Algorithm
Shell Sort

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Shell Sort

Shell Sort(쉘 정렬)는 ‘Donald L. Shell’이라는 사람이 제안한 정렬 알고리즘으로서 삽입 정렬을 보완한 알고리즘입니다.

삽입 정렬이 어느 정도 정렬된 배열에 대해서는 대단히 빠른 것에 착안하여 고안된 알고리즘으로서 기본 삽입 정렬보다 빠른 속도로 동작합니다.

쉘 정렬의 큰 핵심은 데이터를 일정한 수의 부분 집합으로 나누고 해당 부분 집합을 삽입 정렬을 하면서 최종 부분 집합이 0이 될때까지 반복하는 방식입니다.

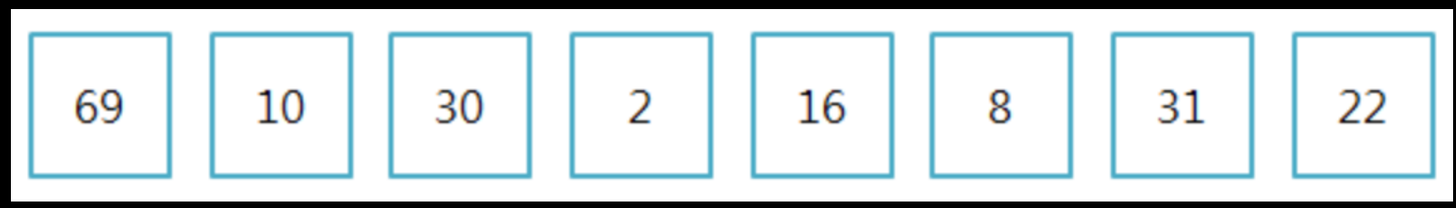
Concept

기본적인 알고리즘의 컨셉을 살펴보면 아래와 같습니다.

1. 리스트를 일정한 기준에 따라서 부분 리스트를 생성(처음에는 N / 2 개의 부분 리스트를 생성함)
2. 각 부분 리스트를 삽입 정렬을 이용하여 정렬
3. 각 부분 리스트가 정렬이 되고 나면 다시 전체 리스트를 더 적은 개수로 부분 리스트로 만듬(이전의 생성한 부분 리스트의 수보다 2배 적은 수로 생성)
4. 2번의 과정을 다시 반복함
5. 다시 부분 리스트를 3번의 규칙대로 생성
6. 부분 리스트 갯수가 0보다 클 때까지 계속 반복하며 정렬을 함

Concept

만약 아래와 같은 수가 있다고 가정합니다.



처음은 원소 갯수(N) / 2개 만큼 부분 리스트를 만듭니다.
아래와 같이 4개의 부분 리스트 쌍을 정합니다.

{69, 16}, {10, 8}, {30, 31}, {2, 22}

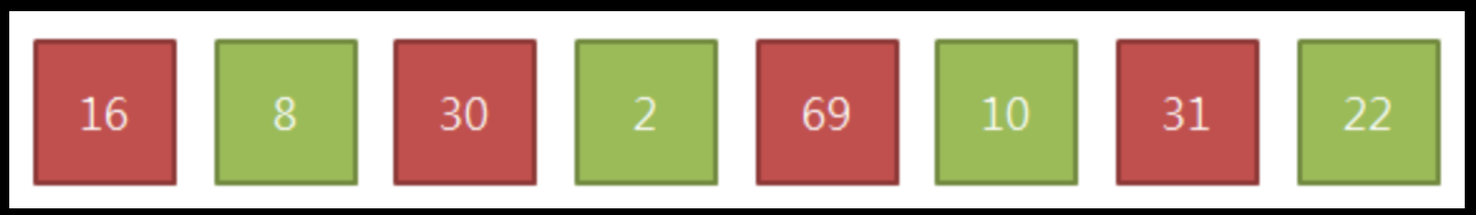
Concept

{69, 16}, {10, 8}, {30, 31}, {2, 22}

해당 부분 리스트의 원소끼리 삽입 정렬을 통하여 정렬하면 아래와 같이 변경됩니다.

{16, 69}, {8, 10}, {30, 31}, {2, 22}

이번에는 부분 집합을 처음보다 2배 작은 2개로 다시 만들어봅니다.



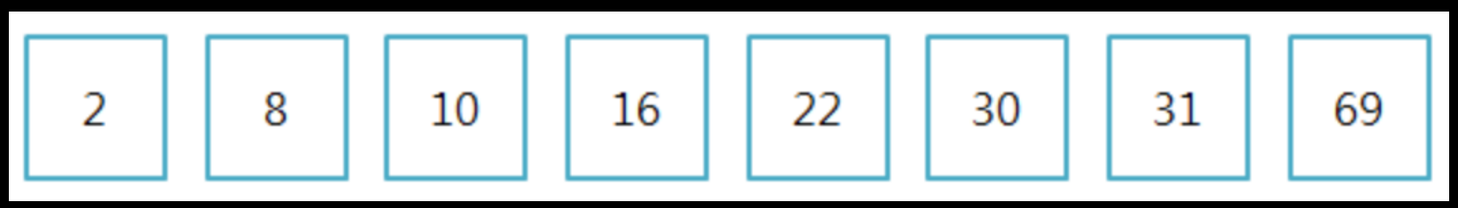
Concept

{16, 30, 69, 31}, {8, 2, 10, 22}

이렇게 생성된 부분 집합을 다시 삽입 정렬로 정렬을 하면 아래와 같이 변경됩니다.

{16, 30, 31, 69}, {2, 8, 10, 22}

이제 다시 마지막으로 부분 집합을 1로 해서 삽입 정렬을 시도하면 아래와 같이 최종 정렬된 배열을 얻을 수 있습니다.



Features

Shell Sort(쉘 정렬)는 아래와 같은 특징을 가진 알고리즘입니다.

1. 쉘 정렬은 삽입 정렬을 활용한 정렬 알고리즘으로서 어느정도 정렬된 상태에서의 삽입 정렬 속도가 빠른 것을 이용한 알고리즘
2. 정렬 속도는 최악의 경우는 O(n^2)이지만 최고의 경우는 O(nlogn)을 가진다.
3. Gap의 규칙에 따라서 속도가 다를 수 있다.
4. 비교적 속도 대비 알고리즘이 간단하여 활용도가 좋다.

Implementation

Swift를 활용하여 쉘 정렬 알고리즘을 살펴보겠습니다.

```
public func shellSort<T : Comparable>(_ array: inout [T]) -> [T] {
    var sublistCount = array.count / 2

    while sublistCount > 0 {
        for pos in 0..<sublistCount {
            insertionSort(&array, start: pos, gap: sublistCount)
        }

        sublistCount = sublistCount / 2
    }

    return array
}
```

Implementation

```
private func insertionSort<T : Comparable>(_ array: inout [T], start: Int, gap:
Int) {
    // start 부터 to 위치까지 gap 만큼의 차이로 루프를 돈다.
    for i in stride(from: (start + gap), to: array.count, by: gap) {
        let currentValue = array[i]
        var pos = i

        while pos >= gap && array[pos - gap] > currentValue {
            array[pos] = array[pos - gap]
            pos -= gap
        }

        array[pos] = currentValue
    }
}
```

Implementation

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

print(shellSort(&array))

// before : [5, 13, 2, 25, 7, 17, 20, 8, 4]
// after : [4, 13, 2, 25, 5, 17, 20, 8, 7]
// before : [4, 13, 2, 25, 5, 17, 20, 8, 7]
// after : [4, 13, 2, 25, 5, 17, 20, 8, 7]
// before : [4, 13, 2, 25, 5, 17, 20, 8, 7]
// after : [4, 13, 2, 25, 5, 17, 20, 8, 7]
// before : [4, 13, 2, 25, 5, 17, 20, 8, 7]
// after : [4, 13, 2, 8, 5, 17, 20, 25, 7]
// before : [4, 13, 2, 8, 5, 17, 20, 25, 7]
// after : [2, 13, 4, 8, 5, 17, 7, 25, 20]
// before : [2, 13, 4, 8, 5, 17, 7, 25, 20]
// after : [2, 8, 4, 13, 5, 17, 7, 25, 20]
// before : [2, 8, 4, 13, 5, 17, 7, 25, 20]
// after : [2, 4, 5, 7, 8, 13, 17, 20, 25]
// [2, 4, 5, 7, 8, 13, 17, 20, 25]
```

References

[1] [Sort] 셸 정렬(Shell Sort)
: https://palpit.tistory.com/127

[2] Shell Sort - 쉘 정렬 : https://dejavuqa.tistory.com/369

[3] Shell Sort : https://victorqi.gitbooks.io/swift-algorithm/shell_sort.html

[4] Shell, merge, heap sort : https://www.slideshare.net/springofmylife/shell-merge-heap-sort-33141131

[5] Shell Sort (쉘 정렬) : https://blastic.tistory.com/127

References

[6] Shell Sort : https://iq.opengenus.org/shell-sort/

[7] Shell Sort Algorithm : https://www.programiz.com/dsa/shell-sort

[8] Implementing Common Sorting Algorithms in Swift : https://agostini.tech/2017/01/10/implementing-common-sorting-algorithms-in-swift/

[9] [알고리즘] 셸 정렬(shell sort)이란 : https://gmlwjd9405.github.io/2018/05/08/algorithm-shell-sort.html

[10] <셸 정렬(shell sort)> 기본 개념 및 알고리즘 : https://mattlee.tistory.com/76

Thank you!