자료구조론 실습
Heap

2017. 05. 12

한양대학교

이주홍

Heap이란?

- 자식보다 자기 자신의 **우선순위가 높은** 완전이진트리

- 우선순위: Max(값이 큰 게 우선)

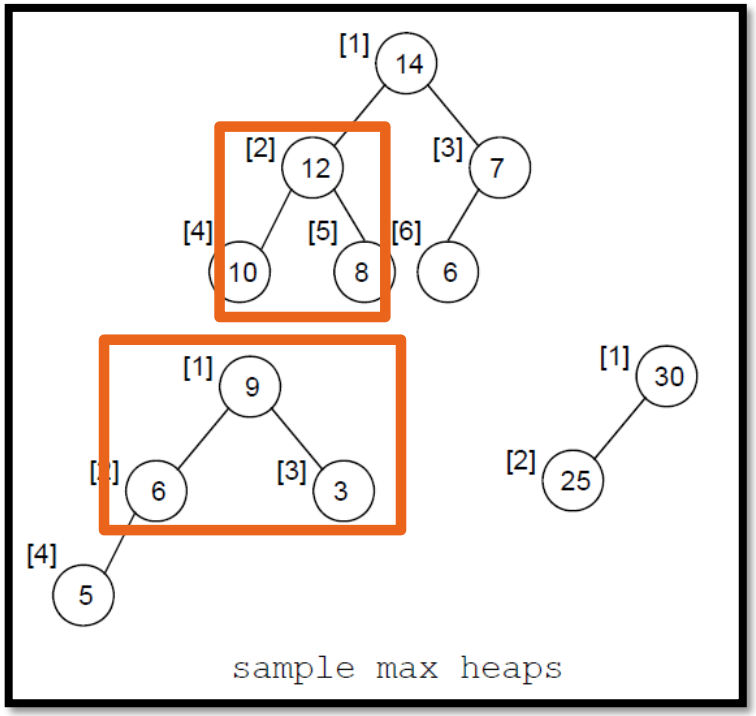
  Min(값이 작은 게 우선)

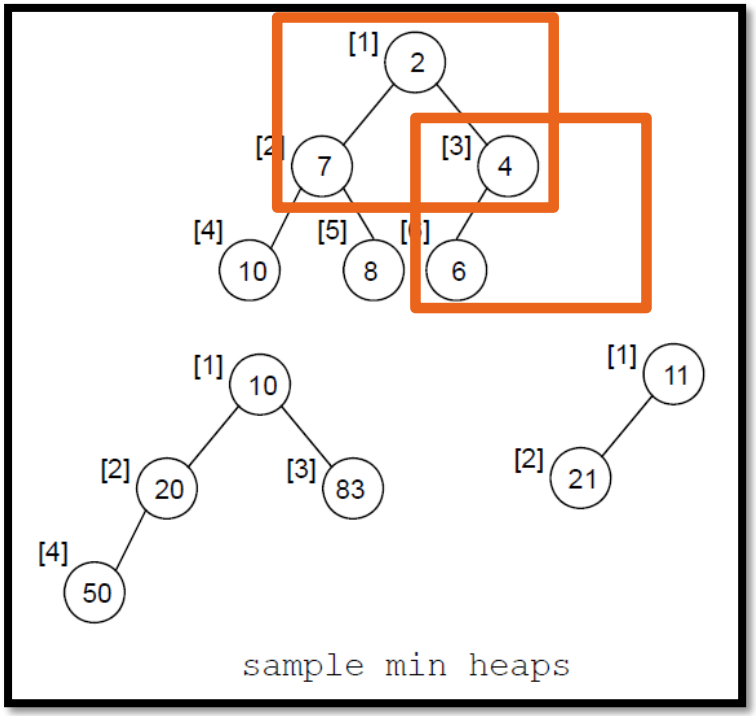
  알파벳 순서 => priority(C) > priority(F)

  ...

Heap이란?

- 자식보다 자기 자신의 **우선순위가 높은** 완전이진트리

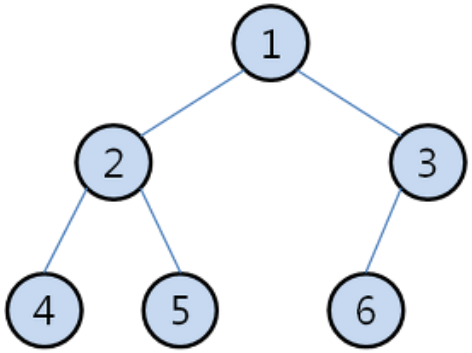




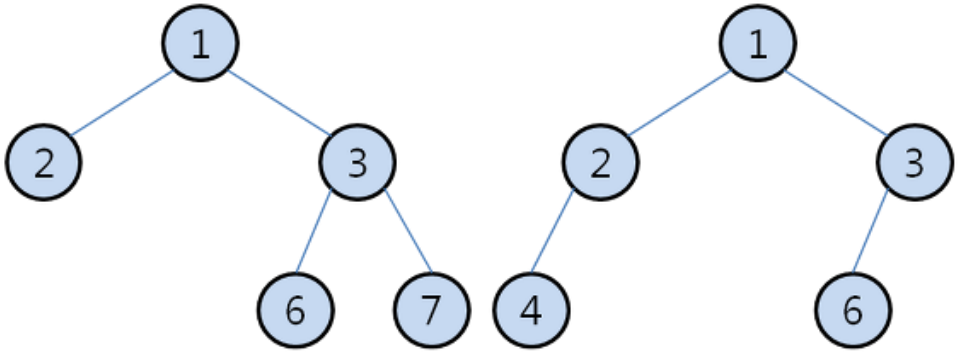
Heap이란?

- 자식보다 자기 자신의 **우선순위가 높은** **완전이진트리**

- 완전이진트리:



완전 이진트리

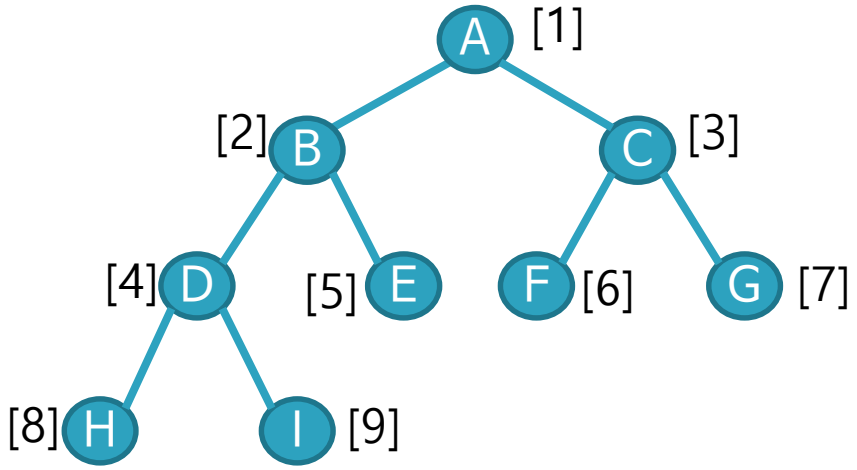


왼쪽부터 채워지지 않거나 빈 공간이 있어 완전 이진트리가 아니다.

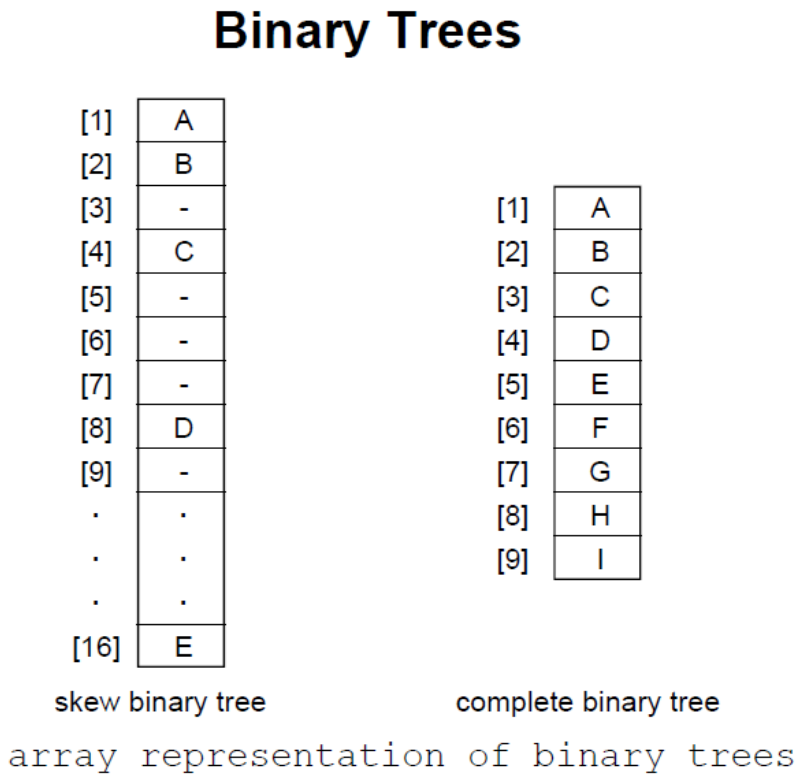
Heap이란?

- 자식보다 자기 자신의 **우선순위가 높은** **완전이진트리**

- 완전이진트리:
- 배열을 통해서 관리!



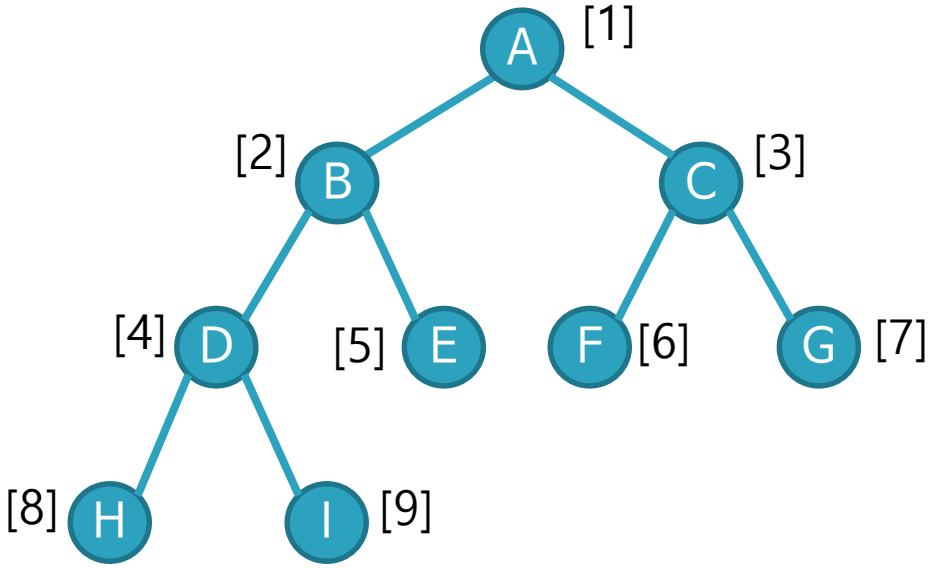
완전이진트리여서 왼쪽, 위부터 순서대로 채워짐.
연속성이 보장되므로 배열이 좋음



배열을 통한 구현의 장점

[1]	A
[2]	B
[3]	C
[4]	D
[5]	E
[6]	F
[7]	G
[8]	H
[9]	I

complete binary tree



- **[i]의 부모는 [i / 2]**
- **[i]의 자식은 [2 * i] and [2 * i + 1]**

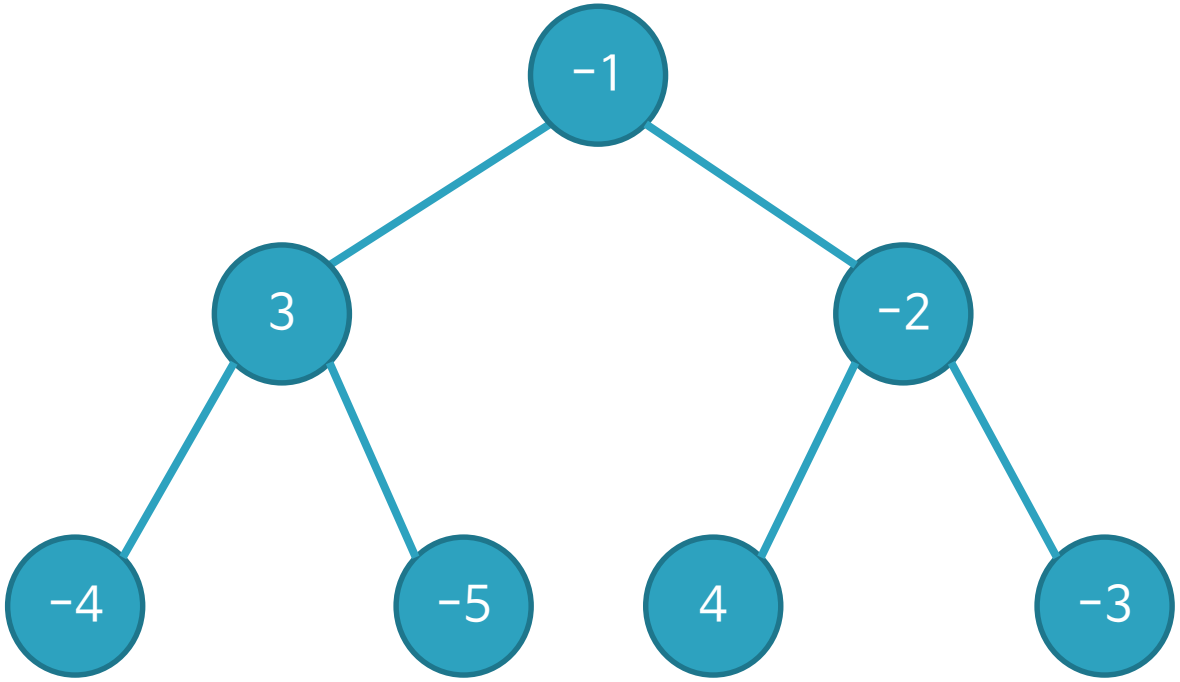
Ex1) E의 부모? => [5 / 2] = [2] = B

Ex2) C의 자식 중 뭐가 클까?

=> if(heap[2 * 3] < heap[2 * 3 + 1])

Assignment 5 - 절대값 Min-Heap

- **절대값의 크기가 작을수록 우선순위가 높은 Heap 구현**
- 즉, 절대값 크기가 작은 원소가 위로
  절대값이 크면 아래에 위치해야 한다.



Max-Heap 구현

- Structure
- Insertion
- Deletion

Max-Heap Structure

```
#define MAX_ELEMENTS 100000
#define HEAP_FULL(n) (n == MAX_ELEMENTS- 1)
#define HEAP_EMPTY(n) (!n)
typedef struct {
    int key;
    /* other field */
} element; // heap의 각 노드 단위는 element라는 구조체
element heap[MAX_ELEMENTS]; // heap이라는 이름의 배열 10만개 생성
int n = 0; // heap의 사이즈
```

* ppt에 오타(MaX_ELEMENTS)

- 힙의 최대 크기(MAX_ELEMENTS)는 넉넉하게 100,000으로
- 원소(노드)는 element라는 구조체이지만, 어차피 int 하나만 사용하기 때문에 element -〉 int 로 바꿔서 과제를 진행해도 무방.

Max-Heap Insertion

```
void insert_max_heap(element item, int *n) {
    int i; // insert될 위치를 찾아줄 변수
    if (HEAP_FULL(*n)) { // Heap이 가득차면 넣을 수 없어
        fprintf(stderr,"The heap is full. \n");
        exit(1);
    }
    i = ++(*n); // 일단 insert하니까 사이즈(n)을 1증가시킴
    while ((i!=1 )&& (item.key > heap[i/2].key)) {
        heap[i] = heap[i/2];
        i /= 2;
    } // while: 부모를 계속 타고 올라가며, 새로 들어온 원소(item)의 위치(i)를 찾는 과정
    heap[i] = item; // 그렇게 해서 찾은 item의 위치(i)에 item을 저장
}
```

Max-Heap Deletion

```
element delete_max_heap(int *n) {
    element item, temp;
    if (HEAP_EMPTY(*n)) {
        fprintf(stderr,"The heap is empty\n");
        exit(1);
    }
    item = heap[1];
    temp = heap[(*n)--];
    parent = 1; child = 2;
    while (child <= *n) {
        /* compare left and right child's key values */
        if ((child < *n) && (heap[child].key <
                             heap[child+1].key))
            child++;
        if (temp.key >= heap[child].key) break;
        /* move to the next lower level */
        heap[parent] = heap[child];
        parent = child;
        child *= 2;
    }
    heap[parent] = temp;
    return item;
}
```

// heap의 루트를 제거하고 해당 값을 리턴

// heap이 비었는지 확인

// 우리 과제와 맞지 않음. 적절한 값을 리턴하도록 수정

// 루트(heap[1])를 삭제하기 전에 미리 item에 빼둠

// 적절한 위치로 옮기기 위해 가장 마지막 값을 빼고 사이즈 1축소

// while: 끝에서 뺀 값을 루트에서부터 자식들을 확인하며 아래로 내리다가 힙 조건에 맞는 적절한 위치를 찾으면 stop

찾은 위치(parent)에 제일 끝에서 뺀 값(temp)를 저장하고, 초기 루트값인 item을 리턴

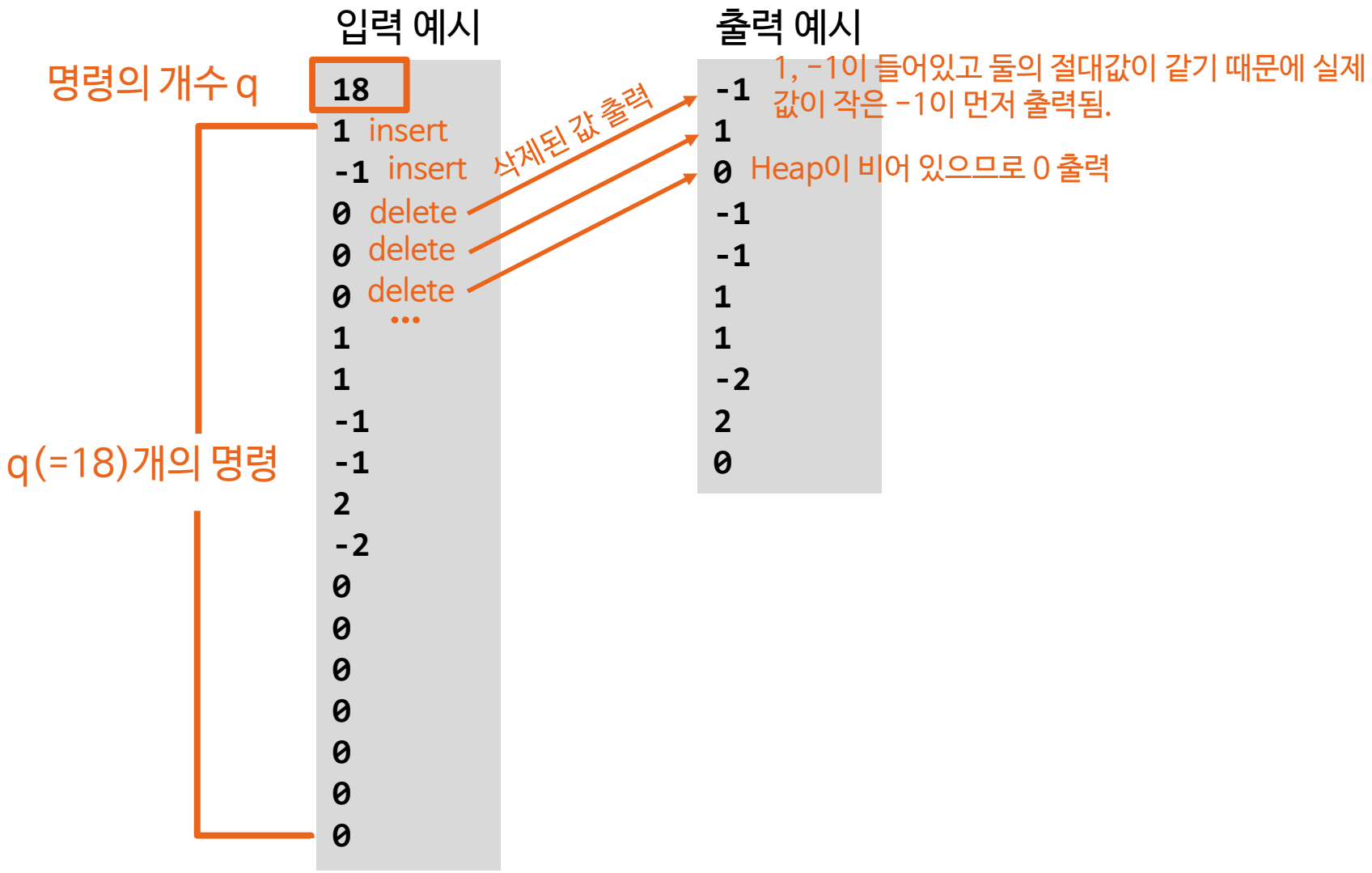
Assignment 5 – 절대값 Min-Heap

- **절대값의 크기가 작을수록 우선순위가 높은 Heap 구현**
- **절대값이 같을 경우 실제 값이 작은 게 우선순위가 높다.**
  priority(-2) > priority(2). 즉 -2가 먼저 delete됨.

- **입출력 설명**
  - 명령의 개수 **q**($\leq 100{,}000$)가 첫째 줄에 입력되고
    둘째 줄부터 q+1째 줄에 걸쳐 q개의 명령이 주어진다.
    각 명령은 (int범위 내의) 하나의 정수로 구성된다.

  - 0이 아닌 정수일 경우, 해당 숫자를 Heap에 Insert한다.
  - 0일 경우, Heap에서 Delete를 하고 해당 값을 출력한다.
    단, Heap이 비어 있을 경우 0을 출력한다.

Assignment 5 - 절대값 Min-Heap



Assignment 5 – 절대값 Min-Heap

- 제출방식: Assignment5 폴더 만들고
  Assignment5_학번.c 파일 저장
- GitLab(https://hconnect.hanyang.ac.kr/)으로 제출

- 제출기한: 5월 18일 23시 59분

감사합니다