3주차 힙, 힙정렬

iming03 · 방금 전

알고리즘 힙 힙정렬

문제: 백준 2075번

2075번 제출 맞힌 사람 숏코딩 재채점 결과 채점 현황 질문 게시판

N번째 큰 수

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	12 MB (하단 참고)	26503	10621	7693	39.011%

문제

N×N의 표에 수 N^2개 채워져 있다. 채워진 수에는 한 가지 특징이 있는데, 모든 수는 자신의 한 칸 위에 있는 수보다 크다는 것이다. N=5일 때의 예를 보자.

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

이러한 표가 주어졌을 때, N번째 큰 수를 찾는 프로그램을 작성하시오. 표에 채워진 수는 모두 다르다.

입력

첫째 줄에 N(1 ≤ N ≤ 1,500)이 주어진다. 다음 N개의 줄에는 각 줄마다 N개의 수가 주어진다. 표에 적힌 수는 -10억보다 크거나 같고, 10억보다 작거나 같은 정수이다.

출력

첫째 줄에 N번째 큰 수를 출력한다.

예제 입력 1 복사

```
5
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
```

예제 출력 1 복사

```
35
```

1. 개념 정리하기

사용될 핵심개념: 힙, 힙정렬

힙(heap): 내부노드에 키를 저장하며 다음 두 가지 속성을 만족하는 이진트리

1. **힙순서(heap-order)**: 루트를 제외한 모든 내부노드 v에 대해, **key(v)≥key(parent(v))**
2. **완전이진트리(complete binary tree)**: **힙의 높이를 h**라 하면
    1. i = 0, … , h-2에 대해, 깊이 i인 노드가 **2^i**개 존재
    2. 깊이 h-1에서, **내부노드들은 외부노드들의 왼쪽**에 존대

힙의 마지막 노드(last node): 깊이 **h-1**의 가장 오른쪽 내부노드

힙에 삽입

- 우선순위 큐 ADT의 메소드 insertItem은 힙에 키 k를 삽입하는 것에 해당
- 삽입 알고리즘의 세 단계
    1. 삽입노드 z, 즉 새로운 마지막 노드를 찾는다
    2. k를 z에 저장한 후 expandExternal(z)작업을 사용하여 z을 내부노드로 확장
    3. 힙순서 속성을 복구

Upheap

- 새로운 키 k가 삽입된 후, 힙순서 속성이 위배될 수 있다.
- 알고리즘 Upheap은 삽입노드로부터 상향경로를 따라가며 키 k를 교환함으로써 힙순서 속성을 복구
- Upheap은 키 k가 루트에, 또는 부모의 키가 k보다 작거나 같은 노드에 도달하면 정지

- 힙의 높이는 O(log n)이므로 Upheap은 O(log n)시간에 수행

힙으로부터 삭제

- 우선순위 큐 ADT의 메소드 removeMin은 힙으로부터 루트 키를 삭제하는 것에 해당
- 삭제 알고리즘의 세 단계
  1. 루트 키를 마지막 노드 w의 키로 대체
  2. reduceExternal(w) 작업을 사용하여 그의 자식들을 외부노드로 축소
  3. 힙순서 속성을 복구

Downheap

- 루트 키를 마지막 노드의 키로 대체한 후, 힙순서 속성이 위배될 수 있다.
- 알고리즘 downheap은 루트로부터 하향 경로를 따라가며 키 k를 교환함으로써 힙순서 속성을 복구
- downheap은 키 k가 잎에, 또는 자식의 키가 k보다 크거나 같은 노드에 도달하면 정지
- 힙의 높이는 O(log n)이므로 downheap은 O(log n) 시간에 수행

힙 정렬

- **Heap sort의 성능 향상을 위한 두가지 개선점**
  1. 제자리 힙 정렬은 heap sort의 공간 사용을 줄인다.
  2. 상향식 힙 생성은 heap sort의 속도를 높힌다.

2. 전략

> **속도를 높이기 위해 상향식 힙으로 풀어야겠다고 생각했다.**

상향식 힙생성

- log n 단계만을 사용하여 주어진 n개의 키를 저장하는 힙 생성 가능
- 단계 i에서, 각각 2^i-1개의 키를 가진 두 개의 힙을 2^i -1개의 키를 가진 힙으로 합병
- 이후 힙순서 속성 복구를 위해 downheap 수행

"상향식"이라 불리는 이유?

- 각 재귀호출이 힙인 부트리를 반환하는 방식 때문에
- 힙화(heapification)는 외부노드에서 시작하여, 각 재귀호출이 반환함에 따라 트리 위쪽으로 진행됨

- 때문에 **힙화한다(heapify)**고 말하기도 함

3. 코드

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

//전역변수
int* H = NULL;
int size = 0;

//힙 정렬
void downHeap(int i) {
    int bigger;
    int left = 2 * i;
    int right = 2 * i + 1;

    if (size < left) { return; } //자식이 없는 상태

    bigger = left;

    if (right <= size) {
        if (H[right] > H[bigger]) { //더 크면 오른쪽 자식 선택
            bigger = right;
        }
    }

    //선택한 자식과 현재 노드 비교
    if (H[bigger] <= H[i]) { return; }
    //교환
    int tmp = H[i];
    H[i] = H[bigger];
    H[bigger] = tmp;

    downHeap(bigger);
}
//상향식 힙 (재귀)
int rBuildHeap(int i) {
    if (i > size) { return size; }

    rBuildHeap(i * 2);
    rBuildHeap(i * 2 + 1);

    downHeap(i);

    return size;
}
//최대값 제거, 반환 함수
int removeMax() {
    int key = H[1];
    H[1] = H[size--];
    downHeap(1);
    return key;
}
```

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

    H = (int*)malloc(sizeof(int) * (size+1));

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

    rBuildHeap(1);

    for (int i = 0; i < N - 1; i++) {
        removeMax(); //N-1번만 제거
    }
    printf("%d\n", removeMax()); //제거, 출력

    free(H);

    return 0;
}
```

4. 오류사항들과 후기...

와.. 이 코드 완성하는데 진짜 오래 걸렸당.. 난 바보인가?

맨 처음에 실습코드만 살짝 변형할 생각으로 insertItem()으로 삽입하고 (왜인지 모름, 약간 큐처럼 생각했나봄 넣고 빼고를 함수로 멋있게~ 요런 느낌„하하..~~사실 강의 안듣고 막 찾은 정보로 해보려고 해서 그럼„„ 결국 강의 다듣고 빨리 품 ^^„ 그래도 총 시간은 엄청 오래걸림.~~)

오류 1: 시간초과

답이 나오는데 자꾸 시간초과인 거임... 진짜 왤까 계속 고민하다가 결국 상향식 힙생성으로 풀어야지 결심하게 됨.. ~~그리고 강의 들음~~

오류 2: 컴파일에러와 런타임에러„„„„„„„„„„„„„„„

downHeap()을 downheap으로 써서 그랬다... 백준이 알려줘서 너무 좋음!
런타임에러는 전역변수 사이즈 지정할 때 size로 했는데 기존 코드대로 n으로 되어있어서 그랬다.

문제	결과	메모리	시간	언어	코드 길이
2075	맞았습니다!!	9908 KB	532 ms	C99 / 수정	1073 B
2075	런타임 에러 (Segfault)			C99 / 수정	1077 B
2075	런타임 에러 (Segfault)			C99 / 수정	1069 B
2075	컴파일 에러			C99 / 수정	1069 B

2075	시간 초과			C99 / 수정	1058 B
2075	런타임 에러 (OutOfBounds)			C99 / 수정	1275 B
2075	런타임 에러 (OutOfBounds)			C99 / 수정	1286 B
2075	시간 초과			C99 / 수정	1260 B
2075	컴파일 에러			C99 / 수정	1274 B
2075	시간 초과			C99 / 수정	1260 B
2075	시간 초과			C99 / 수정	1260 B
2075	시간 초과			C99 / 수정	1260 B

> 후기: 다음부터는 강의를 바로바로 듣자! ^^

강민돌

민돌이의 공부

이전 포스트

2주차 우선순위 큐

0개의 댓글

댓글을 작성하세요

댓글 작성