백준 | 2696번 중앙값 구하기

khi8930 · 방금 전

백준 스터디

2023 Algorithm

목록 보기 5/5

- 소요시간 : 2시간 이상
- 자료 참고 여부 : △ (힙에 대한 내용은 찾아봄 그러나 문제 풀이에 직접적 이용 X)
- 체감 난이도 : 上 → 中

📌 2696번 문제

2696번 제출 맞힌 사람 숏코딩 재채점 결과 채점 현황 내 제출 난이도 기여 질문 게시판

중앙값 구하기 성공 다국어

☆ 한국어

골드 II

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	7280	3393	2672	49.190%

문제

어떤 수열을 읽고, 홀수번째 수를 읽을 때 마다, 지금까지 입력받은 값의 중앙값을 출력하는 프로그램을 작성하시오.

예를 들어, 수열이 1, 5, 4, 3, 2 이면, 홀수번째 수는 1번째 수, 3번째 수, 5번째 수이고, 1번째 수를 읽었을 때 중앙값은 1, 3번째 수를 읽었을 때는 4, 5번째 수를 읽었을 때는 3이다.

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

1. 미친듯한 삽질 (배열로 구현 → 힙으로 구현해보려 시도 → 힙으로 선택정렬 구현 시도 → 힙으로 삽입 정렬 구현)
2. 시간초과의 늪 (9번의 시도 중 5번이 시간초과)
3. 너무 어렵게 생각했다. 오히려 조금 더 편하게 생각했으면 빨리 풀렸을텐데

코드 전문

```
#pragma warning(disable:4996)
#pragma warning(disable:4013)
#include<stdio.h>
#include<string.h>
#include <stdlib.h>
#define MAX_ELEMENT 10001
```

```c
typedef struct {
    int key;
} element;

typedef struct {
    element heap[MAX_ELEMENT];
    int heap_size;
} HeapType;

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// 선택 정렬 → 사용 Xxx
void NodeSwap(HeapType* h, int NodeSite) {
    int i = NodeSite; // 노드 자리
    int left = i * 2;
    int right = i * 2 + 1;

    int largest = i;

    if (left <= h->heap_size && h->heap[left].key > h->heap[largest].key) {
        largest = left;
    }
    if (right <= h->heap_size && h->heap[right].key > h->heap[largest].key) {
        largest = right;
    }

    if (largest != i) {
        swap(&(h->heap[i].key), &(h->heap[largest].key));
        for (int j = 0; j < (h->heap_size); j++) {
            printf("%d ", h->heap[j].key);
        }
        printf("\n");
        NodeSwap(h, largest);
    }
}

// 삽입정렬
void  NodeAdd(HeapType* h, int n) {

    for (int i = 0; i < n; i++) {
        if (h->heap[n-1].key > h->heap[i].key) {
            swap(&(h->heap[n-1].key), &(h->heap[i].key));
        }
    }
}

int main() {


    int n, m;
    int cnt = 0;

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &m);
        printf("%d\n", (m / 2) + 1);

        HeapType* p = (HeapType*)malloc(sizeof(HeapType));

        for (int j = 0; j < m; j++) {
            scanf("%d", &(p->heap[j].key));
        }

        for (int j = 0; j < m; j++) {
            NodeAdd(p, j+1);
            if (j % 2 == 0) {
                printf("%d ", p->heap[j / 2].key);
                cnt++;
                if (cnt == 10) {
                    printf("\n");
```

```
                    cnt = 0;
                }
            }
        }

        cnt = 0;
        free(p);
        printf("\n");
    }

}
```

코드 해석

```
typedef struct {
    int key;
} element;

typedef struct {
    element heap[MAX_ELEMENT];
    int heap_size;
} HeapType;
```

- 힙의 구조를 사용했다.
  - struct element : 각 노드의 값을 저장
  - struct HeapType : 전체 힙 구조체와 힙의 크기를 저장
    - 최대 10001개의 값을 가질 수 있는 struct element heap 배열
    - heap 안에 몇 개의 값이 들어가는지 저장하는 int heap_size

```
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// 삽입정렬
void  NodeAdd(HeapType* h, int n) {

    for (int i = 0; i < n; i++) {
        if (h->heap[n-1].key > h->heap[i].key) {
            swap(&(h->heap[n-1].key), &(h->heap[i].key));
        }
    }
}
```

- void swap() : 포인터 주소를 바꿔줘야 두 값이 제대로 바뀌기 때문에 포인터 값을 받아 교환하는 함수
- void NodeAdd() : 힙으로 삽입정렬 구현
  - 현재 받은 힙(h)의 n번째 값의 key가 앞선 n-1번째 key보다 크다면 두 값 교환

```
int main() {


    int n, m;
    int cnt = 0;

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &m);
```

```
        printf("%d\n", (m / 2) + 1);

        HeapType* p = (HeapType*)malloc(sizeof(HeapType));

        for (int j = 0; j < m; j++) {
            scanf("%d", &(p->heap[j].key));
        }

        for (int j = 0; j < m; j++) {
            NodeAdd(p, j+1);
            if (j % 2 == 0) {
                printf("%d ", p->heap[j / 2].key);
                cnt++;
                if (cnt == 10) {
                    printf("\n");
                    cnt = 0;
                }
            }
        }

        cnt = 0;
        free(p);
        printf("\n");
    }
}
```

- m 값은 항상 홀수이기 때문에, 나오는 중앙값의 개수는 항상 m/2개
- 힙 작동 Logic
    1. 힙 p를 할당 받는다.
    2. m만큼 힙 p의 key 값을 입력 받는다.
    3. for문을 이용하여 m번의 NodeAdd(삽입정렬)을 실행한다.
    4. (j%2 == 0) 일시에, 즉 홀수번째 요소가 NodeAdd를 실행하고 나면 지금까지 정렬된 힙의 키 값 중 중앙값을 출력한다.
        4-1. 만약 출력하는 개수가 10개가 넘어간다면 엔터키를 출력한다.

📌 총평

너무 많은 케이스들을 시도했다. 생각하기 싫어서 무작정 시도해보고 결과를 보는 버릇을 고쳐야한다. 최대한 효율적이게 문제 풀려고 노력해야겠다.

우선순위 큐를 배열형식으로만 구현해봐서 '힙'이 뭔지 공부하고 문제를 풀다보니 오래걸렸다. 실행시간이 1초로 한정되어있어 실행시간을 줄이기 위해서는 힙을 시도해보지 않을 수 없었다.

자료구조에서도 삽입 정렬을 무작정 버블정렬 돌려서 풀었는데, 그것도 너무 비효율적이다. 위 방법을 이용해서 배열을 사용한 삽입정렬도 구현해볼것 !

수 많은 힙에 대한 구조와 구현법을 찾아봤지만 뭐가 맞는지 아직 모르겠다.. 이번 주 자료구조 실습하면서 좀 더 알아볼 것 ..

Hyangim

코딩 공부 중인 대학생 🤦‍♀️

이전 포스트
백준 | 1991번 트리 순회

0개의 댓글

댓글을 작성하세요

댓글 작성