9주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

1.

다음과 같은 linked list 자료구조를 만들어 랭킹 시스템에 사용하였다.

typedef struct \_Node {

    char name[NAMELEN];

    int score;

    struct \_Node \*link;

} Node;

Node \*head = NULL;

Node \*tail = NULL;

linked list의 경우 배열과 달리 물리적으로 연속된 메모리에 정보가 저장되지 않는다. 따라서 원소를 삽입 또는 삭제할 경우, 배열에 비해 시간 소모가 매우 적다는 장점을 가지고 있다. 특히 랭킹 시스템의 경우 값을 계속 맨 뒤에 추가하는 것이 아니라, 순위에 따라 중간에 삽입하게 되는 경우도 많으므로 linked list 자료구조가 효율적이라고 생각했다.

다음은 rank 함수의 부분들이다.

1-1. 옵션 1

*//4-1. 메뉴1: X, Y를 입력받고 적절한 input인지 확인 후(X<=Y), X와 Y사이의 rank 출력*

if (ch == '1') {

    echo();

    printw("X: ");

    scanw("%d", &X);

    printw("Y: ");

    scanw("%d", &Y);

    noecho();

    if(X>Y || X<1 || score\_number < Y) {

        printw("search failure: no rank in the list\n");

*// return;*

    }

    else {

        int count = 1;

        printw("      name      |   score\n");

        printw("----------------------------\n");

        for(Node \*ptr = head; ptr != NULL; ptr = ptr->link) {

            if(X<=count && count<=Y) {

                printw("%-16s| %d\n", ptr->name, ptr->score);

            }

            else if(Y<count) break;

            count++;

        }

    }

}

X, Y값을 입력 받는다. 이후 X, Y가 조건을 만족하지 않는 경우 에러 메시지를 출력한다. 만약 조건을 만족한다면 ptr 포인터로 linked list를 순회하면서 X와 Y 사이의 랭킹을 출력한다.

시간복잡도: 최악의 경우 linked list의 원소 전체를 순환해야 한다. linked list의 원소 수를 n이라 할 때 O(n)의 시간복잡도를 가진다.

공간복잡도: 전역변수를 사용하며 상수개의 변수를 사용하므로 O(1)이다.

1-2. 옵션2

*//4-2. 메뉴2: 문자열을 받아 저장된 이름과 비교하고 이름에 해당하는 리스트를 출력*

else if ( ch == '2') {

    char str[NAMELEN+1];

    int check = 0;

    echo();

    printw("input the name: ");

    scanw("%s", str);

    noecho();

    printw("      name      |   score\n");

    printw("----------------------------\n");

    Node \*ptr = head;

    int finddata = 0;

    for(; ptr!=NULL; ptr=ptr->link) {

*// ptr이 가리키는 노드의 이름과 입력받은 이름이 동일하다면*

        if(strcmp(ptr->name, str) == 0) {

            printw("%-16s| %d\n", ptr->name, ptr->score);

            finddata = 1;

        }

    }

    if(finddata == 0) {

        printw("search failure: no name in the list\n");

    }

}

찾을 str를 입력 받는다. ptr 포인터로 linked list를 순환하면서 str과 동일한 이름의 노드가 있는 경우 해당 정보를 출력한다. 만약 해당 정보를 하나도 찾지 못했다면 실패 메시지를 출력한다.

시간복잡도: str과 같은 이름을 가진 모든 랭킹 정보를 출력해야 하므로 linked list 전체를 순환해야 한다. 따라서 linked list의 원소 수를 n이라 할 때 O(n)의 시간복잡도를 가진다.

공간복잡도: 전역변수를 사용하며 상수개의 변수를 사용하므로 O(1)이다.

1-3. 옵션3

*//4-3. 메뉴3: rank번호를 입력받아 리스트에서 삭제*

else if ( ch == '3') {

    int num;

    echo();

    printw("input the rank: ");

    scanw("%d", &num);

    noecho();

    if(num<1 || num>score\_number) {

        printw("\nsearch failure: the rank not in the list\n");

    }

    else {

        Node \*ptr = head;

        Node \*pre = NULL;

        for(int i=1; i<num; i++) {

            if(ptr==NULL) break; *// Err*

            pre = ptr;

            ptr = ptr->link;

        }

        if(pre == NULL) { *// linked list의 맨 첫 노드를 삭제하는 경우*

            head = ptr->link;

        }

        else {

            pre->link = ptr->link;

        }

        free(ptr);

        score\_number--;

        printw("\nresult: the rank deleted\n");

    }

}

삭제할 랭킹인 num를 사용자로부터 입력 받는다. num이 조건에 부합하지 않는다면 실패 메시지메 출력한다. 만약 조건에 부합한다면 포인터 ptr이 삭제할 노드를 가리킬 때까지 ptr을 linked list에서 순환시킨다. 삭제할 노드를 가리키게 되었을 때, 만약 pre가 NULL인 경우 head가 ptr->link를 저장하도록 한다. pre가 NULL이 아니라면 pre의 link에 ptr의 link를 저장한 다음 ptr를 free한다. 이후 score\_number를 -1한 뒤, 성공 메시지를 출력한다.

시간복잡도: 최악의 경우 삭제할 노드가 맨 마지막에 존재하므로 linked list 전체를 순환해야 한다. 따라서 linked list의 원소 수를 n이라 할 때 O(n)의 시간복잡도를 가진다.

공간복잡도: 전역변수를 사용하며 상수개의 변수를 사용하므로 O(1)이다.

2.

1번에서 살펴본 결과처럼 linked list를 사용하면 삽입, 삭제, 조회 등의 과정을 O(n)의 시간복잡도로 끝내는 것이 가능하다. 또한 배열의 경우 배열 중간의 값을 삭제하거나 삭제할 경우 해당 값 뒤의 모든 값을 1칸 씩 앞으로 당겨오거나 뒤로 밀어야 되기 때문에 시간이 많이 소요된다. 하지만 linked list의 경우 다음 노드의 주소를 저장하는 방식으로 연결되어 있기 때문에, 뒤의 모든 값을 옮기지 않고 변화가 생긴 노드의 앞 뒤 노드의 값만 수정해주면 된다는 점을 알 수 있었다.