Data Structure #7

Heap

2020년 1학기

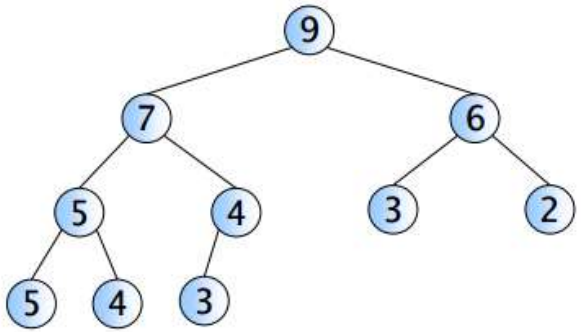
Intro.

- **실습주제**
  - Heap
    - insert
    - delete

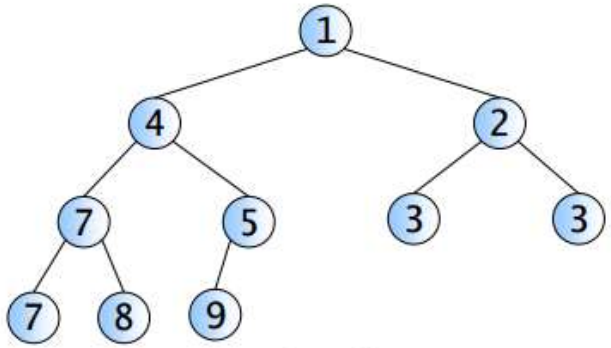
Heap

- **정의**
  - 완전이진트리로서 각 노드에 있는 키들이 다음 조건을 만족해야 한다.
    - max heap : 부모노드.key ≥ 자식노드.key
    - min heap : 부모노드.key ≤ 자식노드.key
- **목적**
  - Heap은 여러 개의 값들 중에서 가장 큰 값이나 가장 작은 값을 빠르게 찾아내도록 만들어진 자료구조임.

↓아래의 그림을 보면 heap tree에서는 중복된 값을 허용함에 유의



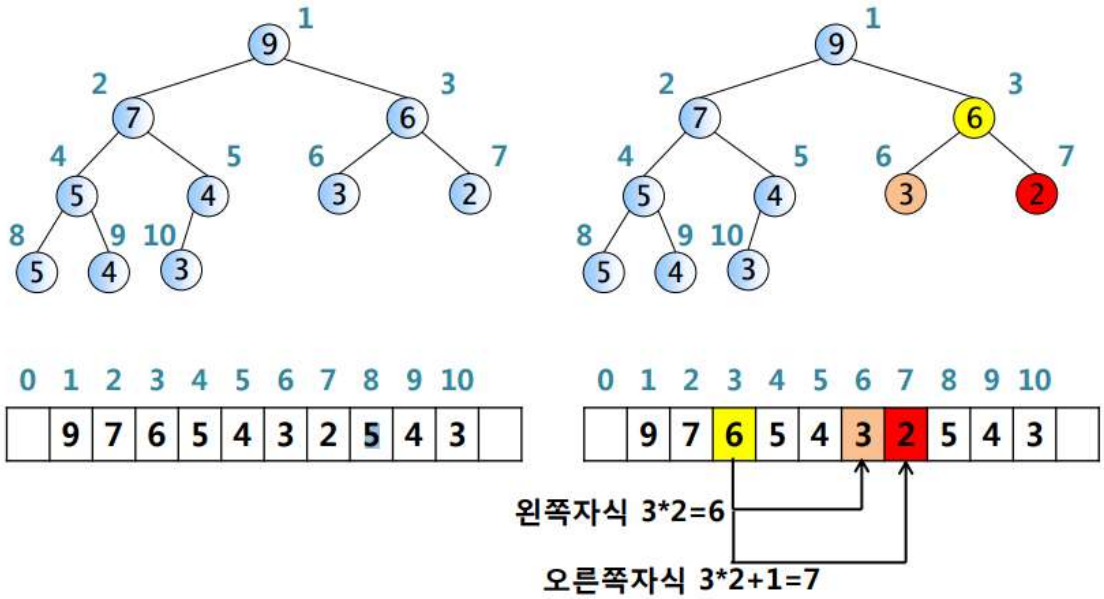
Max heap



Min heap

배열로 구현한 Heap

- **Heap은 배열을 이용하여 효율적으로 구현**
  - 완전이진트리이므로 각 노드에 번호 부여 가능
  - 이 번호를 배열의 인덱스로 사용
  - 빈 공간의 개수를 최소화함
- **부모노드와 자식노드를 찾기가 쉽다**
  - 왼쪽 자식의 인덱스 = (부모의 인덱스) * 2
  - 오른쪽 자식의 인덱스 = (부모의 인덱스) * 2 + 1
  - 부모의 인덱스 = (자식의 인덱스) / 2

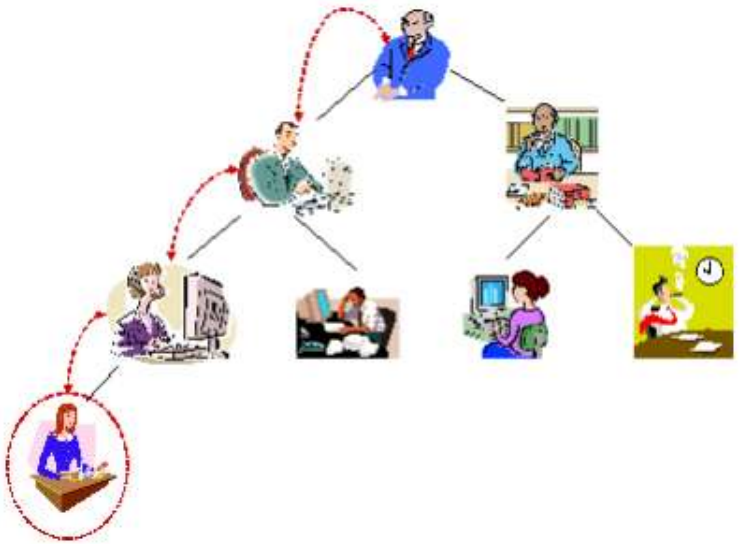


최대힙(max heap) (개인 실습#1)

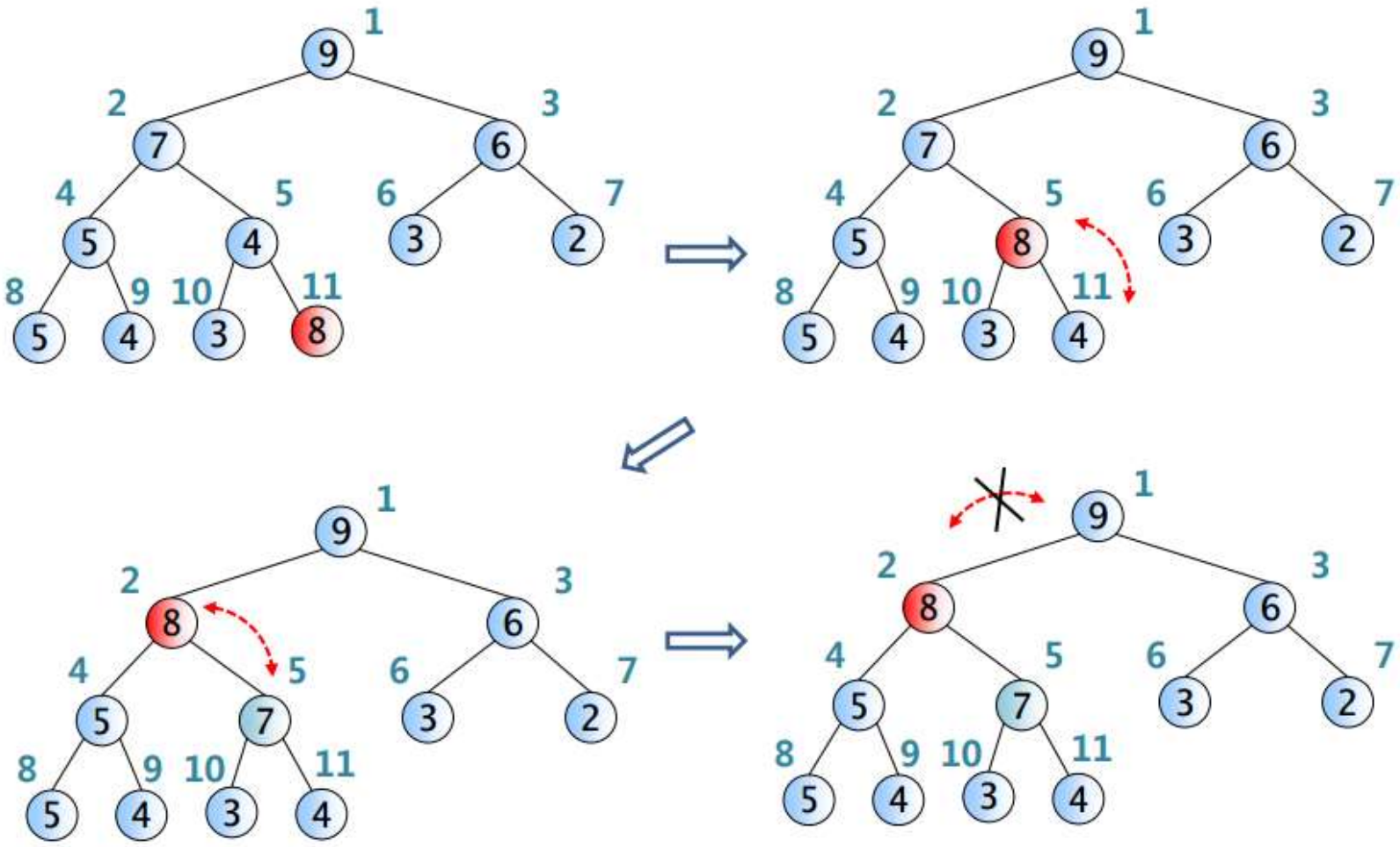
- **Binary max heap 를 구현하기 위한 요소**
  - 삽입(insert): 새로운 키를 heap에 삽입.
  - 삭제(delete): heap에서 루트의 값을 지우고, 최대 heap이 되도록 재구성 하는 연산.
  - 출력: heap의 전체를 출력.

heap에서의 삽입

- **Max heap 자료구조에서 새로운 노드를 삽입하는 것은 신입 사원을 일단 말단 위치에 배정한 뒤에, 상사와의 능력을 비교해서 위로 승진시키는 것과 비슷함**
  - 새로운 요소를 heap의 마지막 노드 다음에 삽입
  - 삽입 후에 새로운 노드를 부모 노드와 비교하여 heap의 속성을 만족할 때까지 교환을 계속함 (bubbling up전략)



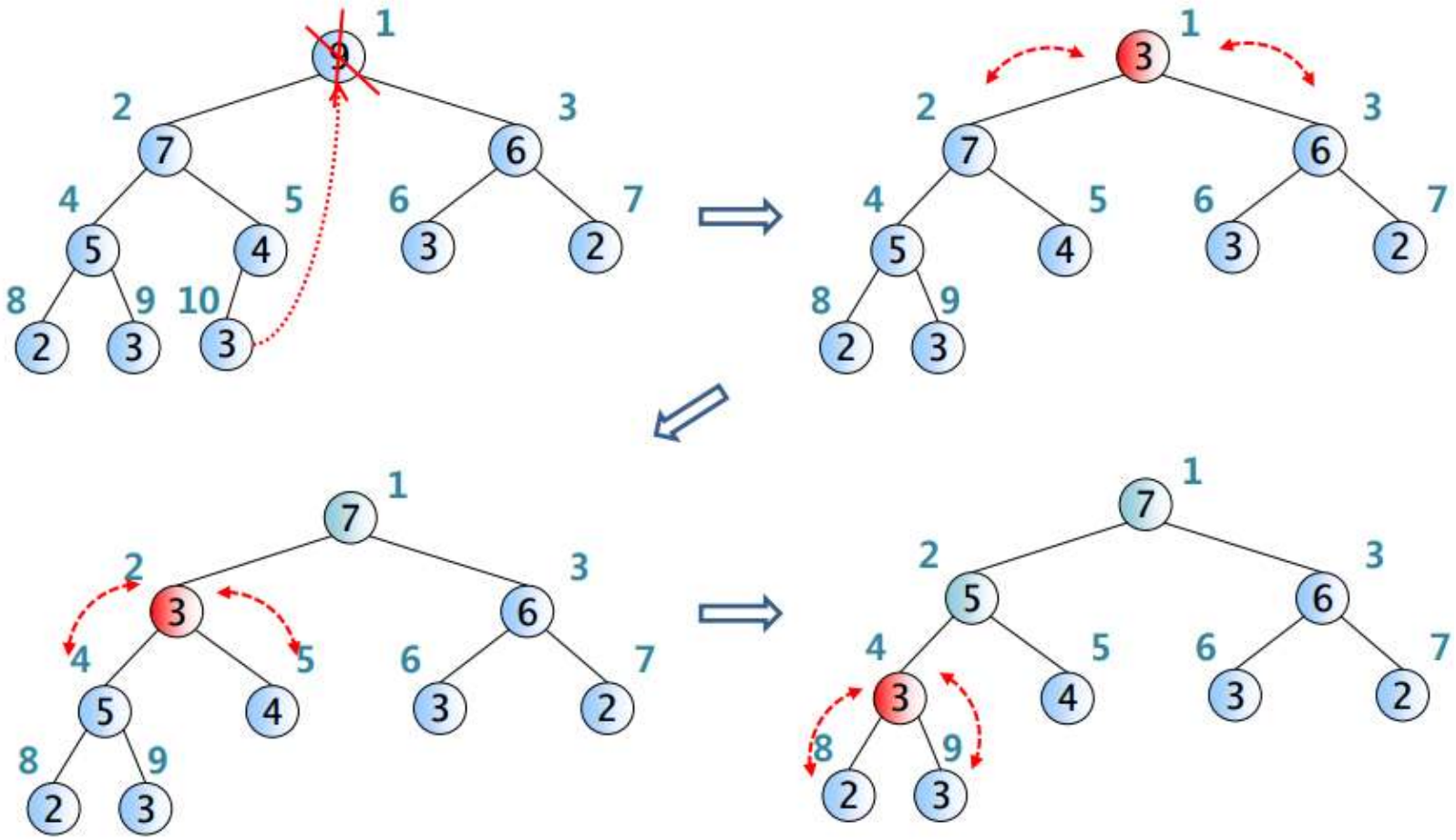
bubbling up전략 예제



heap에서의 삭제

- **Max heap에서의 삭제는 가장 큰 키 값을 가진 노드, 즉 루트노드가 삭제된다.**
  - 루트노드 내용을 삭제
  - 마지막 노드를 루트로 이동
  - Heap 조건이 만족될 때까지 루트부터 단말까지 교환을 계속함 (trickle down 전략)

trickle down 전략 예제



Max heap 구현

- 구조체

```
typedef struct {
        int heap[MAX_ELEMENT];
        int heap_size;
} HeapType;
```

Max heap 구현

- 초기화 함수

```
void init(HeapType *h)
{
        h->heap_size =0;
};
```

- Swap 함수

```
void swap(int *a, int *b){
        int tmp = *a;
        *a = *b;
        *b = tmp;
}
```

Max heap 구현

– 출력 함수

```
void print_heap(HeapType *h)
{
        int i;
        int level=1;
        printf("\n====================");
        for(i=1;i<=h->heap_size;i++){
                if( i == level ) {
                        printf("\n");
                        level *= 2;
                }
                printf("\t%d", h->heap[i]);
        }
        printf("\n====================");
}
```

Max heap 구현

- **삽입 알고리즘(참고용)**
  - insert_max_heap(A, key)

```
1.  heap_size ← heap_size +1;
2.  i ← heap_size;
3.  A[i] ← key;
4.  while i ≠ 1 and A[i] > A[PARENT(i)] do
5.          A[i] ↔ A[PARENT(i)];
6.          i ← PARENT(i);
```

Max heap 구현

- insert 연산

```
void insert_max_heap(HeapType *h, element item)
{.
  // heap의 크기를 하나 증가 시킨다.

  // 새로운 노드에 item을 삽입

  //  트리 상단으로 가면서 부모 노드와 비교

}
```

Max heap 구현

- **삭제 알고리즘 (참고용)**
  - delete_max_heap(A)

```
1.  item ← A[1];
2.  A[1] ← A[heap_size];
3.  heap_size  ← heap_size -1;
4.  i ← 2;
5.    while i <= heap_size do
6.      if i < heap_size and A[i+1] > A[i]
7.              then largest  ← i+1;
8.              else largest  ← i;
9.      if A[PARENT(largest)] > A[largest] then break;
10.     A[PARENT(largest)] ↔ A[largest];

11. return item;
```

Max heap 구현

- delete 연산

```
int delete_max_heap(HeapType *h)
{
   //루트 노드의 값을 미리 저장, 마지막 노드를 루트로 이동시킨다

   // while() start:
      //현재 노드가 가진 자식 중 더 큰 자식 노드를 만나면 child++한다.

      //자식과 부모를 비교하여 부모 노드가 더 작거나 같으면 루프를 나온다.

      //자식을 부모의 위치로 이동시킨다.
      // 부모와 자식을 한 Level 아래로 이동

    //while() end:
    return result;
}
```

Max heap 구현

– 메인함수

```
#include <stdio.h>
#define MAX_ELEMENT 10
void main()
{
HeapType heap;
        init(&heap);

        insert_max_heap(&heap, 10);
        insert_max_heap(&heap, 5);
        insert_max_heap(&heap, 30);
        insert_max_heap(&heap, 15);
        insert_max_heap(&heap, 25);
        print_heap(&heap);

        int d1 = delete_max_heap(&heap);

        printf("\n <del. %d > \n", d1);
        print_heap(&heap);
}
```

제출

- **제출**
  - 개인실습 (#1)
    - 오늘 자정까지 제출 (~ 2020/5/1 23:59)
    - Max Heap
  - 과제
    - Binary Min Heap 구현 (lab7.docx)
    - 다음 주 목요일 자정까지 제출 (~ 2020/5/7 23:59)

과제

- **Lab7.docx**
  - CreateHeap (n x)
    - X size의 새로운 힙 생성 (X는 heap의 최대 크기)
  - Insert (i x)
    - 새로운 키 x를 min heap에 삽입
  - DeleteMin (d)
    - root node에 있는 min key 삭제
  - PrintHeap (p)
    - 전체 min heap 출력 (level)

과제

- **구조체**

```
struct HeapStruct {
        int Capacity;
        int Size;
        ElementType *Elements;
};
```

- **함수**
  - HeapStruct* CreateHeap(int heapsize);
  - void Insert(HeapStruct heap, ElementType value);
  - ElementType DeleteMin(HeapStruct heap);
  - void PrintHeap(HeapStruct heap);

과제

input.txt

```
n 7
d
p
i 5
i 12
i 35
i 1
i 36
p
d
p
i 29
i 50
i 5
i 24
p
```

```
heap is empty
heap is empty

1
5       35
12      36

5
12      35
36
heap is full

5
12      5
36      29      50      35
```