백준 2075: N번째 큰 수

zeong · 방금 전

c언어 백준 알고리즘스터디 힙 힙정렬

백준 2075 N보다 큰 수

[문제 요약]

12	7	9	15	5
13	8	11	19	6
21	10	26	31	16
48	14	28	35	25
52	20	32	41	49

다음과 같은 표가 있을 때, N번째 큰 수를 찾기.

[주의 사항]

1. 표에 적힌 수는 -10억보다 크거나 같고, 10억보다 작거나 같은 정수
2. 시간제한이 1초임
3. N의 범위는 1 ≤ N ≤ 1,500
4. 1번 주의사항을 유의해서 푸는 것이 중요함 -> 음수 입출력 신경쓰기

[개념]

① ② ③

@ 힙의 개념

완전 이진 트리로, 우선순위 큐를 위하여 만들어진 자료구조
여러 값들 중에서 최댓값이나 최솟값을 빠르게 찾아내도록 만들어짐.
힙은 일종의 반정렬 상태를 유지함

큰 값이 상위 레벨에 있고 작은 값이 하위 레벨에 있다는 정도(=트리)
힙 트리에서는 중복된 값을 허용함. (이진 탐색 트리에서는 중복된 값을 허용하지 않음)

@ 힙의 종류

① 최대 힙
key (부모 노드) >= key (자식 노드)

② 최소 힙
key(부모 노드) <= key(자식 노드)

@ 힙(heap)의 구현

표준적으로 힙을 배열로 저장함
구현을 쉽게 하기 위하여 배열의 첫 번째 인덱스인 0은 사용되지 않음 H[0]
특정 위치의 노드 번호는 새로운 노드가 추가되어도 변하지 않음

@ 힙에서의 부모 노드와 자식 노드의 관계

왼쪽 자식의 인덱스 = (부모의 인덱스) *2*
오른쪽 자식의 인덱스 = (부모의 인덱스) 2 + 1
부모의 인덱스 = (자식의 인덱스) / 2

[문제 해결 과정]

이 문제도 시간제한이 1초이기 때문에, 기본적인 방식으로는 풀 수 없는 문제라고 생각했다.
힙으로 푸는 문제여서, 처음에는 힙과 배열을 이용해 문제를 풀었지만 시간초과가 발생했다.
그래서 나는 문제를 다시 풀어보았다.

재출을 했는데 틀렸습니다가 떴다.
다른 예시들은 다 잘 출력되어서 혹시하고 음수값을 넣어봤는데, 그게 문제였다. 테스트 케이스로 2, 0 -1 -2 -3을 넣어봤는데 -2와 -3은 잘 인식되는 반면, -1은 계속 0으로 인식되는 것이었다.
0 앞뒤로 입력하는 숫자가 잘 인식되지 않는 것 같다. (사실 잘 모르겠음)
틀렸습니다를 보고 코드를 세부적으로 출력해가며 어느 부분에서 오류가 발생했는지를 찾게 되었고, downHeap에서 오류가 발생했다는 것을 알게되었다.
이 문제를 해결하자 또 다른 문제가 발생했다.

N이 3이상이 되면 다른 숫자가 출력되는 것이었다. 그냥 아예
N이 5일 때가 맞으면, N이 4일 때가 틀리고,

N이 3일 때가 맞으면, N이 5일 때가 틀리고
아주 다 제멋대로다..
계속 고쳐도 뭐가 문제인지 잘 모르겠다.

하루의 절반을 갈아넣어 결과를 얻어냈다.
하지만 난 아직도 뭐가 문제였는지 잘 모르겠다.
이것저것 고치다 보니까 답이 나왔다.
맞았습니다를 봤지만 조금 찝찝하다.
다음에 다시 풀어봐야겠다..

[코드]

```
//2075_5.c -오류 발생 후 버린 코드들 중 가장 근접한 코드

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

int N, * H, cnt = 0, c;

void printHeap() {
    //출력 예시 확인
    for (int i = 1; i <= N * N; i++) {
        printf(" %d", H[i]);
    }
}

int removeMax() {
    int key = H[1];
    H[1] = H[N];
    //printHeap();
    //printf("\n####%d %d \n", N, c);
    //downHeap(1);
    N--;
    //printHeap();
    //c--;
    //printf("\n/////////////////////\n");
    downHeap(1);
    return key;
}

int upHeap(int i) {
    int tmp;
    if (i == 1) {
        return;
    }
    int p = i / 2;
    if (H[i] <= H[p]) {
        return;
    }
    else {
        tmp = H[i];
        H[i] = H[p];
        H[p] = tmp;
    }
```

```
    upHeap(p);
}

int downHeap(int i) {
    int l = 2 * i;
    int r = 2 * i + 1;
    int tmp;
    if (l > N) {
        return;
    }
    int big = l;


    if ((H[r] > H[big]) && (N >= big)) {
        big = r;
    }


    //printf("//// %d %d %d %d %d %d\n", i, l, r, big, cnt, N);
    //printf("/ %d %d %d %d\n", H[i], H[l], H[big], H[r]);
    //printHeap();


    if (H[i] < H[big]) {
        tmp = H[i];
        H[i] = H[big];
        H[big] = tmp;
    }

    //printf("\n/ %d %d\n", H[i], H[big]);
    //printHeap();
    //printf("\n\n");

    downHeap(big);
}


void rBuildHeap(int i) {
    //재귀 방식으로 상향식 힙 생성
    if (i > N) {
        return;
    }
    rBuildHeap(2 * i);
    rBuildHeap(2 * i + 1);
    downHeap(i);
    return;
}

int buildHeap() {
    for (int i = N; i > 0; i--) {
        downHeap(i);
        //printf("///%d %d\n", H[i], i);
    }
    return;
}



int main() {
    scanf("%d", &N);
    H = (int*)malloc(sizeof(int) * (N * N));
```

```
        int B = N, num, f;
        c = N * N;
        for (int i = 1; i <= N * N; i++) {
            cnt++;
            scanf("%d", &H[cnt]);
            //cnt++;
            upHeap(cnt);
        }
        printHeap();
        printf("\n");

        //rBuildHeap(1);
        //buildHeap();

        //printf("\n");
        //printHeap();
        //printf("\n//%d\n", B);

        for (int i = 1; i <= B; i++) {
            f = removeMax();
            printHeap();
            printf("\n//%d\n", i);
            printf("///%d %d %d\n", N, B, f);
            if (i == B) {
                printf("%d", f);
                break;
            }
        }
        //printf("\n");
        //printHeap();
        //printf("%d", H[1]);
    }


    //2075_8_final.c -맞았습니다
    #include <stdio.h>
    #include<string.h>
    #include<stdlib.h>
    #pragma warning(disable:4996)

    int *H, n, cnt = 0;

    int removeMax() {
        int key = H[1];
        H[1] = H[n];
        n--;
        downHeap(1);
        return key;
    }

    int upHeap(int i) {
        int tmp;
        if (i == 1) {
            return;
        }
        int p = i / 2;
        if (H[i] <= H[p]) {
            return;
        }
        tmp = H[i];
        H[i] = H[p];
        H[p] = tmp;
```

```
        upHeap(p);
    }

    void printHeap() {
        for (int i = 1; i <= n*n; i++) {
            printf(" %d", H[i]);
        }printf("\n");
    }

    int downHeap(int i) {
        int l = 2 * i;
        int r = 2 * i + 1;
        int tmp;
        if (l > n) {
            return;
        }
        int big = l;

        if (H[r] > H[big] && n >= r) {
            big = r;
        }

        if (H[i] < H[big])
        {
            tmp = H[i];
            H[i] = H[big];
            H[big] = tmp;
        }
        else {
            return;
        }
        downHeap(big);
    }

    int main()
    {
        int N;
        scanf("%d", &N);
        int f, b = N;
        n = N * N;

        H = (int*)malloc(sizeof(int) * n);

        for (int i = 1; i <= n; i++) {
            scanf("%d", &H[i]);
            cnt++;
            upHeap(i);
        }

        //printHeap();

        for (int i = 1; i <= b; i++) {
            f = removeMax();
            //printf("//%d %d %d %d\n", b, n, i, f);
            if (i == b) {
                printf("%d", f);
            }
        }
    }
```

효정

이전 포스트
백준 2696: 중앙값 구하기

0개의 댓글

댓글을 작성하세요

댓글 작성