실험 PRJ-1 2주차 기본 테트리스 프로그램 예비보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**1. 목적**

2주차 실습에서 구현할 랭킹 시스템의 자료를 이해하고, 이를 구현하기 위한 자료구조를 2가지 이상 생각한다. 또한 생각한 자료구조를 통해 새로운 랭킹을 삽입, 삭제할 경우 필요한 시간, 공간 복잡도를 계산한다. 마지막으로 어떻게 정렬된 랭킹을 얻을 수 있을지 생각해본다.

**2. 랭킹 시스템을 위한 자료구조와 삽입·삭제에서의 시·공간 복잡도**

이번 실습에서는 createRankList, rank, newRank, writeRankFile의 총 4가지 함수를 작성해야 한다. 이번 실습에서는 자료구조의 자유도를 부여하기 때문에, 개인적으로 일반적인 list와 linked list, 이진 트리 세 가지를 고려사항으로 가졌다. 우선 처음으로 개인적인 구현 가능성에 대해서 판단했다. 사실 이진 트리 형태는 제작해본 적이 없기 때문에 어려울 것이라 생각했다. 따라서 일반적인 list와 linked list 두 가지가 고려대상이 되었다.

우선 일반적인 list는 binary search를 사용할 수 있다는 장점이 존재했다. Binary search를 통해, 데이터의 위치를 출력할 때나 새로운 데이터를 삽입할 때, 빠른 속도로 새로운 데이터의 위치를 찾는 것이 가능했다. 하지만 문제는 새로운 데이터를 삽입, 혹은 삭제할 때 새로운 데이터의 위치 뒤쪽의 데이터를 모두 한 칸씩 옮겨야 한다는 단점이 존재했다. 결과적으로 시간 복잡도는 O(log n+n) -> O(n) 형태가 된다. 여기에서 n은 자료구조의 원소 수이다.

반대로 linked list는 삽입, 삭제의 용이성에 관련된 장점이 존재했다. 삽입, 삭제를 진행할 때 새로운 데이터의 앞, 뒤 노드와의 관계만 정리해주면 쉽게 삽입, 삭제가 가능했다. 하지만 linked list는 binary search를 진행할 수 없다는 단점이 있다. Binary search를 적용하기 위해서는, list의 중간값이 무엇인지 파악해야 하는데, 중간값을 파악하기 위해서 linked list를 읽어야 되기 때문이다. 결과적으로 O(log n) 형태가 아닌, O(nlog n) 형태가 되어, binary search를 적용할 때, 오히려 linear search보다 비효율적인 형태가 만들어지게 된다. 결과적으로 삽입, 또는 삭제를 진행할 때의 시간 복잡도는 linear search 일 때 O(n)이 된다.

**3. 정렬된 랭킹을 얻는 방법**

정렬된 랭킹 또한 위의 내용과 비슷하게, linear 형태로 진행할 예정이다. 그 이유로는 정렬된 list에서 각 원소를 하나하나 출력해야 할 때, 가장 간단하면서, 최선의 방법이 linear이기 때문이다.

따라서 일반적인 리스트에서는 list[X]~list[Y] 를 출력하는 형태로 진행하며, linked list에서는 head부터 시작해서 X를 구해내며, 이후 Y까지 출력해내는 형식으로 진행할 수 있다.