9주차 예비보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

1. 2주차 실습에 구현하는 랭킹 시스템에 대한 자료를 읽어보고, 이를 구현하기 위한

자료구조를 2가지 이상 생각한다.

랭킹 시스템에는 사용자의 이름과 점수가 담겨 있는 자료구조가 필요하다. 배열(array)와 연결 리스트(linked list)를 생각하게 되었다. linked list의 경우 struct { name; score; link}를 사용하면 되고, 배열의 경우는 이 link를 제외한 구조체를 통해서 배열을 만들면 된다.

2. 생각한 각 자료구조에 대해서 새로운 랭킹을 삽입 및 삭제를 구현하기 위한 pseudo

code를 작성하고, 시간 및 공간 복잡도를 계산한다.

|  |
| --- |
| Insert( insertnode)  If (ranking)  For(;node;node=node->link)  If (insertnode->score 와 node->score 비교)  //적절한 부분에 삽입  Else  insertnode; |

|  |
| --- |
| Delete ( int )  Int count = 0;  While(node->link)  If(count == int)  //이 경우 입력한 값과 ranking 순위가 동일할 경우 그 ranking을 삭제한다.  Node = node -> link;  Count ++; |

linked list의 경우 0번째(처음 사용자)가 입력될 때 첫번째 node가 생성이 된다. n번째가 입력되는 상황을 가정한다면(이때 node들은 내림차순으로 정의되어 있다고 가정하자) n번째 node는 첫번째 node부터 link를 따라 읽다가, n번째 node의 score보다 작은 노드가 있다면, 이 노드를 A라하자. A 이전 노드 -> link = n번째 node , n번째 node -> link = A를 통해 insert하게 된다. 이렇게 insert하는 과정은 첫번째 node부터 순차적으로 읽어서 삽입하는 경우인데, 최악의 경우 전체의 노드 개수 만큼 진행을 해야 한다. 따라서 시간 복잡도는 O(전체 노드의 개수)이다. 공간 복잡도의 경우에는 전체의 노드 개수만큼 데이터를 필요로 하기 때문에, O(전체 노드의 개수이다)

linked list의 경우 삭제를 설명하자면, 입력받은 숫자에 해당하는 랭킹을 지워야 한다. 이 또한 node를 순차적으로 읽는 과정이 필요해 시간 복잡도는 O(전체 노드의 개수)이다. 위와 마찬가지로 공간 복잡도의 경우 linked list를 전체 노드의 개수만큼 가지고 있기 때문에 O(전체 노드의 개수)이다.

배열의 경우 insert의 기능이 조금 더 까다롭다.

배열의 경우, 배열에 값을 집어넣기 전 배열의 모든 요소들을 이 값과 비교하여 내림차순으로 넣는 과정이 필요하다. 만약 사이에 집어넣어야 하는 경우 배열을 한 칸씩 뒤로 밀고 새로운 값을 그 칸에 저장하면 된다. 이렇게 delete할 때에는 무조건 배열의 크기가 사용자가 저장하는 횟수보다 크거나 같아야 한다.

만약 배열의 크기가 ARRAY\_SIZE라면, 공간 복잡도는 이 배열의 크기가 변하지 않으니 O(ARRAY\_SIZE)이다. 시간 복잡도는 예를 들어 저장해야 할 곳은 i번째라고 히고, 현재 저장된 값들의 개수를 j라고 할 때 시간 복잡도는 미는 만큼 계산하게 되어 O(I + j) 이다. i번째까지 읽고, j만큼 밀기 때문이다.

삭제의 경우. 랭킹의 index를 활용해 쉽게 delete할 수 있다. 따라서 시간 복잡도는 O(1)이고, 공간 복잡도는 배열의 크기인 O(ARRAY\_SIZE)이다.

3. 생각한 각 자료구조에서 사용자가 부분적으로 확인하길 원하는 정렬된 랭킹(x~y위, x≤y, x, y는 정수)의 정보를 얻는 방법을 간략히 요약해서 pseudo code로 작성하고, 시간 및 공간 복잡도를 계산한다.

linked list의 경우 일전에 설명한 delete 함수의 pseudo code를 살펴보면, count라는 변수가 있는데, 이 변수를 이용해서 x와 count의 값이 같아지면 y값과 같아질 때까지 ranking을 출력하면 된다. 이 시간 복잡도의 경우는 y번째 node까지 읽으면 되기 때문에 O(y)이고, 공간 복잡도는 기존에 있는 전체 노드의 개수를 가지기 때문에 공간 복잡도는 O(전체 노드의 개수)이다.

array의 경우. delete와 마찬가지로 배열의 index값 곧 ranking을 의미하기 때문에 입력된 x부터 y까지의 데이터를 읽으면 된다. 따라서 시간 복잡도는 O(y-x)이고, 위와 마찬가지로 이미 고정된 배열의 크기인 ARRAY\_SIZE가 있어, 공간 복잡도는 O(ARRAY\_SIZE)이다.

마지막으로 두 자료구조는 무엇이 항상 좋다고 말할 수 없다. linked list의 경우 insert에서 array처럼 값을 한 칸 씩 밀지 않아도 되며, 저장하는 횟수가 제한되어 있지 않다는 점에서 용이하지만, array는 정렬된 인덱스에서 ranking을 추출하기 쉽다는 장점이 있다. 하지만, linked list를 활용해 실습을 진행할 예정이다.