자료구조응용

14. Binary Search Tree, Winner Tree

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

(1) 실행순서

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

```
printf("random number generation (1 ~ %d)\n", MAX_SIZE);
printf("%s","the number of nodes in BST (less than and equal to 50) : " );
scanf_s("%d", &n);
printf("%s", "seed : " );
scanf_s("%u", &seed);
printf("\ncreating a BST from random numbers\n");
srand(seed);
```

② 1~500 범위의 난수를 생성하여 노드의 key와 item 필드 값으로 동일하게 사용(키값과 항목 값이 같음), 난수가 발생 되는 순서대로 출력 할 것

※ 이진탐색트리의 key 값은 중복이 허용되지 않음을 주의

③ ②의 key, item을 사용하여 이진탐색트리에 노드를 하나 추가함

④ ②~③ 과정을 n번 수행하여 이진탐색트리를 구성

※ 난수발생 순서대로 노드를 추가해야 함

⑤ 탐색할 key를 입력받아서 이진탐색하여 그 결과를 출력한다.
탐색과정을 출력하시오

※ Program 5.15 혹은 5.16

⑥ 이진탐색트리를 구성하고 있는 노드의 **key값을 오름차순으로 정렬되도록 출력**함

※ inorder traversal 사용

(2) 구현세부사항(참조)

```
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

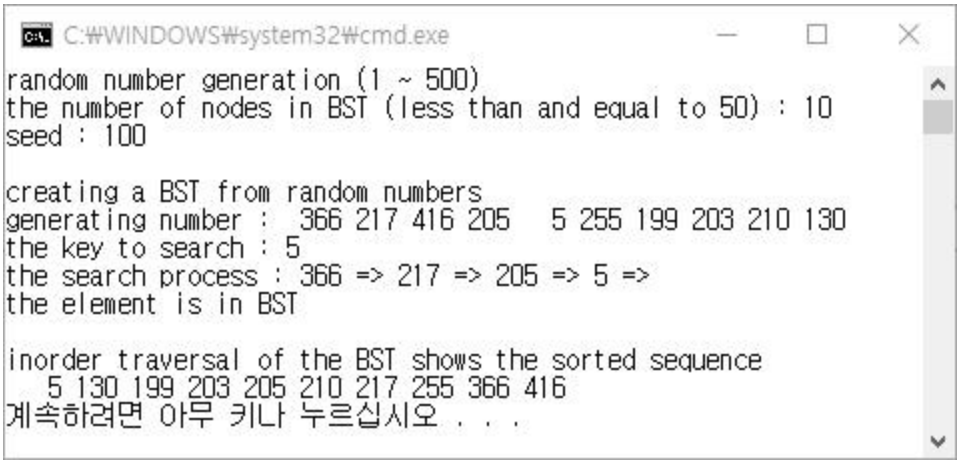
```
element* iterSearch(treePointer tree, int k)
{/* return a pointer to the element whose key is k,  if
    there is no such element, return NULL. */
  while (tree) {
    if (k == tree→data.key) return &(tree→data);
    if (k < tree→data.key)
      tree = tree→leftChild;
    else
      tree = tree→rightChild;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

Program 5.16: Iterative search of a binary search tree

```
void insert(treePointer *node, int k, iType 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

(3) 실행 예



2. [승자트리를 이용한 정렬] *k* 개의 레코드(node)를 가지는 승자트리(winner tree)의 초기생성 함수를 작성하여 정렬을 수행하라. 이때 *k*는 2의 누승 (power of 2)임을 가정하라. 단 승자는 키 값이 작은 노드이다.

(1) 실행순서

① 난수생성을 위한 **seed**와 **k**를 입력받는다.(1번 문제 참조)

② 1~500 사이에서 발생시킨 *k* 개의 난수를 key로 사용하여 순서대로 배열에 저장한다.

※ 각 key는 중복 가능하다.

③ ②에서 생성한 키 데이터에 대해 **초기 승자트리**를 구성한다.

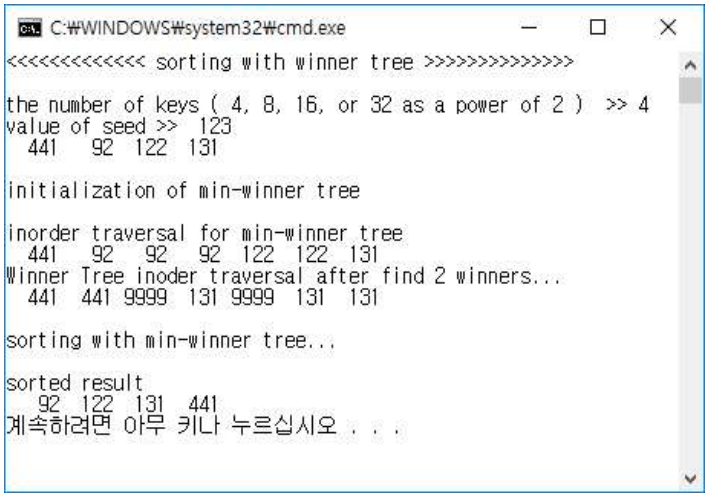
※ winner tree 는 완전이진트리이며 노드 레벨에 따라 배열에 순차적으로 저장된다.

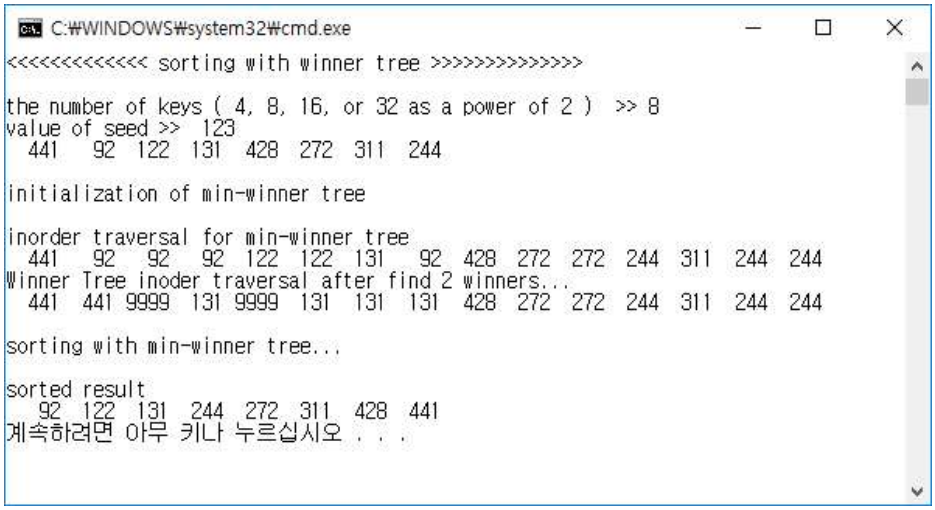
④ 승자트리에 대해 inorder traversal을 수행하여 키 값을 출력한다.

⑤ 승자트리를 사용한 정렬을 수행한다.

※ { 최소키를 sorted 배열에 저장 -> 무한대를 의미하는 임의의 값으로 최소키를 치환 -> 승자트리를 재구성}, 이 과정을 k번 반복함

⑥ 2개의 승자를 출력 후 승자 트리를 inorder traversal을 수행하여 키 값을 출력한다.


C:₩WINDOWS₩system32₩cmd.exe
<<<<<<<<<<<<< sorting with winner tree >>>>>>>>>>>>>
the number of keys (4, 8, 16, or 32 as a power of 2) >> 4
value of seed >> 123
441 92 122 131
initialization of min-winner tree
inorder traversal for min-winner tree
441 92 92 92 122 122 131
Winner Tree inoder traversal after find 2 winners...
441 441 9999 131 9999 131 131
sorting with min-winner tree...
sorted result
92 122 131 441
계속하려면 아무 키나 누르십시오 . . .


C:₩WINDOWS₩system32₩cmd.exe
<<<<<<<<<<<<< sorting with winner tree >>>>>>>>>>>>>
the number of keys (4, 8, 16, or 32 as a power of 2) >> 8
value of seed >> 123
441 92 122 131 428 272 311 244
initialization of min-winner tree
inorder traversal for min-winner tree
441 92 92 92 122 122 131 92 428 272 272 244 311 244 244
Winner Tree inoder traversal after find 2 winners...
441 441 9999 131 9999 131 131 131 428 272 272 244 311 244 244
sorting with min-winner tree...
sorted result
92 122 131 244 272 311 428 441
계속하려면 아무 키나 누르십시오 . . .

3. [승자트리를 이용한 정렬] 10개의 정수형 원소로 구성된 k 개의 run에 대하여 승자트리(winner tree)를 이용하여 정렬을 수행하라. 이때 k는 2의 누승 (power of 2)임을 가정하라. 단 승자는 키 값이 작은 노드이다.

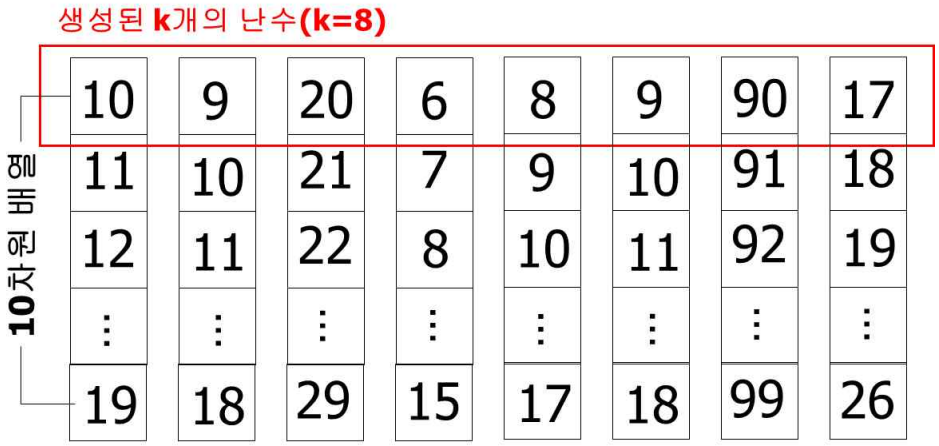
(1) 실행순서

① 난수생성을 위한 seed와 k를 입력받는다.

② k x 10의 2차원 배열을 생성한다.

③ 1~500 사이에서 발생시킨 k 개의 난수를 key로 사용하여 각 레코드에 k부터 k+9까지의 숫자를 순서대로 배열에 저장한다.

※ 각 key는 중복 가능하다.



k = 8일 때의 난수 생성 후 레코드의 예

④ ②에서 생성한 키 데이터에 대해 **초기 승자트리**를 구성한다.

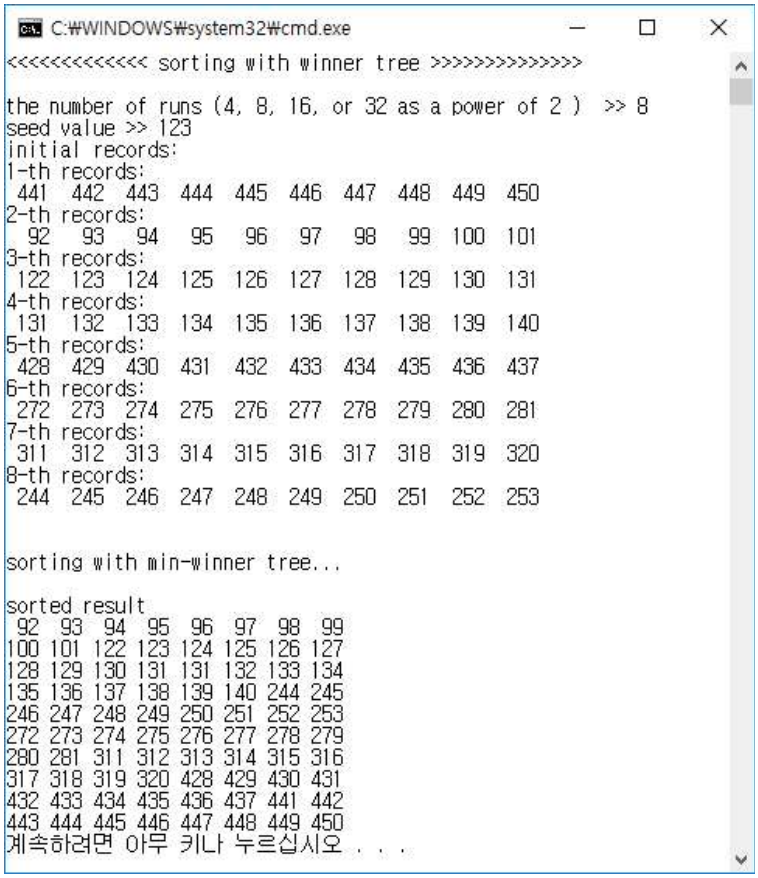
⑤ 승자트리를 사용한 정렬을 수행한다.

※ { 최소키를 sorted 배열에 저장 -> 최소키가 최종 승자로 선택된 레코드에는 다음 키 값으로 nums 배열의 값을 치환(레코드의 키 값을 다 소진한 경우에는 무한대를 의미하는 임의의 값으로 최소키를 치환) -> 승자트리를 재구성 } 이 과정을 k * 10번 반복함

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

⑥ 정렬된 결과를 출력 출력한다.

(2) 실행 예


C:₩WINDOWS₩system32₩cmd.exe
<<<<<<<<<<<< sorting with winner tree >>>>>>>>>>>>
the number of runs (4, 8, 16, or 32 as a power of 2) >> 8
seed value >> 123
initial records:
1-th records:
441 442 443 444 445 446 447 448 449 450
2-th records:
92 93 94 95 96 97 98 99 100 101
3-th records:
122 123 124 125 126 127 128 129 130 131
4-th records:
131 132 133 134 135 136 137 138 139 140
5-th records:
428 429 430 431 432 433 434 435 436 437
6-th records:
272 273 274 275 276 277 278 279 280 281
7-th records:
311 312 313 314 315 316 317 318 319 320
8-th records:
244 245 246 247 248 249 250 251 252 253
sorting with min-winner tree...
sorted result
92 93 94 95 96 97 98 99
100 101 122 123 124 125 126 127
128 129 130 131 131 132 133 134
135 136 137 138 139 140 244 245
246 247 248 249 250 251 252 253
272 273 274 275 276 277 278 279
280 281 311 312 313 314 315 316
317 318 319 320 428 429 430 431
432 433 434 435 436 437 441 442
443 444 445 446 447 448 449 450
계속하려면 아무 키나 누르십시오 . . .

■ 제출 형식

- 공학인증 시스템(ABEEK)에 과제를 올릴 때 제목:
- 1차 제출: 학번 이름 DS-14(1), 2차 제출: 학번 이름 DS-14(2)
- 솔루션 이름 : DS-14
- 프로젝트 이름 : 1, 2, 3
- 실행화면을 캡쳐하여 한글파일에 추가 후 솔루션 폴더에 포함.
- 한글 파일명 : 학번_이름_실습결과.hwp
- 솔루션 폴더를 압축하여 게시판에 제출할 것.
- 솔루션 압축 파일 명:

  1차 제출: 학번_이름_DS_14(1).zip, 2차 제출: 학번_이름_DS_14(2).zip

- 제출은 2회 걸쳐 가능(수정 시간 기준으로 처리)