알고리즘 스터디

2019-03-09 (1회차)

1. 주제 : 정렬
2. 개요 : 정렬 알고리즘의 종류와 개념을 알아보고. 기본 구현을 실제로 진행해본다
3. 내용 :

* Bubble Sort

: 정렬이 되는 모양이 마치 물속에서 거품이 올라오듯이 정렬되는 알고리즘으로, 배열의 처음부터 인접한 두 항의 값을 비교하면서 배열의 끝까지 이동. 마지막 두 원소끼리의 비교가 끝나면 마지막 항의 원소는 자기 자리를 찾았음으로. 마지막 원소를 제외하고 처음부터 인접한 두 항의 값을 비교하면서 이동하는 작업을 반복한다. 시간 복잡도는 O(N2)이 된다.

* Selection Sort

: 전체 배열에서 가장 큰(또는 가장 작은) 원소를 선택(Selection)을 한 뒤 이 원소를 배열의 끝 부분부터 채워 넣고 이 원소를 제외한 나머지 n-1개의 원소에 대해서도 같은 방법을 반복하며 정렬을 실행한다. 시간 복잡도는 O(N2) 이 된다.

* Insertion Sort

: 2번쨰 원소부터 시작하여 배열의 앞쪽으로 이동한다. 이때 처음 시작한 원소의 값을 key 값으로 기억하고 이 값을 각 원소들과 비교하면서 key보다 크면(또는 작으면) 뒤쪽으로 밀어내는 작업을 반복하다가 key보다 작은 값이 나오면 그 위치가 key값이 들어갈 위치가 된다. 이 작업을 3번째, 4번째..원소로 반복하며 마지막 원소까지 반복하여 정렬을 한다. 시간 복잡도는 O(N2)이 된다.

* Shell Sort

: 삽입 정렬의 성능을 개선한 방법으로, key인 원소가 앞쪽으로 이동해야 하는 거리가 멀 경우 시간이 오래 걸리는 단점을 보완한 방법이다. 전체 n 개의 원소를 k값으로 나누어 n/k개의 부분 리스트를 생성한 뒤, 각 부분 리스트에 대한 삽입 정렬을 실행한다. 이 작업이 끝나면 k값을 더 작은 값으로 변경하여 새로운 n/k개의 부분 리스트를 생성하고. 각 부분 리스트를 삽입 정렬하는 것을 k가 1이 될 때까지 계속한다. K가 1이 되면 일반 삽입 정렬을 구현한 것과 같이 동작하게 되는데, 이때 이 전에 부분적으로 정렬이 된 상태이므로 원소가 배열의 끝에서 끝까지 멀리 이동하는 case 를 개선할 수 있다. K값을 구하는 방법으로는 처음에는 n을 2로 나누어서 구하고, k값이 짝수일때보다 홀수일 때 효율적으로 프로그램이 동작하기 때문에 n/2가 짝수인 경우 (n/2)+1값이 초기 k값이 된다. 처음 부분 리스트들에 대한 정렬을 완료하고 난 뒤, 다시 k값을 2로 나누어 가며(짝수가 되면 홀수로 바꿔주며) k가 1이 될때까지 부분 리스트들에 대한 정렬을 실행한다.

* Merge Sort

: 하나의 정렬되지 않은 배열을 두개의 작은 배열로 나누어 정렬한 후 합치면서 정렬하는 알고리즘이다. 전형적인 Divide&Conquer 방식으로 재귀 함수를 사용하여 작성할 수 있다. 알고리즘은 구체적으로 배열의 크기를 기준으로 두개의 배열로 나누는 부분과 나누어진 두개의 배열을 대상으로 재귀 함수를 이용하여 정렬하는 부분 마지막으로 정렬된 두개의 작은 배열을 앞에서부터 차례대로 비교하며 작은 값을 먼저 입력하는 합병 과정을 거친다. 전형적인 O(nlogn)의 시간복잡도를 가지는 알고리즘이나 똑같은 크기의 배열을 하나 더 필요로 하여 메모리 효율 면에서 불리하다.

* Quick Sort

: 주어진 입력 리스트에서 pivot이라는 키 값을 정하고 정렬 기준이 오름차순일 경우 그 값보다 작은 값을 왼쪽에, 그 값보다 큰 값을 오른쪽에 정렬한다. 그리고 pivot을 기준으로 분리된 두 개의 서브 리스트들을 위의 방법과 동일하게 재귀적으로 각각 재배열하는 과정을 수행하는 방식이다. 우리는 이것을 분할과 정복 방식이라고 한다.

 1. 정렬할 데이터 리스트에서 pivot하나를 선택한다.

 2. 분할 :　pivot보다 작은 수들은 pivot기준으로 왼쪽에, 큰 수들은 오른쪽에 정렬한다.

 3. 재귀적 수행

위 방법으로 정렬을 하면 평균 연산 시간에 있어서 내부 정렬 방법 중에서 실제 수행 속도가 가장 빠른 정렬 알고리즘이다. 이 정렬의 시간복잡도는 O(nlogn)이다.

* Heap Sort

: 힙(heap)은 2진트리의 계층적인 구조로 루트를 제외한 모든 값이 자신보다 큰 값을 부모로 가질 때 max-heap 그 반대로 작은 값을 부모로 가질 때 min-heap 이라고 부르는 자료구조이다. 최솟값 혹은 최댓값을 조회하기 위하여 고안되었으며 원하는 값이 반환된 후 힙을 빠져나갔을 때에도 내부적으로 그 구조를 유지해야한다. 이러한 힙은 배열을 통해 구현하며 주로 배열의 0번 인덱스를 제외하고 1번을 루트로 하여 구현한다. 자신의 인덱스를 2로 나눈 몫을 자신의 부모로 하게 구현한다. 힙을 구현하는데 걸리는 시간은 O(n)이다. 이렇게 완성된 힙의 특성을 통하여 쉽게 정렬을 구현할 수 있는데, 최댓값을 순서대로 반환하면 정렬된 배열이 완성된다. 완성된 힙에서 그 값을 전부 빼내는 과정은 O(nlogn) 으로 무작위의 배열을 힙으로 만들고 정렬하는 과정은 O(n + nlogn) => O(nlogn) 으로 결국 big-O 개념상 O(nlogn) 의 시간복잡도를 가진다. 힙은 주로 큰 값이나 작은 값을 차례대로 이용할 때, 배열을 전부 탐색할 필요가 없거나 상대적으로 느린 속도로 값을 읽어가도 될 때 유용하게 사용할 수 있다.