10주차 결과보고서

전공 : 컴퓨터공학 학년 : 2학년 학번 : 20151616 이름 : 최승환

1. 2주차 실습에 구현하는 랭킹 시스템에 대한 자료를 읽어보고, 이를 구현하기 위한 자료구조를 2가지 이상 생각한다.

일단 랭킹 시스템을 구현하기 위한 자료구조는 크게 두가지를 생각해볼 수 있다.

첫번째로는 array를 생각할 수 있다. array를 이용하면 push나 pop을 할 때 이미 sort되어 있는 배열에서 적당한 위치를 검색해 삽입하거나 삭제해줘야 한다. 이 경우 크기를 비교해야 하기 때문에 binary search 알고리즘을 사용해 자신이 들어가야 할 위치를 찾고, 이 위치에 삽입을 해줘야 한다. 하지만 array의 경우 삽입과 삭제가 모두 복잡하다. realloc을 이용해 계속 메모리 크기를 할당해줘야 하기 때문이다.

두번째로는 linked list를 생각할 수 있다. linked list를 이용하면 push나 pop을 할 때 시작하는 노드부터 끝나는 노드까지 모두 검색해 자신의 자리를 찾아야 한다. 이 경우 검색에는 더 많은 시간이 걸릴 수 있으나 이를 삽입하는 건 array와 비교했을 때 훨씬 효율적이다.

2. 생각한 자료구조에 대해서 새로운 랭킹을 삽입 및 삭제를 구현하기 위한 pseudo code를 작성하고, 시간 및 공간 복잡도를 계산한다.

array - binary search

while (first <= last) {  
 if (array[middle] < search)  
 first = middle + 1;   
 else if (array[middle] == search) {  
 printf("%d found at location %d.\n", search, middle+1);  
 break;  
 }  
 else  
 last = middle - 1;  
   
 middle = (first + last)/2;  
 }

이 경우 시간복잡도는 O(log(n))이 된다.

추가나 삭제는 realloc을 사용한다.

linked list

void insert(NODE\* root,int k)

{

NODE\* temp ,\*new\_node,\*prev;

for (temp=root; temp->value >= k; prev=temp,temp=temp->link);

new\_node=~; //create new node

new\_node ->value = k;

prev->link = new\_node;

new\_node->link = temp;

}

void delete(NODE\* root,int k)

{

NODE\* temp,\*prev;

for (temp=root;temp->value != k;prev=temp,temp=temp->link);

prev->link = temp->link;

free(temp);

}

최악의 경우 시간복잡도는 O(n)이다.

3. 생각한 각 자료구조에서 사용자가 부분적으로 확인하길 원하는 정렬된 랭킹(x~y위, x<=y, x, y는 정수)의 정보를 얻는 방법을 간략히 요약해서 pseudo code를 작성하고, 시간 및 공간 복잡도를 계산한다.

array의 경우 x와 y를 입력 받은 후 x에서 y까지 출력하는 방식이다. 이 경우 y-x이고 최대 입력의 경우 시간복잡도는 O(n)이다. pseudo code의 경우 매우 간단하므로 생략하도록 하겠다.

linked list의 경우

void find(NODE\* root,int x,int y)

{

int i;

NODE\* temp=root;

for (i=1;i<x;temp=temp->link);

for (i=x;i<=y;temp=temp->link)

{

//현재 인덱스 값과 비교하는 노드의 값 비교

}

}

이 시간 복잡도는 O(n)이고 공간복잡도는 시간복잡도에 비례한다.