실험 PRJ-1 2주차 랭킹 시스템 예비보고서

전공: 국제한국학 학년: 4 학번: 20181202 이름 : 김수미

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

1) Linked List (연결리스트)

데이터 필드(필요한 값을 저장)와 포인터 필드(다음 노드를 가리킨다)로 이루어진 구조체 노드를 여러 개 연결해서 리스트 형태로 사용하는 자료구조이다. 게임 결과에 따른 플레이어의 점수를 오름차순 또는 내림차순으로 정렬해 저장할 수 있으며, 새로운 점수를 저장할 때에도 순서에 알맞은 자리를 찾아 저장할 수 있다. 리스트의 길이를 처음부터 정해놓지 않아도 된다는 점과 기존에 있던 자료를 삭제해도 공간의 낭비가 발생하지 않는다는 점에서 사람들이 플레이하면 할수록 변하는 게임의 점수 리스트를 관리하기에 효율적인 자료구조이다.

2) Array (배열)

만들 수 있는 가장 큰 배열을 생성한 다음 해당 배열 안에서 랭킹 데이터를 관리하는 방법이다. 인덱스번호로 Access가 가능해 어떤 등수에 누가 몇 점 인지 아주 쉽게 확인할 수 있으나, 하지만 할당 후 사용하지 않는 메모리가 많을 수 있고, 랭킹 값을 삭제하는 경우 배열의 요소들을 한 칸 씩 앞으로 당겨서 재배치해야 하기 때문에 시간/공간적으로 연결리스트보다 효율적이지 못하다.

2. 생각한 각 자료구조에 대해서 새로운 랭킹의 삽입 및 삭제를 구현하기 위한 pseudo code를 작성하고, 시간 및 공간 복잡도를 계산한다

1) Linked List

① 새로운 랭킹(점수)의 삽입

가장 첫번째로 등록되는 점수 = 시작 노드가 된다.

```
while (다음 노드가 NULL이 아니면) {// 마지막 노드(더이상 연결된 노드가 없는) 도착시 루프 중단
        // 시작 노드부터 차례대로 훑으며 새 점수 노드가 들어갈 자리를 찾는다
        현재 노드 바로 다음에 연결된 데이터 값을 확인한다
        if (지금 삽입하고자 하는 점수가 현재 노드와 바로 다음 노드의 사이 값) {
                // 위 조건문에서의 '바로 다음 노드'는 NULL일 수도 있다.
                새로운 노드를 생성한다
                새로운 노드의 데이터필드에 점수를 저장
                현재 노드의 바로 다음에 방금 새로 만든 노드 연결
                원래 현재 노드 바로 다음이었던 노드는(없을 수도 있음), 새로 만든 노드 바로 뒤에 연결
                노드 총 개수를 나타내는 변수 +1 } }
```

〉〉 시/공간 복잡도 : O(랭킹 개수), O(랭킹 개수)

전공: 국제한국학 학년: 4 학번: 20181202 이름 : 김수미

② 기존 랭킹 삭제

index = 1; / 예를 들어 지우고자 하는 랭킹 = N 일 때

while (다음 노드가 NULL이 아니면) {

// 가장 처음 노드(1위)부터 훑어가며 지우고자 하는 순서(랭킹)의 노드를 찾는다

if(index == N-1) { // 지우고자 하는 랭킹을 찾은 경우

N-1번 노드는 따로 저장해두고 N번 노드로 이동해 메모리 할당 해제 해주기

N-1번째 노드가 가리키는 노드를 원래 N+1 번째 였던 노드로 바꿔준다

(N+1 번째는 NULL 일 수도 있다) } index++; }

〉〉 시/공간 복잡도 : O(N), O(랭킹 개수)

2) Array (배열)

① 새로운 랭킹(점수)의 삽입

현재 랭킹에 존재하는 점수의 개수 = X개, 현재 삽입하고자 하는 점수가 N점일 경우

배열 rank 생성(플레이어 이름(string)과 점수(int)로 구성된 구조체 배열)(크기는 아주 크게 정적 할당)

가장 마지막(rank[X])에 삽입될 수 있는지 먼저 체크. 가능하면 아래 과정은 수행하지 않아도 된다.

```
for( int i =  0 ; i 〈 X ; i++ ) {
        // 가장 처음 노드(1위)부터 훑어가며 새 점수를 삽입할 수 있는 자리를 찾는다
        if( rank[i] 〉 N && rank[i+1] 〈 N ) {
                // rank[i+1] ~ rank[마지막] 까지 한 칸씩 뒤로 밀어낸다
                // 가장 마지막에 수행되는 것 : rank[i+2] = rank[i+1]
                for ( int j = X ; j 〉 i+1 ; j-- ) rank[j] = rank[j-1]
                rank [i+1] = N;  X++; // 새 점수 삽입
                break; } }
```

〉〉 시/공간 복잡도 : O(X), O(배열의 최대 크기)

② 기존 랭킹 삭제

현재 랭킹에 존재하는 점수의 개수 = X개, 지우고자 하는 랭킹 = N 일 때

→ 배열의 N+1번째 요소 ~ 마지막 요소들을 한칸씩 앞으로 당겨주면 된다.

```
for( int i = N-1 ; i 〈 X-1 ; i++ ) rank[N] = rank[N+1];
X--;
```

〉〉 시/공간 복잡도 : O(X-N), O(배열의 최대 크기)

실험 PRJ-1 2주차 랭킹 시스템 예비보고서

전공: 국제한국학 학년: 4 학번: 20181202 이름 : 김수미

3. 생각한 각 자료구조에서 사용자가 부분적으로 확인하길 원하는 정렬된 랭킹(X~Y, X≤Y / X, Y는 정수)의 정보를 얻는 방법을 간략히 요약해서 pseudo code로 작성하고, 시간 및 공간 복잡도를 계산한다.

1) Linked List (연결리스트)

먼저 처음 노드부터 차례대로 훑어가며 X번째 노드를 찾고, Y번째가 될 때 까지 이동하며 각 노드의 정보를 출력해주면 된다.

```
index = 1;
while(1) {
        첫번째 노드부터 차례대로 이동
        if (index == X) {
                while(index <= Y) 노드의 플레이어와 점수 정보 출력(또는 return)
                break; // 랭킹 X~Y의 출력이 완료되었으므로 while 루프 종료
        } i++; }
```

〉〉 시/공간 복잡도 : O(Y), O(랭킹 개수)

2) Array (배열)

플레이어 이름과 점수를 저장하는 구조체로 이루어진 배열 rank가 있을 때

배열의 인덱스 X-1 부터 Y-1 까지의 요소를 출력(또는 return)해주기만 하면 된다.

```
for ( int i = X-1 ; i < Y ; i++) {
        rank[i]에 저장된 플레이어 이름과 점수 출력 }
```

〉〉 시/공간 복잡도 : O(Y-X+1), O(배열의 최대 크기)