

## 실습8 Heap



# 실습8 힙프 정렬

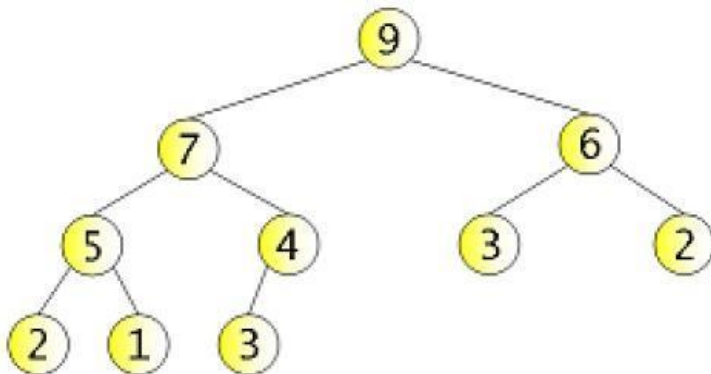
# 실습 목적 및 힙(Heap)의 이해

## ■ 실습 목적

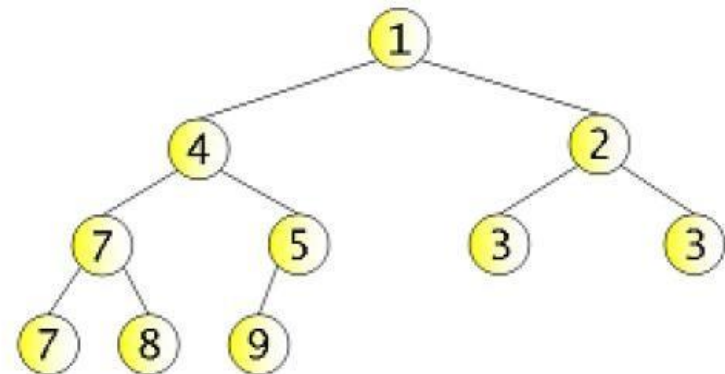
- 힙의 특성을 이해한다.
- 힙의 특성을 활용한 정렬 알고리즘인 힙정렬을 구현해보고, 힙정렬의 효율을 확인해 본다.

## ■ 힙(Heap)

- 여러 개의 값들 중에서 가장 큰 값이나 가장 작은 값을 빠르게 찾으도록 만들어진 자료 구조
- 최대 힙(max heap)
  - 부모 노드의 키 값이 자식 노드의 키 값보다 크거나 같은 완전 이진 트리
- 최소 힙(min heap)
  - 부모 노드의 키 값이 자식 노드의 키 값보다 작거나 같은 완전 이진 트리



최대 힙



최소 힙

# 실습 목적 및 힙프(Heap)의 이해

## ■ 힙프정렬

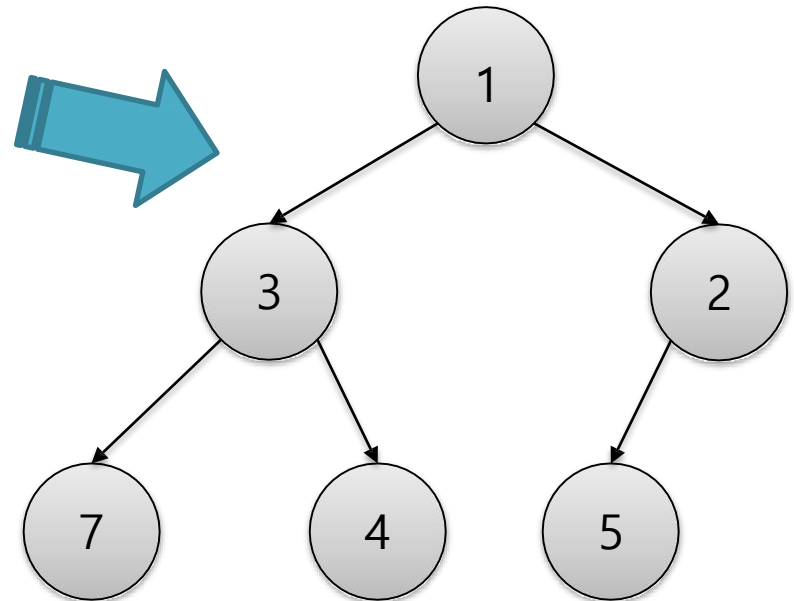
- 힙프를 이용한 정렬 방법
- $O(n\log n)$  시간 소모

## ■ 힙프정렬 방법

- **STEP 1** : 정렬해야 할  $n$ 개의 요소들을 힙프에 삽입
- **STEP 2** : 힙프의 삭제 연산을 수행해가며, 삭제 된 요소를 순차적으로 정리



- 각 요소들을 순차적으로 힙프에 삽입
- 각 요소에 대해  $O(\log n)$ 의 시간이 소모되며,  $n$ 개의 요소에 대해  $O(\log n)$ 의 시간이 소모됨



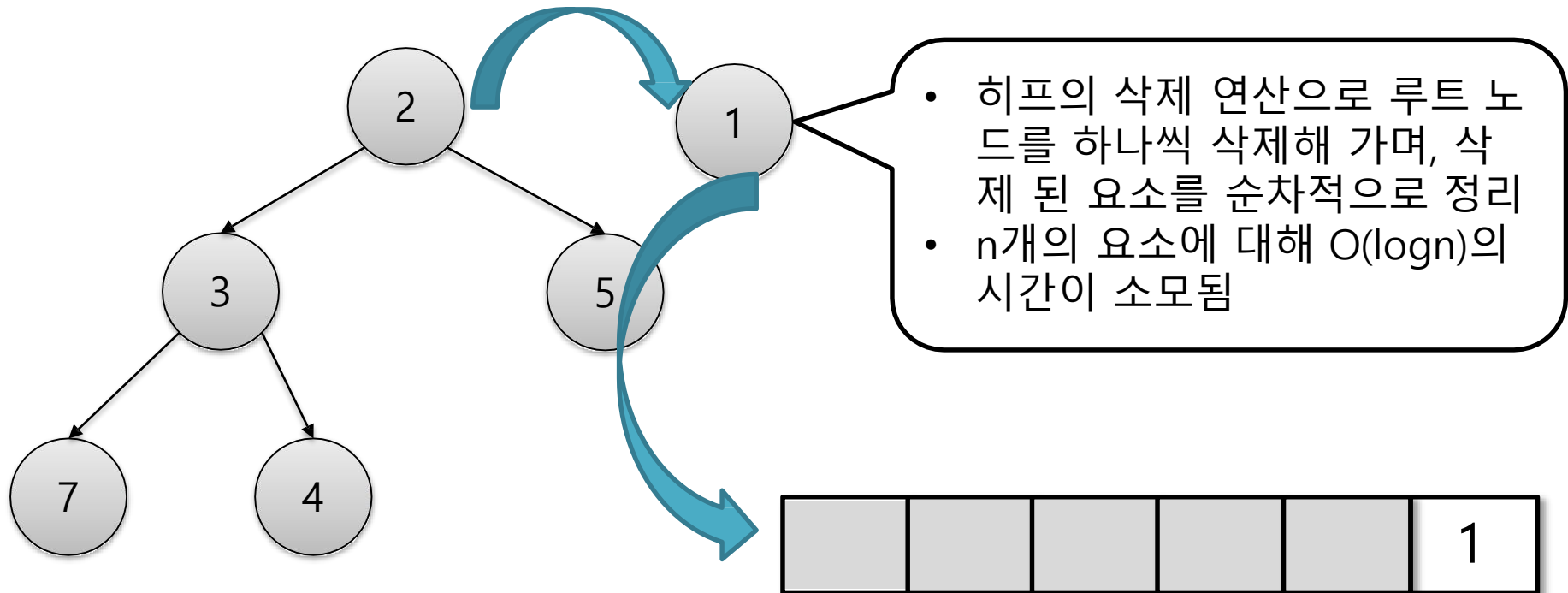
# 실습 목적 및 힙프(Heap)의 이해

## ■ 힙프정렬

- 힙프를 이용한 정렬 방법
- $O(n \log n)$  시간 소모

## ■ 힙프정렬 방법

- **STEP 1** : 정렬해야 할  $n$ 개의 요소들을 힙프에 삽입
- **STEP 2** : 힙프의 삭제 연산을 수행해가며, 삭제 된 요소를 순차적으로 정리



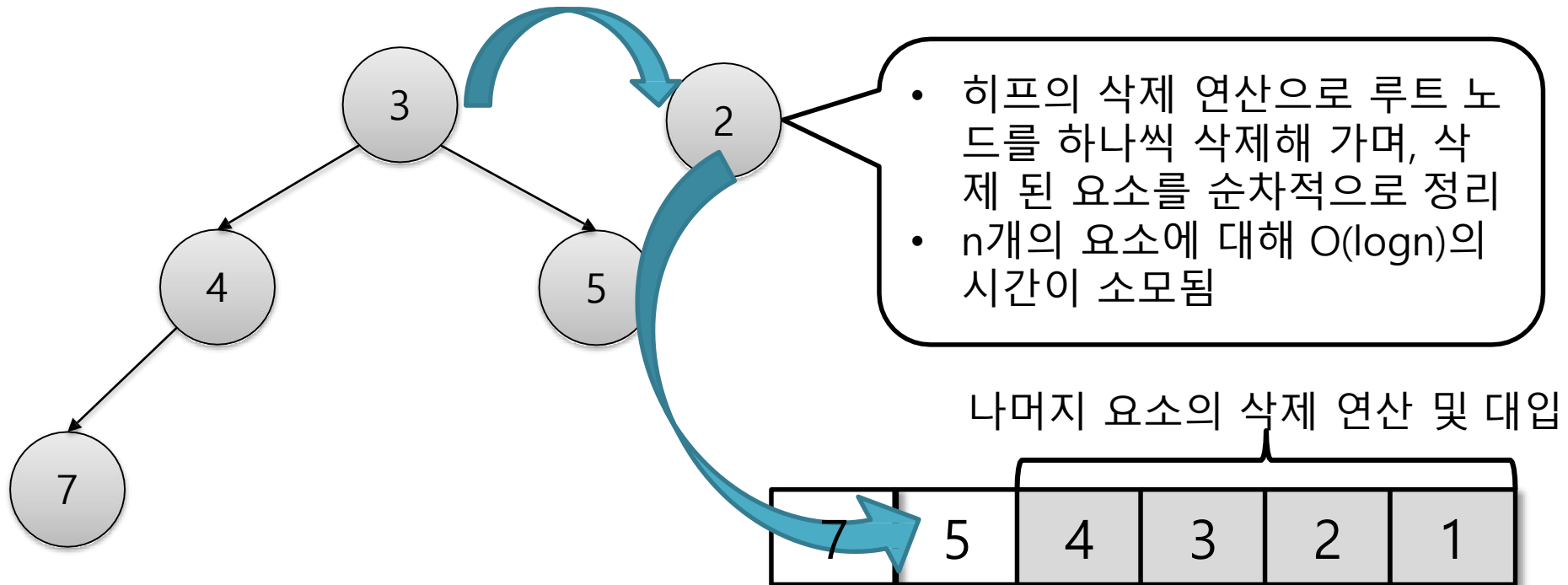
# 실습 목적 및 힙프(Heap)의 이해

## ■ 힙프정렬

- 힙프를 이용한 정렬 방법
- $O(n\log n)$  시간 소모

## ■ 힙프정렬 방법

- **STEP 1** : 정렬해야 할  $n$ 개의 요소들을 힙프에 삽입
- **STEP 2** : 힙프의 삭제 연산을 수행해가며, 삭제 된 요소를 순차적으로 정리



# 실습 문제 - 힙정렬

## ■ 실습 프로그램 설명

- 주어진 배열을 최대 힙정렬로 정렬한 뒤, 정렬 결과를 확인한다.
- 10의 크기를 가지는 정수 배열의 각 요소를 랜덤하게 생성한다.
- 생성된 배열을 최대/최소 힙정렬로 정렬한 뒤, 정렬 결과를 확인한다.

## ■ 주어진 함수

- 힙 함수
  - void init (HeapType \*h)
  - void insert\_max\_heap(HeapType \*h, element item)
  - element delete\_max\_heap(HeapType \*h)
  - void insert\_min\_heap (HeapType \*h, element item)
  - element delete\_min\_heap (HeapType \*h)
  - void MinheapSort(element a[], int n)
  - void MaxheapSort(element a[], int n)
- 기타 함수
  - void verifyMaxheapSort (element \* e)
  - void verifyMinheapSort (element \* e)
    - 정렬이 올바르게 수행되었는지 확인하는 함수
- 힙 함수에 대한 설명은 Heap 강의노트 참조

# 실습 문제 - 힙정렬

## ■ 실습 문제 : 힙정렬의 구현 및 사용

- 정수형 배열을 정렬하는 힙정렬을 구현해 본다.
- 최대 힙정렬과 최소 힙정렬 두 가지 방식으로 정렬 후 결과를 확인해 본다.

```
void main()
{
    element e[10] = { 0 };
    int i,num=0;

    long seconds = (long)time(NULL);
    srand(seconds);

    //rand()함수로 임의의 수를 e배열에 삽입
    for (i = 0; i < 10; i++) {
        num = rand() % 9 + 1;
        e[i].key = num;
    }

    printf("정렬 전 키 값 출력: ");
    for(i=0;i<10;i++) printf("%3d", e[i].key);
    printf("\n");
    verifyMaxheapSort(e);

    flag=0;
    MaxheapSort(e,10);
    printf("Maxheap정렬 후 키 값 출력: ");
    for(i=0;i<10;i++) printf("%3d", e[i].key);
    printf("\n");

    verifyMaxheapSort(e);
    if(flag==0) printf("Maxheap정렬이 확인되었습니다.\n\n");
}
```

```
//rand()함수로 임의의 수를 e배열에 삽입
for (i = 0; i < 10; i++) {
    num = rand() % 9 + 1;
    e[i].key = num;
}
printf("정렬 전 키 값 출력: ");
for(i=0;i<10;i++) printf("%3d", e[i].key);
printf("\n");
verifyMinheapSort(e);

flag=0;
MinheapSort(e,10);
printf("Minheap정렬 후 키 값 출력: ");
for(i=0;i<10;i++) printf("%3d", e[i].key);
printf("\n");

verifyMinheapSort(e);
if(flag==0) printf("Minheap정렬이 확인되었습니다.\n\n");
}
```



# 실습 문제 - 힙정렬

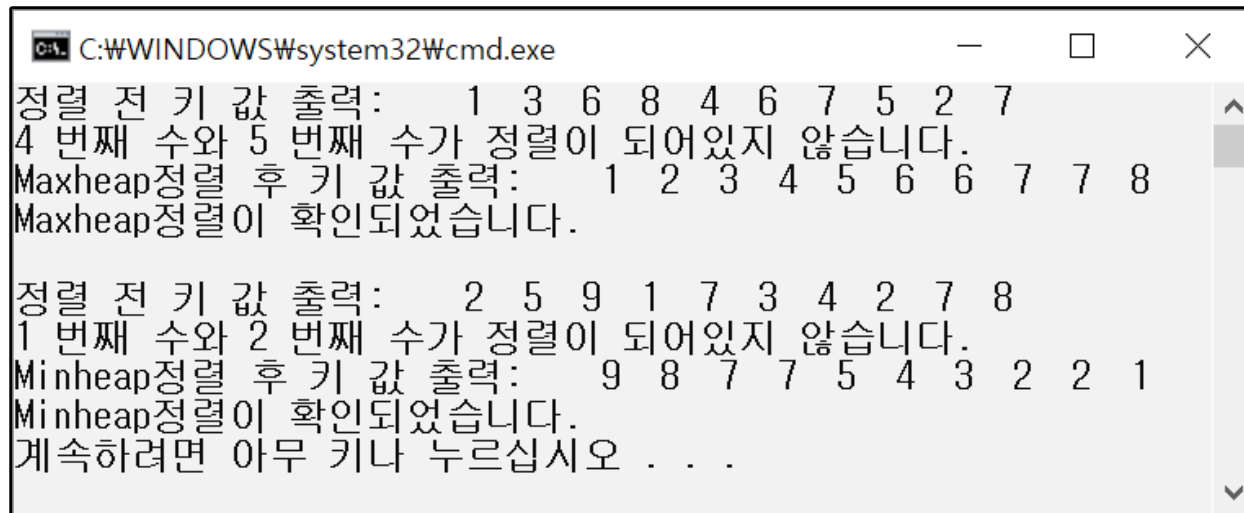
## ■ 실습 문제 : 힙정렬의 구현 및 사용

- 정수형 배열을 정렬하는 힙정렬을 구현해 본다.
- 최대 힙정렬과 최소 힙정렬 두 가지 방식으로 정렬 후 결과를 확인해 본다.

## ■ 메인 함수 설명

- 각 정렬에 대하여, 정렬되지 않은 배열 -> 정렬 -> 정렬 확인을 반복한다.

## ■ 결과 화면



```
C:\WINDOWS\system32\cmd.exe

정렬 전 키 값 출력: 1 3 6 8 4 6 7 5 2 7
4 번째 수와 5 번째 수가 정렬이 되어있지 않습니다.
Maxheap정렬 후 키 값 출력: 1 2 3 4 5 6 6 7 7 8
Maxheap정렬이 확인되었습니다.

정렬 전 키 값 출력: 2 5 9 1 7 3 4 2 7 8
1 번째 수와 2 번째 수가 정렬이 되어있지 않습니다.
Minheap정렬 후 키 값 출력: 9 8 7 7 5 4 3 2 2 1
Minheap정렬이 확인되었습니다.
계속하려면 아무 키나 누르십시오 . . .
```

# 실습 문제 - 힙정렬

## ■ 실습 문제 : 힙정렬의 구현 및 사용

- 정수형 배열을 정렬하는 힙정렬을 구현해 본다.
- 최대 힙정렬과 최소 힙정렬 두 가지 방식으로 정렬 후 결과를 확인해 본다.

```
void heapSort(element a[], int n)
{
    int i;
    HeapType h;

    initHeap(&h);
    ① // a[] 에서 각 요소를 h(heap)에 삽입하세요.

    ② // h(heap)에서 삭제 연산을 하면서,
    // 삭제 연산으로 얻어온 값을 a[]에 순차적으로 대입하세요.
}
```

## ■ 구현 함수 설명

- ① : a[] 배열의 요소를 순차적으로 h(힙)에 삽입한다.
- ② : h(힙)에서 삭제 연산을 수행하면서 나오는 각 요소를 a[] 배열에 순차적으로 대입한다.
- Hint – Heap 강의노트에 있는 heap\_sort, insert\_max\_heap, delete\_max\_heap을 활용하세요.

Thank You  
Q&A