

PBL 과제 3

1조(2018011872 안승우, 2020080164 강경석)

최대 힙

1. 문제

널리 잘 알려진 자료구조 중 최대 힙이 있다. 최대 힙을 이용하여 다음과 같은 연산을 지원하는 프로그램을 작성하시오.

배열에 자연수 x 를 넣는다.

배열에서 가장 큰 값을 출력하고, 그 값을 배열에서 제거한다.

프로그램은 처음에 비어있는 배열에서 시작하게 된다.

첫째 줄에 연산의 개수 N ($1 \leq N \leq 100,000$)이 주어진다. 다음 N 개의 줄에는 연산에 대한 정보를 나타내는 정수 x 가 주어진다. 만약 x 가 자연수라면 배열에 x 라는 값을 넣는(추가하는) 연산이고, x 가 0이라면 배열에서 가장 큰 값을 출력하고 그 값을 배열에서 제거하는 경우이다. 입력되는 자연수는 231보다 작다.

입력에서 0이 주어진 회수만큼 답을 출력한다. 만약 배열이 비어 있는 경우인데 가장 큰 값을 출력하라고 한 경우에는 0을 출력하면 된다.

2. 핵심 소스코드(주석포함)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#define MAX_ELEMENT 100001

typedef struct {
    int key;
} element;

typedef struct {
    int heap[MAX_ELEMENT]; //값을 저장할 배열 heap을 만들고 heap의 크기를 결정
    //하는 heap_size를 포함하는 구조체
    int heap_size;
} HeapType;

HeapType* create() { //heap을 생성하는 함수
    return (HeapType*)malloc(sizeof(HeapType));
}
```

```

void init(HeapType* h) { //heap을 초기화 시키는 함수
    h->heap_size = 0;
}

int is_empty(HeapType* h) { //heap이 0인 경우에 0을 리턴하는 함수
    if (h->heap_size == 0)
        return 0;
    return 1;
}

void insert_max_heap(HeapType* h, int item) { //값을 삽입하는 함수
    int i;
    i = ++(h->heap_size); //값을 삽입하므로 heap_size의 값을 증가시킨다
    while ((i != 1) && (item > h->heap[i / 2])) { //i가 1이 아니고 부모 노드보다 크면 부모 노드의 값을 자식노드의 위치에 저장한 후 i를 2로 나눠 반복한다.
        h->heap[i] = h->heap[i / 2];
        i /= 2;
    }
    h->heap[i] = item; //주어진 값을 알맞은 위치에 저장
}

int delete_max_heap(HeapType* h) { //값을 삭제하는 함수
    if (is_empty(h) == 0) { //heap이 비어있으면 0 출력
        return 0;
    }
    int parent, child;
    int item, temp;
    item = h->heap[1];
    temp = h->heap[(h->heap_size)--]; //temp에 배열의 맨 마지막 값을 저장한 후에 heap_size를 1 감소시킨다.
    parent = 1;
    child = 2;
    while (child <= h->heap_size) { //child가 heap_size보다 커질때까지 반복
        if ((child < h->heap_size) && //child가 heap_size보다 작고 child노드의 값이 child+1노드의 값보다 작다면 child 값을 증가시킨다.
            (h->heap[child] < h->heap[child + 1]))
            child++;
        if (temp >= h->heap[child]) break; //temp의 값이 child의 값보다 크다면 즉, 맨처음 child의 값이 2였으므로 노드 2,3보다 크다면 break한다.
        h->heap[parent] = h->heap[child]; //1번 노드에 증가시킨 child노드 값을 저장하고 parent를 child로 바꾼뒤 child를 2배 해서 반복한다.
        parent = child;
    }
}

```

```

        child *= 2;
    }
    h->heap[parent] = temp;    //빈공간 parent에 temp의 값을 저장한다.
    return item;
}

int main(void) {
    HeapType* h = create();    //heap 생성
    init(h);    //heap 초기화
    int N, x;
    scanf("%d", &N);    //연산의 개수 입력

    for (int i = 0; i < N; i++) {    //N번 반복
        scanf("%d", &x);    //x의 값 입력
        if (x == 0) {    //x가 0일 때
            printf("%d\n", delete_max_heap(h));    //아니면 heap에서 가장 큰
값 삭제
        }
        else {
            insert_max_heap(h, x);    //x가 자연수 이면 heap에 x 삽입
        }
    }
    return 0;
}

```

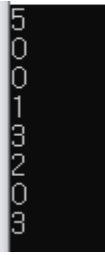
3.알고리즘 설명

이 알고리즘은 최대 힙으로 최대 힙은 최대 트리이면서 완전이진트리이다. 최대 힙은 각 노드의 키값이 자식 노드의 키값 보다 작지 않은 트리이다. 또한 완전이진트리이므로 노드를 삽입할 때 왼쪽부터 삽입하는 특징을 가지고 있다.

4.해결방법 및 절차

배열에서 최대 힙을 이용하여 자연수를 삽입하고 삭제하는 프로그램을 작성하라고 했으므로 최대 힙을 사용하는데 여기서 x=0일 때 삭제하고 x가 자연수이면 삽입하라고 했으므로 조건에 맞춰 main함수를 작성해 준다.(단 x에 값이 없으면 0이 출력되도록 한다.

5.실행화면

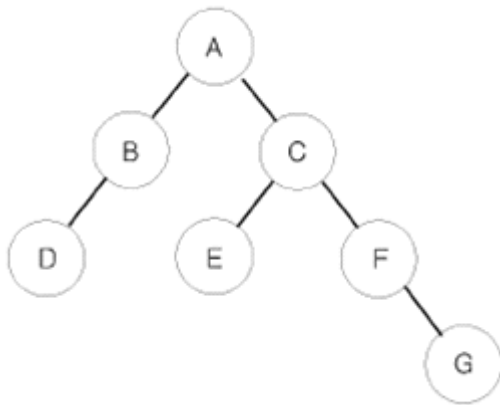


보면 연산 개수를 5개로 해준 후 아무것도 안 들어있을 때 0을 입력하면 0이 출력되고 1 3 2를 입력한 후 0을 입력하니 최댓값인 3이 삭제되는 것을 볼 수 있다.

트리 순회

1. 문제

이진 트리를 입력받아 전위 순회(preorder traversal), 중위 순회(inorder traversal), 후위 순회(postorder traversal)한 결과를 출력하는 프로그램을 작성하시오.



예를 들어 위와 같은 이진 트리가 입력되면,

전위 순회한 결과 : ABDCEFG // (루트) (왼쪽 자식) (오른쪽 자식)

중위 순회한 결과 : DBAECFG // (왼쪽 자식) (루트) (오른쪽 자식)

후위 순회한 결과 : DBEGFCA // (왼쪽 자식) (오른쪽 자식) (루트)

가 된다.

첫째 줄에는 이진 트리의 노드의 개수 $N(1 \leq N \leq 26)$ 이 주어진다. 둘째 줄부터 N 개의 줄에 걸쳐 각 노드와 그의 왼쪽 자식 노드, 오른쪽 자식 노드가 주어진다. 노드의 이름은 A부터 차례대로 영문자 대문자로 매겨지며, 항상 A가 루트 노드가 된다. 자식 노드가 없는 경우에는 .으로 표현된다.

첫째 줄에 전위 순회, 둘째 줄에 중위 순회, 셋째 줄에 후위 순회한 결과를 출력한다. 각 줄에 N 개의 알파벳을 공백 없이 출력하면 된다.

2.핵심 소스코드(주석포함)

```
#define _CRT_SECURE_NO_WARNINGS //scanf 반환값 무시
#include <stdio.h>
#include <stdlib.h>

typedef struct Node { //문자 값 data와 2개의 링크를 가지고 있는 구조체
    char data;
    struct Node* left, * right;
}TreeNode;

TreeNode* NewNode(char data) { //data를 값으로 가지는 node 생성
    TreeNode* New;
    New = (TreeNode*)malloc(sizeof(TreeNode));
    New->data = data;
    New->left = New->right = NULL;
    return New;
}

TreeNode* search_Node(TreeNode* H, char data) { //탐색 함수
    if (H != NULL) {
        if (H->data == data) {
            return H; //탐색에 성공
        }
        else {
            TreeNode* tmp = search_Node(H->left, data); //왼쪽 자식 탐색
            if (tmp != NULL) {
                return tmp;
            }

            return search_Node(H->right, data); //오른쪽 자식 탐색
        }
    }
    return NULL; //탐색에 실패하면 NULL 반환
}

void insert_Node(TreeNode* H, char A, char B, char C) {
    H->data = A; //노드에 A삽입
    if (B != '.') {
        H->left = NewNode(B); //왼쪽 자식에 node B생성
    }
    if (C != '.') {
        H->right = NewNode(C); //오른쪽 자식에 node C생성
    }
}
```

```

preorder(TreeNode* root) { //전위 순회
    if (root) {
        printf("%c", root->data); // 노드 방문
        preorder(root->left); // 왼쪽서브트리 순회
        preorder(root->right); // 오른쪽서브트리 순회
    }
}

inorder(TreeNode* root) { //중위 순회
    if (root) {
        inorder(root->left); // 왼쪽서브트리 순회
        printf("%c", root->data); // 노드 방문
        inorder(root->right); // 오른쪽서브트리 순회
    }
}

postorder(TreeNode* root) { //후위 순회
    if (root) {
        postorder(root->left); // 왼쪽 서브 트리 순회
        postorder(root->right); // 오른쪽 서브 트리 순회
        printf("%c", root->data); // 노드 방문
    }
}

int main() {
    TreeNode* H = NewNode(NULL);
    TreeNode* tmp;
    int N;
    int i;
    scanf("%d", &N); //노드의 개수 입력
    getchar(); //버퍼에 남아있는 \n 제거
    for (i = 0; i < N; i++) {
        char A, B, C;
        scanf("%c %c %c", &A, &B, &C);
        getchar(); //버퍼에 남아있는 \n 제거
        tmp = search_Node(H, A); //A 탐색
        if (tmp != NULL) //A가 있으면 tmp에 노드 삽입
            insert_Node(tmp, A, B, C);
        else //A가 없으면 h에 노드 삽입
            insert_Node(H, A, B, C);
    }
    preorder(H); //전위 순회
}

```

```

printf("\n");
inorder(H);    //중위 순회
printf("\n");
postorder(H);  //후위 순회
}

```

3.알고리즘 설명

이 알고리즘은 포인터를 이용하여 부모 노드가 자식 노드를 가리키게 해 트리를 만든 것이다. 트리를 순회하고자 하므로 3가지의 트리 순회 함수가 있고 노드를 입력받아 삽입해야 하므로 노드 생성 함수와 노드 삽입함수 노드 탐색함수가 있다.

4.해결방법 및 절차

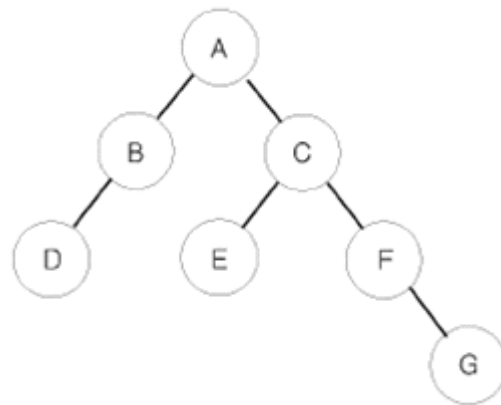
노드의 개수를 입력받고 노드를 입력받아야 하는데 그러면 중간에 \n이 남아있으므로 getchar() 함수를 사용해 지워준다. 문자를 입력받은 뒤 탐색 함수를 이용해 부모 노드가 있으면 그 부모 노드 밑에 자식 노드를 삽입해주고 없으면 그냥 h에다 삽입해 준다. 노드를 입력받은 뒤 전위 순회 중위 순회 후위 순회를 실행시켜 준다.

5.실행 결과

```

7
A B C
B D . F
C E F
E . .
F . G
D . .
G . .
ABDCEFG
DBACEFG
DBEGFCA

```



이 그림대로 전위 순회 중위 순회 후위 순회를 해보면

전위 순회: ABDCEFG

중위 순회: DBACEFG

후위 순회: DBEGFCA 이고

위의 값과 같은 것을 알 수 있다.

하노이 탑 이동 순서

1.문제

세 개의 장대가 있고 첫 번째 장대에는 반경이 서로 다른 n 개의 원판이 쌓여 있다. 각 원판은 반경이 큰 순서대로 쌓여있다. 이제 수도승들이 다음 규칙에 따라 첫 번째 장대에서 세 번째 장대로 옮기려 한다.

한 번에 한 개의 원판만을 다른 탑으로 옮길 수 있다.

쌓아 놓은 원판은 항상 위의 것이 아래의 것보다 작아야 한다.

이 작업을 수행하는데 필요한 이동 순서를 출력하는 프로그램을 작성하라. 단, 이동 횟수는 최소가 되어야 한다.

첫째 줄에 첫 번째 장대에 쌓인 원판의 개수 N ($1 \leq N \leq 20$)이 주어진다.

첫째 줄에 옮긴 횟수 K 를 출력한다.

두 번째 줄부터 수행 과정을 출력한다. 두 번째 줄부터 K 개의 줄에 걸쳐 두 정수 A B 를 빈칸을 사이에 두고 출력하는데, 이는 A 번째 탑의 가장 위에 있는 원판을 B 번째 탑의 가장 위로 옮긴다는 뜻이다.

2.핵심 소스 코드(주석 포함)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include<math.h>
void hanoi_tower(int n, int from, int tmp, int to) { // 막대 from에 쌓여있는 n개의 원판을
    // 막대 tmp를 사용하여 막대 to로 옮긴다.
    if (n == 1) { //가장 위에 있는 원판을 from에서 to로 옮긴다.
        printf("%d %d\n", from, to);
    }
    else {
        hanoi_tower(n - 1, from, to, tmp); //tmp를 임시버퍼로 사용해 원판을
        //from에서 to로 옮긴다.
        printf("%d %d\n", from, to); //가장 큰 원판을 from에서 to로 옮긴다.
        hanoi_tower(n - 1, tmp, from, to); //tmp를 임시버퍼로 사용해 원판을
        //from에서 to로 옮긴다.
    }
}
int main(void) {
    int N,k;
    scanf("%d", &N);

    k = pow(2, N) - 1; //하노이 탑의 옮긴 횟수는 2^n-1이다.
    printf("%d\n", k);
    hanoi_tower(N, 1, 2, 3); //N개의 하노이 탑을 옮긴다.
    return 0;
}
```



```
}
```

3.알고리즘 설명

재귀를 이용해서 하노이 탑의 이동을 구현했다.

4.해결방법 및 절차

처음에 하노이 탑의 옮긴 횟수를 출력해야 하므로 pow함수를 이용해 출력해주고 hanoi_tower 함수를 이용해 주어진 조건에 맞게 원판이 이동하도록 한다.

5.실행 결과

```
3
7
1 3
1 2
3 2
1 3
2 1
2 3
1 3
```

원판이 3개이면 총 7회 움직이고 위에 출력된 데로 원판을 움직여 보면 하노이 탑을 이동 시킬 수 있는 것을 알 수 있다.

수 정렬하기 (2750)

(1) 소개

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N ($1 \leq N \leq 1,000$)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

예제 입력 1

```
5
5
2
3
4
1
```

예제 출력 1

```
1
```

2
3
4
5

(2) Source Code(소스코드) 및 설명

```
// 수 정렬하기
// 2750
// 강경석작성

#include<stdio.h>
int main(void) {
    int arr[1000];
    int n, i, j, temp;

    // 입력.
    scanf("%d", &n);
    for(i=0; i<n; i++) {
        scanf("%d", &arr[i]);
    }
    // 검사.
    for(i=0; i<n; i++) {
        for(j=0; j<n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }

    for(i=0; i<n; i++) {
        printf("%d\n", arr[i]);
    }

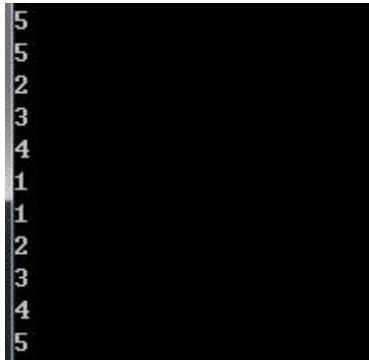
    return 0;
}
```

<알고리즘 및 해결방법 설명>

본 문제에서는 정수를 n 개를 입력을 받고 오름차순으로 정렬후 수를 출력하는 문제이다.

정렬 방법으로는 자료구조중 버블정렬을 사용하여 index n 과 $n-1$ 요소를 비교하여 정렬하였다. 이후 정렬된 array를 출력해주었다.

(3) Screenshot of the Results



(4) 문제를 해결하면서 느낀점

제약조건이 1000 미만이라서 1 초라는 시간의 제약조건에 크게 제약을 받지 않았던 것 같다. 그래서 쉬운 버블정렬법을 이용하여 정렬하였다.

수 정렬하기 2 (2751)

(1) 소개

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	256 MB	123514	32450	22057	30.019%

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N ($1 \leq N \leq 1,000,000$)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

예제 입력 1

```
5
5
4
3
2
1
```

예제 출력 1

```
1
2
3
4
5
```

(2) Source Code(소스코드) 및 설명

```
// 수 정렬하기
// 2750
// 강경석작성
```

```
#include<stdio.h>
int main(void) {
    int arr[1000];
    int n, i, j, temp;
```

```

// 입력.
scanf("%d", &n);
for(i=0; i<n; i++) {
    scanf("%d", &arr[i]);
}
// 검사.
for(i=0; i<n; i++) {
    for(j=0; j<n-i-1; j++) {
        if(arr[j] > arr[j+1]) {
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}

for(i=0; i<n; i++) {
    printf("%d\n", arr[i]);
}

return 0;
}

```

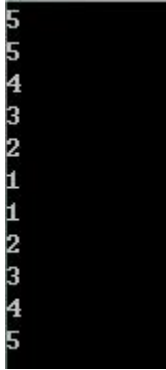
<알고리즘 및 해결방법 설명>

본 문제에서는 정수를 n개를 입력을 받고 오름차순으로 정렬후 수를 출력하는 문제이다.

정렬 방법으로는 자료구조중 mergesort(병합정렬)을 사용하였다.

merge sort로 정렬이후 출력한다.

(3) Screenshot of the Results



```
5
5
4
3
2
1
1
2
3
4
5
```

(4) 문제를 해결하면서 느낀점

본 문제는 입력이 1,000,000 까지 받아야 하는 제약조건이 있으므로 시간제한 2 초를 고려하여야 한다.

$O(N^2)$ 시간복잡도를 가지는 정렬 알고리즘을 사용을 하면 시간초과가 발생이 된다.

처음에는 이 사실을 모르고 버블정렬, 선택 정렬 등을 사용하여 풀고자 하였지만, 시간초과가 발생이 되어 퀵정렬을 통해 풀고자 하였다.

하지만 퀵정렬 알고리즘 또한 시간초과가 발생이 되었다. 이 부분에서 며칠 동안 헤매었던 것 같다. 이후 병합정렬을 통해 해결을 하였다.

퀵 정렬이 왜 시간초과가 발생이 되는지 원인을 분석한 결과 퀵정렬의 최악의 경우의 시간복잡도 때문이었다. 퀵정렬은 $O(N\log N)$ 의 평균 시간복잡도를 가지지만 최악의 경우는 $O(N^2)$ 이기 때문에 시간초과가 발생이 된다.

K 번째 수 11004

(1) 소개

문제

수 N 개 A_1, A_2, \dots, A_N 이 주어진다. A 를 오름차순 정렬했을 때, 앞에서부터 K 번째 있는 수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N ($1 \leq N \leq 5,000,000$)과 K ($1 \leq K \leq N$)이 주어진다.

둘째 줄에는 A_1, A_2, \dots, A_N 이 주어진다. ($-109 \leq A_i \leq 109$)

출력

A 를 정렬했을 때, 앞에서부터 K 번째 있는 수를 출력한다.

예제 입력 1

5 2

4 1 2 3 5

예제 출력 1

2

(2) Source Code(소스코드) 및 설명

```
// 백준수정렬
```

```
// 11004
```

```
// 강경석작성
```

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

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

```
void MergeTwoArea(int arr[], int left, int mid, int right)
```

```
{
```

```
int fIdx = left
```

```
int rIdx = mid + 1;
```

```
int i;
```

```
int* sortArr = (int*)malloc(sizeof(int) * (right + 1));
```

```
int sIdx = left
```

```
while (fIdx <= mid && rIdx <= right)
```

```
{
```

```

if (arr[fIdx] <= arr[rIdx]) {
    sortArr[sIdx] = arr[fIdx++];
}
else
{
    sortArr[sIdx] = arr[rIdx++];
}
sIdx++;
}

```

```

if (fIdx > mid)
{
    for (i = rIdx; i <= right i++, sIdx++) {
        sortArr[sIdx] = arr[i];
    }
}
else
{
    for (i = fIdx; i <= mid i++, sIdx++) {
        sortArr[sIdx] = arr[i];
    }
}

```

```

for (i = left i <= right i++) {
    arr[i] = sortArr[i];
}
free(sortArr);
}

```

```

void MergeSort(int arr[], int left, int right) {
    int mid;

```

```

    if (left < right)
    {
        mid = (left + right) / 2;

```

```

        MergeSort(arr, left, mid);
        MergeSort(arr, mid + 1, right);

```

```

        MergeTwoArea(arr, left, mid, right);
    }

```



```

}

int arr[5000000] = { 0 };
int main(void)
{

int i, n, index;

scanf("%d %d", &n, &index);

for (i = 0; i < n; i++)
{
scanf("%d", &arr[i]);
}

MergeSort(arr, 0, n - 1);

printf("%d", arr[index - 1]);

return 0;
}

```

<알고리즘 및 해결방법 설명>

본 문제에서는 정수를 n개를 입력을 받고 오름차순으로 정렬후 k번째 수를 출력하는 문제이다.

정렬 방법으로는 자료구조중 mergesort(병합정렬)을 사용하였다.
merge sort로 정렬이후 출력한다.

(3) Screenshot of the Results

백준 채점결과 성공처리 되었다.



```

5 2
4 1 2 3 5
2

```

(4) 문제를 해결하면서 느낀점

정렬방법에는 다양한 방법이 있다. 그중에서 수업시간에 배운 병합정렬을 이용하여 문제를 해결해보았다. 기존에 아는 선택정렬, 버블정렬등보다는 구현이 힘들지만, 월등히 빠른 빅오시간을 보이는 합병정렬을 통해 구현을 하니 뿌듯하였다. 본 문제에서는 연산을 많이 하는 부분이 아니라 크게 와닿지는 않았지만 기존에 아는 정렬법 이외에 효과적인 정렬법을 통해 정렬을 할수 있어 뿌듯하였고, 다음에 기회가 된다면 시간을 비교하여 체감해보고 싶다.

깃허브 주소:

<https://github.com/KANGKYEONGSEOK/Data-Structures-and-Algorithms.git>

<https://github.com/jayce288/Data-structures-and-algorithms.git>