9주차 예비보고서

전공: 경제학과 학년: 4학년 학번: 20180501 이름: 김연수

1. 자료구조 고민

랭킹 시스템은 다음 4가지 기능을 갖춘다. 랭킹 목록 생성, 랭킹 정보 출력, 랭킹 정보 추가, 랭킹 정보 저장. 즉 create, read, update, store의 기능을 모두 갖춰야 한다. 이를 구현하기 위한 자료구조로 LinkedList와 배열을 생각할 수 있다.

LinkedList는 구조체를 연결해 공간복잡도의 절약을 기대할 수 있을 것이다. 보통 LinkedList는 시간보다는 공간의 제약이 많을 때 유용하다. data를 삽입하고 조회하는데 걸리는 시간 복잡도가 배열보다 더 오래걸린다. 하지만 미리 공간을 확보해 놓을 필요가 사라지므로 공간 복잡도 면에서는 성능 향상을 기대할 수 있다.

반면, 배열은 시간복잡도 측면에서 LinkedList보다 유리하다. 배열은 조회하는 데 걸리는 시간복잡도가 1이다. 바로 address에 접근해서 조회할 수 있다. 반면 LinkedList는 해당 address의 위치를 찾기 위해서 LinkedList의 연결을 따라 탐색해야 한다. 배열을 사용한다면 시간 복잡도 측면에서 성능향상을 기대할 수 있을 것이다.

1. 삽입 및 삭제의 pseudo code, 시간 및 공간 복잡도

* Linked List

삽입 :

let rank\_node is structure for rank Linked List.

define temp\_node is rank\_node pointer and dynamic allocate space to temp\_node as much as size of rank\_node

put data in temp\_node.

let current\_node is first node of Linked List.

travling the LinkedList, find position that temp\_node would be inserted. and insert the temp\_node in that place.

삭제 :

Initialize temporary pointers for current and previous nodes

Iterate through the linked list until the desired rank number is reached

Check if the current node exists

If the previous node is NULL (i.e., the head node), update the head to point to the next node

Otherwise, update the previous node's next pointer to skip the current node

Free the memory of the current node

Print a message stating that the rank has been deleted

If the current node does not exist, print a message stating that the rank is not in the list

삽입 및 삭제 시간복잡도 : O(n) , first 또는 last element의 경우에는 O(1) 시간 복잡도를 가질 수 있음

* 배열.

삽입 :

define array of rank\_name nested array with size of (number of rank. \* (char\*))

define array of rank\_score with size of (number of rank. \* int)

search the position that data is inserted in in rank\_score array with order score decreased. and get the index.

insert the name to rank\_name array and the score to rank\_score separately.

move the remain element that is located behind the index.

삭제 :

find the index that is gonna deleted with the score that is ordered by decreasing already.

erase the name that is located in that index at rank\_name.

erase the score that is located in that index at rank\_score.

fill the blank field in these array by score decreased order.

시간 복잡도: 삽입 및 삭제 둘다 O(n)

공간 복잡도: Linked List and array 둘 다 O(n) 하지만 LinkedList의 경우 공간할당이 정해져 있지 않고 유동적이기 때문에 공간을 절약할 수 있음.

하지만 LinkedList도 마냥 좋은 것은 아니다. 그 이유가 LinkedLlist를 구현하기 위해서 structure 안에 nextnode의 pointer를 저장해야 하는 데, 이 때문에 메모리가 낭비되는 측면이 있다.

1. 정보를 불러오는 기능 pseudocode, 시간 및 공간 복잡도

LinkedList에서 실행하는 경우 ;

let temp is head node

if x <= y

for i from X to Y inclusive

if i is between x and y (excluding y)

print the name and score of the current node (temp)

if i is greater or equal to y

break the loop

move to the next node in the LinkedList (temp)

else

print an error message stating search failure: no rank in the list

시간 복잡도 : for문으로 Linked List를 탐색 하면서 x보다 같거나 크고 y보다 작은 rank에 위치한 node의 정보를 출력하기 때문에 최악의 경우 O(n) 최선의 경우 O(1)의 시간 복잡도를 가진다.

공간 복잡도 : LinkedList의 경우 공간복잡도는 O(n)이다. 정보 조회시 늘어나는 공간이나 줄어드는 공간은 없다.

array에서 실행하는 경우 :

Assume rankArray is the array containing rank information

if x <= y

for i from X to Y inclusive

if i is between x and y (excluding y)

print the name and score of the rank at index i in rankArray

if i is greater or equal to y

break the loop

else

print an error message stating search failure: no rank in the list

시간 복잡도 : 위 LinkedList와 유사한 시간 복잡도를 가진다(O(n)). 다만 조금 더 빠를 것으로 예상한다. LinkedList는 해당 인덱스에 접근하기 위해서 head노드부터 차례대로 탐색해야하지만, array는 그럴 필요없이 바로 접근이 가능하다.

공간 복잡도 : O(n)의 공간복잡도를 가진다.