Algorithm
Merge Sort

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Merge Sort

Merge Sort(병합 정렬)는 삽입이나 선택 정렬보다 빠른 정렬 알고리즘으로서 분할 및 정복 과정을 통하여 동작하는 알고리즘입니다.

병합 정렬은 재귀 용법을 활용할 정렬 알고리즘으로서, 전체 원소를 가장 작은 단위로 분할한 후 분할한 원소를 다시 병합하면서 정렬하는 분할정복(Divide and Conquer) 방식을 사용합니다.

Concept

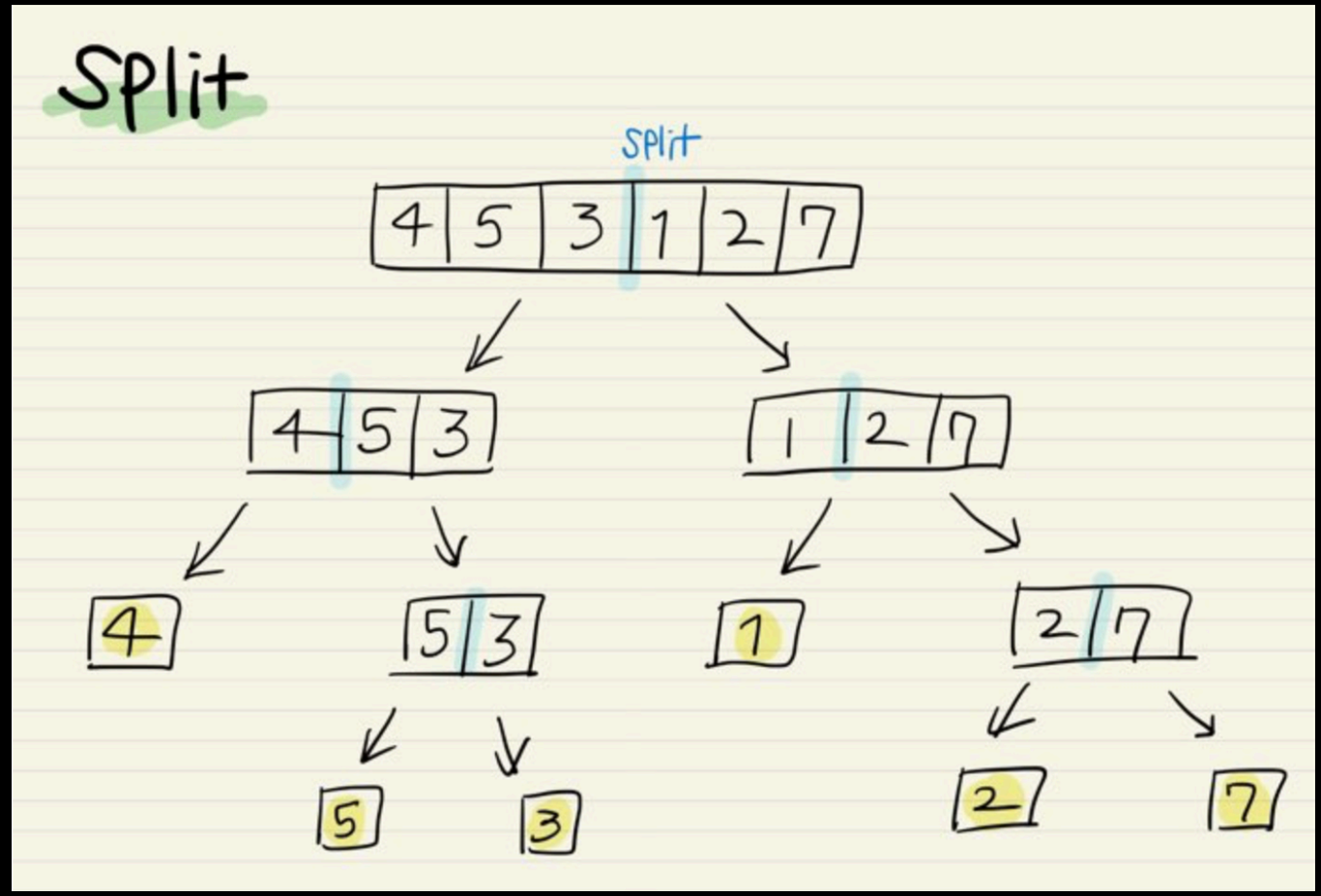
기본적인 알고리즘의 컨셉을 살펴보면 아래와 같습니다.

병합 정렬은 크게 아래의 동작을 통하여 정렬을 실행합니다.

1. 리스트의 길이가 1 이하이면 이미 정렬된 것으로 보고 그대로 데이터(리스트)를 리턴, 그렇지 않은 경우 아래의 과정 수행
2. 분할(Divide) : 정렬되지 않은 리스트를 절반으로 잘라서 두개의 리스트로 나눈다.
3. 정복(Conquer) : 각 부분 리스트를 재귀적으로 합병 정렬을 이용하여 정렬한다.
4. 결합(Combine) : 두 부분 리스트를 다시 하나의 정렬된 리스트로 병합한다.

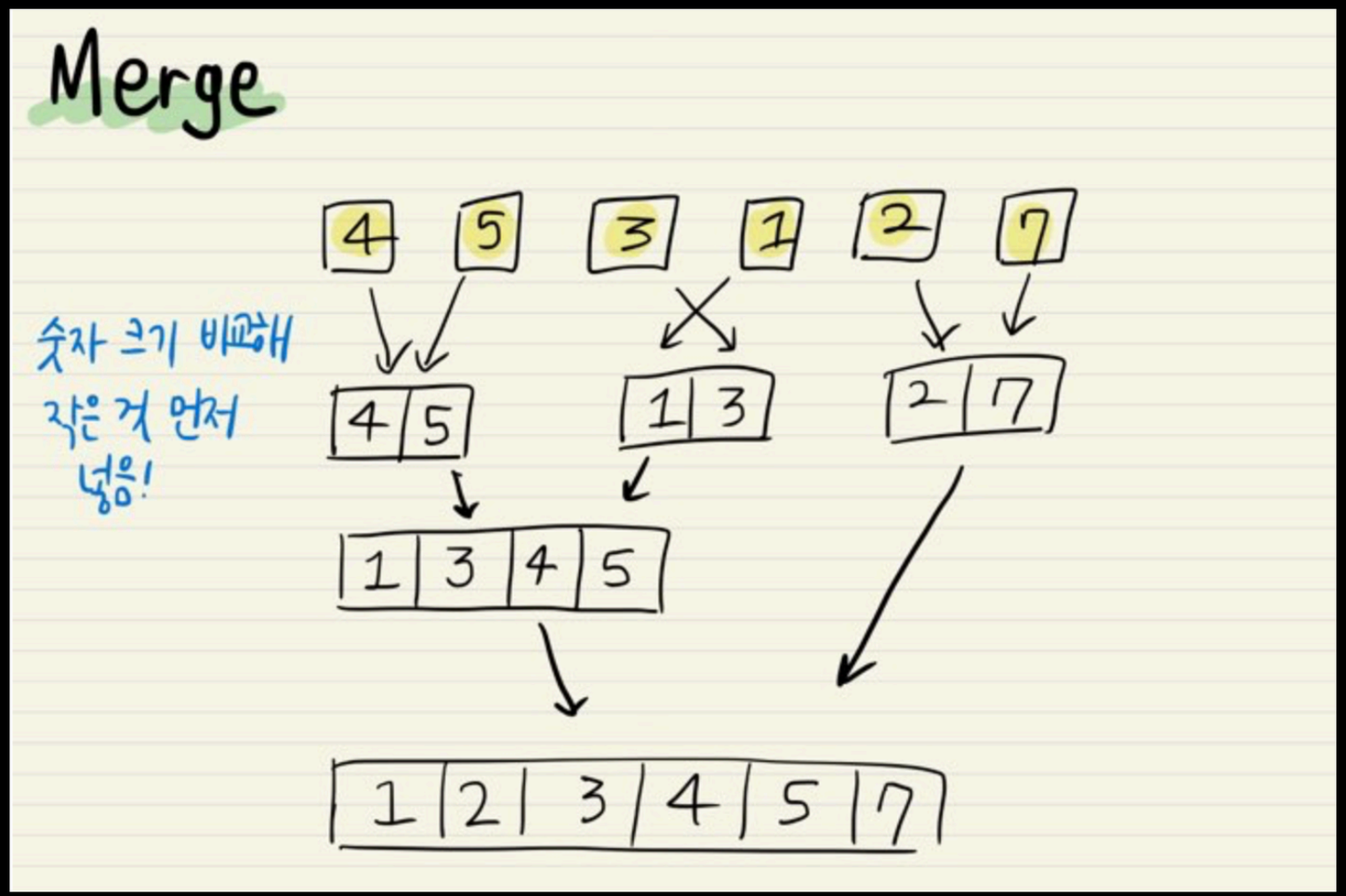
Concept

Divide(Split) :
데이터가 1개일때 해당 데이터 리턴
그외에는 왼쪽과 오른쪽으로 나누어 반드로 쪼갠다. 데이터가 1개가 될때까지



Concept

Merge :
왼쪽 데이터와 오른쪽 데이터를 비교하여 정렬하여 병합, 더이상 병합할 데이터가 없으면 종료



Features

Merge Sort(병합 정렬)는 아래와 같은 특징을 가진 알고리즘입니다.

1. 기본적인 삽입, 선택, 버블 정렬에 비하여 속도가 빠름
2. 데이터를 분할, 정복, 병합의 과정을 통하여 최종 정렬
3. 분할 및 병합 과정을 재귀 호출을 통하여 구현
4. 병합 정렬은 평균적으로나 최악의 경우에도 O(n(log n))의 성능을 가짐
5. 병합 정렬을 수행하기 위한 공간 복잡도는 O(n), 배열을 통한 구현 시에는 O(n) 이지만 연결 리스트로 구현할 시에는 실제로는 O(1)의 공간 사용
6. 실제 사용 시에는 보편적으로 퀵 정렬이나 힙 정렬에 비해 느린 경우가 많다.

Implementation

Swift를 활용하여 병합 정렬 알고리즘을 살펴보겠습니다.

```
public func mergeSort<T: Comparable>(_ array: [T]) -> [T] {
    if array.count < 2 {
        print("split - array : \(array)")
        return array
    }
    let center = array.count / 2

    print("split - array : \(array)")

    return merge(leftHalf: mergeSort([T](array[0..<center])),
                 rightHalf:mergeSort([T](array[center..<array.count])))
}
```

Implementation

```
private func merge<T: Comparable>(leftHalf: [T], rightHalf: [T]) -> [T] {
    var leftIndex = 0
    var rightIndex = 0
    var tempList = [T]()

    //tempList.reserveCapacity(leftHalf.count + rightHalf.count)

    while leftIndex < leftHalf.count && rightIndex < rightHalf.count {
        if leftHalf[leftIndex] < rightHalf[rightIndex] {
            tempList.append(leftHalf[leftIndex])
            leftIndex = leftIndex + 1
        } else if leftHalf[leftIndex] > rightHalf[rightIndex] {
            tempList.append(rightHalf[rightIndex])
            rightIndex = rightIndex + 1
        }
        else {
            tempList.append(leftHalf[leftIndex])
            tempList.append(rightHalf[rightIndex])
            leftIndex = leftIndex + 1
            rightIndex = rightIndex + 1
        }
    }

    tempList += leftHalf[leftIndex..<leftHalf.count]
    tempList += rightHalf[rightIndex..<rightHalf.count]

    print("merge - array : \(tempList)")

    return tempList
}
```

Implementation

```
let numbers = [-1, 0, 1, 2, 5, 13, 15, 20, 68, 59, 51, 45, 77]
print(mergeSort(numbers)) // [-1, 0, 1, 2, 5, 13, 15, 20, 45, 51, 59, 68, 77]

// split - array : [-1, 0, 1, 2, 5, 13, 15, 20, 68, 59, 51, 45, 77]
// split - array : [-1, 0, 1, 2, 5, 13]
// split - array : [-1, 0, 1]
// split - array : [-1]
// split - array : [0, 1]
// split - array : [0]
// split - array : [1]
// merge - array : [0, 1]
// merge - array : [-1, 0, 1]
// split - array : [2, 5, 13]
// split - array : [2]
// split - array : [5, 13]
// split - array : [5]
// split - array : [13]
// merge - array : [5, 13]
// merge - array : [2, 5, 13]
// merge - array : [-1, 0, 1, 2, 5, 13]
// split - array : [15, 20, 68, 59, 51, 45, 77]
// split - array : [15, 20, 68]
// split - array : [15]
// split - array : [20, 68]
// split - array : [20]
// split - array : [68]
// merge - array : [20, 68]
// merge - array : [15, 20, 68]
// split - array : [59, 51, 45, 77]
// split - array : [59, 51]
// split - array : [59]
// split - array : [51]
// merge - array : [51, 59]
// split - array : [45, 77]
// split - array : [45]
// split - array : [77]
// merge - array : [45, 77]
// merge - array : [45, 51, 59, 77]
// merge - array : [15, 20, 45, 51, 59, 68, 77]
// merge - array : [-1, 0, 1, 2, 5, 13, 15, 20, 45, 51, 59, 68, 77]
// [-1, 0, 1, 2, 5, 13, 15, 20, 45, 51, 59, 68, 77]
```

References

[1] [Swift]MergeSort : http://minsone.github.io/programming/merge-sort-in-swift

[2] Merge Sort In Swift : http://www.thomashanning.com/merge-sort-in-swift/

[3] [Swift] Merge Sort(합병 정렬) : https://m.blog.naver.com/PostView.nhn?blogId=qkrgustnrk&logNo=220956604972&proxyReferer=https:%2F%2Fwww.google.com%2F

[4] Merge Sort in Swift : https://medium.com/@notestomyself/merge-sort-in-swift-ae33679251e7

[5] [Algorithm] Merge Sort, 병합 정렬 : https://velog.io/@delmasong/Algorithm-Merge-Sort-병합-정렬

References

[6] Implementing Merge Sort in Swift : https://mikebuss.com/2016/04/28/merge-sort/

[7] [Swift 자료구조 ch10] 합병 정렬 (Merge Sort) : https://kor45cw.tistory.com/246

[8] What is Merge Sort? : https://www.codingame.com/playgrounds/506/sorting-in-swift/merge-sort

[9] Swift 병합 정렬 : https://www.hohyeonmoon.com/blog/swift-merge-sort/

[10] Merge Sort : https://victorqi.gitbooks.io/swift-algorithm/merge_sort.html

Thank you!