<알고리즘>

2장 요약문

Divide and Conquer

분할정복 알고리즘. 알고리즘에 대해서 한 번이라도 찾아본 적이 있다면 들어본 적이 있을 것이다. 이름에서부터 알 수 있듯이 주어진 문제를 작아지는 방향으로 분할하고, 분할된 것을 계산하여 그 결과를 나중에 합치는 것이다. 어느 정도까지 분할하고, 분할하여 계산한 결과를 어떻게 합치는 지가 이 장에서의 주목해야 할 내용이다.

Binary Search, MergeSort, QuickSort, StoogeSort, Large Integer Multipication, Matrix Multipication 이 모든 알고리즘이 모두 DAC, 분할정복 알고리즘이다.

당연하게도 모두 조금씩 차이점이 있지만 기본적인 원리는 DAC를 따라간다.

이진 탐색의 경우 찾고자 하는 target 값과 주어진 문제의 중간 값을 계속해서 비교하며 그 크고 작음에 따라 재귀호출로 분할하여 다른 한 쪽은 탐색하지 않는 방법이다.

MergeSort의 경우 주어진 문제를 충분히 작아질 때까지 분할하고 분할된 상태로 정렬을 하고 합병을 하여 모든 합병이 이루어졌을 때 완벽하게 정렬된다. 분할하는 방법이 반으로 자르는 것이기 때문에 간단한 편이고, 균등하게 분할된다는 장점이 있지만 추후에 기술할 quickSort에는 없는 합병 과정이 필요하다는 것이 단점이 된다.

QuickSort의 경우 합병과정은 없지만 분할과정이 복잡하고 균등하지 않을 수 있다. 피버값을 기준으로 한 쪽으로 작은 값을, 다른 한 쪽으로 큰 값을 두고, 나누어진 영역을 다시 피버값으로 분할해가는 것이기 때문이다.

StoogeSort의 경우 알고리즘이 매우 신기하다. 전체를 2/3으로 나누고 그것을 세 번 반복하여 정렬하는 것이다. 다른 두 정렬 알고리즘의 장점을 가져온 것인데도 성능이 좋지 않다는 것도 무척 신기한 일이다.

행렬의 곱의 경우 단순한 아이디어가 성능개선에 많은 도움을 준다. N by n 행렬의 경우 곱셉을 할 때 무조건 n^3의 시간 복잡도가 걸리는 것을 다른 원소들을 덧셈으로 먼저 계산해둠으로써 곱셉 횟수를 줄이고 시간 복잡도를 줄일 수 있다. 대표적으로 Strassen의 알고리즘이 꽤 중요하다. 행렬 곱 알고리즘의 시간 복잡도를 줄이는 여러 알고리즘이 있지만, 줄어드는 차수에 비해 계수가 매우 크므로 실효성이 떨어진다는 것이 인상깊다.

몰랐던 정렬 알고리즘에 대해서 배운 것이 가장 좋았던 것 같다. 배운 김에 코드로 작성해서 시간 복잡도에 따라 변화가 체감되는지 확인하고 싶었지만, 어지간한 크기로는 체감이 되지 않았다.