9주차 예비보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

1) Sorted Linked List

점수를 기준으로 내림차순으로 정렬된 단순 연결리스트로, 사용자의 이름과 점수를 저장한다.

다음과 같이 전역변수로 선언된 연결리스트의 포인터(Root)와 저장된 노드의 개수를 저장하는 변수(score\_number)로 랭킹 데이터를 삽입, 삭제 및 원하는 범위의 노드를 출력할 수 있다.

typedef struct \_Node {

char name[NAMELEN + 1];

int score;

struct \_Node\* next;

} Node;

Node\* Root = NULL;

int score\_number = 0;

연결리스트 포인터 Root는 처음에 내용이 비어있는 헤드 노드를 할당받는다.

데이터 삽입시 새로운 노드를 생성한 뒤, 입력된 점수와 이름으로 노드의 데이터를 채운 다음 헤드 노드 다음 노드부터 마지막 노드까지 연결리스트를 탐색하면서 입력된 노드의 점수가 탐색중인 노드의 점수 이상일 때 해당 위치에 새로운 노드를 삽입한다.

(연결리스트에 헤드 노드밖에 없는 경우 헤드 노드 다음에 바로 새로운 노드를 삽입한다)

데이터 삭제시 헤드 노드 다음 노드부터 연결리스트를 탐색한다. 삭제할 노드의 위치에 도달하면 해당 노드를 삭제한다.

데이터 출력시 헤드 노드 다음부터 마지막 노드까지 탐색하면서 x~y번째에 해당하는 노드의 정보를 출력한다.

연결리스트 형식으로 랭킹 데이터를 저장할 경우, 저장되는 랭킹 데이터의 개수가 n개일 때, 공간복잡도 O(n)을 갖는다.

2) Sorted Array

점수를 기준으로 내림차순으로 정렬된 동적 할당된 1차원 배열로, 사용자의 이름과 점수를 저장한다.

다음과 같이 전역변수로 선언된 배열의 포인터(Arr)와 배열의 크기(capacity), 배열에 저장된 데이터의 개수(size)로 랭킹 데이터를 삽입, 삭제 및 원하는 범위의 원소를 출력할 수 있다.

typedef struct \_Data {

char name[NAMELEN + 1];

int score;

}Data;

int size = 0, capacity = 1;

Data\* Arr = NULL;

배열 포인터 Arr는 처음에 크기가 1인 Data타입 배열을 할당받는다.

데이터 삽입시 저장된 데이터의 개수+1이 배열의 크기와 같은 경우 (배열에 데이터가 꽉 찬 경우) realloc을 사용해 배열의 크기를 2배로 늘린다.

그런 다음, 배열의 원소를 순회하면서 (인덱스 0~size-1까지) 입력된 점수가 배열의 원소의 점수 이상인 인덱스를 찾아서 loc에 저장한다. (없으면 loc = size)

마지막으로 배열의 loc부터 size-1까지의 모든 원소들을 한 칸씩 모두 다음 인덱스로 옮긴 다음 Arr[loc]에 입력된 데이터를 저장한다.

데이터 삭제시 삭제할 위치의 다음 원소부터 데이터가 저장된 마지막 원소까지 모두 한 칸씩 이전 인덱스로 옮긴다.

데이터 출력시 배열의 시작원소부터 데이터가 저장된 마지막 원소까지 탐색하면서 인덱스가 x-1~y-1인 원소의 정보를 출력한다.

배열 형식으로 랭킹 데이터를 저장할 경우, 저장되는 랭킹 데이터의 개수가 n개 일 때

공간복잡도는 O(n)을 갖는다.

**2.**

1) Sorted Linked List

List\_Insert(name, score)

Node\* temp = (Node\*)malloc(sizeof(Node))

temp->score = score

strcpy(temp->name, name)

temp->next = NULL

if (score\_number == 0)

Root->next = temp

else

Node\* ptr = Root->next, \* trail = Root->next

while (true)

if (ptr == NULL)

trail->next = temp

break

else if (ptr == trail && temp->score >= ptr->score)

temp->next = ptr

Root->next = temp

break

else if (temp->score >= ptr->score)

temp->next = ptr

trail->next = temp

break

trail = ptr

ptr = ptr->next

score\_number++

: 연결리스트의 삽입함수이다. 최악의 경우, 연결리스트에 존재하는 모든 노드를 순회해야 하므로 노드의 개수를 n개라 할 때, 시간복잡도는 O(n)이다.

최대 1개의 노드를 추가로 생성하므로, 이 함수의 공간복잡도는 O(1)이다.

List\_Delete(index)

if (index <= 0 || index > score\_number)

return

count = 0

Node\* ptr = Root, \* trail = Root

for (; ptr != NULL; ptr = ptr->next, count++)

if (index == count)

trail->next = ptr->next

free(ptr)

score\_number--

break

trail = ptr

: 연결리스트의 삭제함수이다. (index번째 노드를 삭제함)

최악의 경우, 연결리스트에 존재하는 모든 노드를 순회해야 하므로

노드의 개수를 n개라 할 때, 시간복잡도는 O(n)이다.

함수 내에서 추가적인 공간 할당이 없으므로 이 함수의 공간복잡도는 O(1)이다.

2) Sorted Array

Arr\_Insert(name, score)

if (size+1 == capacity)

capacity \*= 2

Arr = (Data\*)realloc(Arr, capacity \* sizeof(Data))

for (loc = 0; loc < size; loc++)

if (score >= Arr[loc].score)

break

for (i = size - 1; i >= loc; i--)

Arr[i+1] = Arr[i]

Arr[loc].score = score

strcpy(Arr[loc].name, name)

size++

: 배열의 삽입함수이다. 최악의 경우, 배열의 데이터가 저장된 모든 원소를 순회하고, 이동시켜야 하므로 배열에 저장된 데이터들의 총 개수(size)가 n이라 할 때 시간복잡도는 O(n)이다.

배열은 최소 size개 만큼의 원소를 가지고, 부족하면 내부에서 2\*n개의 공간을 재할당 하므로 공간복잡도는 O(n)이다.

Arr\_Delete(index)

index--

if (index < 0 || index >= size)

return

for (i = index; i < size; i++)

Arr[i] = Arr[i + 1]

size--

: 배열의 삭제함수이다. (index번째 원소를 삭제함)

최악의 경우 배열의 데이터가 저장된 모든 원소를 이동시켜야 하므로 배열에 저장된 데이터들의 총 개수(size)가 n이라 할 때 시간복잡도는 O(n)이다.

함수 내에서 추가적인 공간 할당이 없으므로 이 함수의 공간복잡도는 O(1)이다.

**3.**

1) Sorted Linked List

List\_Print(x, y)

if (x > y)

return

if (x < 1)

x = 1

if (y > score\_number)

y = score\_number

count = 0

Node\* ptr = Root

for (; ptr != NULL && count <= y; ptr = ptr->next, count++)

if (count >= x && count <= y)

printf("%s %d\n", ptr->name, ptr->score)

: 연결리스트의 출력함수이다. (x번째 ~ y번째 노드를 출력함)

x번째 노드부터 y번째 노드까지 출력하고, 각각의 출력은 연속적으로 수행되므로

시간복잡도는 O(y-x)이고, 공간복잡도는 O(1)이다.

2) Sorted Array

Arr\_Print(x, y)

x--, y--

if (x > y)

return

if (x < 0)

x = 0

if (y >= size)

y = size-1

for (i = x; i <= y; i++)

printf("%s %d\n", Arr[i].name, Arr[i].score)

: 배열의 출력함수이다. (x번째 ~ y번째 원소를 출력함)

x번째 원소부터 y번째 원소까지 차례대로 출력하고, 각각의 출력은 O(1)의 시간복잡도를 가지므로 이 함수의 시간복잡도는 O(y-x)이고, 공간복잡도는 O(1)이다.