

자료구조응용

16. 최대힙, 이진탐색트리 (10점)

2022.5.4.

1. 다음 입력파일의 데이터를 사용하여 최대힙(Max Heap)에 대한 실습을 수행한다.(3점)

input.txt : 10 40 30 5 12 6 15 9 60

[실행순서]

- ① 파일입력을 받으면서 최대힙을 구성한다.
매 입력마다, 구성된 최대힙의 배열원소를 인덱스 순서대로 출력한다.
- ② 최대힙의 최대값을 연속으로 원소개수만큼 삭제한다.
매 삭제마다, 재구성된 최대힙의 배열원소를 인덱스 순서대로 출력한다.

[구현 전, 노트에 연습하기]

- ① 위 실행순서 ①의 최대힙이 만들어지는 과정을 최대힙 그림으로 보여라.
 - ② 위 실행순서 ②의 삭제연산 과정을 최대힙 그림으로 보여라.
 - ③ ①②의 각 트리 결과에 대해 배열로 표시해 보라.
- ※ 각각의 Key 추가/삭제에 대해 결과 트리만 표시하면 됨
※ 노트에 직접 그려서 스캐닝 혹은 사진을 찍어서 보고서에 포함

[구현 세부사항]

```
#define MAX-ELEMENTS 200 /* maximum heap size+1 */
#define HEAP-FULL(n) (n == MAX-ELEMENTS-1)
#define HEAP-EMPTY(n) (!n)
typedef struct {
    int key;
    /* other fields */
} element;
element heap[MAX-ELEMENTS];
int n = 0;
```

```
void push(element item, int *n)
{
    /* insert item into a max heap of current size *n */
    int i;
    if (HEAP-FULL(*n)){
        fprintf(stderr, "The heap is full. \n");
        exit(EXIT_FAILURE);
    }
    i = ++(*n);
    while ((i != 1) && (item.key > heap[i/2].key)) {
        heap[i] = heap[i/2];
        i /= 2;
    }
    heap[i] = item;
}
```

Program 5.13: Insertion into a max heap

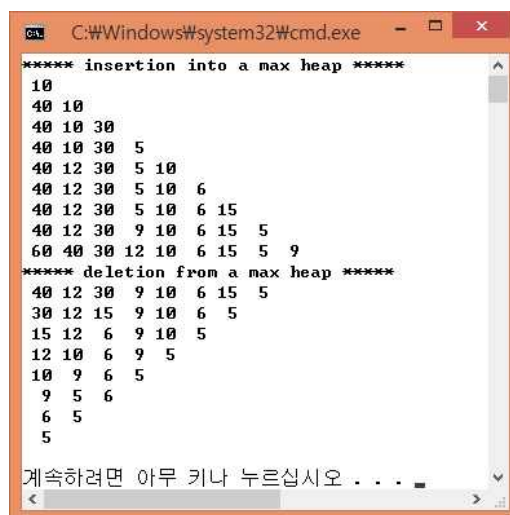
```

element pop(int *n)
{
    /* delete element with the highest key from the heap */
    int parent, child;
    element item, temp;
    if (HEAP_EMPTY(*n)) {
        fprintf(stderr, "The heap is empty\n");
        exit(EXIT_FAILURE);
    }
    /* save value of the element with the highest key */
    item = heap[1];
    /* use last element in heap to adjust heap */
    temp = heap[(*n)--];
    parent = 1;
    child = 2;
    while (child <= *n) {
        /* find the larger child of the current parent */
        if ((child < *n) && (heap[child].key < heap[child+1].key))
            child++;
        if (temp.key >= heap[child].key) break;
        /* move to the next lower level */
        heap[parent] = heap[child];
        parent = child;
        child *= 2;
    }
    heap[parent] = temp;
    return item;
}

```

Program 5.14: Deletion from a max heap

[실행예]



```

C:\Windows\system32\cmd.exe
***** insertion into a max heap *****
10
40 10
40 10 30
40 10 30 5
40 12 30 5 10
40 12 30 5 10 6
40 12 30 5 10 6 15
40 12 30 9 10 6 15 5
60 40 30 12 10 6 15 5 9
***** deletion from a max heap *****
40 12 30 9 10 6 15 5
30 12 15 9 10 6 5
15 12 6 9 10 5
12 10 6 9 5
10 9 6 5
9 5 6
6 5
5
계속하려면 아무 키나 누르십시오 . . .

```

2. 다음과 같이 임의의 노드 n개로 구성된 이진탐색트리(binary search tree)를 생성하는 프로그램을 작성하라. (7점)

(1) 실행순서

① 난수생성을 위한 **seed**와 이진탐색트리의 노드 개수(n)를 입력받음

※ scanf

② 1~500 범위의 난수를 생성하여 노드의 key와 item 필드 값으로 동일하게 사용

※ 이진탐색트리의 key 값은 중복이 허용되지 않음을 주의

③ ②의 key, item을 사용하여 이진탐색트리에 노드를 하나 추가함

※ **Program 5.17. insert**

※ Program 5.17에서 사용된 **modifiedSearch**함수는 Program 5.16을 수정함

④ ②~③ 과정을 n번 수행하여 이진탐색트리를 구성

※ 난수발생 순서대로 노드를 추가해야 함

⑤ 탐색할 key를 입력받아서 이진탐색하여 그 결과를 출력한다.

※ Program 5.15 혹은 5.16

⑥ 이진탐색트리를 구성하고 있는 노드의 **key값을 오름차순으로 정렬되도록 출력함**

※ **inorder traversal** 사용

(2) 구현세부사항

```
typedef int iType;
typedef struct{
    int key;
    iType item;
}element;
typedef struct node *treePointer;
typedef struct node{
    element data;
    treePointer leftChild, rightChild;
}node;
```

```

element* search(treePointer root, int key)
{
    /* return a pointer to the element whose key is k, if
       there is no such element, return NULL. */
    if (!root) return NULL;
    if (k == root->data.key) return &(root->data);
    if (k < root->data.key)
        return search(root->leftChild, k);
    return search(root->rightChild, k);
}

```

Program 5.15: Recursive search of a binary search tree

```

element* iterSearch(treePointer tree, int k)
{
    /* return a pointer to the element whose key is k, if
       there is no such element, return NULL. */
    while (tree) {
        if (k == tree->data.key) return &(tree->data);
        if (k < tree->data.key)
            tree = tree->leftChild;
        else
            tree = tree->rightChild;
    }
    return NULL;
}

```

Program 5.16: Iterative search of a binary search tree

```

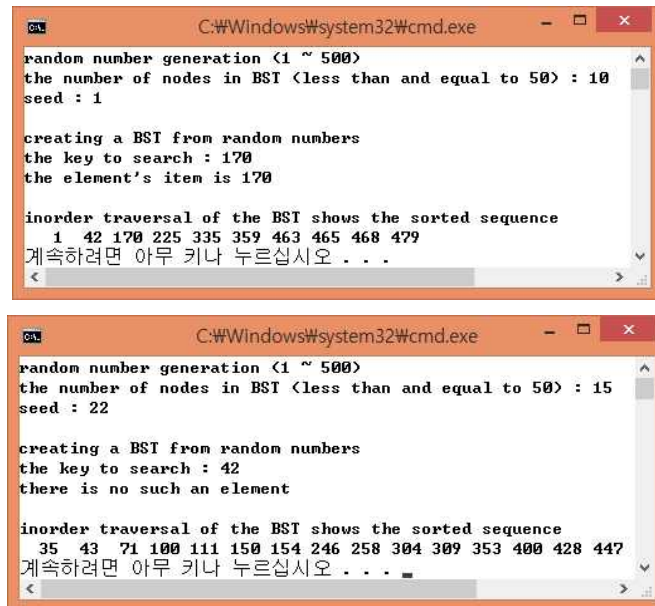
void insert(treePointer *node, int k, itemType theItem)
{
    /* if k is in the tree pointed at by node do nothing;
       otherwise add a new node with data = (k, theItem) */
    treePointer ptr, temp = modifiedSearch(*node, k);
    if (temp || !(*node)) {
        /* k is not in the tree */
        MALLOC(ptr, sizeof(*ptr));
        ptr->data.key = k;
        ptr->data.item = theItem;
        ptr->leftChild = ptr->rightChild = NULL;
        if (*node) /* insert as child of temp */
            if (k < temp->data.key) temp->leftChild = ptr;
            else temp->rightChild = ptr;
        else *node = ptr;
    }
}

```

Program 5.17: Inserting a dictionary pair into a binary search tree

modifiedSearch 알고리즘 (Program 5.16 iterSearch의 수정)	
1. 만약 BST가 empty라면 NULL을 반환한다.	
2. empty BST가 아닌 한 다음 과정을 반복한다.	
① 루트 키 값이 탐색키 k와 같으면 NULL을 반환한다.	
② 만약 k가 루트 키 값보다 작으면, 왼쪽 부트리의 루트를 새로운 루트로 만든다. 그렇지 않으면, 루트의 오른쪽 부트리의 루트를 새로운 루트로 한다.	
3. 2의 탐색 과정동안 만난 마지막 노드에 대한 포인터를 반환한다.	
※ 마지막 노드 : non-leaf 혹은 leaf node 일수 있음	

(3) 실행 예



```
C:\Windows\system32\cmd.exe
random number generation (1 ~ 500)
the number of nodes in BST (less than and equal to 50) : 10
seed : 1

creating a BST from random numbers
the key to search : 170
the element's item is 170

inorder traversal of the BST shows the sorted sequence
1 42 170 225 335 359 463 465 468 479
계속하려면 아무 키나 누르십시오 . . .

C:\Windows\system32\cmd.exe
random number generation (1 ~ 500)
the number of nodes in BST (less than and equal to 50) : 15
seed : 22

creating a BST from random numbers
the key to search : 42
there is no such an element

inorder traversal of the BST shows the sorted sequence
35 43 71 100 111 150 154 246 258 304 309 353 400 428 447
계속하려면 아무 키나 누르십시오 . . .
```

[도전 문제] 2번 결과에 대해 이진탐색트리의 삭제연산을 구현해 보라!!! (제출 안 함)

■ 제출 형식

- 솔루션 이름 : DS 15
- 프로젝트 이름 : 1, 2
- 각 소스파일에 주석처리
“학번 이름”
“본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”
- 실행화면을 캡처한 보고서를 작성 후 pdf 파일로 변환하여 솔루션 폴더에 포함
- 솔루션 정리 메뉴를 수행 후 전체 솔루션을 “학번.zip”으로 압축하여 제출

■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 1차 마감 (LMS 과제 마감일) : 수업일 자정
- 2차 마감 (LMS 과제 열람기한) : 수업 익일 자정(만점의 50%, 반올림)
- 1, 2차 마감 이외의 제출은 허용하지 않습니다. (이메일 제출 불가!)