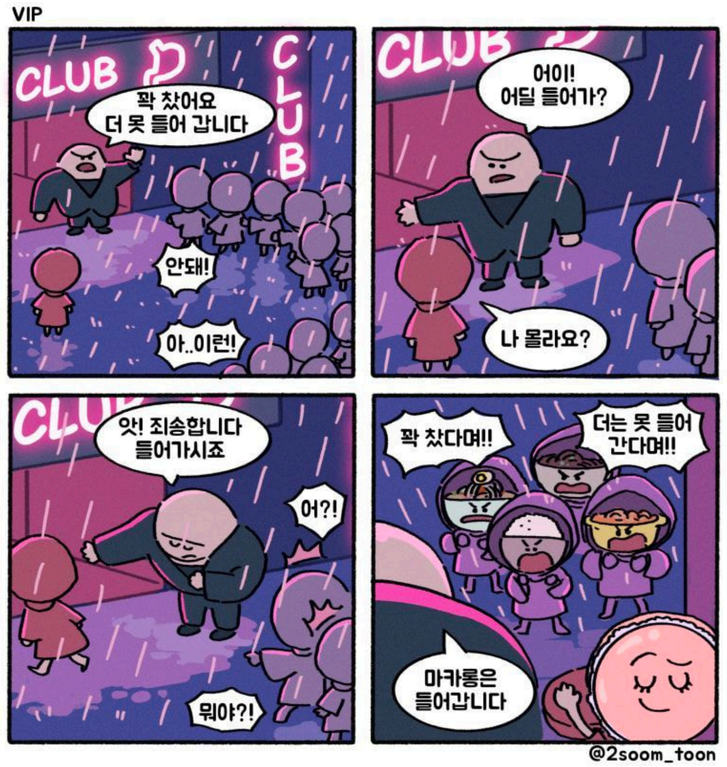
programmers

# 힙

코딩테스트 광탈방지 A to Z : JavaScript - 이선협 @kciter

# 우선순위 큐

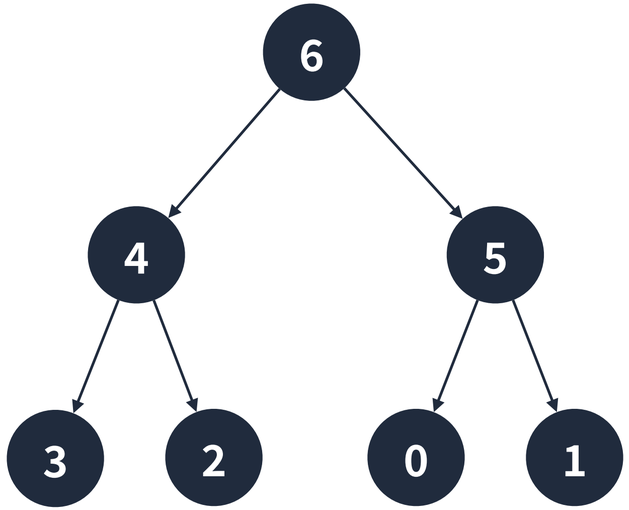
FIFO인 큐와 달리 우선 순위가 높은 요소가 먼저 나가는 큐



우선순위 큐는 자료구조가 아닌 **개념**이다.

# 힙

이진 트리 형태를 가지며 우선순위가 높은 요소가 먼저 나가기 위해
요소가 삽입, 삭제 될 때 바로 정렬되는 특징이 있다.




| Array | | 6 | 4 | 5 | 3 | 2 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|

# 우선순위 큐 != 힙

# 힙의 특징

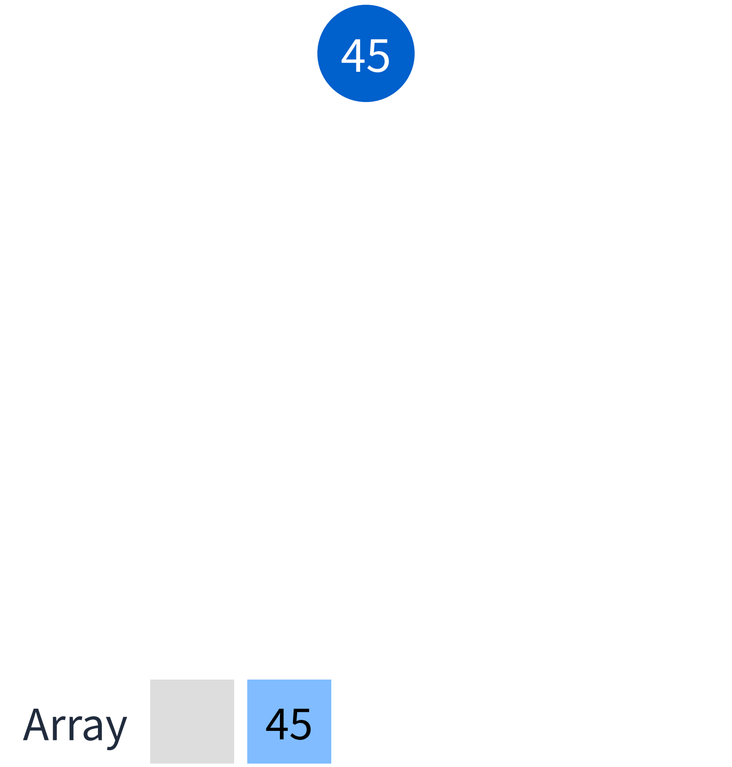
- 우선순위가 높은 요소가 먼저 나가는 특징을 가진다.
- 루트가 가장 큰 값이 되는 최대 힙(Max Heap)과 루트가 가장 작은 값이 되는 최소 힙(Min Heap)이 있다.
- 아쉽게도 자바스크립트에선 직접 구현해서 사용해야 한다.

# Heap 요소 추가

# 힙 요소 추가 알고리즘

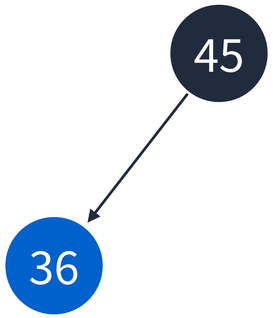
- 요소가 추가될 때는 트리의 가장 마지막에 정점에 위치한다.
- 추가 후 부모 정점보다 우선순위가 높다면 부모 정점과 순서를 바꾼다.
- 이 과정을 반복하면 결국 가장 우선순위가 높은 정점이 루트가 된다.
- 완전 이진 트리의 높이는 log N이기에 힙의 요소 추가 알고리즘은 O(log N) 시간복잡도를 가진다.

# Step 1




최대 힙에 45를 추가

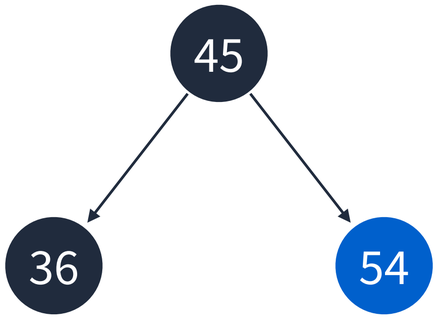
# Step 2





36을 추가

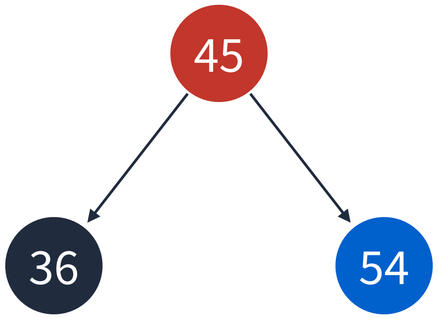
# Step 3





54를 추가

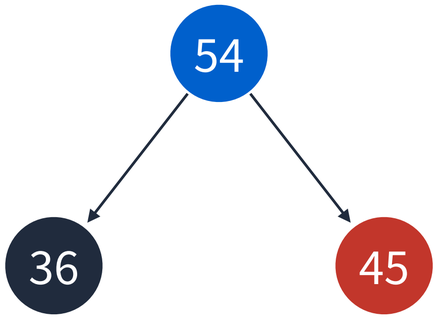
# Step 3





최대 힙에서 45보다 54가 우선순위가 더 높다

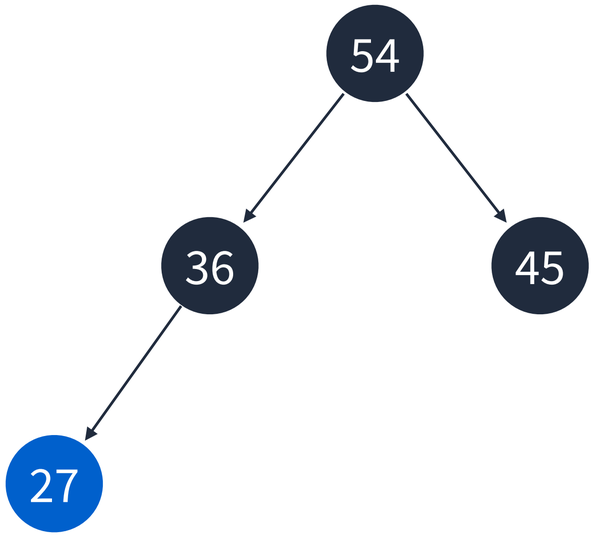
# Step 4





45와 54 정점을 바꾼다

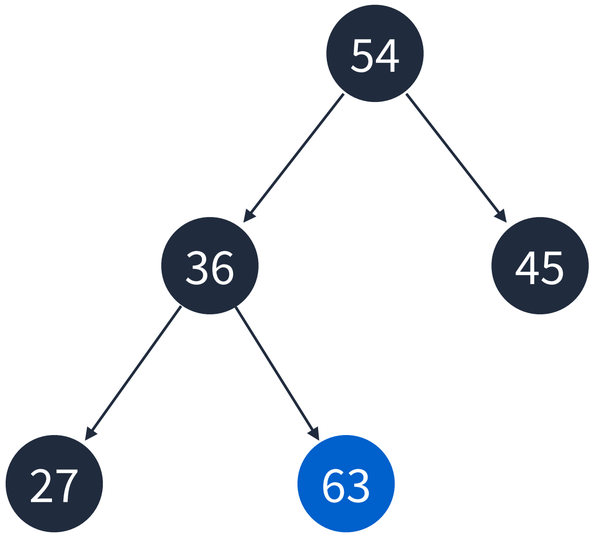
# Step 5





27을 추가

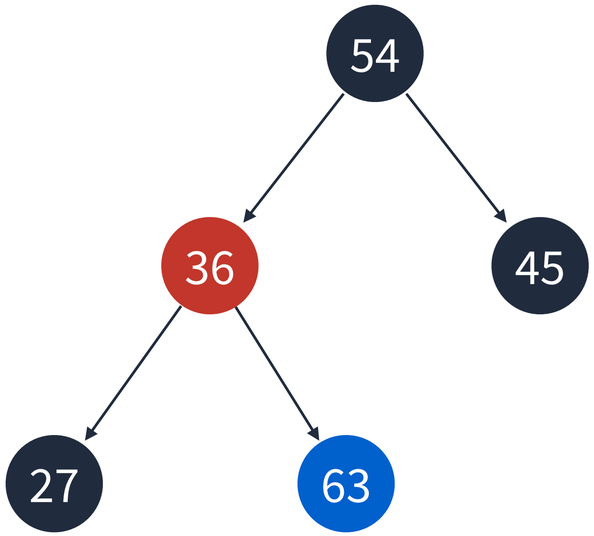
# Step 6




| Array | | 54 | 36 | 45 | 27 | 63 |
|---|---|---|---|---|---|---|

63을 추가

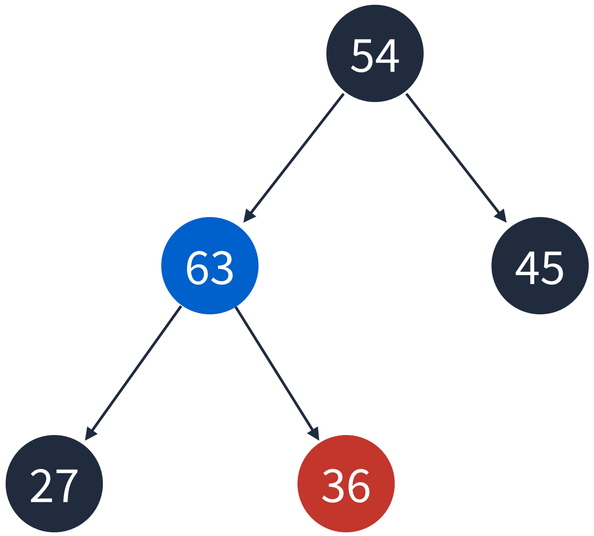
# Step 6





36보다 63이 우선순위가 더 높다

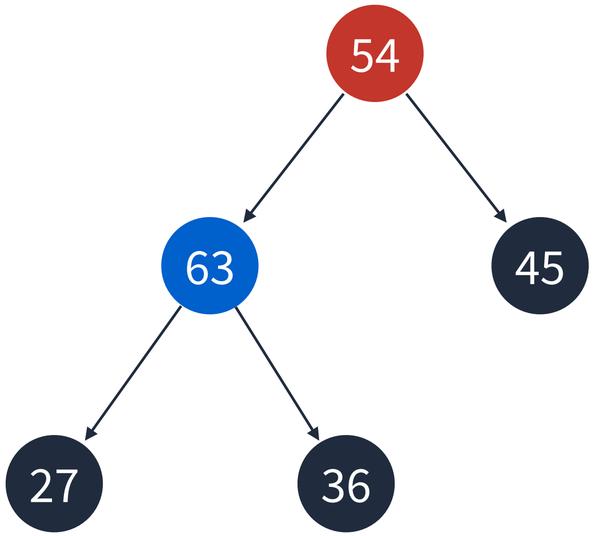
# Step 7





36과 63 정점을 바꾼다

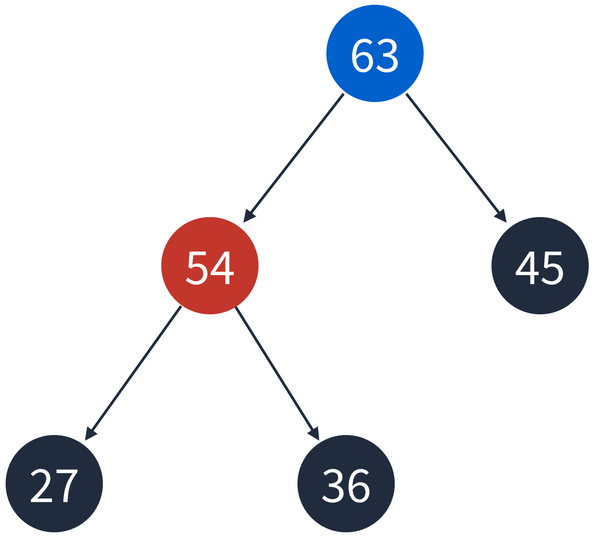
# Step 7





54보다 63이 더 우선순위가 높다

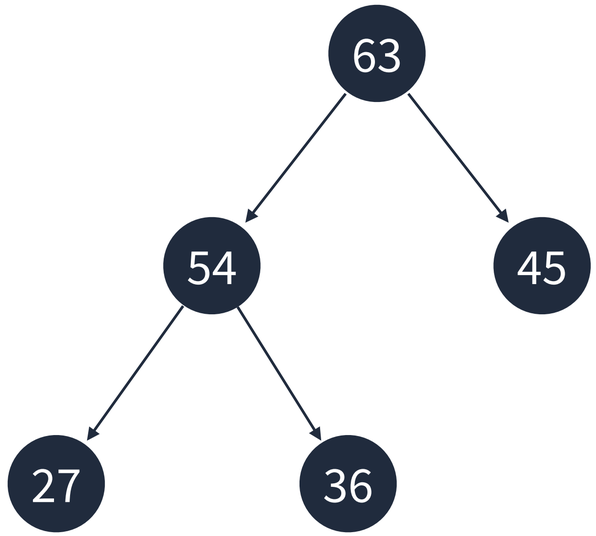
# Step 8





53와 63 정점을 바꾼다

# Step 8




| Array | | 63 | 54 | 45 | 27 | 36 |
|---|---|---|---|---|---|---|

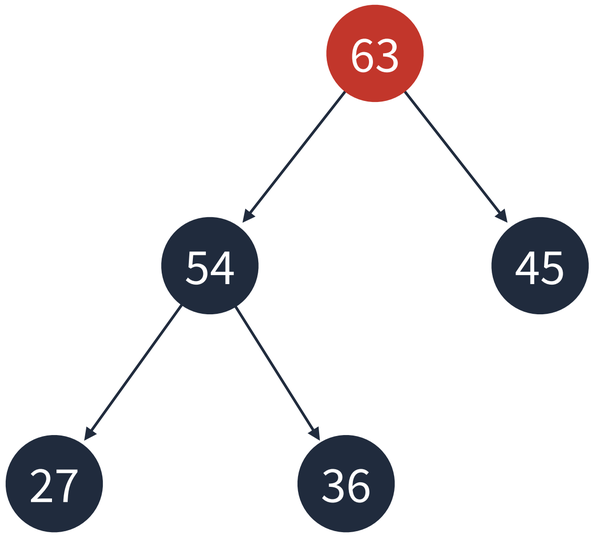
최대 힙 완성!

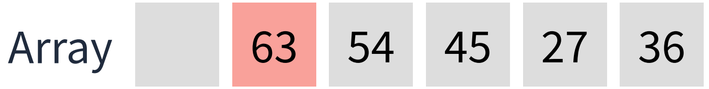
# Heap 요소 제거

# 힙 요소 제거 알고리즘

- 요소 제거는 루트 정점만 가능하다.
- 루트 정점이 제거된 후 가장 마지막 정점이 루트에 위치한다.
- 루트 정점의 두 자식 정점 중 더 우선순위가 높은 정점과 바꾼다.
- 두 자식 정점이 우선순위가 더 낮을 때 까지 반복한다.
- 완전 이진 트리의 높이는 log N이기에 힙의 요소 제거 알고리즘은 O(log N) 시간복잡도를 가진다.

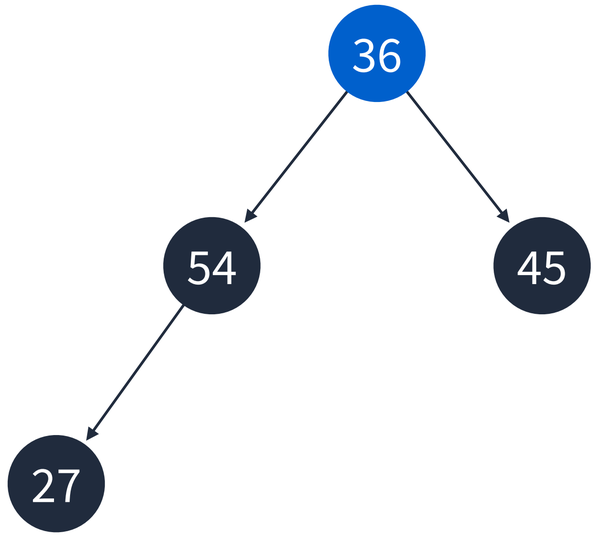
# Step 1




루트 정점 63을 제거한다

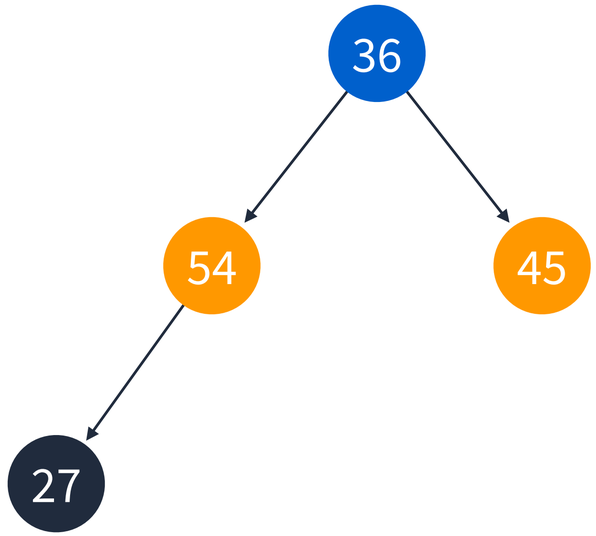
# Step 2





가장 마지막 정점 36을 루트에 위치한다

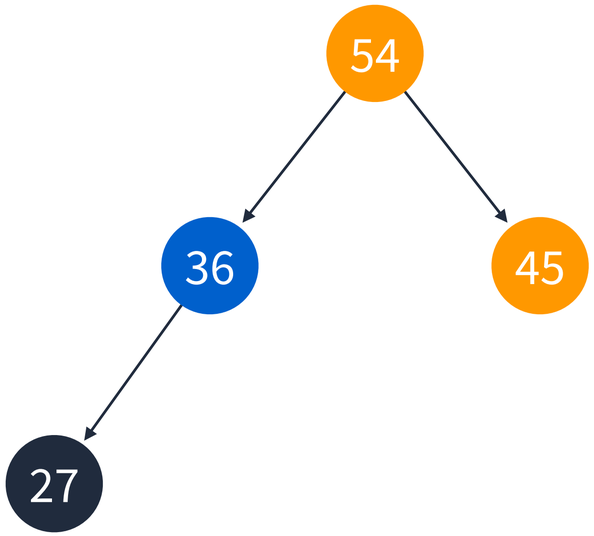
# Step 2





자식 정점 54와 45를 비교한다

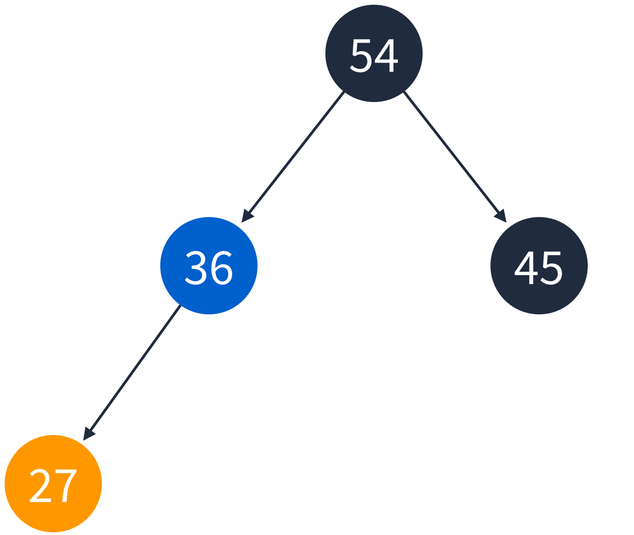
# Step 3





우선순위가 더 높은 54와 바꾼다

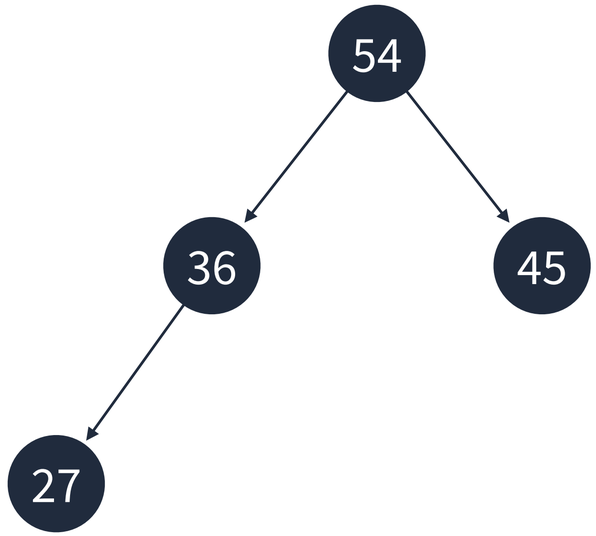
# Step 4





자식 정점 27과 비교한다. 오른쪽 자식은 없기에 27하고만 비교한다

# Step 4




| Array | | 54 | 36 | 45 | 27 |
|---|---|---|---|---|---|

자식 정점의 우선순위가 더 낮기에 바꾸지 않고 끝낸다

# JavaScript에서 사용법

# 힙 요소 추가

```
class MaxHeap {
  constructor() {
    this.heap = [null];
  }

  push(value) {
    this.heap.push(value);
    let currentIndex = this.heap.length - 1;
    let parentIndex = Math.floor(currentIndex / 2);

    while (parentIndex !== 0 && this.heap[parentIndex] < value) {
      const temp = this.heap[parentIndex];
      this.heap[parentIndex] = value;
      this.heap[currentIndex] = temp;

      currentIndex = parentIndex;
      parentIndex = Math.floor(currentIndex / 2);
    }
  }
}
```

```
const heap = new MaxHeap();
heap.push(45);
heap.push(36);
heap.push(54);
heap.push(27);
heap.push(63);
console.log(heap.heap);
// Result is [null, 63, 54, 45, 27, 36]
```

# 힙 요소 추가

```
class MaxHeap {
  constructor() {
    this.heap = [null];
  }

  push(value) {
    this.heap.push(value);
    let currentIndex = this.heap.length - 1;
    let parentIndex = Math.floor(currentIndex / 2);

    while (parentIndex !== 0 && this.heap[parentIndex] < value) {
      const temp = this.heap[parentIndex];
      this.heap[parentIndex] = value;
      this.heap[currentIndex] = temp;

      currentIndex = parentIndex;
      parentIndex = Math.floor(currentIndex / 2);
    }
  }
}
```

```
const heap = new MaxHeap();
heap.push(45);
heap.push(36);
heap.push(54);
heap.push(27);
heap.push(63);
console.log(heap.heap);
// Result is [null, 63, 54, 45, 27, 36]
```

# 힙 요소 추가

```
class MaxHeap {
  constructor() {
    this.heap = [null];
  }

  push(value) {
    this.heap.push(value);
    let currentIndex = this.heap.length - 1;
    let parentIndex = Math.floor(currentIndex / 2);

    while (parentIndex !== 0 && this.heap[parentIndex] < value) {
      const temp = this.heap[parentIndex];
      this.heap[parentIndex] = value;
      this.heap[currentIndex] = temp;

      currentIndex = parentIndex;
      parentIndex = Math.floor(currentIndex / 2);
    }
  }
}
```

```
const heap = new MaxHeap();
heap.push(45);
heap.push(36);
heap.push(54);
heap.push(27);
heap.push(63);
console.log(heap.heap);
// Result is [null, 63, 54, 45, 27, 36]
```

# 힙 요소 추가

```
class MaxHeap {
  constructor() {
    this.heap = [null];
  }

  push(value) {
    this.heap.push(value);
    let currentIndex = this.heap.length - 1;
    let parentIndex = Math.floor(currentIndex / 2);

    while (parentIndex !== 0 && this.heap[parentIndex] < value) {
      const temp = this.heap[parentIndex];
      this.heap[parentIndex] = value;
      this.heap[currentIndex] = temp;

      currentIndex = parentIndex;
      parentIndex = Math.floor(currentIndex / 2);
    }
  }
}
```

```
const heap = new MaxHeap();
heap.push(45);
heap.push(36);
heap.push(54);
heap.push(27);
heap.push(63);
console.log(heap.heap);
// Result is [null, 63, 54, 45, 27, 36]
```

# 힙 요소 제거

```
pop() {
  const returnValue = this.heap[1];
  this.heap[1] = this.heap.pop();

  let currentIndex = 1;
  let leftIndex = 2;
  let rightIndex = 3;
  while (
    this.heap[currentIndex] < this.heap[leftIndex] ||
    this.heap[currentIndex] < this.heap[rightIndex]
  ) {
    if (this.heap[leftIndex] < this.heap[rightIndex]) {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[rightIndex];
      this.heap[rightIndex] = temp;
      currentIndex = rightIndex;
    } else {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[leftIndex];
      this.heap[leftIndex] = temp;
      currentIndex = leftIndex;
    }
    leftIndex = currentIndex * 2;
    rightIndex = currentIndex * 2 + 1;
  }

  return returnValue;
}
```

```
// Heap state: [null, 63, 54, 45, 27, 36]
const array = [];
array.push(heap.pop()); // 63
array.push(heap.pop()); // 54
array.push(heap.pop()); // 45
array.push(heap.pop()); // 36
array.push(heap.pop()); // 27
console.log(array);
// Reulst is [63, 54, 45, 36, 27] - Heap Sort!
```

# 힙 요소 제거

```
pop() {
  const returnValue = this.heap[1];
  this.heap[1] = this.heap.pop();

  let currentIndex = 1;
  let leftIndex = 2;
  let rightIndex = 3;
  while (
    this.heap[currentIndex] < this.heap[leftIndex] ||
    this.heap[currentIndex] < this.heap[rightIndex]
  ) {
    if (this.heap[leftIndex] < this.heap[rightIndex]) {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[rightIndex];
      this.heap[rightIndex] = temp;
      currentIndex = rightIndex;
    } else {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[leftIndex];
      this.heap[leftIndex] = temp;
      currentIndex = leftIndex;
    }
    leftIndex = currentIndex * 2;
    rightIndex = currentIndex * 2 + 1;
  }

  return returnValue;
}
```

```
// Heap state: [null, 63, 54, 45, 27, 36]
const array = [];
array.push(heap.pop()); // 63
array.push(heap.pop()); // 54
array.push(heap.pop()); // 45
array.push(heap.pop()); // 36
array.push(heap.pop()); // 27
console.log(array);
// Reulst is [63, 54, 45, 36, 27] - Heap Sort!
```

# 힙 요소 제거

```
pop() {
  const returnValue = this.heap[1];
  this.heap[1] = this.heap.pop();

  let currentIndex = 1;
  let leftIndex = 2;
  let rightIndex = 3;
  while (
    this.heap[currentIndex] < this.heap[leftIndex] ||
    this.heap[currentIndex] < this.heap[rightIndex]
  ) {
    if (this.heap[leftIndex] < this.heap[rightIndex]) {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[rightIndex];
      this.heap[rightIndex] = temp;
      currentIndex = rightIndex;
    } else {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[leftIndex];
      this.heap[leftIndex] = temp;
      currentIndex = leftIndex;
    }
    leftIndex = currentIndex * 2;
    rightIndex = currentIndex * 2 + 1;
  }

  return returnValue;
}
```

```
// Heap state: [null, 63, 54, 45, 27, 36]
const array = [];
array.push(heap.pop()); // 63
array.push(heap.pop()); // 54
array.push(heap.pop()); // 45
array.push(heap.pop()); // 36
array.push(heap.pop()); // 27
console.log(array);
// Reulst is [63, 54, 45, 36, 27] - Heap Sort!
```

# 힙 요소 제거

```
pop() {
  const returnValue = this.heap[1];
  this.heap[1] = this.heap.pop();

  let currentIndex = 1;
  let leftIndex = 2;
  let rightIndex = 3;
  while (
    this.heap[currentIndex] < this.heap[leftIndex] ||
    this.heap[currentIndex] < this.heap[rightIndex]
  ) {
    if (this.heap[leftIndex] < this.heap[rightIndex]) {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[rightIndex];
      this.heap[rightIndex] = temp;
      currentIndex = rightIndex;
    } else {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[leftIndex];
      this.heap[leftIndex] = temp;
      currentIndex = leftIndex;
    }
    leftIndex = currentIndex * 2;
    rightIndex = currentIndex * 2 + 1;
  }

  return returnValue;
}
```

```
// Heap state: [null, 63, 54, 45, 27, 36]
const array = [];
array.push(heap.pop()); // 63
array.push(heap.pop()); // 54
array.push(heap.pop()); // 45
array.push(heap.pop()); // 36
array.push(heap.pop()); // 27
console.log(array);
// Reulst is [63, 54, 45, 36, 27] - Heap Sort!
```

# 힙 요소 제거

```
pop() {
  const returnValue = this.heap[1];
  this.heap[1] = this.heap.pop();

  let currentIndex = 1;
  let leftIndex = 2;
  let rightIndex = 3;
  while (
    this.heap[currentIndex] < this.heap[leftIndex] ||
    this.heap[currentIndex] < this.heap[rightIndex]
  ) {
    if (this.heap[leftIndex] < this.heap[rightIndex]) {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[rightIndex];
      this.heap[rightIndex] = temp;
      currentIndex = rightIndex;
    } else {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[leftIndex];
      this.heap[leftIndex] = temp;
      currentIndex = leftIndex;
    }
    leftIndex = currentIndex * 2;
    rightIndex = currentIndex * 2 + 1;
  }

  return returnValue;
}
```

```
// Heap state: [null, 63, 54, 45, 27, 36]
const array = [];
array.push(heap.pop()); // 63
array.push(heap.pop()); // 54
array.push(heap.pop()); // 45
array.push(heap.pop()); // 36
array.push(heap.pop()); // 27
console.log(array);
// Reulst is [63, 54, 45, 36, 27] - Heap Sort!
```

# 힙 요소 제거

```
pop() {
  const returnValue = this.heap[1];
  this.heap[1] = this.heap.pop();

  let currentIndex = 1;
  let leftIndex = 2;
  let rightIndex = 3;
  while (
    this.heap[currentIndex] < this.heap[leftIndex] ||
    this.heap[currentIndex] < this.heap[rightIndex]
  ) {
    if (this.heap[leftIndex] < this.heap[rightIndex]) {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[rightIndex];
      this.heap[rightIndex] = temp;
      currentIndex = rightIndex;
    } else {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[leftIndex];
      this.heap[leftIndex] = temp;
      currentIndex = leftIndex;
    }
    leftIndex = currentIndex * 2;
    rightIndex = currentIndex * 2 + 1;
  }

  return returnValue;
}
```

```
// Heap state: [null, 63, 54, 45, 27, 36]
const array = [];
array.push(heap.pop()); // 63
array.push(heap.pop()); // 54
array.push(heap.pop()); // 45
array.push(heap.pop()); // 36
array.push(heap.pop()); // 27
console.log(array);
// Reulst is [63, 54, 45, 36, 27] - Heap Sort!
```

# 힙 요소 제거

```
pop() {
  const returnValue = this.heap[1];
  this.heap[1] = this.heap.pop();

  let currentIndex = 1;
  let leftIndex = 2;
  let rightIndex = 3;
  while (
    this.heap[currentIndex] < this.heap[leftIndex] ||
    this.heap[currentIndex] < this.heap[rightIndex]
  ) {
    if (this.heap[leftIndex] < this.heap[rightIndex]) {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[rightIndex];
      this.heap[rightIndex] = temp;
      currentIndex = rightIndex;
    } else {
      const temp = this.heap[currentIndex];
      this.heap[currentIndex] = this.heap[leftIndex];
      this.heap[leftIndex] = temp;
      currentIndex = leftIndex;
    }
    leftIndex = currentIndex * 2;
    rightIndex = currentIndex * 2 + 1;
  }

  return returnValue;
}
```

```
// Heap state: [null, 63, 54, 45, 27, 36]
const array = [];
array.push(heap.pop()); // 63
array.push(heap.pop()); // 54
array.push(heap.pop()); // 45
array.push(heap.pop()); // 36
array.push(heap.pop()); // 27
console.log(array);
// Reulst is [63, 54, 45, 36, 27] - Heap Sort!
```

programmers

# 힙

코딩테스트 광탈방지 A to Z : JavaScript - 이선협 @kciter