Algorithm
Heap Sort

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Heap Sort

Concept

Features

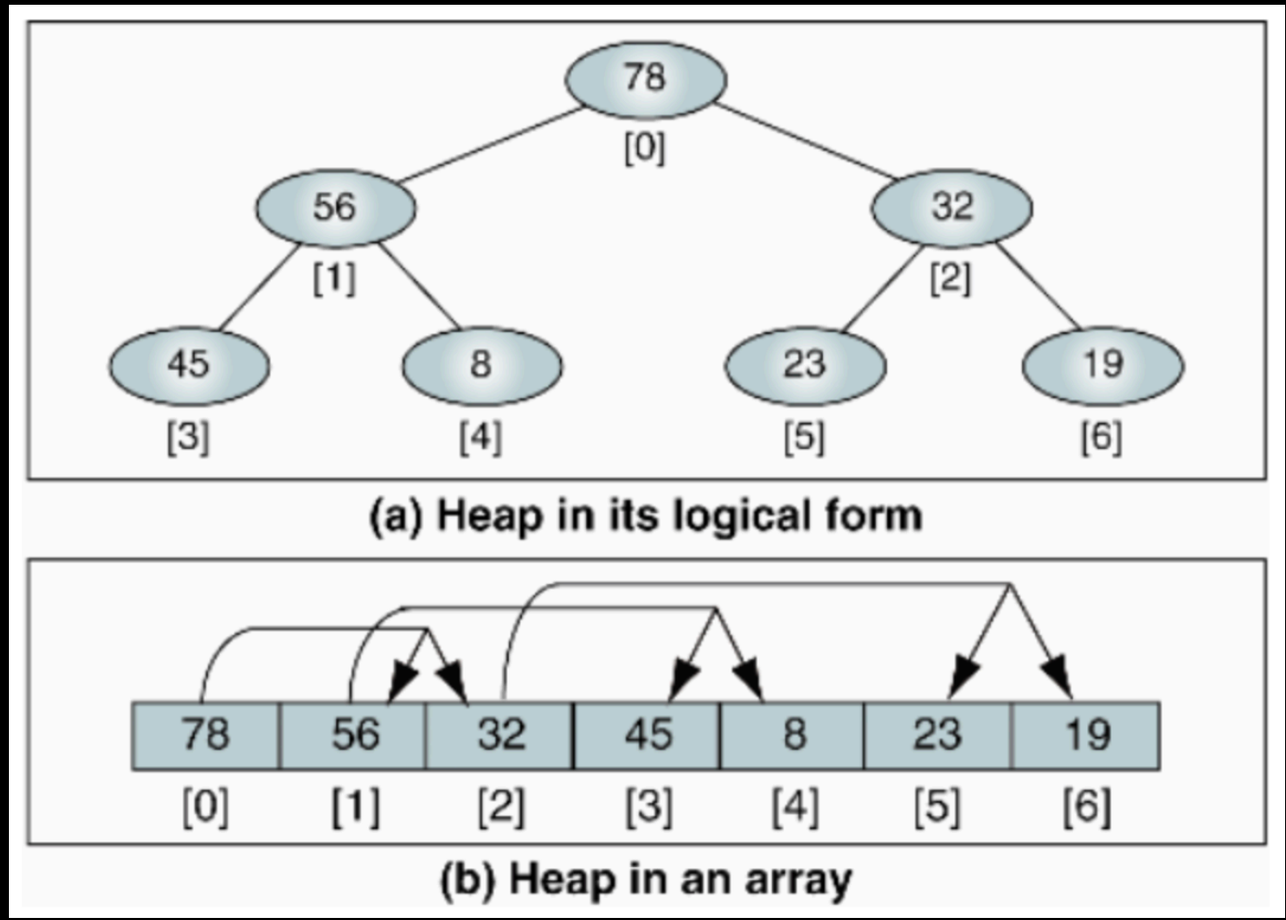
Implementation

References

Heap Sort

Heap Sort(힙 정렬)는 힙 트리를 활용하여 데이터를 정렬을 하는 정렬 알고리즘입니다.

힙 정렬을 이해하기 위해서는 기본적으로 이진 트리 구조와 최대 힙에 대한 이해가 필요합니다.



Concept

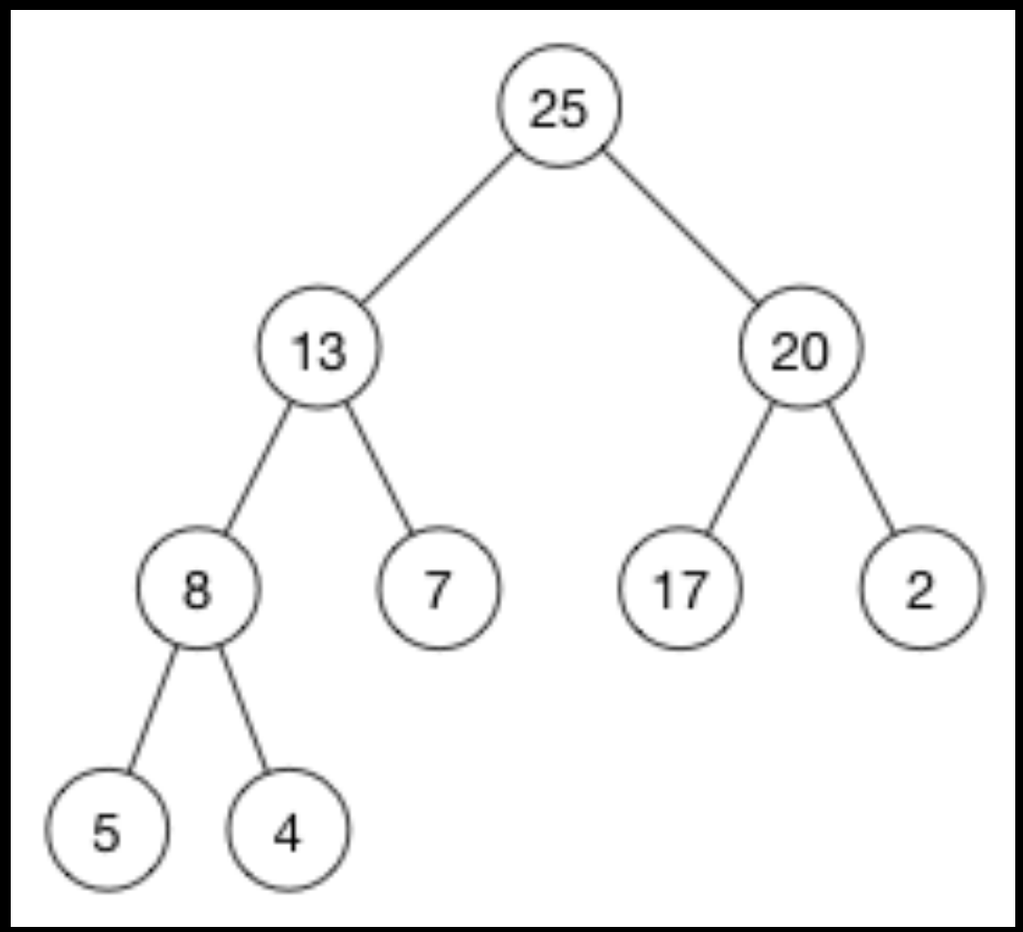
기본적인 알고리즘의 컨셉을 살펴보면 아래와 같습니다.

1. 주어진 데이터를 최대 힙 배열로 만든다.
2. 힙에서 최대값(첫번째 요소)을 마지막 값과 바꾼다.
3. 힙의 크기가 1이 줄어든 것으로 간주하고 마지막 값을 제외하고 다시 최대 힙 배열로 재구성한다. 루트 노드만 힙 조건을 만족하지
4. 2 ~ 3번을 반복하면서 힙의 크기가 1이 될때까지 반복합니다.

Concept

아래와 같은 **최대 힙 트리**가 있다고 가정합니다.
아래의 힙을 배열로 표시하면 아래와 같습니다.

[25, 13, 20, 8, 7, 17, 2, 5, 4]



Concept

[25, 13, 20, 8, 7, 17, 2, 5, 4]

위의 배열은 사실 루트 노드만 제외하고는 정렬이 되었다고 할 수 없는 상태입니다.

이제 루트 노트를 제일 끝으로 보내보겠습니다.

[4, 13, 20, 8, 7, 17, 2, 5, 25]
  *   *

위와 같이 끝에 부분의 요소와 첫번째 요소를 바꾸면 마지막이 가장 큰 수가 됩니다.

이제 제일 마지막 수를 제외하고서 다시 최대힙 정렬을 해보겠습니다.

Concept

[20, 13, 17, 8, 7, 4, 2, 5 | 25]

다시 마지막을 제외하고 최대힙으로 바꾸고 나면 루트 노드(첫번째)는 가장 큰 수가 되었습니다.

바로 이렇게 최대힙은 특징인 최대값은 항상 루트에 있다는 성질을 이용하여 점차 범위를 좁혀가면서 반복적으로 정렬을 하는 것이 힙 정렬의 특징입니다.

이제 계속 힙 정렬 과정을 반복해보면 아래와 같이 진행됩니다.

Concept

[5, 13, 17, 8, 7, 4, 2 | 20, 25]
[17, 13, 5, 8, 7, 4, 2 | 20, 25]
[2, 13, 5, 8, 7, 4 | 17, 20, 25]
[13, 8, 5, 2, 7, 4 | 17, 20, 25]
[4, 8, 5, 2, 7 | 13, 17, 20, 25]
[8, 7, 5, 2, 4 | 13, 17, 20, 25]
[4, 7, 5, 2 | 8, 13, 17, 20, 25]
[7, 4, 5, 2 | 8, 13, 17, 20, 25]
[2, 4, 5 | 7, 8, 13, 17, 20, 25]
[5, 4, 2 | 7, 8, 13, 17, 20, 25]
[2, 4 | 5, 7, 8, 13, 17, 20, 25]
[4, 2 | 5, 7, 8, 13, 17, 20, 25]
[2 | 4, 5, 7, 8, 13, 17, 20, 25]
[2, 4, 5, 7, 8, 13, 17, 20, 25]

Features

Heap Sort(힙 정렬)는 아래와 같은 특징을 가진 알고리즘입니다.

1. 이진 트리 중에서 최대 힙 트리를 활용한 정렬 알고리즘
2. 최악의 경우에도 시간 복잡도가 O(nlogn)
3. 병합 정렬과 다르게 추가적인 배열(공간)이 필요없다.
4. 같은 값을 가진 원소들에 대해서는 순서가 바뀔 수 있는 불안정 정렬이다.

Implementation

Swift를 활용하여 힙 정렬 알고리즘을 살펴보겠습니다.

```swift
func heapify<T : Comparable>(_ array: inout [T], _ i: Int, _ size: Int) {
    var largest = i
    let left = i * 2 + 1
    let right = i * 2 + 2

    if left < size && array[largest] < array[left] {
        largest = left
    }

    if right < size && array[largest] < array[right] {
        largest = right
    }

    if largest != i {
        swap(&array, largest, i)
        heapify(&array, largest, size)
    }
}
```

Implementation

```
func swap<T : Comparable>(_ array: inout [T] , _ i: Int, _ j: Int) {
    let temp = array[i]
    array[i] = array[j]
    array[j] = temp
}

func buildHeap<T : Comparable>(_ array: inout [T]) {
    var i = array.count/2
    while i >= 0 {
        heapify(&array, i, array.count)
        i -= 1
    }
}
```

Implementation

```
func heapSort<T : Comparable>(_ array: inout [T]) -> [T] {
    buildHeap(&array)

    var size = array.count
    var i = size - 1

    while i >= 1 {
        swap(&array, i, 0)
        print("\(array)")

        i -= 1
        size -= 1

        heapify(&array, 0, size)

        print("\(array)")
    }

    return array
}
```

Implementation

```
var array = [ 5, 13, 2, 25, 7, 17, 20, 8, 4 ]

print(heapSort(&array))
// [4, 13, 20, 8, 7, 17, 2, 5, 25]
// [20, 13, 17, 8, 7, 4, 2, 5, 25]
// [5, 13, 17, 8, 7, 4, 2, 20, 25]
// [17, 13, 5, 8, 7, 4, 2, 20, 25]
// [2, 13, 5, 8, 7, 4, 17, 20, 25]
// [13, 8, 5, 2, 7, 4, 17, 20, 25]
// [4, 8, 5, 2, 7, 13, 17, 20, 25]
// [8, 7, 5, 2, 4, 13, 17, 20, 25]
// [4, 7, 5, 2, 8, 13, 17, 20, 25]
// [7, 4, 5, 2, 8, 13, 17, 20, 25]
// [2, 4, 5, 7, 8, 13, 17, 20, 25]
// [5, 4, 2, 7, 8, 13, 17, 20, 25]
// [2, 4, 5, 7, 8, 13, 17, 20, 25]
// [4, 2, 5, 7, 8, 13, 17, 20, 25]
// [2, 4, 5, 7, 8, 13, 17, 20, 25]
// [2, 4, 5, 7, 8, 13, 17, 20, 25]
// [2, 4, 5, 7, 8, 13, 17, 20, 25]
```

References

[1] [Swift Algorithm Club 번역] 힙 정렬 (Heap Sort) : https://oaksong.github.io/2018/04/12/swift-algorithm-club-ko-heap-sort/

[2] Data Structures and Algorithms in Swift: Heap Sort : https://www.raywenderlich.com/53-data-structures-and-algorithms-in-swift-heap-sort

[3] Sorting Algorithms: Implementing Heap Sort Using Swift : https://medium.com/appcoda-tutorials/sorting-algorithms-implementing-heap-sort-using-swift-f24e6868ad28

[4] 힙정렬(Heap Sort) 알고리즘 :: 마이구미 : https://mygumi.tistory.com/310

[5] Heap Sort algorithm: Swift & Objective-C implementations : http://www.marioeguiluz.com/blog/2015/05/14/heap-sort-algorithm-swift-objective-c-implementations/

References

[6] Algorithms: Heapsort in Swift : https://laptrinhx.com/algorithms-heapsort-in-swift-106676493/

[7] Generics in Swift: Merge sort, Quicksort, Insertion sort and Heapsort : https://jkbdev.wordpress.com/2015/01/19/merge-sort-quicksort-insertion-sort-and-heapsort-using-generics-in-swift/

[8] Heap Sort Algorithm : **https://www.programiz.com/dsa/heap-sort**

[9] Heap Sort 정렬 알고리즘 (개념 / 시간복잡도 -O(nlogn)) : https://zeddios.tistory.com/56

[10] [알고리즘] 힙 정렬 (HEAP SORT) 개념과 코드 구현 : https://reakwon.tistory.com/43

Thank you!