**<정렬 알고리즘>**

1. 종류 : Selection Sort, Bubble Sort, Quick Sort, Insertion Sort, Merge Sort, Heap Sort, Radix Sort
2. 알고리즘 별 시간 복잡도

O() : Bubble Sort, Selection Sort, Insertion Sort, Quick Sort

O() : Heap Sort, Merge Sort

O() : Radix Sort [k는 자릿수이다. 자릿수가 적은 4바이트 정수에서나 제대로 된 성능을 낼 수 있다는 제약이 있다.]

1. 설명
2. Bubbble Sort

• 인접한 두 개의 원소를 비교하여 자리를 바꾸는 방식이다.

• 첫 번째 원소부터 마지막까지 반복하여 끝나면 Ascending이면 가장 큰 원소가, Descending이면 가장 작은 원소가 마지막 자리로 정렬된다.

• 첫 번째 원소부터 인접한 원소끼리 계속 자리를 바꾸면서 맨 마지막 자리로 이동해가는 모습이 물 속에서 물 위로 올라오는 물방울 모양과 같다고 해서 Bubble 정렬이라고 한다.

1. Selection Sort

• 전체 원소들 중에서 기준 위치에 맞는 원소를 선택하여 자리를 바꾸는 방식이다.

• 순서 : 전체 원소들 중에서 가장 작은 원소를 찾아서 선택해 첫 번째 원소와 자리를 바꾼다. 두 번째, 세 번째… N번째까지 이 과정을 반복하면서 정렬이 완성된다.

1. Insertion Sort

• 정렬돼 있는 부분집합에 정렬할 새로운 원소의 위치를 삽입하는 방법이다.

• 정렬할 자료를 2개의 부분집합 A(정렬된 앞 부분의 원소들), B(아직 정렬되지 않은 나머지 원소들)로 가정한다.

<1> 정렬되지 않은 B의 원소를 하나씩 꺼내면서 이미 정렬된 A의 마지막 원소부터 비교하면서 위치를 찾아 삽입한다.

<2> 이런 방법으로 삽입 정렬을 반복하면서 부분집합 A의 원소는 하나씩 늘고 B의 원소는 하나씩 감소하게 된다.

<3> B가 공집합이 되면 정렬이 완성된 것이다.

1. Quick Sort

• 정렬할 전체 원소에 대해서 정렬을 수행하지 않고, 기준 값을 중심으로 왼쪽 부분 집합과 오른쪽 부분 집합으로 분할하여 정렬하는 방법이다.

• 왼쪽 부분 집합에는 기준 값보다 작은 원소를 이동시키고, 오른쪽에는 기준 값보다 큰 원소들을 이동시킨다.

• 기준 값 : pivot, 일반적으로 전체 원소 중에서 가운데에 위치한 원소를 선택한다.

• 분할(divide) : 정렬할 자료들을 기준 값을 중심으로 2개의 부분 집합으로 분할한다.

• 정복(conquer) : 원소들 중에서 기준 값(pivot)보다 작은 원소들은 왼쪽 부분 집합으로, 기준 값보다 큰 원소들은 오른쪽 부분 집합으로 정렬한다. 부분 집합의 크기가 1이하로 충분히 작지 않으면 순환 호출을 이용하여 다시 분할하여 정렬한다.

• 주의사항 : Quick Sort 같은 경우 전체적으로 보게 되면 시간 복잡도가 으로 실행 시간이 준수한 편이지만 기준 값(pivot)에 따라서 크게 달라진다. 적당히 이상적인 값이라면 이지만 최악으로 기준 값을 선택할 경우 이라는 시간 복잡도를 가지게 된다.

1. Heap Sort

• 최대 힙 트리나 최소 힙 트리를 구성해 정렬을 하는 방법으로 내림차순(Descending) 정렬을 위해서는 최대 힙을 구송하고, 오름차순(Ascending) 정렬을 위해서는 최소 힙을 구성한다.

• 순서

<1> n개의 노드에 대한 완전 이진 트리를 구성한다. 이때 루트 노드부터 부노드, 왼쪽 자노드, 오른쪽 자노드 순으로 구성한다.

<2> 최대 힙을 구성한다. 최대 힙이란 부노드가 자노드보다 큰 트리를 말하는데, 단말 노드를 자노드로 가진 부노드로부터 구성하며 아래부터 루트까지 올라오며 순차적으로 만들어 갈 수 있다.

<3> 가장 큰 수(루트에 위치)를 가장 작은 수와 교환한다.

<4> <2>와 <3>의 과정을 반복한다.

1. Merge sort

• 2개 이상의 자료를 Ascending이나 Descending으로 재 배열하는 것이다. 여러 개의 정렬된 자료의 집합을 합쳐서 한 개의 정렬된 집합으로 정렬하는 것이다. 따라서 부분 집합으로 분할(divide)하고, 각 부분집합에 대해서 정렬 작업을 완성(conquer)한 후에 정렬된 부분 집합들을 다시 결합(combine)하는 분할 정복(divide and conquer) 기법을 사용한다.

• 병합 정렬 방법의 종류는 2개의 정렬된 자료의 집합을 결합하여 하나의 집합으로 만드는 병합 방법인 2-way 병합과 똑같지만 n개의 자료 집합을 다루는 n-way 병합이 있다.

• 2-way 병합 정렬 순서

<1> 분할(divide) : 입력 자료를 같은 크기의 부분집합 2개로 분할한다.

<2> 정복(conquer) : 부분집합의 원소들을 정렬한다. 부분집합의 크기가 충분히 작지 않으면 순환 호출을 이용하여 다시 분할 정복 기법을 적용한다.

<3> 결합(combine) : 정렬된 부분 집합들을 하나의 집합으로 결합한다.

<4> <1> -> <2> -> <3> 의 과정을 반복 수행하면서 정렬을 완성시킨다.

1. Radix Sort

• 원소의 키 값을 나타내는 기수를 이용한 정렬 방법이다. 정렬한 원소의 키 값에 해당하는 버킷(bucket)에 원소를 분배하였다가 버킷의 순서대로 원소는 꺼내는 방법을 반복하면서 정렬한다. (원소의 키를 표현하는 기수만큼의 버킷을 사용한다 ex) 10진수로 표현된 키 값을 가진 원소들을 정렬할 때에는 0부터 9까지 10개의 버킷을 사용)

• 순서

<1> 키 값의 일의 자리에 대해서 기수 정렬을 수행한다.

<2> 다음 단계에서는 키 값의 십의 자리에 대해서 정렬을 수행한다.

<3> 그 다음 단계에서는 백의 자리에 대해서 정렬을 수행한다.

<4> <1>, <2>, <3> 에서 진행되었던 것처럼 자릿수 만큼 반복하여 정렬을 한다.

참고 링크:

<https://medium.com/@joongwon/%EC%A0%95%EB%A0%AC-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-%EA%B8%B0%EC%B4%88-805391cb088e>