07 프로그래밍 프로젝트

01 **순차 탐색이란 정렬되지 않은 배열에서 탐색하고자 하는 숫자를 배열의 각 요소와 순차적으로 비교하여 탐색하는 방법을 말한다. 이진 탐색 트리의 성능을 일반 배열에서의 순차 탐색과 비교하여 보자**

(1) 다음과 같은 순차탐색 프로그램을 작성하고 10000-100000개 정도의 랜덤한 정수를 배열에 저장한 후에 순차 탐색의 수행 시간을 측정하라. 순차 탐색의 시간 복잡도는 탐색 연산은 이 된다.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define TRUE 1
#define FALSE 0
#define MAX_ARRAY_SIZE 10000
int list[MAX_ARRAY_SIZE];    // 1차원 배열
//
int seq_search(int key)
{
 int i;
 for(i=0; i<MAX_ARRAY_SIZE; i++)
     if(list[i]==key)
         return TRUE;
 return FALSE;
}
//
int main()
{
 int i;
 time_t start, finish;

 start=clock();
 for(i=0;i<MAX_ARRAY_SIZE;i++){
     list[i]=MAX_ARRAY_SIZE*rand()/(double)RAND_MAX;
 }
 for(i=0;i<MAX_ARRAY_SIZE;i++){
     seq_search(i);
 }
 finish=clock();
 printf("%d\n", finish-start);
}
```

(2) 이진 탐색 트리를 이용하여 동일한 실험을 반복하라. 어떤 결과가 얻어지는가? 그 이유는 무엇인가?

(3) 이진 탐색 트리에 입력하는 데이터를 최악의 경우로 만들고 다시 실험을 반복하라. 어떤 결과가 얻어지는가?

02 사용자로부터 정수들을 입력받아 이진 탐색 트리 안에 저장하고 다음과 같은 기능을 하는 프로그램을 작성하라.

```
**************
i: 입력
d: 삭제
s: 탐색
v: 순회
n: 트리의 높이를 구한다.
c: 노드의 개수를 계산한다.
t: 단말 노드의 개수를 출력한다.
m: 가장 큰 값을 출력한다.
n: 가장 작은 값을 출력한다.
x: 노드를 전부 삭제
p: 출력
q: 종료
**************
```

- 입력(i): 사용자로부터 숫자를 입력받아 탐색 트리 안에 저장한다. 본문의 소스를 참조하라.
- 삭제(d): 사용자로부터 숫자를 입력받아 탐색 트리로부터 숫자를 삭제한다. 본문의 소스를 참조하라.
- 탐색(s): 사용자로부터 숫자를 입력받아 탐색 트리를 탐색하여 숫자의 존재여부를 표시한다(있음, 없음). 본문의 소스를 참조하라.
- 높이(h): 현재 생성된 이진 탐색 트리의 높이를 반환한다. 본문의 소스를 참조하라.
- 노드의 개수(c): 현재 생성된 이진 탐색 트리안의 노드의 개수를 반환한다. 본문의 소스를 참조하라.
- 단말 노드의 개수(t): 현재 생성된 이진 탐색 트리안의 단말 노드의 개수를 반환한다. 본문의 소스를 참조하라.
- 순회(v): 현재 생성된 이진 탐색 트리를 표준적인 3가지 방법으로 순회하고 순회한 결과를 화면에 출력한다. 본문의 소스를 참조하라.
- 최대숫자(m): 탐색 트리안의 가장 큰 숫자를 표시한다. 함수 `get_maximum(TreeNode *root)`를 작성한다. 이진 탐색 트리에서 가장 큰 값은 가장 오른쪽에 있다. 따라서 오른쪽 자식 링크를 따라서 링크가 NULL이 될 때까지 가장 오른쪽으로 가면 된다. 다음은 유사 코드로 표시된 알고리즘이다.

알고리즘 7.12 이진 탐색 트리에서 최대값 탐색 알고리즘

```
get_maximum(x)
while RIGHT(x) ≠ NULL
    do x ← RIGHT(x);
return DATA(x);
```

- 최소숫자(n): 탐색 트리안의 제일 작은 숫자를 표시한다. 함수 `get_minimum(TreeNode *root)`를 작성하라. 최대값 구하는 알고리즘과 매우 비슷하다. 다음은 유사 코드이다.

알고리즘 7.13 이진 탐색 트리에서 최소값 탐색 알고리즘

```
get_minimum(x)
while LEFT(x) ≠ NULL
    do x ← LEFT(x);
return DATA(x);
```

- 모두 삭제(x): 트리안의 노드를 모두 삭제한다. 함수 delete_all(TreeNode *root)를 작성하라. 노드를 모두 삭제하기 위해서는 노드들을 모두 방문하여야 할것이다. 방문하면서 노드들을 삭제하면 된다. 여기서 노드를 삭제한다는 의미는 노드들이 차지하고 있는 메모리 공간을 해제시켜서 되돌려주면 된다. 3가지의 순회 방법 중에서 가장 적절한 것은 후위순회이다. 먼저 서브 트리안의 노드들을 삭제한 다음에 루트 노드를 삭제하는 편이 안전하다. 따라서 알고리즘은 다음과 같이 된다.

알고리즘 7.14 이진 탐색 트리에서 전체 노드 삭제 알고리즘

```
delete_all(x)

if  x ≠NULL
    delete_all(LEFT(x));
    delete_all(RIGHT(x));
    free(x);
```

03 1번의 이진 탐색 트리 프로그램을 이용하여 학생들과 관련된 자료를 저장하고 탐색하는 프로그램을 개발하여 보자. 하나의 학생은 학번(정수), 이름(문자열), 주소(문자열), 소속학과(정수)의 정보를 가지고 있다. 이들 정보를 학번을 키로 하여 이진 탐색 트리에 저장하고 다음과 같은 메뉴가 가능하도록 프로그램을 작성하라. 학번 순으로 출력하는 것은 이진 탐색 트리의 중위 순회시 정렬된 숫자가 얻어지는 것을 이용하여 구현하라.

```
학생 정보 검색 프로그램
**************
1. 학생 정보 입력
2. 학생 정보 삭제
3. 학생 정보 탐색
4. 학생 정보 학번순으로 출력
5. 현재 저장된 학생들의 총 숫자
6. 전부 삭제
7. 종료
**************
```