**7주차 Tetris-2 예비보고서**

전공: 수학과 학년: 4 학번: 20161255 이름: 장원태

1. **2주차 실습에 구현하는 랭킹 시스템에 대한 자료를 읽어보고, 이를 구현하기 위한 다양한 자료구조를 2가지 이상 생각한다.**

다음의 두 가지 자료구조를 생각했다.

1. 배열

배열을 2행 n열 표라고 생각하면, 두 행에 이름과 점수를 저장하는 형태로 자료구조를 구축할 수 있다. 이는 이름과 점수를 저장하는 구조체를 선언한 뒤, 그 구조체 형식을 가지는 배열을 만들 수 있다. 이 경우, 파일에 포함되어 있을 랭킹 정보의 수 + 1만큼의 크기를 가지는 배열을 만들고, 거기 있는 모든 정보를 저장한 뒤 다음 정보를 저장할 수 있다. 단, 점수 순으로 배열해야 하므로 이 과정에서 순서를 변경하는 과정이 필요하다.

1. 링크드 리스트(linked list)

두 가지 정보와, 다음 노드를 가리킬 포인터를 포함하는 ‘노드(node)’를 정의하여 이들을 연결하는 형태로 자료구조를 구축할 수 있다. 노드 포인터를 정의한 뒤, 이름과 점수, 다음 노드에 대한 포인터를 저장할 노드를 구조체로서 타입 정의(typedef)할 수 있다. 이 경우, 파일의 정보들을 노드를 이용하여 모두 저장하되, 점수가 큰 순서대로 저장되게 해야 하며, 추가할 정보 역시 크기에 맞게 저장되어야 한다.

1. **생각한 각 자료구조에 대해서 새로운 랭킹을 삽입 및 삭제하기 위해 필요한 시간 및 공간 복잡도를 계산한다.**
2. 배열

새로운 랭킹을 삽입하는 것에 대한 pseudo code는 다음과 같다.

|  |
| --- |
| For (i=0; rank[i].score > score; i++);  For (j=n-1; j>i; j--) 다음 인덱스로 데이터 이동;  현재 i 인덱스에 데이터 저장; |

두 번의 for문을 거치되 각각 단일 for문으로 작용하므로, 시간 복잡도는 배열의 크기인 n+1에 의해서만 영향을 받는다. 따라서, 시간 복잡도는 O(n)이 된다. 공간 복잡도의 경우, 크기가 n+1인 배열을 다루므로 O(n)이 된다.

이어서 랭킹 삭제의 pseudo code는 다음과 같다.

|  |
| --- |
| For (i=삭제할 인덱스; i<n-1; i++) 다음 인덱스의 정보 끌어옴;  삭제할 인덱스에 대한 배열을 null로 설정; |

이 과정의 경우 삭제할 인덱스와 n-1의 차이만큼 for문을 진행하므로, 시간 복잡도는 O(n-(삭제할 인덱스))이다. 공간 복잡도의 경우, 크기가 n인 배열을 다루므로 O(n)이 된다.

1. 링크드 리스트

랭킹을 삽입하는 코드에 대한 pseudo code는 다음과 같다.

|  |
| --- |
| Node \*new;  new 노드에 이름과 점수 저장;  node \*prev=head; node \*temp=head->link;  while (temp->link != NULL || temp->score < score){  prev = temp; temp = temp->link; // 노드 한칸씩 이동  }  New->link = prev->link;  Prev->link = new; |

시간 복잡도의 경우, while문으로 진행하는 것이므로 O(1)이 된다. 공간 복잡도의 경우, n+1개의 노드를 다루므로 O(n)이 된다.

랭킹을 삭제하는 코드에 대한 pseudo code는 다음과 같다.

|  |
| --- |
| Node \*temp=head;  Node \*prev;  While (temp->score != score || temp != NULL){  Prev = temp; Temp = temp->link;  }  Prev->link = temp->link;  Free(temp); |

시간 복잡도의 경우, while문을 진행해야 하므로 O(1)이 된다. 공간 복잡도의 경우, 크기가 n인 링크드 리스트를 다뤄야 하므로 O(n)이 된다.

1. **생각한 각 자료구조에 대해서 어떻게 정렬된 랭킹(x~y위, x≤y, x, y는 정수)을 얻을 수 있을지에 대해서 생각해보고, 그 방법에 대해서 기술하시오.**
2. 배열

배열은 앞선 삽입과 삭제 과정에서 이미 정렬되어 있으므로, x위의 데이터는 배열의 x-1 인덱스에 있을 것이다. 따라서, 배열의 x-1 인덱스부터 y-1 인덱스까지의 데이터를 얻으면 된다.

1. 링크드 리스트

링크드 리스트 역시 앞선 삽입과 삭제 과정에서 이미 정렬되어 있으며, 서로 link를 통해 연결되어 있는 상태이다. 배열과 비슷하게, x위의 데이터는 x번째 노드에 있을 것이다. 따라서, 헤드 노드로부터 (x-1)회만큼 링크를 타고 이동한 뒤, 그 노드부터 (y-x)회만큼 링크를 타며 하나씩 데이터를 얻으면 된다.