9주차 예비 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20221559 이름: 박준우

1. 랭킹 시스템을 구현하기 위한 자료구조를 2가지 이상 생각한다.

자료가 정렬되어 있어야 하므로, 트리 구조를 생각해 봤으나, 값으로 찾는게 아닌, 순서로 찾는 상황에서는 linked list나 그냥 배열을 사용하는 것이 가장 간단하고, 효율적일 것이라 생각했다.

배열의 경우 사용할 구조체의 형태

Struct element{

Int score;

String username;

}

Linked list의 경우

Struct Node{

Element e;

Node \*next;

}

배열의 크기를 어떻게 정의할 지, 정해진 크기를 넘었을 때의 동작 등의 고민도 필요해 보인다.

2 - 1. linked list의 경우

1) 삽입

점수를 기준, 내림차순 정렬 된 상태로 linked list 형태로 저장되어 있으면, 삽입하고자 하는 값이 나올 때 까지 (NULL을 만나도 중단) while loop를 돌린 후 삽입하면 된다.

insertRank(int score, string username)

make new node(score, username)

while(curr->next != NULL && curr->next->score > new->score) curr = curr->next;

new->next = curr->next;

curr->next = new->next;

시간 복잡도는 존재하는 랭크의 개수에 비례할 것이다. O(rankcount)

삽입 자체의 공간 복잡도는 O(1), 자료형 전체는 O(rankcount)

2) 삭제

이름으로 찾아서 삭제하는 것이 아닌, 몇 번째 순서의 아이템을 삭제하는 것이므로 간단하다.

Deleterank(int input)

if ( input > rankcount ) fail;

temp = 0

while(temp < input - 1) :

(삭제할 노드의 이전노드까지 이동, 맨 앞 노드이면 따로 정의)

curr = curr->next;

tempnode = curr->next;

curr->next = tempnode->next;

delete(tempnode);

입력받은 번호의 node까지 순차 탐색 해야 하므로, 최악의 경우를 고려했을 때 O(rankcount)의 시간복잡도를 가지고, 삭제 자체의 공간복잡도는 O(1)이다.

2-2. 배열의 경우

1) 삽입

삽입할 위치를 찾고 (정렬 되어 있으므로 이분 탐색 등을 이용하면 log 시간 내 가능), 해당 위치부터 뒤의 자료들을 한 칸씩 밀어야 함.

Insertrank(int score, string username)

Rankcount++;

for i = 0 to rankcount – 1

if rank[i].score >= score

prev = rank[i]

set rank[i] newrank

for j = i + 1 to rankcount

temp = rank[j]

rank[j] = prev

prev = temp

break;

삽입 시 찾는 것에서 rankcount만큼, 뒤의 배열 전체를 미는 것에서 rankcount만큼씩 해서, O(rankcount^2)의 시간복잡도를 가지고, 삽입 자체의 공간 복잡도는 선언된 배열만 사용하므로, O(1)이 된다.

2) 삭제

삭제할 index가 주어지므로, 해당 인덱스 이후를 한칸씩 앞으로 당기면 된다.

Deleterank(int idx)

If idx > rankcount : fail;

for i = idx To rankcount – 1;

rank[i] = rank[i+1]

시간 복잡도는 뒤의 배열을 한 칸씩 당기는 for loop의 횟수가 rankcount에 bound 되므로, O(rankcount)이다. 공간 복잡도는 기존 배열을 그대로 사용하므로 O(1)이다.