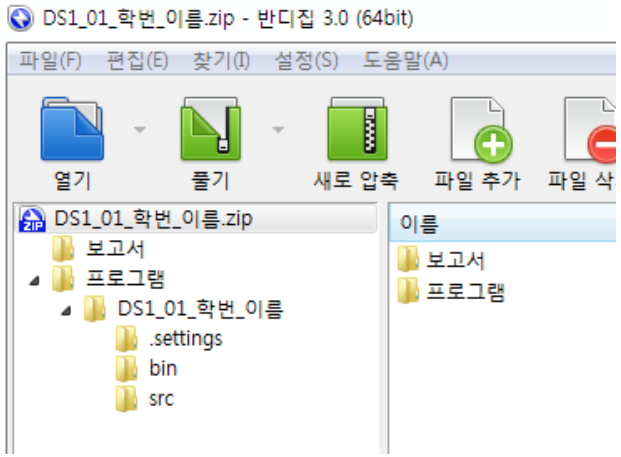
- pineai@cnu.ac.kr
  - 메일 제목 : [0X]DS2_11_학번_이름
    - 양식에 맞지 않는 메일 제목은 미제출로 간주됨
    - 앞의 0X는 분반명 (오전10시 : 00반 / 오후4시 : 01반)

- 제출 기한
  - 11월 19일(화) 23시59분까지
  - 시간 내 제출 엄수

  - 제출을 하지 않을 경우 0점 처리하고, 숙제를 50% 이상 제출하지 않으면 F 학점 처리하며, 2번 이상 제출하지 않으면 A 학점을 받을 수 없다.

과제 제출

- 파일 이름 작명 방법
  - DS2_11_학번_이름.zip
  - 폴더의 구성
    - DS2_11_학번_이름
      - 프로그램
        - 프로젝트 폴더 / 소스
        - 메인 클래스 이름 : DS2_11_학번_이름.java
      - 보고서
        - 이곳에 보고서 문서 파일을 저장한다.
        - 입력과 실행 결과는 화면 image로 문서에 포함시킨다.
        - 문서는 pdf 파일로 만들어 제출한다.



보고서 작성 방법

- 겉장
  - 제목: 자료구조 실습 보고서
  - [제xx주] 숙제명
  - 제출일
  - 학번/이름
- 내용
  1. 프로그램 설명서
     1. 주요 알고리즘 /자료구조 /기타
     2. 함수 설명서
     3. 종합 설명서 : 프로그램 사용방법 등을 기술
  2. 구현 후 느낀 점 : 요약의 내용을 포함하여 작성한다.
  3. 실행 결과 분석
     1. 입력과 출력 (화면 capture : 실습예시와 다른 예제로 할 것)
     2. 결과 분석

     -------- 표지 제외 3장 이내 작성 --------
  4. 소스코드 : 화면 capture가 아닌 소스를 붙여넣을 것

     소스는 장수 제한이 없음.

[제 11 주 실습] 끝