7주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181294 이름: 임승섭

**1.**

1 - 1 랭킹 시스템의 자료구조

랭킹 시스템을 나타내는 자료구조로는 LinkedList를 이용했다. 각 노드에는 점수를 나타내는 변수 data 와 이름을 나타내는 변수 name이 있고, 다음 노드를 가리키는 포인터 next가 있다. 링크드리스트의 가장 앞 노드로 HEAD를 미리 설정하여 새로 생기는 노드들을 그 뒤에 이어 붙였다. 뒤에 붙어있는 노드들의 개수, 즉 랭킹의 개수를 나타내는 변수 rank\_num도 미리 선언하였다.

1 - 2 createRankList() : rank.txt 파일을 읽어들여, linkedlist를 만든다.

(1). 파일을 연다.

(2). 변수들을 선언한다.

num : 저장된 랭크의 개수

name : 새로운 노드에 저장될 이름

data : 새로운 노드에 저장될 점수

curr : 새로운 노드의 직전 노드. 처음은 HEAD를 가리키게 한다.

(3). 파일의 첫 줄에 써있는 랭킹의 개수를 읽는다. (num)

(4). 개수가 0이 아니면 그 개수만큼 for loop을 진행한다.

(5). 각 loop마다 한 줄씩 읽어들이며, 새로운 node를 생성하여 node의 이름과 점수로 설정한다. 새로운 node의 next는 일단 NULL로 선언한다. curr 의 next를 새로운 node로 해주고, curr은 그 새로운 노드를 가리키게 한다. 이를 통해 curr은 loop이 돌 때마다 한 칸씩 이동한다.

(6). 파일을 닫는다.

(\*) 시간복잡도&공간복잡도

rank\_num의 개수만큼의 loop이 돌면서 새로운 노드를 뒤에 붙인다. loop은 이곳에밖에 없고, 나머지는 O(1)의 시간복잡도를 가지므로 함수의 시간복잡도는 O(rank\_num)이라고 할 수 있다.

노드들은 총 rank\_num개가 있고, 각 노드마다 int data, char name[NAEMLEN+1]의 정보가 존재한다. 이름을 저장하는 문자열이 NAMELEN + 1만큼의 공간을 차지하므로, 공간복잡도는 O(rank\_num \* NAMELEN)이라고 할 수 있다.

1 - 3. rank() : rank 메뉴를 열어, 원하는 명령에 대한 출력을 하도록 한다.

(1). createRankList() 함수를 실행시켜, rank.txt 파일에 있는 정보를 읽는다.

(2). 변수들을 선언한다.

X, Y : 1번 명령에서 랭크의 범위를 나타내는 변수로 사용. 일단 -1로 초기화한다.

ch : 명령 번호를 저장할 문자. 1, 2, 3 중 하나로 읽어올 예정이다.

(3). clear() 화면을 깨끗하게 지운다.

(4). 1, 2, 3번 메뉴를 출력한다.

(5). 3개의 메뉴 중 어떤 것을 선택했는지 ch에 저장한다. (wgetch 이용)

(6 - 1). 1번 메뉴를 선택한 경우 : 사용자가 원하는 범위 내의 랭크들을 출력한다.

- X와 Y를 읽어온다.

- 만약, X와 Y가 둘 다 제대로 읽었는데 X가 Y보다 크면 에러 메세지를 출력한다. 또, X가 rank의 개수보다 클 때도 에러 메세지를 출력한다.

- 정상적으로 출력 가능하면, 총 4가지 케이스로 나누어서 알고리즘을 짰다. X, Y를 둘 다 입력하지 않은 경우, curr를 이용하여 처음 node부터 맨 마지막 node까지 while loop을 통해 이동하면서 각 노드의 name과 data를 출력하였다. X는 입력하지 않고, Y는 입력한 경우, 마찬가지로 curr을 이용하여 처음 node부터 Y번째 node까지 while loop을 통해 이동하면서 각 노드의 name과 data를 출력하였다. Y번째에서 멈출 때에는 count 변수를 이용해 초기값 0에서 loop을 돌 때마다 1씩 추가되는 count가 Y와 같아질 때 loop을 멈추도록 하였다. X는 입력하고, Y는 입력하지 않은 경우, 우선 위와 마찬가지로 count 변수를 이용하여 X번째 node 까지 이동하였다. 이 과정에서는 출력하지 않는다. 이후 X번째부터 마지막 노드까지 curr를 이동시키며 노드의 정보를 출력한다. X, Y를 둘 다 입력한 경우, 위와 마찬가지로 X번째 노드까지 이동하고, X번째 노드부터 Y번째 노드까지 curr를 이동시키며 정보를 출력시켰다.

(6 - 2). 2번 메뉴를 선택한 경우 : 사용자가 입력한 문자열과 같은 이름의 랭크들을 출력한다.

- 사용자가 입력한 이름 (str)을 읽어온다.

- curr2 노드를 정보를 담고 있는 첫 번째 노드로 선언한다. (HEAD->next)

- while loop을 이용해 curr2 노드를 맨 마지막 노드까지 이동시키며, curr2 노드의 이름과 str이 같으면 curr2의 이름과 점수를 출력하도록 한다. 이 때마다 check 변수(초기값 0) 을 1씩 추가해준다.

- loop이 끝난 후 check이 여전히 0이라면, 같은 이름이 하나도 없다는 뜻이기 때문에 search failure message를 출력한다.

(6 - 3). 3번 메뉴를 선택한 경우 : 사용자가 입력한 순위의 랭크 정보를 지운다.

- 사용자가 입력한 순위(num)을 읽어온다.

- 만약 num이 랭크들의 개수(rank\_num)보다 크거나 0보다 작다면 찾을 수가 없기 때문에 에러 메세지를 출력한다.

- 그렇지 않으면, 우선 prev3 노드를 새로 선언하여 지우고 싶어하는 노드 직전 노드를 가리키도록 한다. 처음은 HEAD로 선언하고 for loop 을 이용해 이동하였다. prev3 노드의 다음 노드를 curr3 노드로 해준다. curr3 노드를 없앨 예정이기 때문에 prev3노드의 next를 curr3의 next로 바꿔누고 curr3의 메모리를 해제한다. 이 과정이 끝나면 rank\_num을 1 줄여준다.

- 삭제가 완료되었다는 메세지를 출력하고, 새 정보를 rank.txt 파일에 적어준다. writeRankFile() 함수를 이용하였다.

(\*) 시간복잡도&공간복잡도

ch가 1, 2, 3일 때를 각각 분석한다. 먼저 ch가 1일 때, worst case는 X와 Y를 모두 입력하지 않았을 때이다. 리스트의 모든 노드의 정보를 출력해야 하기 때문에 loop이 가장 오래 걸리게 된다. 이 때 count가 0부터 rank\_num까지 1씩 증가하기 때문에 시간복잡도는 O(rank\_num)이다. 다음 ch가 2일 때, 이때는 무조건 curr2 노드가 HEAD부터 마지막 노드까지 탐색하게 되고, 그 개수는 rank\_num이다. 따라서 시간복잡도는 O(rank\_num)이다. 마지막으로 ch가 3일 때, worst case는 맨 마지막 노드를 삭제하려고 할 때이다. 그러면 for loop으로 i가 0부터 rank\_num까지 가게 된다. 따라서 시간복잡도는 O(rank\_num)이다. 3가지 케이스에서 모두 시간복잡도가 갖고, 3개로 나뉘기 전의 명령들은 O(1)이기 때문에 함수의 시간복잡도는 O(rank\_num)이다.

새로 추가되는 공간 메모리는 없기 때문에 공간복잡도는 O(1)이다.

1 - 4. writeRankFile() : 현재 linkedlist의 정보를 rank.txt 파일에 업데이트시킨다.

(1). 파일을 연다.

(2). 파일 첫 줄에 적혀있는 랭크의 개수를 읽어와서, 현재와 같으면 따로 업데이트할 필요가 없다.

(3). curr 를 HEAD->next로 선언하여 정보가 들어있는 첫 번째 노드로 선언한다.

(4). curr를 맨 마지막 노드까지 이동시키며 노드의 이름과 점수를 출력한다.

(5). 파일을 닫는다.

(\*) 시간복잡도&공간복잡도

노드의 정보를 파일에 출력하는 과정에서 while loop이 돈다. 이 때 모든 노드의 정보를 출력하기 때문에 시간복잡도는 O(rank\_num)이다.

새로 추가되는 공간 메모리는 없기 때문에 공간복잡도는 O(1)이다.

1 - 5. newRank(int score) : gameover 시 그때까지의 score를 사용자가 입력한 이름과 함께 새로 저장한다.

(1). createRankList()함수를 실행시켜서 현재 랭킹 시스템이 있는 linkedlist 를 만든다.

(2). 사용자가 입력한 이름 str을 입력받는다.

(3). 새로운 노드를 생성해서 이름은 str, 점수는 게임 종료 시의 점수로 저장한다.

(4). 새로운 노드의 위치를 찾기 위한 노드로 prev와 curr를 이용한다. 처음에 prev는 HEAD, curr은 그 다음 노드로 선언한다.

(5). 만약 curr이 NULL이라면 현재 저장된 랭크가 없다는 뜻이기 때문에 맨 처음 노드를 newnode로 해준다.

(6). 그렇지 않으면 curr이 NULL이 될 때까지 while loop을 돈다. 그 과정에서 만약 newnode의 점수가 curr의 점수보다 큰 경우가 생기면, prev와 curr 사이에 newnode를 저장하고 loop을 끝낸다.

(7). loop이 끝났는데 curr이 NULL이라면, 맨 끝까지 newnode의 점수보다 작은 노드가 없다는 뜻이기 때문에 맨 마지막 노드로 newnode를 저장한다.

(8). rank\_num을 1 추가해주고, 새로운 정보를 rank.txt 파일에 저장한다. writeRankfile() 함수를 이용한다.

(\*) 시간복잡도&공간복잡도

점수를 추가할 때, 내림차순으로 알맞은 위치에 노드를 추가하기 위해 while loop을 돈다. worst case는 현재까지 있는 모든 점수보다 낮은 점수를 저장해야 할 때 이다. 첫 번째부터 마지막 노드까지 모든 노드의 점수를 비교해야 하기 때문에 시간복잡도는 O(rank\_num)이다.

새로 추가한 노드는 결국 하나밖에 없기 때문에 공간복잡도는 문자열의 길이만 고려한다. 따라서 공간복잡도는 O(NAMELEN)이다.

**2.**

Linkedlist 노드를 선언할 때, 전역변수로 생성하려고 하면 오류가 나온다는 점을 알게 되었다. 원래는 struct Node를 만들면서 HEAD 노드를 바로 만들고, 메모리 할당 후 HEAD->data 가 rank\_num을 의미하도록 하려고 했다. HEAD 노드는 말 그대로 노드들의 맨 앞을 의미하기 때문에 data 값이 없기 때문이다. 이렇게 하면 rank\_num이라는 새로운 변수도 만들 필요가 없다고 생각했다. 하지만 “error : initializer element is not constant” 에러가 계속 발생하였고, HEAD 노드의 메모리 할당은 createRankList() 함수 내에서 하니까 해결되었다. 어쩔 수 없이 int rank\_num 변수도 새로 만들었다.

ncurses library에 대한 기초 사용법을 익힐 수 있었다. 이번 프로젝트 전에는 한 번도 써보지 못한 방법이라서 많이 헤멨지만, 강의자료 pdf를 읽어가며 printw, getch, clear 등의 함수를 사용하다 보니 현재는 자연스럽게 사용할 수 있게 되었다.