

자료구조응용

2. Binary Search Tree, Winner Tree

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

(1) 실행순서

① 난수생성을 위한 seed와 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 modifiedSearch함수는 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;
}tNode;
```

```

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

```

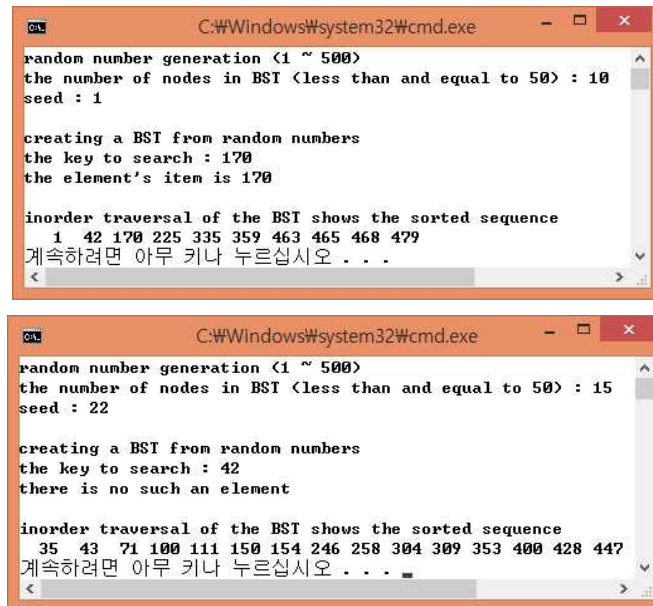
void insert(treePointer *node, int k, itemType theItem)
{
    /* if k is in the tree pointed at by node do nothing;
       otherwise add a new node with data = (k, theItem) */
    treePointer ptr, temp = modifiedSearch(*node, k);
    if (temp || !(*node)) {
        /* k is not in the tree */
        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 . . .
```

2. [승자트리를 이용한 정렬] k 개의 레코드를 가지는 승자트리(winner tree)의 초기생성 함수를 작성하여 정렬을 수행하라. 이때 k 는 2의 거듭제곱 (power of 2)임을 가정하라.

(1) 실행순서

① 난수생성을 위한 seed와 k를 입력받는다.

② 1~100 사이에서 발생시킨 k 개의 난수를 key로 사용하여 순서대로 배열에 저장한다.

※ 각 key는 중복 가능하다.

③ ②에서 생성한 키 데이터에 대해 초기 승자트리를 구성한다. ※ initWinner()

※ winner tree 는 완전이진트리이며 노드 레벨에 따라 배열에 순차적으로 저장된다.

※ winner tree 의 각 노드는 키 값(혹은 레코드)에 대한 포인터(혹은 인덱스)만을 가진다.

※ initWinner()는 recursive postorder traversal 형태로 작성한다.

④ 승자트리에 대해 inorder traversal을 수행하여 키 값을 출력한다. ※ inorder()

⑤ 승자트리를 사용한 정렬을 수행한다.

※ { 최소키를 sorted 배열에 저장 -> 무한대를 의미하는 임의의 값으로 최소키를 치환 -> 승자트리를 재구성(adjustWinner()) } 이 과정을 k 번 반복함

※ adjust 시, 치환된 키의 index --> parent index --> sibling index를 구할 수 있음. 치환된 키와 sibling 키의 비교를 루트 방향으로 수행함

⑥ 정렬된 결과배열의 인덱스 순서대로 키 값을 출력한다.

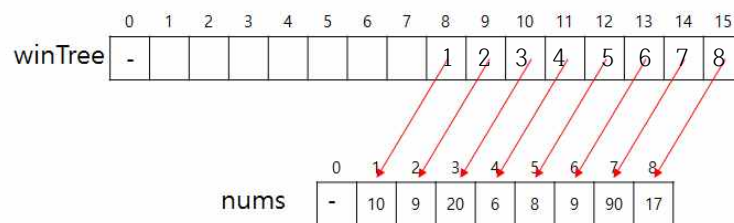
(2) 구현세부사항

① 함수정의 및 트리선언

```
#define MAX_SIZE 100
#define INF_NUM 10000

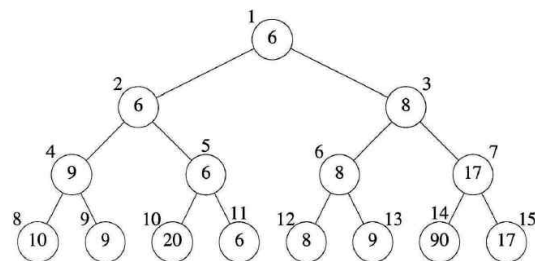
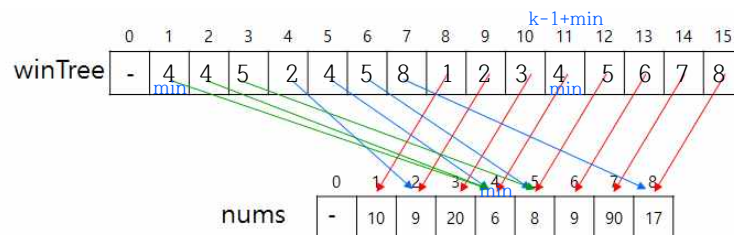
// min - winner tree
int nums[ MAX_SIZE+1 ] = { 0 };           // keys to sort
int winTree[ 2*MAX_SIZE ] = { 0 };       // winner tree
int sorted[ MAX_SIZE+1 ] = { 0 };        // sorted result
int initWinner(int cur, int k, int winTree[]);
void adjustWinner(int min, int k, int winTree[]);
void inorder(int root, int k, int winTree[]);
```

② k=8일 경우, 초기 min - winner tree 구성 예



↓ **initWinner(1, k, winTree);**

※ 각자 필요한 형태로 함수정의 후 호출



(3) 실행 예

```
C:\Windows\system32\cmd.exe
```

```
<<<<<<<<<<<< sorting with winner tree >>>>>>>>>>>>>>  
the number of keys < 8, 16, or 32 as a power of 2 > >> 8  
random numbers to use as key values (1 ~ 100)  
seed >> 1  
    42 68 35   1 70 25 79 59  
  
initialization of min-winner tree  
  
inorder traversal for min-winner tree  
    42 42 68   1 35   1   1   1 70 25 25 25 79 59 59  
  
sorting with min-winner tree...  
  
sorted result  
    1 25 35 42 59 68 70 79  
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
```

```
<<<<<<<<<< sorting with winner tree >>>>>>>>>>>>>>>  
  
the number of keys ( 8, 16, or 32 as a power of 2 ) >> 16  
random numbers to use as key values (1 ~ 100)  
seed >> 11  
75 49 37 90 12 24 35 86 39 20 27 86 95 51 8 33  
  
initialization of min-winner tree  
  
inorder traversal for min-winner tree  
75 49 49 37 37 37 90 12 12 12 24 12 35 35 86 8 39 20 20 20 27 27 86 8 95 51 5  
1 8 8 8 33  
  
sorting with min-winner tree...  
  
sorted result  
8 12 20 24 27 33 35 37 39 49 51 75 86 86 90 95  
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
```

```
<<<<<<<<< sorting with winner tree >>>>>>>>>>
```

the number of keys < 8, 16, or 32 as a power of 2 >> 32
random numbers to use as key values <1 ~ 100>
seed >> 1
42 68 35 1 70 25 79 59 63 65 6 46 82 28 62 92 96 43 28 37 92 5 3 54 93 83 2
2 17 19 96 48 27

initialization of min-winner tree

inorder traversal for min-winner tree
42 42 68 1 35 1 1 70 25 25 25 79 59 59 1 63 63 65 6 6 6 46 6 82 28 2
8 28 62 62 92 1 96 43 43 28 28 28 37 3 92 5 5 3 3 3 54 3 93 83 83 17 22
17 17 17 17 19 19 96 19 48 27 27

sorting with min-winner tree...

sorted result
1 3 5 6 17 19 22 25 27 28 28 35 37 42 43 46 48 54 59 62 63 65 68 70 79 82 8
3 92 92 93 96 96

계속하려면 아무 키나 누르십시오 . . .

■ 제출 형식

- 제출 : 12월 6일자 채점서버에 소스 업로드(12월 19일까지)
- 실행화면을 캡처하여 한글파일 보고서(DS 2_학번.hwp)에 추가 후 과제에 제출 (12월 19일 까지)
- 각 소스파일에 주석처리 추가
“학번 이름”
“본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”

■ 주의

- **소스복사로는 실력향상을 기대할 수 없습니다!!!**
- 먼저 제출한 학생은 남은 시간 동안 자료구조 관련 개인학습을 하거나 동료를 도와 줄 것
- 수강생 끼리 서로 물어보고 논의를 해도 됨
- 채점서버에 제출하지 않는 경우 최종 점수의 0점으로 처리함
(사용법을 잘 모르겠다면 개인적으로 튜터나 TA를 찾아와서 물을 것)