안녕하세요, 동계 대학생 S/W 알고리즘 특강의 일곱번째 시간인 오늘은 Heap 에 대해 다루어보도록 하겠습니다.

1. 기초 강의

동영상 강의 컨텐츠 확인 > 6. Heap

Link :
https://swexpertacademy.com/main/learn/course/subjectDetail.do?courseId=CONTENTS_REVIEW&subjectId=AYVXof8qQrgDFARs

※ 출석은 강의 수강 내역으로 확인합니다.

2. 실전 강의

힙은 우선순위 큐를 위해 만들어진 자료구조로, 모든 노드에 대해서 부모와 자식 간에 일정한 대소 관계가 성립하는 완전이진트리 입니다. 부모가 가지는 값은 자식보다 무조건 크거나 작아야 합니다. 부모가 자식보다 큰 값을 가지고 있으면 최대 힙, 작은 값을 가지고 있으면 최소 힙입니다. 전체 자료를 정렬하는 것이 아니라 가장 큰 값만 몇 개 필요한 경우 주로 사용되며, B 형 시험에서 단골로 나오는 자료구조이기 때문에 완벽하게 이해하고 필요할 경우 빠르게 구현할 수 있도록 연습하는 것이 좋습니다.

2.1. 배열을 이용한 힙 구현

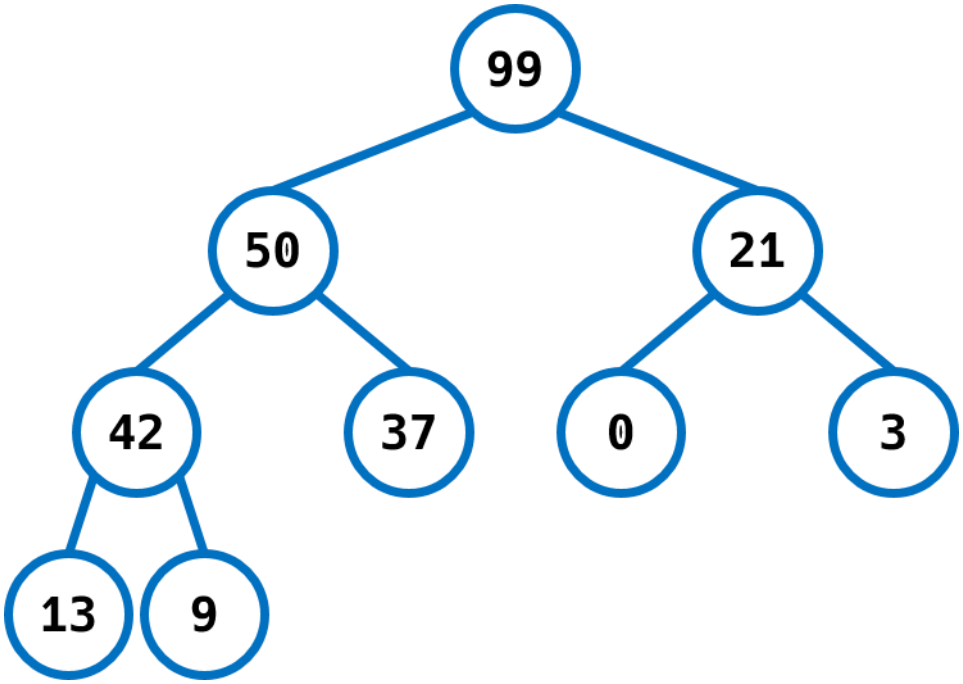
힙은 완전이진트리이므로 배열로 구현하면, 다음과 같이 간단한 수식을 이용하여 부모, 자식 노드에 접근할 수 있습니다.

◆ 루트 노드의 인덱스가 0 으로 시작하는 경우
- 부모의 인덱스 = (자식의 인덱스 - 1) / 2
- 왼쪽 자식 인덱스 = (부모의 인덱스) * 2 + 1
- 오른쪽 자식 인덱스 = (부모의 인덱스) * 2 + 2

◆ 루트 노드의 인덱스가 1 로 시작하는 경우 (0 번 노드는 버림)
- 부모의 인덱스 = (자식의 인덱스) / 2
- 왼쪽 자식 인덱스 = (부모의 인덱스) * 2
- 오른쪽 자식 인덱스 = (부모의 인덱스) * 2 + 1

힙을 포함해서 배열로 구현하는 완전이진트리는 보통 루트 노드의 인덱스로 1 을 사용합니다. 노드 하나의 메모리만 버리면 부모, 자식의 인덱스를 구하는 방법이 더 짧아지므로 편합니다.

1 번 노드를 루트로 하는 최대 힙의 예시입니다.

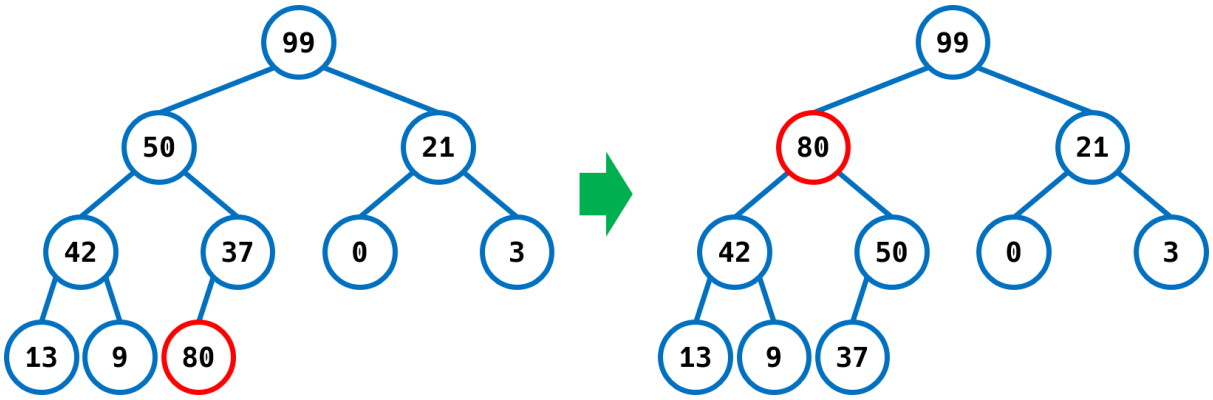


index	0	1	2	3	4	5	6	7	8	9
value		99	50	21	42	37	0	3	13	9

배열을 이용한 완전이진트리는 1 차원 배열에 저장된 데이터를 마치 트리 구조처럼 사용합니다.

2.2. 삽입

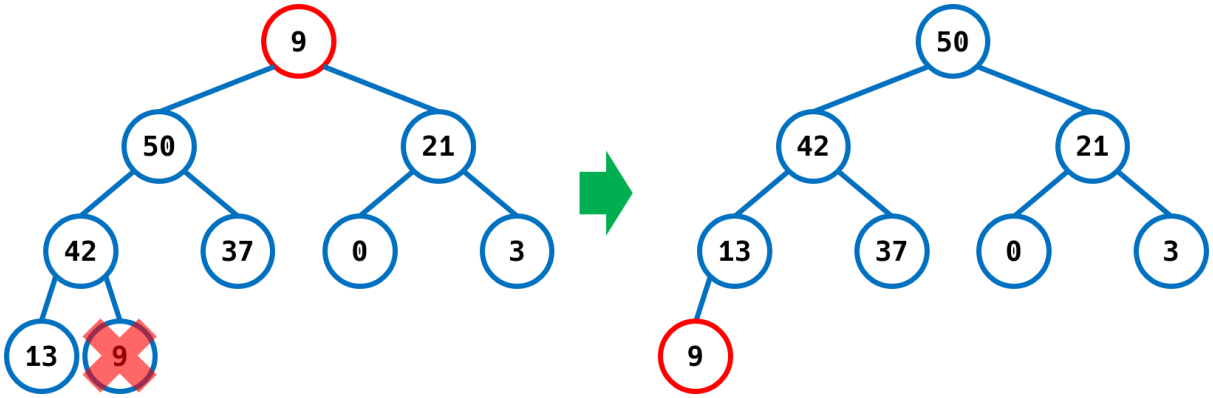
배열의 맨 마지막에 데이터를 삽입하고, 힙 구조를 계속 유지하도록 부모와 값을 바꾸면서 올라갑니다. 힙의 삽입을 거품이 올라가는 모양과 유사하여 Bubble up 이라 합니다.



2.3. 삭제

루트 노드에 위치하는 제일 큰 데이터(최소 힙일 경우 제일 작은 데이터)를 제거한 뒤, 맨 마지막

데이터를 루트 노드로 옮기고, 힙 구조를 계속 유지하도록 자식과 값을 바꾸면서 내려갑니다. 자식과 값을 비교할 때는 둘 중 더 큰 값(최소 힙일 경우 작은 값)과 비교합니다. 힙의 삭제를 거품이 내려가는 모양과 유사하여 Bubble down 이라 합니다.



완전이진트리는 height balanced 한 트리이므로 높이가 O(logN)입니다. 삽입과 삭제 모두 최악의 경우에 루트부터 리프까지의 값을 모두 비교하기 때문에 시간복잡도는 O(트리의 높이), 즉 O(logN)입니다.

2.4. 구현 예

```
#include <algorithm>

#include <cassert>


class MaxHeap {

#define parent (i >> 1)

#define left (i << 1)

#define right (i << 1 | 1)


        static constexpr size_t MAX_N = 100000;


        int data[MAX_N + 1];

        size_t size;
```

```
public:
	MaxHeap() = default;

	// x 삽입
	void push(int x) {
		data[++size] = x;
		for (int i = size; parent != 0 && data[parent] < data[i]; i >>= 1) {
			std::swap(data[parent], data[i]);
		}
	}

	// 최대값 리턴
	int top() const {
		assert(size != 0);
		return data[1];
	}

	// 최대값 삭제
	void pop() {
		assert(size != 0);
		data[1] = data[size--];
		for (size_t i = 1; left <= size;) {
			if (left == size || data[left] > data[right]) {
				if (data[i] < data[left]) {
					std::swap(data[i], data[left]);
```

```
                    i = left;
                } else {
                    break;
                }
            } else {
                if (data[i] < data[right]) {
                    std::swap(data[i], data[right]);
                    i = right;
                } else {
                    break;
                }
            }
        }
    }

#undef parent
#undef left
#undef right
};
```

2.5. 구현 연습

간단한 구현 연습 문제가 준비되어 있습니다. 동계 대학생 S/W 알고리즘 특강 - 기본 문제에서 아래 문제를 풀어주세요.

1. 기초 Partial Sort 연습

구현 연습 관련 아래의 영상도 학습해주세요.

동영상 강의 컨텐츠 확인 > 기초 Partial Sort 연습

※ 출석은 강의 수강 내역으로 확인합니다.

3. 기본 문제

- 힙
- 보급로
- 중간값 구하기
- 수 만들기

4. 응용 문제

- Social Media