[알고리즘] 우선순위 큐

imhyun · 방금 전 · 비공개

알고리즘 우선순위 큐

우선순위 큐

https://www.acmicpc.net/problem/2696

> 문제 핵심 : 수열이 입력될 때마다 해당 수열을 읽고, 홀수번째 수를 읽을 때 지금까지 입력받은 값의 중앙값을 출력하는 문제이다. 중앙값이란, 주어진 값들을 크기의 순서대로 정렬했을 때 가장 중앙에 위치하는 값을 의미한다. 따라서, **우선순위 큐를 정렬**하는 문제이다.

방법

테스트 케이스의 개수에 따라 수열이 주어지면, 중앙값의 개수를 출력하고 중앙값을 차례대로 공백으로 구분하여 출력하는 것을 반복한다. (이때, 중앙값 출력은 한 줄에 최대 10개씩이다.)

1. 원소 개수와 중앙값 개수의 관계

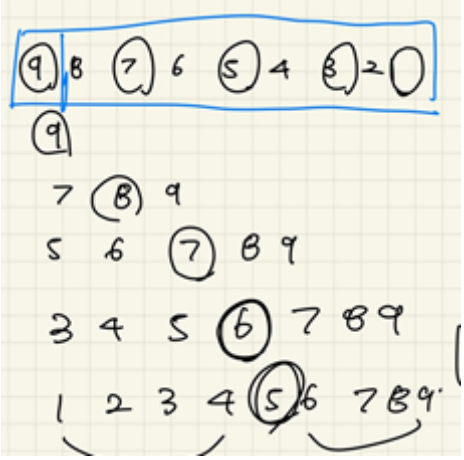
$$중앙값개수 = (원소개수 + 1)/2$$

2. 우선 순위 큐의 정렬
   - 현재 홀수번째일 때 중앙값 출력
   - 중앙값의 위치

$$중앙값위치 = (1 + 정렬횟수)/2$$

   - 열번째 출력인지 카운트를 통해 줄 구분

위에서 제시한 방법을 따라 우선순위 큐 정렬을 진행해주면 끝이 난다. 이 때, 우선순위 큐 정렬에는 크게 선택 정렬과 삽입 정렬이 있다. 추가적으로, 주어진 순서 내에서 외부의 제 2 공간을 사용하지 않고 제자리에서 수행한다면 메모리를 아낄 수 있는 제자리 정렬 방법이 있다.



예제 입력 케이스 과정을 직접 순서에 따라 적어주었을 때, 배열 중 왼쪽은 정렬된 상태의 힙을 가상의 공간으로 생각했고, 나머지는 정렬되지 않는 리스트로 생각해주었다. 즉 가상의 공간인 힙은 들어올 때 정렬된 상태로 생각해주었다. 따라서 **제자리 삽입 정렬**을 통해 구현하기로 하였다.

제자리 삽입 정렬 알고리즘

```
Alg inPlaceInsertionSort(A)
    input array A of n keys
    ouput sorted array A

1. for pass<-1 to n-1
        save<-A[pass]
        while((j>=0)&(A[j]>save))
            A[j+1]<-A[j]
            j<-j-1
        A[j+1]<-save
2. return
```

이 때 강조할 점은 자리를 ***뒤로 밀어줄 때마다 save를 넣는 것***이 아니라, 마지막에 반복문이 끝나고 **최종적으로** 넣는다는 것이다. 예시로 들면, 동물의 키를 재는 알고리즘이라면 그 동물을 계속 cage에 넣고 빼고 하는 것이 아니라, 한 동물의 키를 미리 재놓고, 줄자로만 판단하는 것이라고 생각하면 된다.

```c
#include <stdio.h>
#include <stdlib.h>
#pragma warning(disable:4996)

void makeArray(int*, int);
void printArray(int*, int);
void printMedianNum(int);
void printMedian(int *, int);

int main() {
    int T, M;
    int* A = NULL;

    int i, j,k;
    int save;
    int cnt = 1;
    int Medianloc = 0;

    //테스트 케이스의 개수
    scanf("%d", &T);


    //테스트 케이스만큼 반복
    for (k = 0; k < T; k++) {
        scanf("%d", &M);

        //M 개의 정보를 저장하는 연결 리스트 생성
        A = (int*)malloc(sizeof(int) * M);
        makeArray(A, M);

        printMedianNum(M);

        //첫번째는 반드시 출력하게 됨.
        printf("%d ", A[0]);

        //제자리 삽입 정렬
        cnt = 1;
        for (i = 1; i <= M - 1; i++) {
            j = i - 1;
            save = A[i];

            while (j >= 0 && save < A[j]) { //추가될 아이 비교하기
                A[j + 1] = A[j];
                j--;
            }

            A[j + 1] = save;

            //중앙값 찾아주기
            if ((i + 1) % 2 != 0) {
                cnt++;
                Medianloc = (0 + i) / 2;
                printf("%d ", A[Medianloc]);
                if (cnt % 10 == 0)
```

```
        printf("\n");



        free(A);

    }

    return 0;
}

void makeArray(int* A, int n) { //난수발생을 통해 초기화
    int i;
    int N;

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


void printArray(int* A, int n) {  //정렬 결과 출력.
    for (int i = 0; i < n; i++) {
        printf(" %d", A[i]);
    }
}

void printMedianNum(int M) {
    int Medianloc = (1 + M) / 2;

    printf("%d\n", Medianloc);
}
```

박시현

이전 포스트
[알고리즘] 연결 리스트와 큐

댓글을 작성하세요

댓글 작성