

1. [실습] 선택한 자료구조(링크드 리스트)의 효율성 및 알고리즘

(1) Game over 이후, 새로운 랭킹 정보(사용자 이름과 점수)가 등록될 때 시간 및 공간 복잡도

play() 함수에서 Game Over인 경우 newRank()를 호출해 새로운 랭킹 정보를 등록한다.

```
void play(){
    . . .
    printf("GameOver!!");
    refresh();
    getch();
    newRank(score);
}
```

newRank() 함수에서는 점수를 parameter로 받고, 사용자에게 이름을 입력받아서 새로운 랭킹 정보를 등록한다. 먼저 해당 이름과 점수를 가진 새 노드를 생성한다. 기존 랭킹 리스트가 비어있는 경우, head 노드로 해당 노드를 설정하고, 비어있지 않은 경우 점수를 비교해가며 새 노드가 삽입되어야 할 위치를 찾는다. 새 노드가 head 노드로 삽입되어야 하는 경우와 그렇지 않은 경우로 구분하여 새 노드를 삽입한다. 삽입 이후에는 리스트의 전체 노드 개수(score_number)를 증가시키고, wrtieRankFile()을 호출한다.

```
void newRank(int score){
    char str[NAMELEN+1];
    int i, j;
    clear();

    Node* new_node;

    echo(); //turn on
    printf("your name: ");
    scanw("%s", str);
    noecho(); //turn off

    new_node = createNode(str, score);

    if (head == NULL) { //empty list
        head = new_node;
    }
    else {
        Node* curr = head;
        Node* prev = NULL;

        while (curr != NULL) { //find the position where new > curr
            if (new_node->score < curr->score) {
                prev = curr;
                curr = curr->next;
            }
            else
                break;
        }
    }
}
```

```

    }
    if (prev == NULL) { //new_node should be the head
        head = new_node;
        new_node->next = curr;
    }
    else { //new_node should be inserted after curr
        new_node->next = prev->next;
        prev->next = new_node;
    }
}

score_number++;
writeRankFile();
}

```

따라서 새로운 랭킹 정보(사용자 이름과 점수)가 등록될 때 시간 복잡도는 $O(n)$ 이고, 공간 복잡도는 $O(n)$ 이다. (n 은 노드의 전체 개수)

(2) 원하는 랭킹 범위를 입력 받은 뒤, 랭킹 추출 과정에서 탐색 및 랭킹 추출의 시간 및 공간 복잡도

rank() 함수에서 1을 입력받은 경우 rankXY() 함수를 호출해 해당 기능을 수행한다.

```

void rank(){
    int ch;
    clear();

    //print rank menu
    printf("1. list ranks from X to Y\n");
    printf("2. list ranks by a specific name\n");
    printf("3. delete a specific rank\n");

    ch = wgetch(stdscr);

    if (ch == '1') {
        rankXY();
    }

    else if ( ch == '2') { //week7 hw
        searchName();
    }

    else if ( ch == '3') { //week7 hw
        deleteRank();
    }

    getch();
}

```

rankXY() 함수에서는 원하는 랭킹 범위를 입력 받은 뒤, 해당 범위의 랭킹을 리스트를 순회하며 데이터를 추출해 출력한다. X를 입력받지 않은 경우 1위부터 출력하고, Y를 입력받지 않은 경우 마지막 순위까지 출력한다. 만약 X가 Y보다 크게 입력받은 경우 실패 문구를 출력하며 종료한다.

```
void rankXY() {
    int X = 1, Y = score_number; //set default
    echo(); //turn on
    printf("X: ");
    scanf("%d", &X);
    printf("Y: ");
    scanf("%d", &Y);
    noecho(); //turn off

    printf("      name      | score  \n");
    printf("-----\n");

    if (X > Y) { //range failure
        printf("\nsearch failure : no rank in the list");
        return;
    }

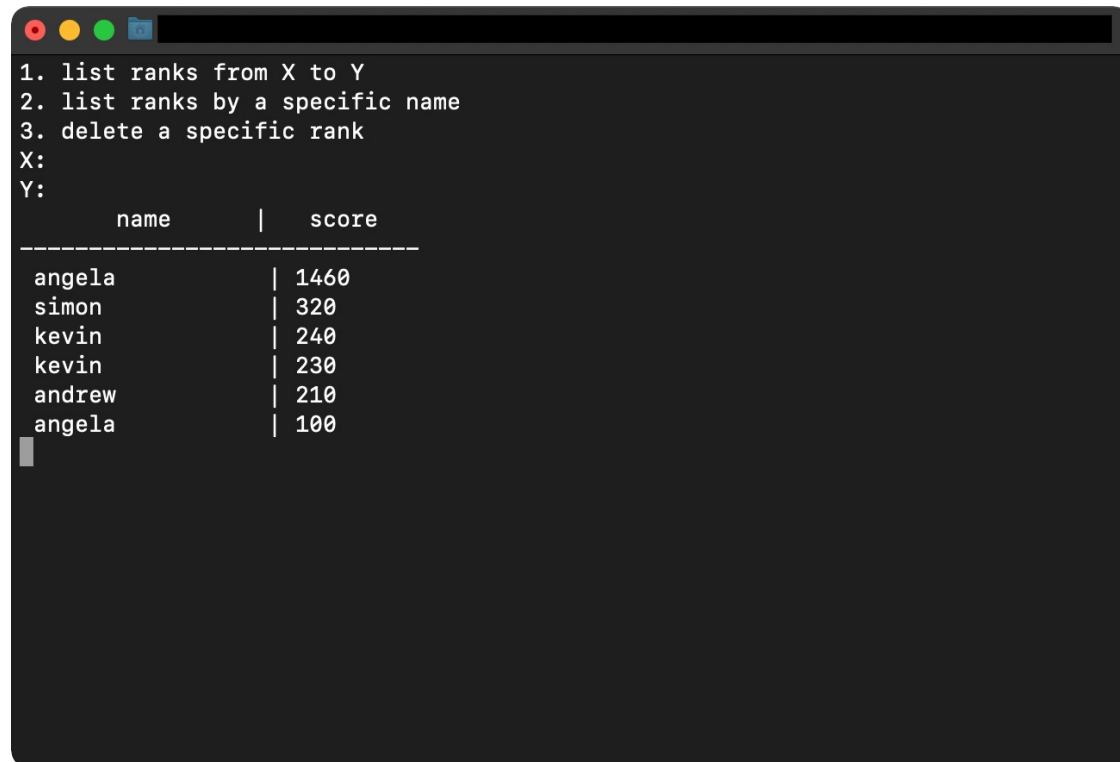
    Node* curr = head;
    int ranking = 1;
    while (curr != NULL) { //print the ranks from X to Y
        if (X <= ranking && ranking <= Y)
            printf(" %-16s | %d\n", curr->name, curr->score);
        else if (ranking > Y)
            break;
        curr = curr->next;
        ranking++;
    }
    return;
}
```

따라서 원하는 랭킹 범위를 입력 받은 뒤, 랭킹 추출 과정에서 탐색 및 랭킹 추출의 시간복잡도는 $O(n)$, 공간 복잡도는 $O(n)$ 이다. (n 은 리스트의 전체 노드 개수)

삽입 및 탐색, 정렬된 상태로 추출하는 기능에 대해 링크드 리스트는 $O(n)$ 의 시간복잡도로 수행이 가능했다. 배열의 경우 정렬된 배열을 사용한다면 삽입할 경우 적절한 위치를 탐색한 뒤에 삽입해야 하며 나머지 원소들을 재배치시켜야 하고, 정렬되지 않은 배열을 사용한다면 정렬된 상태로 추출할 경우 전체 정렬 과정이 필요하다. 반면에 링크드 리스트는 연산 과정을 줄일 수 있기 때문에 배열에 비해 효율적이다.

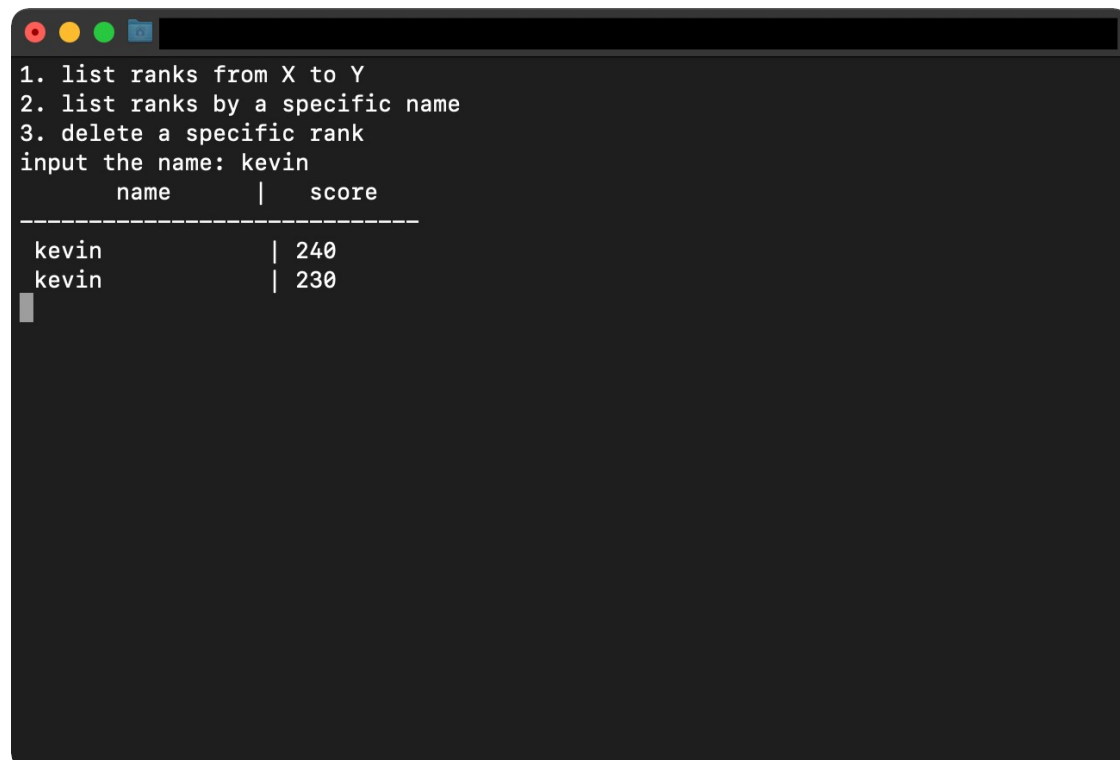
2. [과제] 사용자의 이름을 입력하고 해당하는 랭킹 정보 출력 화면

먼저 아래는 현재 랭킹 정보를 모두 출력한 것이다.



```
1. list ranks from X to Y
2. list ranks by a specific name
3. delete a specific rank
X:
Y:
  name | score
-----
angela | 1460
simon  | 320
kevin   | 240
kevin   | 230
andrew  | 210
angela  | 100
```

다시 rank 메뉴에서 2번을 선택해 아래처럼 "kevin"을 입력해 해당 랭킹 정보를 출력했다.



```
1. list ranks from X to Y
2. list ranks by a specific name
3. delete a specific rank
input the name: kevin
  name | score
-----
kevin  | 240
kevin  | 230
```

3. [과제] 원하는 사용자의 이름을 검색할 때 시간 및 공간 복잡도

2번에서 보인 화면과 같이 사용자의 이름을 검색하여 랭킹을 보여주는 기능은 rank() 함수에서 2를 입력받은 경우 searchName() 함수를 호출해 해당 기능을 수행한다.

```
void rank(){
    . . .
    else if ( ch == '2') { //week7 hw
        searchName();
    }
    . . .
}
```

(rank() 함수 전체 코드는 1-(2)에 보였다.)

searchName() 함수에서는 사용자에게 이름을 입력받아 해당 이름을 가진 노드가 있는지 리스트 전체를 순회하며 찾고, 찾은 경우 출력한다. 해당 이름을 가진 노드가 2개 이상 존재할 수도 있으므로 리스트의 마지막 노드까지 이름을 비교해가며 탐색한다. 노드를 찾지 못 한 경우 실패 문구를 출력한다.

```
void searchName() {
    char str[NAMELEN+1]; //name to search
    int check = 0; //If the name is found in the list, check == 1.

    //Input a user name.
    echo(); //turn on
    printf("input the name: ");
    scanf("%s", str);
    noecho(); //turn off

    printf("      name      |  score  \n");
    printf("-----\n");

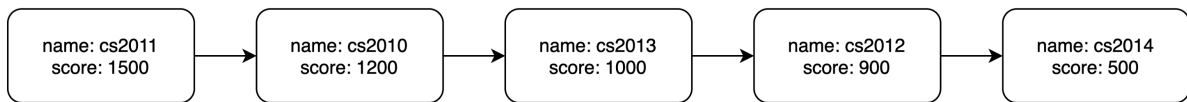
    //Search the user name.
    Node* curr = head;
    while (curr != NULL) {
        if (strcmp(curr->name, str) == 0) {
            printf(" %-16s | %d\n", curr->name, curr->score);
            check = 1;
        }
        curr = curr->next;
    }

    //If matched name is not found
    if (check == 0) {
        printf("\nsearch failure: no information in the list\n");
    }

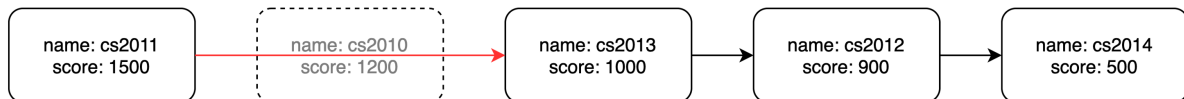
    return;
}
```

따라서 원하는 사용자의 이름을 검색할 때 시간 복잡도는 $O(n)$, 공간 복잡도는 $O(n)$ 이다. (n 은 리스트의 전체 노드 개수)

4. [과제] 선택한 자료구조에 맞춰 삭제 알고리즘을 표현한 그림



두 번째 노드(name: cs2010, score:1200)를 삭제할 경우 다음과 같이 첫 번째 노드의 링크를 삭제 대상 노드의 링크로 만들어준다. 두 번째 노드(삭제 대상 노드)가 할당된 메모리는 해제해 준다.



삭제 함수는 다음과 같이 구현했다.

```

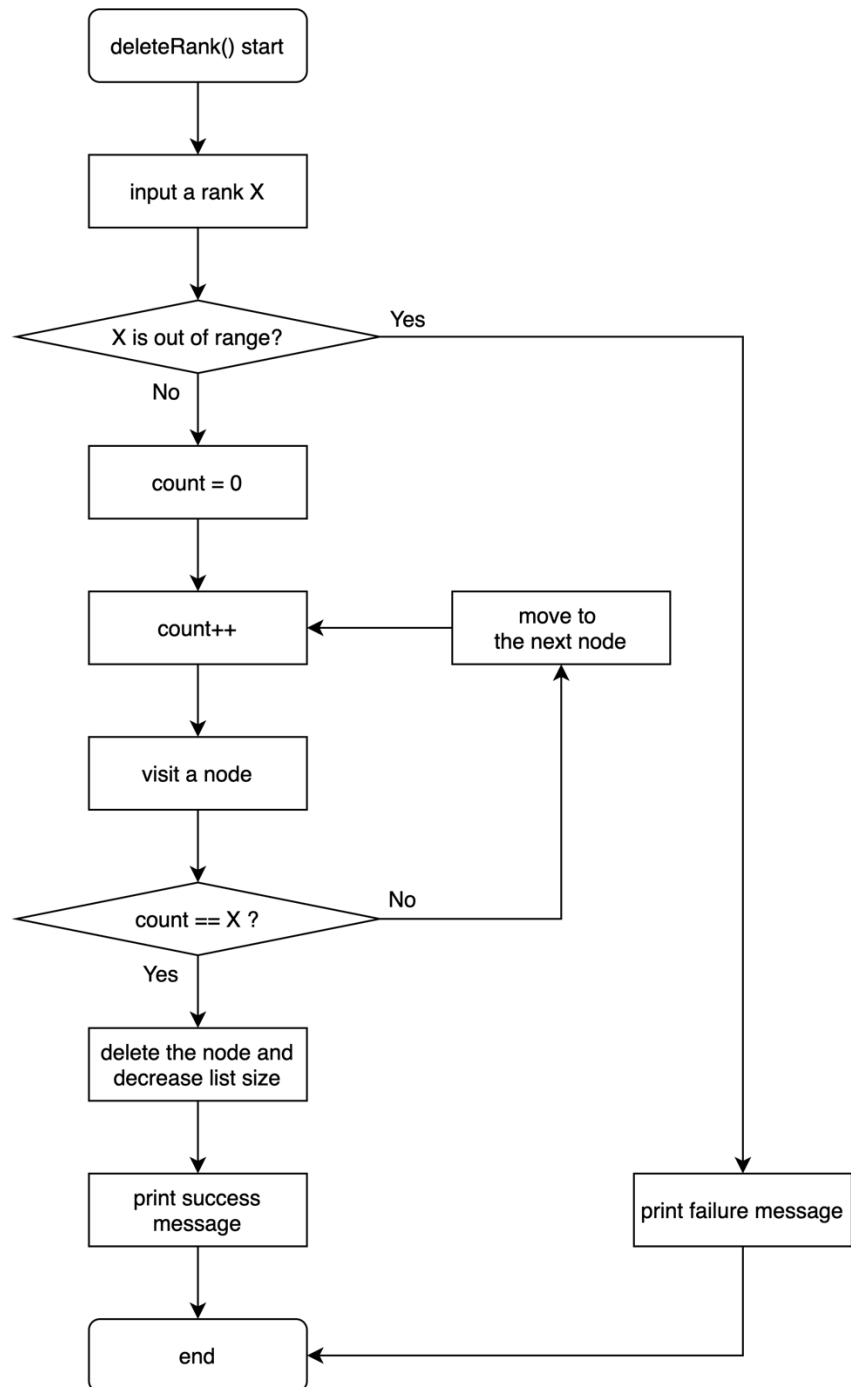
void deleteRank() {
    int num;
    int count = 0;

    //get input rank
    echo(); //turn on
    printf("input the rank: ");
    scanf("%d", &num);
    noecho(); //turn off

    //if num is out of range, print failure message
    if (num <= 0 || num > score_number) {
        printf("\nsearch failure: the rank not in the list\n");
        return;
    }

    //delete rank
    Node* curr = head;
    Node* temp;
    if (num == 1) { //head node deletion
        head = curr->next;
        free(curr);
    }
    else {
        count++;
        while (count < num - 1) { //curr := (node before the target node)
            count++;
            curr = curr->next;
        }
        temp = curr->next;
        curr->next = curr->next->next;
        free(temp);
    }
    score_number--;
    printf("\nresult: the rank deleted\n");
    writeRankFile();
    return;
}
  
```

flow chart로 나타내면 다음과 같다.



5. 본 실험 및 숙제를 통해 습득한 내용

테트리스 게임에 랭킹 정보를 생성, 삭제, 탐색하는 기능을 링크드 리스트로 구현하여 링크드 리스트에 노드를 삽입하고, 삭제하고, 순회하는 방법을 습득했다. 또한 ncurses 라이브러리에 있는 함수들(printw(), scanw(), echo(), noecho() 등)을 사용하는 방법을 습득했다.