CSE2010 자료구조론

Week 8: Priority Queue, Heap 1

ICT융합학부 한진영

우선순위 큐(Priority Queue)

- 우선순위 큐(priority queue)
  - 우선순위를 가진 항목들을 저장하는 큐
  - FIFO 순서가 아니라 **우선 순위**가 높은 데이터가 먼저 나감

우선순위 큐(Priority Queue)

- 우선순위 큐: 가장 일반적인 큐
  - 스택이나 FIFO 큐를 우선순위 큐로 구현할 수 있음

자료구조	삭제되는 요소
스택	가장 최근에 들어온 데이터
큐	가장 먼저 들어온 데이터
우선순위큐	가장 우선순위가 높은 데이터

- 응용분야
  - 시뮬레이션 시스템(여기서의 우선 순위는 대개 사건의 시각)
  - 네트워크 트래픽 제어
  - 운영 체제에서의 작업 스케쥴링

우선순위 큐 ADT

·객체: n개의 element형의 우선 순위를 가진 요소들의 모임

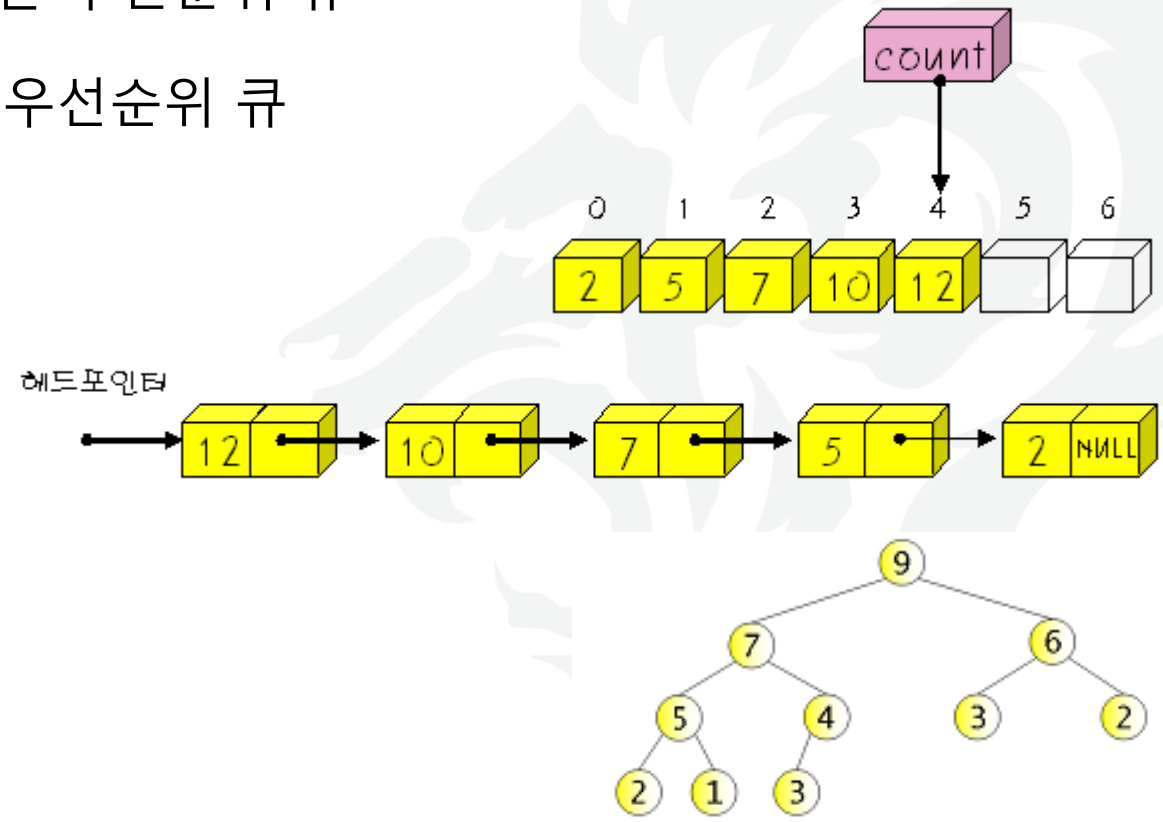
·연산:

- create() ::= 우선 순위큐를 생성
- init(q) ::= 우선 순위큐 q를 초기화
- is_empty(q) ::= 우선 순위큐 q가 비어있는지를 검사
- is_full(q) ::= 우선 순위큐 q가 가득 찼는가를 검사
- insert(q, x) ::= 우선 순위큐 q에 요소 x를 추가
- delete(q) ::= 우선 순위큐로부터 가장 우선순위가 높은 요소를 삭제하고 이 요소를 반환
- find(q) ::= 우선 순위가 가장 높은 요소를 반환

* 가장 중요한 연산은 insert 연산(요소 삽입), delete 연산(요소 삭제)

우선순위 큐 구현 방법(1)

- 배열을 이용한 우선순위 큐
- 연결리스트를 이용한 우선순위 큐
- 힙(heap)을 이용한 우선순위 큐

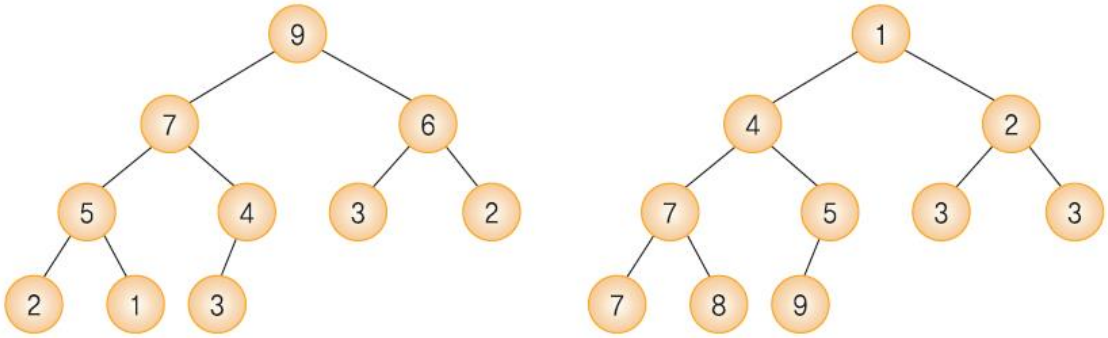


우선순위 큐 구현 방법(2)

표현 방법	삽 입	삭 제
순서없는 배열	O(1)	O(n)
순서없는 연결 리스트	O(1)	O(n)
정렬된 배열	O(n)	O(1)
정렬된 연결 리스트	O(n)	O(1)
힙	O(logn)	O(logn)

힙(Heap)이란?

- 힙: 노드들이 저장하고 있는 키들이 다음과 같은 식을 만족하는 **완전이진트리**
- 최대 힙(max heap)
  - 부모 노드의 키값이 자식 노드의 키값보다 크거나 같은 완전 이진 트리
  - Key(부모노드) ≥key(자식노드)
- 최소 힙(min heap)
  - 부모 노드의 키값이 자식 노드의 키값보다 작거나 같은 완전 이진 트리
  - key(부모노드) ≤key(자식노드)

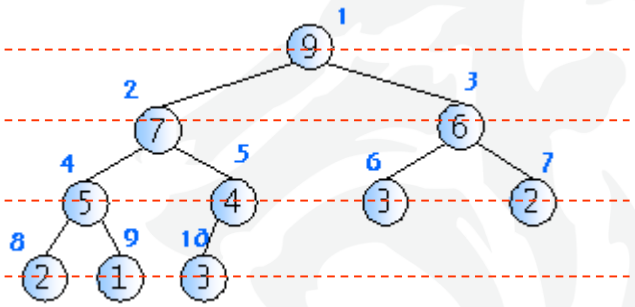


(a) 최대 히프 (b) 최소 히프

힙(Heap)의 높이

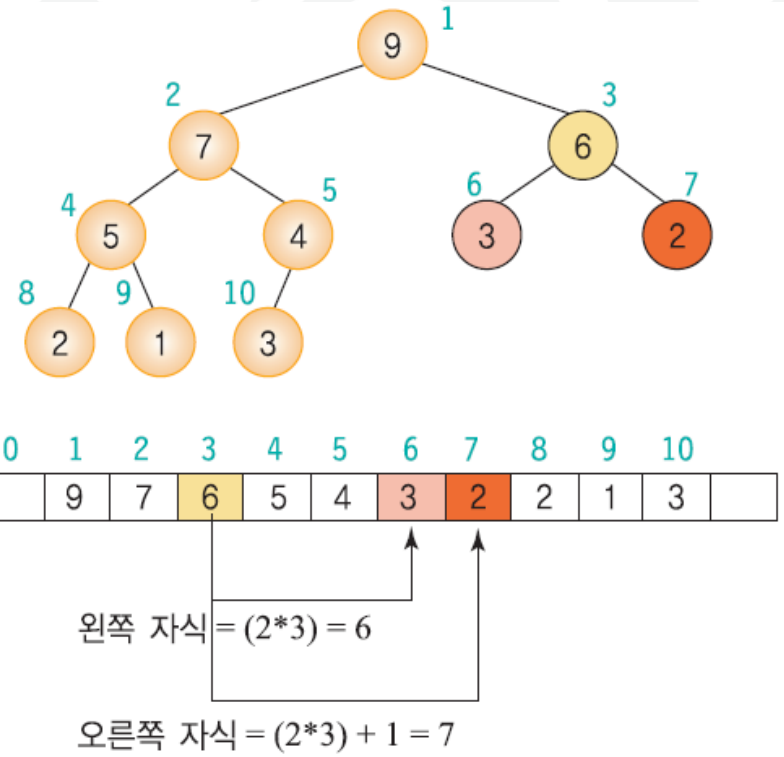
- n개의 노드를 가지고 있는 힙의 높이는 O(logn)
  - 힙은 완전이진트리
  - 마지막 레벨 h을 제외하고는 각 레벨 i에 2^{i-1}개의 노드 존재

깊이	노드의 개수
1	$1=2^0$
2	$2=2^1$
3	$4=2^2$
4	3



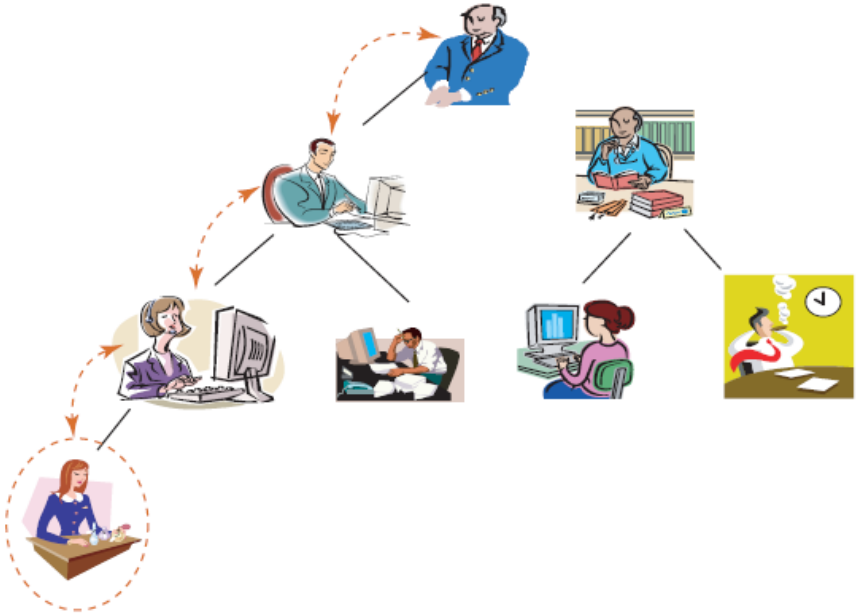
힙(Heap)의 구현

- 배열을 이용한 구현
  - 완전이진트리이므로 각 노드에 번호를 붙임
  - 이 번호를 배열의 인덱스라고 생각
- 부모노드와 자식노드를 찾기가 쉬움
  - 왼쪽 자식의 인덱스 = (부모의 인덱스)*2
  - 오른쪽 자식의 인덱스 = (부모의 인덱스)*2 + 1
  - 부모의 인덱스 = (자식의 인덱스)/2

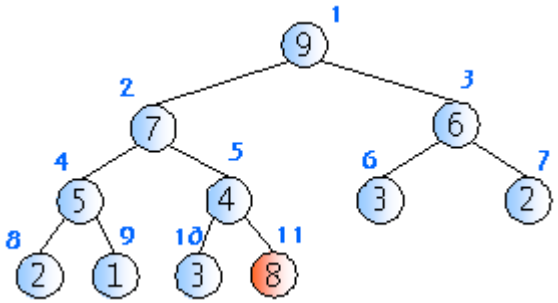


힙(Heap)에서의 삽입

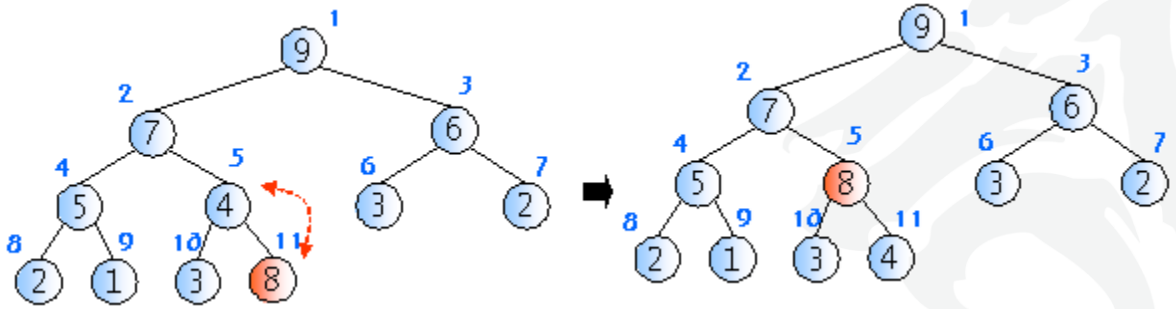
- 삽입 연산
  - 회사에서 신입 사원이 들어오면 일단 말단 위치에 앉힌 다음에, 신입 사원의 능력을 봐서 위로 승진시키는 것과 비슷한 원리
  - 알고리즘
    - (1) 힙에 새로운 요소가 들어 오면, 일단 새로운 노드를 힙의 마지막 노드에 이어서 삽입
    - (2) 삽입 후에 새로운 노드를 부모 노드들과 교환해서 힙의 성질을 만족

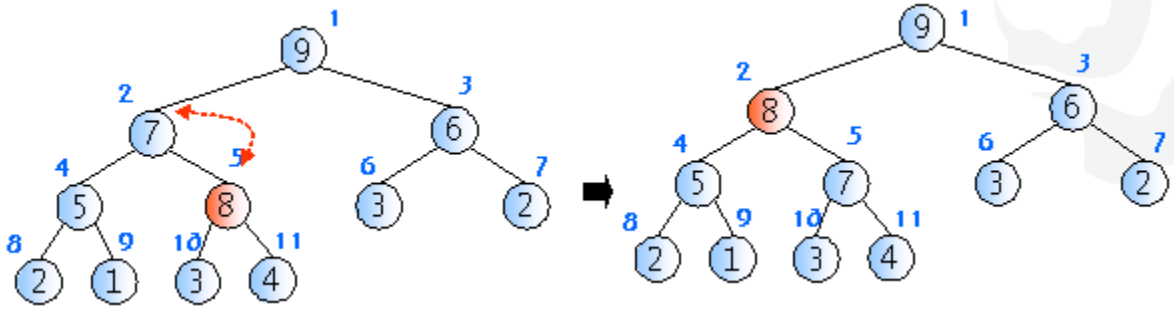


힙(Heap)에서의 삽입: unheap 연산



- 새로운 키 k의 삽입연산후 힙의 성질이 만족되지 않을 수 있음
- Upheap는 삽입된 노드로부터 루트까지의 경로에 있는 노드들을 k와 비교, 교환함으로써 힙의 성질을 복원
- 키 k가 부모노드보다 작거나 같으면 upheap는 종료
- 힙의 높이가 O(logn)이므로 upheap연산은 O(logn)



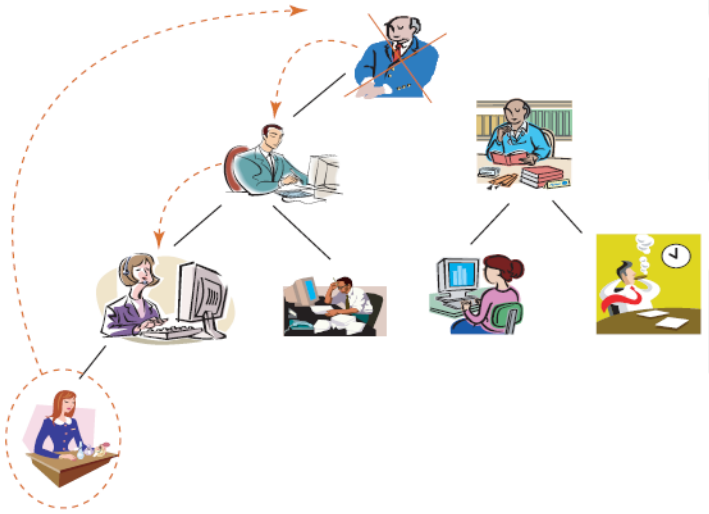


힙(Heap)에서의 삽입 알고리즘

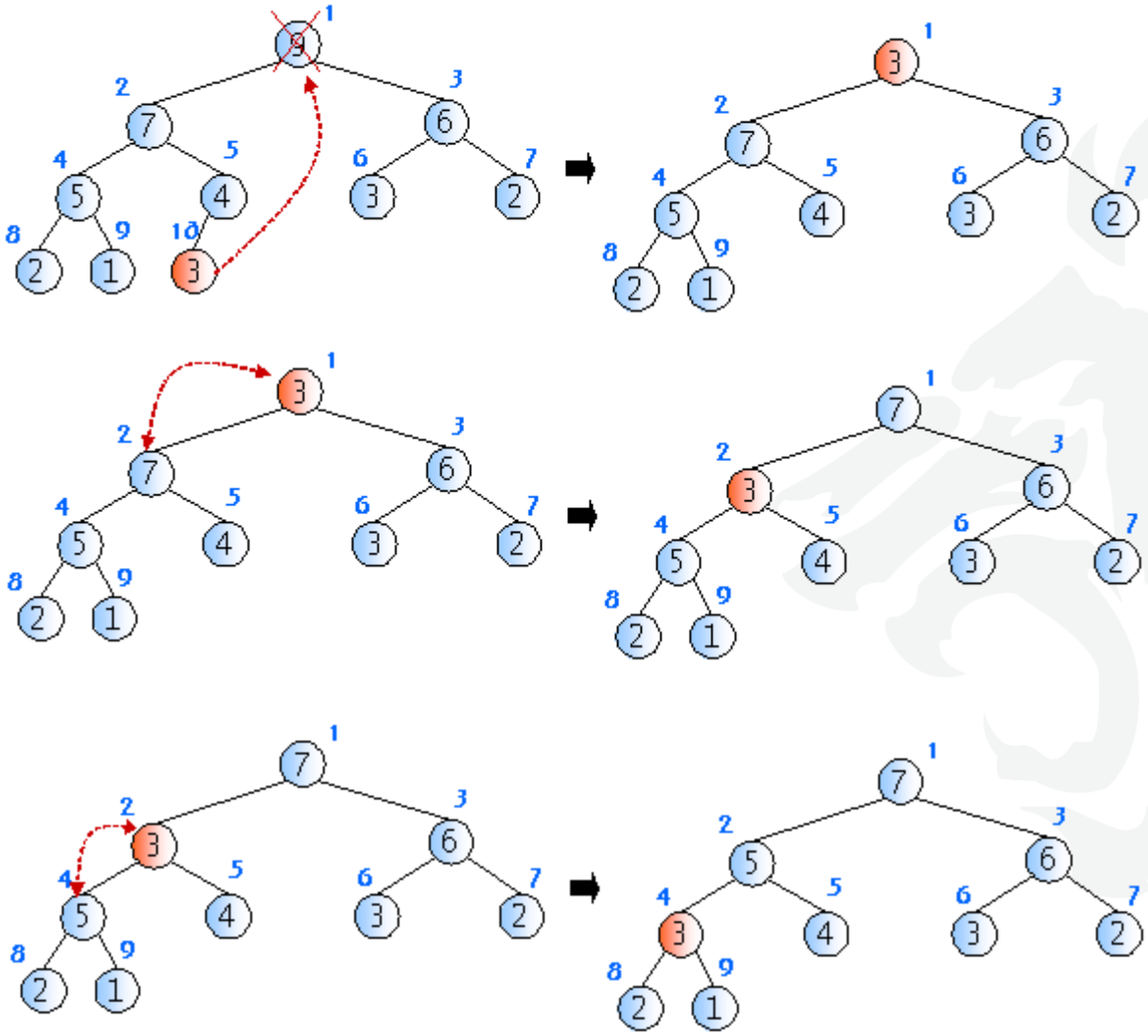
```
insert_max_heap(A, key)


 heap_size ← heap_size + 1;
 i ← heap_size;
 A[i] ← key;
 while i ≠ 1 and A[i] > A[PARENT(i)] do
          A[i] ↔ A[PARENT];
          i ← PARENT(i);
```

힙(Heap)에서의 삭제

- 삭제 연산
  - 최대 힙에서의 삭제는 가장 큰 키값을 가진 노드를 삭제하는 것을 의미 -> 따라서 루트 노드가 삭제
  - 삭제 연산은 회사에서 사장의 자리가 비게 되면 먼저 제일 말단 사원을 사장 자리로 올린 다음에, 능력에 따라 강등시키는 것과 비슷함
  - 알고리즘
    - (1) 루트 노드를 삭제
    - (2) 마지막 노드를 루트 노드로 이동
    - (3) 루트에서부터 단말 노드까지의 경로에 있는 노드들을 교환하여 힙 성질을 만족시킴



힙(Heap)에서의 삭제: downheap 연산



힙의 높이가 O(logn)이므로
downheap 연산은 O(logn)

힙(Heap)에서의 삭제 알고리즘

```
delete_max_heap(A)

item ← A[1];
A[1] ← A[heap_size];
heap_size←heap_size-1;
i ← 2;
while i ≤ heap_size do
      if i < heap_size and A[LEFT(i)] > A[RIGHT(i)]
            then largest ← LEFT(i);
            else largest ← RIGHT(i);
      if A[PARENT(largest)] > A[largest]
            then break;
      A[PARENT(largest)] ↔ A[largest];
      i ← CHILD(largest);

return item;
```

힙(Heap): 복잡도

- 삽입 연산
  - 최악의 경우, 루트 노드까지 올라가야 하므로 트리의 높이에 해당하는 비교 연산 및 이동 연산이 필요 -> O(logn)

- 삭제
  - 최악의 경우, 가장 아래 레벨까지 내려가야 하므로 역시 트리의 높이 만큼의 시간이 걸림 -> O(logn)

Week 8: Priority Queue, Heap 1