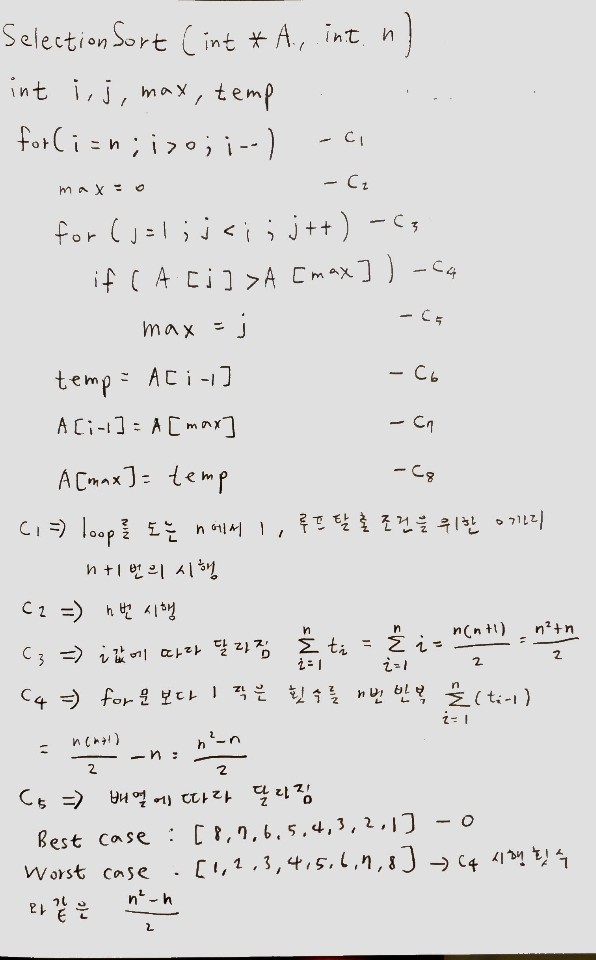
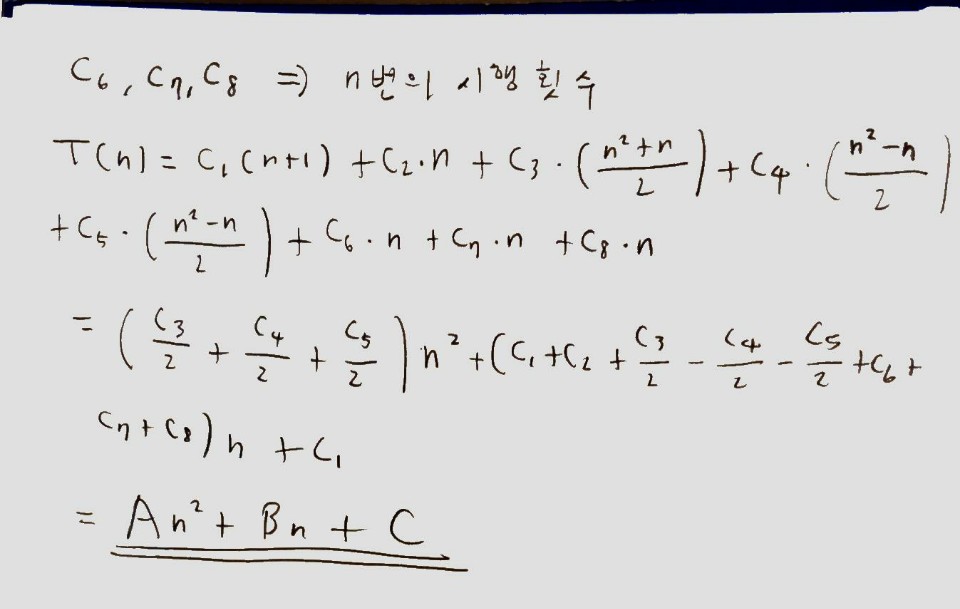
**알고리즘 과제3 2015003654 이호영**

**Selection Sorting의 Worst Case에서의 Time Complexity**





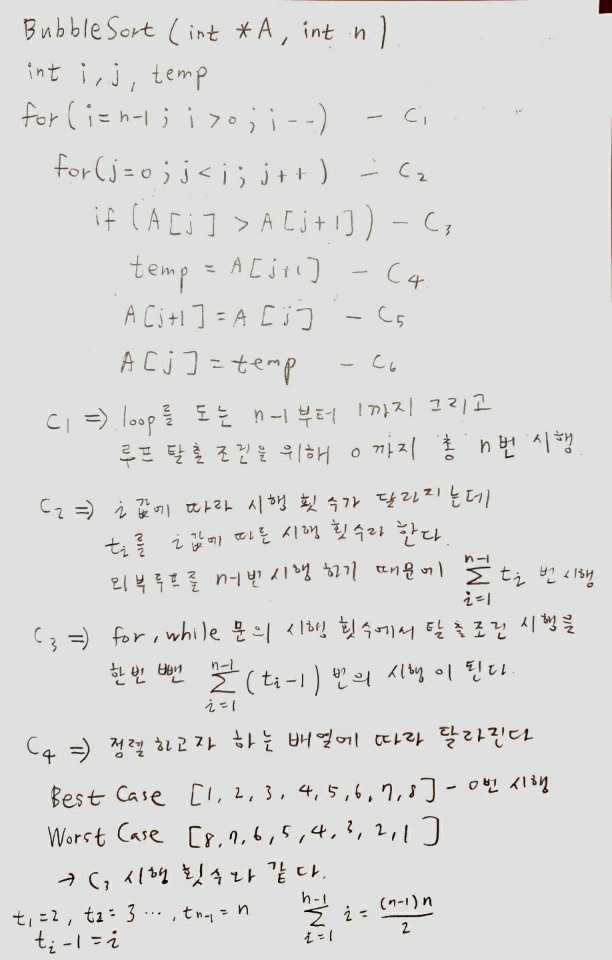
Selection sort에서는 n개의 요소를 가진 배열에서 최대값을 찾은 뒤 배열의 n-1번째 칸에 있는 값과 바꾼다. 따라서 n개의 요소들 중에 최대값을 찾아야한다. n개의 요소를 비교하는 것은 best case든 worst case든 n-1번의 비교를 하게 된다. 위 알고리즘에서는 i가 작아짐에 따라 최대값을 구해야 하는 요소의 개수들이 하나씩 줄게 되어서 결국엔 비교의 횟수는 총 1부터 n-1까지의 합이 된다. 의 비교로 Big O notation에서는 O(n^2)이 된다. 이것은 어떤 배열이 오는지에 따라 다 똑같은 결과이다. 재미있는 사실은 이미 정렬된 [1, 2, 3, 4, 5, 6, 7, 8]의 값이 반대로 정렬된 [8, 7, 6, 5, 4, 3, 2, 1] 보다 running time이 적다는 많다는 것이다. 위 알고리즘에서 max에 j를 할당하는 부분이 루프 두개 안에 있기 때문에 n^2번의 반복을 하게 되어서 외부 루프 안에 있어서 n번의 반복을 하는 Swap 과정보다 더 시간이 오래 걸린다는 것이다.

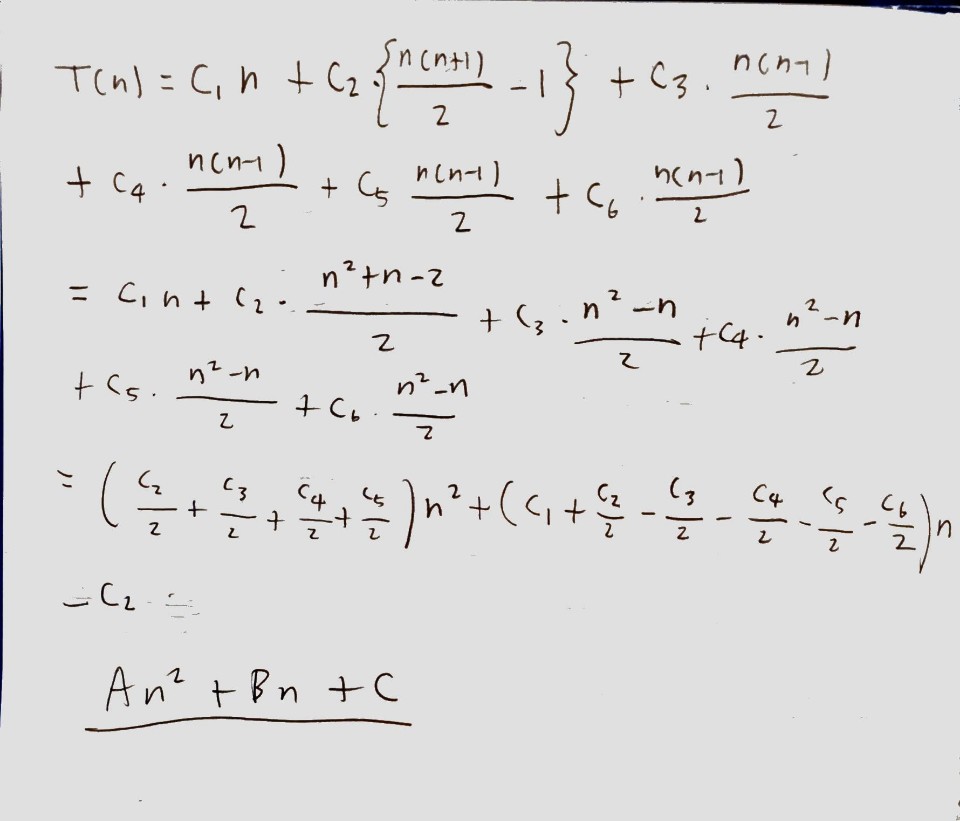
그래서 Worst case는 [1, 2, 3, 4, 5, 6, 7, 8]

이동하는 것에서는 어떤 배열이 오는지에 따라 수행 시간이 달라지는데 알고리즘에서 SWAP하는 마지막 세 줄에서 차이가 난다. 하지만 SWAP의 수행 시간은 상수의 시간이 걸리고 외부 루프가 n번 반복하기 때문에 n의 상수배를 가져서 Time complexity에 큰 영향을 주지 못한다.

여기서 또 재미있는 점은 [8, 7, 6, 5, 4, 3, 2, 1]이 제일 이동을 많이 할 것 같지만 순서를 조금 바꾼 [7, 8, 5, 6, 4, 3, 2, 1]의 이동 횟수가 더 많아 후자가 더 전자보다 running time이 길다.

**Bubble Sorting의 Worst Case에서의 Time Complexity**





Bubble Sort의 알고리즘은 위와 같은데 내부 루프에서 비교가 한 번 일어나게 되는데 첫 번째에는 루프를 n-1번 돌게 된다. 그리고 외부 루프의 i 값이 작아질수록 내부 루프 반복 횟수가 줄어들어서 결국엔 내부 루프의 비교를 한번만 하게 되어 전체적으로는 1부터 n-1까지의 합인 번의 비교를 하게 된다. 정렬이 되어있든 아니든 동일한 비교 횟수를 가지기에 Big O notation은 모두 O(n^2)으로 같다. Running time의 차이는 이동 횟수에 따라 달라지는데 [8, 7, 6, 5, 4, 3, 2, 1]이 [1, 2, 3, 4, 5, 6, 7, 8]보다 이동 횟수가 많다

참고 자료 : 수업 자료.