[백준] 힙과 힙정렬

imhyun · 방금 전

백준 알고리즘 힙 힙정렬

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
67095650	qkrtlgus1120	2075	맞았습니다!!	9908 KB	540 ms	C99 / 수정	1544 B	4분 전

https://www.acmicpc.net/problem/2075

> 문제 핵심: N x N의 표에 수 N^2개가 채워져 있다. 이 때 모든 수는 자신의 한 칸 위에 있는 수보다 크다. 이 때 N번째 큰 수를 찾는 프로그램을 작성하시오.

아이디어

1. 주어진 N x N 표를 통해 최대힙을 생성한다.
   (상향식 힙생성)
2. 만들어진 힙을 통해 오름차순으로 정렬한다.
3. N번째 큰 수를 찾는다.

이 때 주의할 점은, 힙을 생성했다고 배열에서 5번째 수를 출력하면 안 된다. 힙은 단순히 부모 KEY값과의 크기 비교를 한 것이지, 레벨순이 곧 정렬 순이 아니다.

개념

힙이란?

💡 배열을 기반 or 연결 리스트를 기반으로 우선순위 큐를 구현하는 방법은

이라는 단점이 발생할 수 있다.

따라서, 이를 해결하기 위해 힙이라는 자료구조를 이용해 우선 순위 큐를 구현하는 것이 일반적이다.

힙의 조건

> **내부노드**에 키를 저장하며 **아래 두 조건**을 만족하는 **이진트리**
>
> 1. **완전 이진 트리** (Complete binary tree)
>   - 차곡차곡 **빈 틈 없이 노드가 채워진** 이진 트리
>   - 노드가 **위에서 아래**로, **왼쪽에서 오른쪽** 순서대로 채워진 것.
> 2. **힙 순서**
>   - 내부노드 v에 대해, key(v) ≥key(parent(v)) 여야 한다.

배열에 기초한 힙 구현 (순차 배열)

- n개의 키를 가진 힙의 크기 → n의 배열을 사용하여 표현 가능
- 완전이진트리 배열로 저장하는 방법
  - i=0일 때 비어놓기
  - 왼쪽 자식은 2i에 존재
  - 오른쪽 자식은 2i+1
  - 부모는 [i/2]
- 마지막 노드의 첨자 항상 n
  - insertItem : n+1에 삽입
  - removeMin : n의 위치에서 삭제
  - **상수 시간**이 걸린다! **O(1)**

힙정렬

> 힙에 기초한 우선순위 큐를 사용하므로써, n개의 원소로 이루어진 리스트를 **O(nlogn)** 시간에 정렬 가능하다.

- 개선 방법
  - **제자리 힙 정렬**을 통해 공간 사용을 줄인다.
    - H사용하지 않음
  - **상향식 힙생성**을 통해 힙정렬의 속도를 높힌다.
    - 힙생성에서 속도를 줄인다.→ 그래도 점근 속도는 O(nlogn)이지만, CPU Time은 줄어든다.

```
Alg heapSort(L)
    input list L
    output sorted list L
1.H<-empty heap
2.while(!L.isEmpty()) {**힙 생성 단계}**
        k<-L.removeFirst() {O(n)복잡도}
        H.insertItem(k) {O(nlogn)복잡도}
3.while(!H.isEmpty()) {**힙 정렬 시작}**
        k<-H.removeMin()
        L.addLast(k)
4.return
```

상향식 힙생성

- Why?
  - heap-sort의 1기에서, 이전에는 insertItem을 이용하여 힙을 생성함. **O(nlogn)**
- When?
  - 키들이 미리 주어진다면, **O(n)** 시간에 수행하는 상향식 생성 방식

```
Alg buildHeap(L)
    input list L storing n keys
    output heap T storing the keys in L

1. T<-convertToCompleteBinaryTree(L)
2.rBuildHeap(T.root())
3.return T

Alg rBuildHeap(v)
    input node v
    output a heap with root v

1. if(isInternal(v))
        rBuildHeap(leftChild(v))
        rBuildHeap(rightChild(v))
        downHeap(v)
2. return
```

2. k를 저장한 노드를 루트로, 두 개의 힙을 부트리로 하는 새 힙 생성
3. downheap 수행

비재귀적 상향식 힙생성

- When?
  - 정렬되어야 할 리스트가 **배열**로 주어진 경우에만.
  - How?
    - 내부노드를 왼쪽 자식으로 가지는 가장 깊은 내부노드 가운데 가장 오른쪽 노드에서 시작
    - **첨자 [n/2]인 노드**
    - **레벨 순서의 역순**으로

```
Alg buildHeap(L)
    input list A of n keys
    output heap A of size n
```

1. for i <- [n/2] downto 1
   downHeap(i,n)
   2.return

힙으로부터 삭제

> 우선순위 큐 ADT removeMin은 힙으로부터 **루트 키를 삭제**하는 것.

방법

1. **루트 키**를 마지막 노드 w의 키로 **대체**
   - **완전 이진 트리를 유지**하기 위해
2. reduceExternal(z)
   - w와 그의 자식들을 **외부노드로 축소**
3. **힙순서 속성**을 복구

> 루트 키를 마지막 노드의 키로 대체한 후, 힙순서 속성이 위배될 수 있기에 **힙순서 속성을 복구**시키는 방법.
>
> - 방법
>   - 하향 경로를 따라가며 키 k를 교환해 힙순서 속성을 복구
>   - 잎에 도달 or 힙순서 조건을 만족시키면 정지
>   - 자식 노드가 모두 외부 노드라면 끝에 도달했으니 정지
>   - 자식 노드가 **모두 내부 노드**라면 둘 중 **작은 것을 비교해서 교환**해야 함.
> - 시간
>   - 힙의 높이인 **O(logn)** 시간 수행

코드

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


//필요 데이터구조인 정수형 배열과 변수 선언
int*H=NULL;
int size=0; //힙의 크기

//필요한 함수 선언
void DownHeap(int);

//상향식 힙생성
void rBuildHeap(int);

int main() {
    int n;
    int i;

    //N에 대한 정보 입력 받기
    scanf("%d",&n);

    size = n * n;

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

    //예외처리에 대한 조건문
    if (H == NULL) return -1;

    //키들 한꺼번에 입력받기 (이진트리 형식으로 H 구성)
```

```
    //상향식 힙 생성
    rBuildHeap(1);


    //힙정렬 - N번째 큰 수 출력
    int k;

    for ( i = 0; i < n; i++) {
        //n번째 큰수
        k = H[1];

        //교환
        H[1] = H[size];

        //사이즈 줄이기
        size--;

        //힙순서 정렬
        DownHeap(1);
    }

    printf("%d\n",k);

    free(H);
    return 0;
}

//상향식 힙생성
void rBuildHeap(int i) {
    int left = 2 * i;
    int right = 2 * i + 1;

    if (i>size) //종료 조건 : 키가 없는 외부노드로 진입할 때
        return;

    rBuildHeap(left);
    rBuildHeap(right);

    DownHeap(i);
}

void DownHeap(int i) {
    //종료조건 1
    if (2 * i > size) //외부노드일 경우.
        return;

    //바꿔줄 자식 찾기
    int biggerIdx = i * 2;
    if (2*i+1<=size && H[biggerIdx] < H[biggerIdx + 1])
        biggerIdx++;

    //종료조건 2
    if (H[biggerIdx] <= H[i])
        return;
```

```
        H[biggerIdx] = tmp;

        //재귀
        DownHeap(biggerIdx);
    }
```

오래 걸렸던 부분

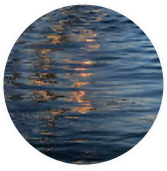
1. H를 **동적할당 할 때 범위**
   => 1부터 size까지 저장을 해 완전이진트리로 만들고 싶기 때문에 size+1만큼 크기의 배열을 만들어야 했다.

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

2. rBuildHeap과 DownHeap 종료조건
   => H의 원소에 접근하려면 우선 자식 노드가 모두 size안에 위치해야 한다. 바꿔줄 자식부터 찾고 비교하게 되면 접근할 수 없는 배열의 원소에 접근하게 될 수 있다. 따라서 **인덱스의 크기를 먼저 비교 후 접근**해야 한다.

개선 방안

현재는 모든 수가 자신의 한 칸 위에 있는 수보다 크다는 것을 이용하지 못 했다. 또한 표의 형태로서 활용하지 않고 단순한 배열로 생각해주었다.



박시현

이전 포스트
[백준] 큐와 트리

댓글을 작성하세요

댓글 작성

관심 있을 만한 포스트

[자료구조] 힙(heap)

이진 힙(binary heap)은 우선순위 큐(priority queue)를 위한 자료구조다. 그런데 왜 우선순위 큐는 기존에 있는 큐와 같은 방식을 이용하지않고 heap이라는 자료구조를 이용하는 것일까? 그에 대한 답은 우선순위 큐라는 이름에서 찾아볼 수 있다. 큐...

2020년 7월 10일 · 0개의 댓글

by **junhok82** 4

우선순위 큐(Prioirity Queue) (1) - 최대 힙 이용

큐는 선입선출, 먼저 들어간 데이터가 먼저 나온다. 우선순위 큐는 들어간 순서에 상관 없이 우선순위가 높은 데이터가 먼저 나온다. (우선순위가 다른 데이터 뿐만 아니라 같은 데이터가 존재할 수도 있다)

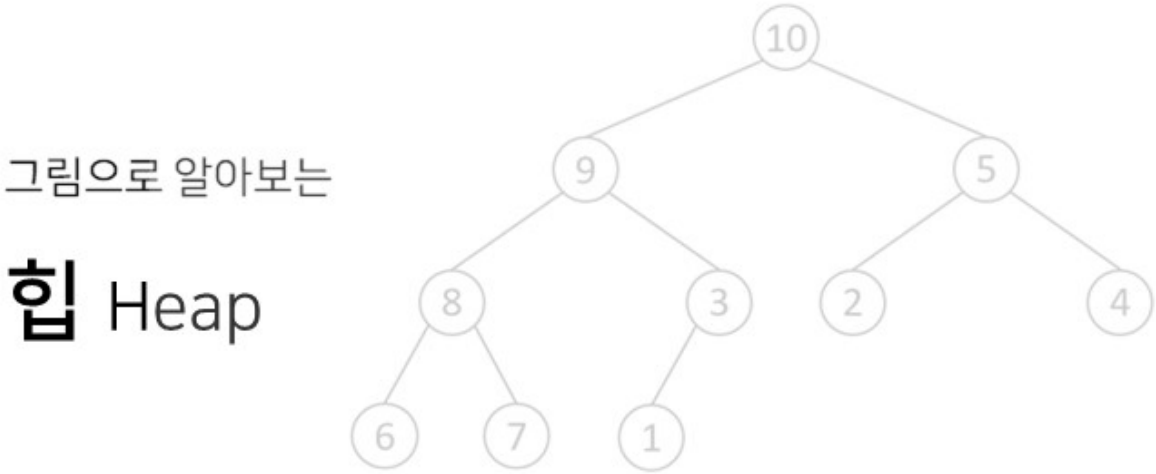
by **holicme7** 3

힙 정렬(Heap Sort) 알고리즘

힙 정렬(Heap Sort) 알고리즘 힙 정렬 알고리즘이란? 최소 힙 트리(내림 차순 정렬)나 최대 힙 트리(오름 차순 정렬)를 구성해 한번에 하나씩 요소를 힙에서 꺼내서 배열의 뒤에서부터 저장한다. 힙 이란? 노드를 삽입할 때 왼쪽부터 차례대로 삽입하는 트리 형태인 완전 이진 트리의 일종으로 최솟값이나 최댓값을 빠르게 찾아내도록 만들어진 자료구조이...

2020년 1월 7일 · 0개의 댓글

by **ssuda** 0



[자료구조] 그림으로 알아보는 힙(Heap)

항상 공부하기로 한 힙(Heap) 자료구조를 이제야 정리하게 되었습니다. 대표적으로 우선순위 큐를 구현하는데 많이 사용한다고 알고만 있었지 정확히 어떤 자료구조인지 잘 몰랐습니다. 이번 기회에 트리, 힙, 우선순위 큐 관련 내용들을 포스팅 해보겠습니다.힙 자료구조는 완...

2020년 12월 16일 · 0개의 댓글

by **emplam27** 9

[자료구조] Heap(힙) - 개념, 종류, 활용 예시, 구현

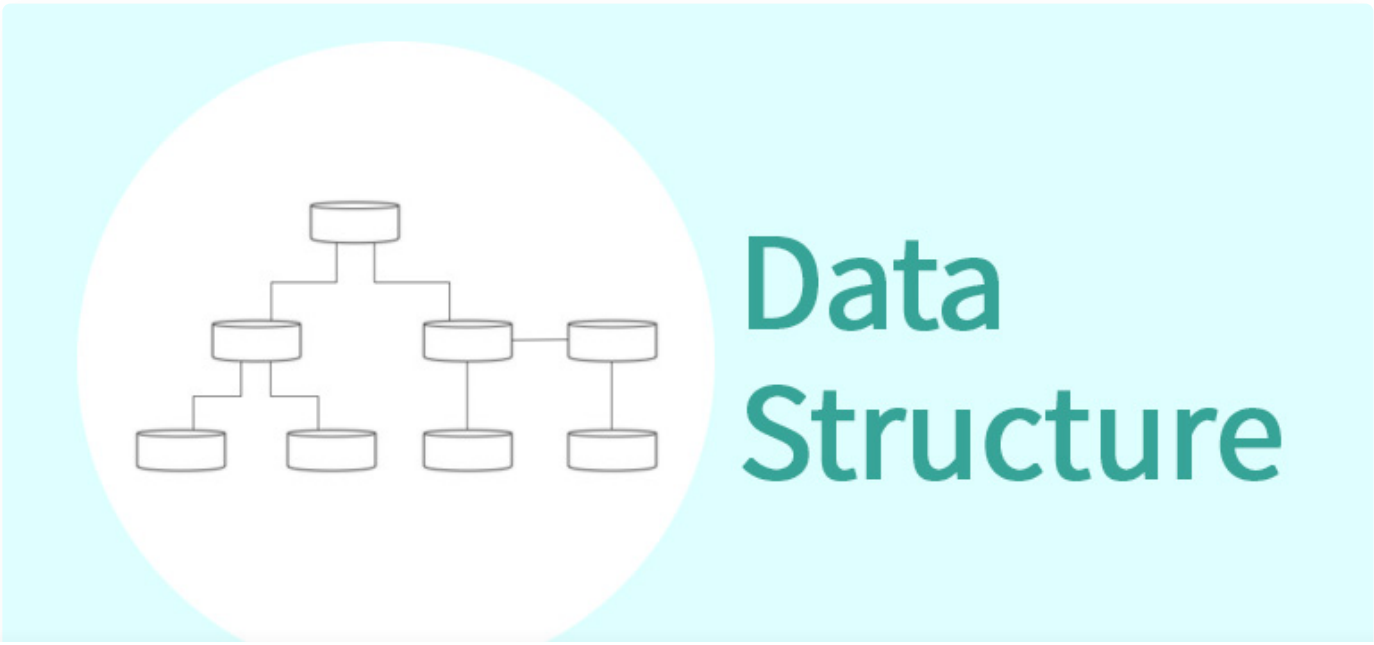
2021년 8월 15일 · 0개의 댓글

알고리즘 5일차 - O(nlogn) 정렬, 힙과 힙정렬 최대힙, 최소힙 - 2부

살려줘그렇다면 힙은 어떻게 사용할 때 유의미하게 사용할 수 있을까??그것이 바로 최소힙과 최대힙이다.최대힙은 힙 원소들 중 가장 큰 값이 맨 위에 있는 것을 의미하며, 최소힙은 반대로 힙 원소들 중 가장 작은 값이 맨 위에 있는 것을 의미한다.최대힙 또는 최소힙은 부모...

2021년 4월 15일 · 1개의 댓글

by **chappi** ♥ 3



[자료구조] 트리와 힙

트리(Tree) 비선형 구조 ⇒ 1:n 관계를 가지는 자료구조 다:다 ⇒ 그래프, 1:다 + 계층 ⇒ 트리 상위 입장에서는 하위가 여러개이지만, 하위 입장에서는 상대가 하나면 트리를 쓸 수 있다. 한 개의 루트 노드만이 존재하며 모든 자식 노드는 한개의 부모 노드만을 ...

2021년 3월 2일 · 1개의 댓글

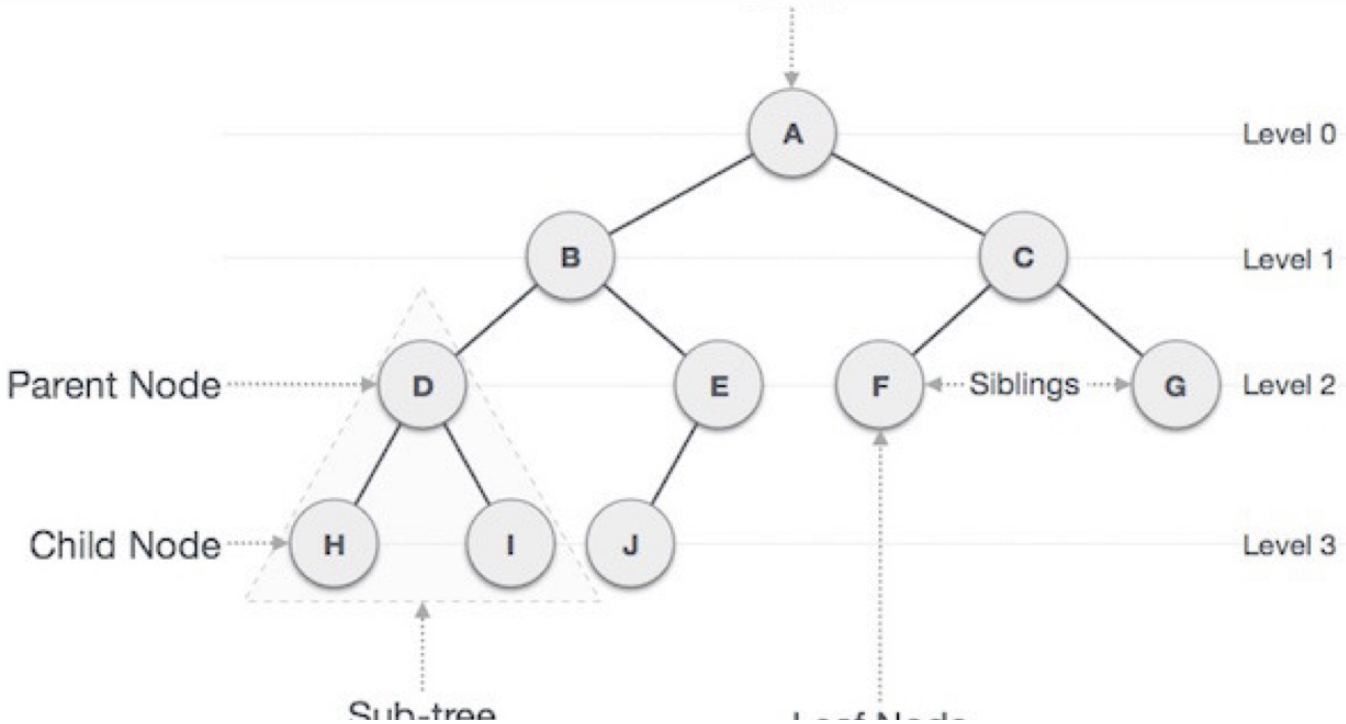
by **humblechoi** 1

[자료구조/알고리즘] - 트리, 힙

데이터 사이의 계층 관계를 노드로 나타낸 자료구조루트 트리의 가장 윗부분에 위치하는 노드를 루트(root)라고 한다. 하나의 트리에는 하나의 루트가 존재한다.리프 트리의 가장 아랫부분에 위치하는 노드를 리프(leaf)라고 한다. 가장 아랫부분의 의미는 물리적으로 ...

2021년 4월 7일 · 0개의 댓글

by **roro** 1



[자료구조]Tree 🎄 🌲 🌳 🌴

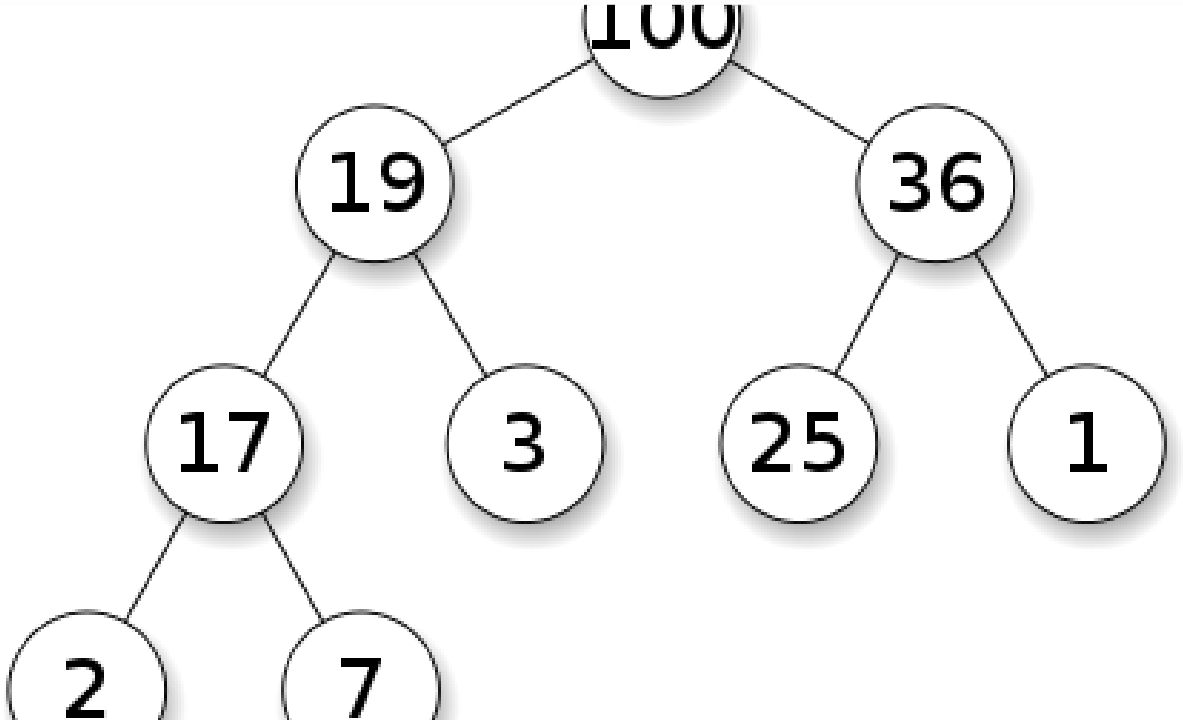
트리는 일반적으로 대상 정보의 각 항목들을 계층적으로 연관되도록 구조화시키고자 할 때 사용하는 비선형 자료구조이다. 데이터 요소들의 단순한 나열이 아닌 부모-자식 관계의 계층적 구조로 표현이 된다. 트리는 그래프의 한 종류이며 사이클이 없다.node: 트리를 구성하고 ...

2020년 4월 5일 · 1개의 댓글

정렬되어 있는 배열이라는 전제 조건이 있음원하는 값(x)을 찾을 때, 배열의 가운데 값(m)을 기준으로 크고 작음을 비교해 탐색하는 것m이 x보다 크면, 배열의 오른쪽을 뚝 잘라 왼쪽에서 다시 가운데 값을 정의(m1)하고 비교, 반복탐색하는 배열이 반씩 줄어들기 때문에...

2020년 7월 28일 · 0개의 댓글

by **matisse** 1



Java Heap & Priority Queue

힙 자료구조와 우선순위 큐

2020년 6월 5일 · 0개의 댓글

by **agugu95** 0