



부산대학교

PUSAN NATIONAL UNIVERSITY

Heaps and Priority Queues

The Smallest One First

Priority Queues and the Binary Heap

DS3600082-140

자료구조

2026학년도 2학기

오용경

데이터사이언스전공

yongkyung.oh@pusan.ac.kr

Priority Queue as an ADT

먼저 꺼낼 것을 값이 정하는 ADT를 정의한다

우선순위 규칙이 꺼낼 원소를 정한다

가방에 7, 3, 12, 1, 9를 넣고 "하나를 꺼내라"고 요청한다

요청	가능 여부	결과
7을 지정해 꺼내기	불가능	임의 삭제는 제공하지 않음
하나 꺼내기	가능	현재 최솟값 1
한 번 더 꺼내기	가능	다음 최솟값 3

꺼낼 대상을 사용자가 지정하지 않고 우선순위 규칙이 정한다

같은 요청이라도 꺼내는 순서가 다르다

7, 3, 12, 1, 9를 넣은 뒤 다섯 번 꺼낸다

꺼내는 차례	FIFO 큐	최소 우선순위 큐
1, 2	7, 3	1, 3
3, 4, 5	12, 1, 9	7, 9, 12

입력은 같아도 큐는 도착 순서, 우선순위 큐는 값의 순서로 꺼낸다

정의

우선순위 큐(priority queue)

값을 넣고 그 가운데 가장 작은 값을 찾고 꺼내는 세 연산을 갖춘 ADT다. 가장 작은 값은 우선순위 기준으로 정하며, 반대로 가장 큰 값을 먼저 꺼내도록 정의할 수도 있다. ADT는 연산을 정할 뿐 값을 어디에 어떻게 담을지는 정하지 않는다.

우선순위 큐는 다음 처리 대상을 고른다

문제	답는 것	우선순위
작업 스케줄링	할 일	급한 정도
응급실	환자	위중도
인쇄 작업 선택	인쇄 작업	쪽수가 적은 순

도착한 차례가 아니라 우선순위가 순서를 정하는 문제들이다

배운 구조로 만들어 보면

이미 아는 구조로도 만들 수 있지만 한쪽이 비싸진다

구현	넣기	최솟값 꺼내기
정렬 안 한 배열	$O(1)$	$O(n)$
정렬된 배열(최솟값을 끝에 둠)	$O(n)$	$O(1)$
BST	$O(\log n)$ 평균	$O(\log n)$ 평균

연산의 이름

하는 일	ADT 연산	Python heapq
넣기	add	heappush
최솟값 보기	find_min	h[0]
최솟값 꺼내기	remove	heappop

이름은 달라도 세 연산이 우선순위 큐를 이룬다

Checkpoint

Priority Queue as an ADT

- 우선순위 큐와 큐는 무엇이 다른가?
- 우선순위 큐의 세 연산은 무엇인가?
- 정렬된 배열로 만들면 어느 연산이 비싸지는가?

Two Conditions

힙이 지키는 두 조건을 나누어 정의한다

정의

binary heap

완전 이진트리의 모양과 힙 성질을 함께 지키는 힙(heap)이며, 흔히 힙이라고 하면 이것을 가리킨다. 모양 조건은 배열에 빈틈없이 담을 수 있게 하고, 힙 성질은 루트에 최솟값이 오게 한다.

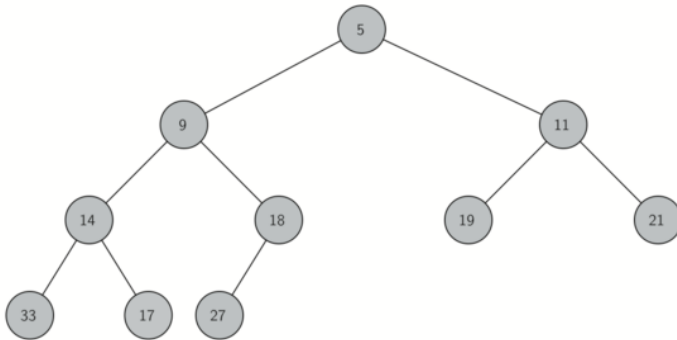
첫째 조건: 모양

정의

완전 이진트리(**complete binary tree**)

마지막 레벨을 제외한 모든 레벨이 가득 차며 마지막 레벨은 왼쪽부터 채워진다. 이 조건을 **structure property**(완전 이진트리여야 한다는 조건)라고 한다. 빈틈이 생기지 않으므로 같은 노드 수의 이진트리 가운데 높이가 가장 낮다.

모양 조건을 그림으로



마지막 레벨이 왼쪽부터 채워져 있고 중간에 빈 자리가 없다.

Source: Miller and Ranum (2013), Binary Heap Implementation.

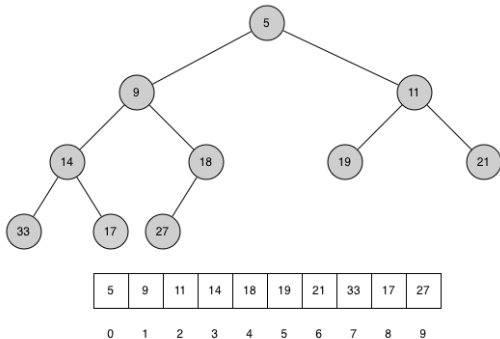
둘째 조건: 순서

정의

힙 성질(heap-order property)

모든 부모 노드의 값이 두 자식 노드의 값보다 작거나 같다는 조건이다. 부모와 자식 노드 사이만 정하고 형제 사이나 서로 다른 갈래 사이는 정하지 않지만, 이 조건만으로 가장 작은 값이 루트에 오는 것이 보장된다.

순서 조건을 그림으로



모든 부모가 자기 자식 노드보다 작지만 형제끼리의 크기 순서는 정해지지 않는다.

Source: Miller and Ranum (2013), Binary Heap Implementation.

정렬된 것이 아니다

관계	힙 성질이 알려 주는 것
두 형제의 크기	정해지지 않는다
두 번째로 작은 값	루트의 두 자식 노드 가운데 하나
세 번째로 작은 값	자식·손자 노드 가운데 있어 비교가 필요

Caution 힙은 전체 정렬이 아니라 최솟값 하나의 위치만 보장한다

최소 힙과 최대 힙

최소 힙(min-heap)과 최대 힙(max-heap)은 부모와 자식의 대소 관계가 반대다

	최소 힙	최대 힙
부모와 자식 노드	부모가 작거나 같다	부모가 크거나 같다
루트에 오는 것	최솟값	최댓값

조건이 부등호만 뒤집으면 나머지 절차는 그대로다

두 조건은 서로 다른 일을 한다

조건	정하는 것	주는 것
structure property	모양	낮은 높이, 배열 표현
힙 성질	값의 배치	루트가 최솟값

모양 조건은 낮은 비용을, 힙 성질은 루트의 최솟값을 보장한다

두 조건을 하나씩만 깨뜨리면

모양과 값의 순서는 서로를 보장하지 않는다

일곱 노드의 상태	모양 조건	힙 성질
마지막 레벨의 왼쪽 칸을 비우고 오른쪽 칸을 채움	위반	부모가 모두 작아도 성립 가능
[1, 5, 6, 4, 3, 8, 9]	성립	5가 자식 4와 3보다 커 위반

완전한 모양과 부모와 자식 사이의 순서를 모두 만족해야 힙이다

BST의 조건과 견주면

둘 다 이진트리이지만 값을 배치하는 규칙이 다르다

	BST	힙
비교하는 관계	왼쪽 · 자기 · 오른쪽	부모 · 자식 노드
중위 순회하면	정렬된 순서	정해진 순서 없음
루트에 오는 것	삽입 순서에 따라 달라짐	최솟값

BST는 전체 순서를, 힙은 최솟값 하나를 보장한다

Checkpoint

Two Conditions

- 힙이 지키는 두 조건은 각각 무엇인가?
- 힙 성질이 정하지 않는 관계는 무엇인가?
- 두 번째로 작은 값은 어디에 있는가?

Storing a Heap in an Array

참조 없이 배열만으로 트리를 담는다

노드를 만들지 않는다

정의

implicit tree

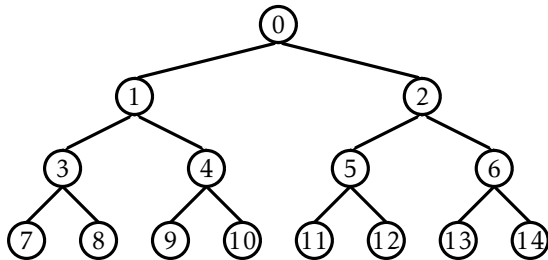
implicit tree(노드와 참조 없이 배열 인덱스만으로 표현한 트리)는 트리를 레벨 순서대로 배열에 늘어놓고 인덱스 계산으로 오르내리는 표현이다. 완전 이진트리는 빈 자리가 없으므로 배열에 빈틈없이 담긴다. 이런 배열 저장은 같은 ADT를 담는 내부 표현의 한 사례다.

완전 이진트리를 레벨 순서로 배열에 담는다

항목	내용
담는 순서	위에서 아래로, 왼쪽에서 오른쪽으로
같은 순서의 순회	레벨 순회
빈칸	모양 조건 덕분에 생기지 않는다

배열의 순서가 곧 레벨 순회 순서이므로 트리를 그대로 되살릴 수 있다

트리의 번호가 곧 배열 인덱스다



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----

노드에 적힌 번호가 그 값을 담는 배열 칸의 인덱스다. 3번 노드의 자식은 7번과 8번, 부모는 1번이다.

인덱스로 오르내리기

Python 리스트처럼 인덱스를 0부터 세는 규약으로 적는다

찾는 것	계산	$i = 3$ 이면
왼쪽 자식 노드	$2i + 1$	7
오른쪽 자식 노드	$2i + 2$	8
부모	$\lfloor (i - 1) / 2 \rfloor$	1

Caution 인덱스를 1부터 세면 식이 $2i$, $2i + 1$, $\lfloor i/2 \rfloor$ 로 달라진다

배열 표현이 주는 것

	노드 기반	배열
답는 것	값과 자식 노드 참조	값을 담은 연속된 칸
부모 찾기	참조가 필요	계산 한 번

배열 한 줄에서 트리 되살리기

[1, 2, 6, 4, 3, 8, 9]가 어떤 트리인지 인덱스로 읽어 본다

인덱스	값	왼쪽 자식 노드	오른쪽 자식 노드
0	1	2 (인덱스 1)	6 (인덱스 2)
1	2	4 (인덱스 3)	3 (인덱스 4)
2	6	8 (인덱스 5)	9 (인덱스 6)
3부터 6까지	4, 3, 8, 9	없음(7 이상)	없음

Checkpoint

Storing a Heap in an Array

- 힙을 배열에 담을 때 어떤 순서로 늘어놓는가?
- 인덱스 i 의 두 자식 노드와 부모는 각각 몇 번인가?
- 모양 조건이 없으면 인덱스 계산에 무슨 일이 생기는가?

Height of a Complete Tree

모양 조건이 높이를 얼마나 낮추는지 확인한다

빈틈이 없으면 얼마나 낮은가

레벨 h 가 담을 수 있는 노드 수는 최대 2^h 개다

레벨	그 레벨의 노드 수	누적
0	1	1
1	2	3
2	4	7
3	8	15

높이와 n 의 관계

높이가 h 이면 레벨 $h - 1$ 까지 가득 차고 레벨 h 에 한 개 이상 있으므로 $2^h \leq n \leq 2^{h+1} - 1$ 이다

n	높이
1, 2, 4, 8, 16	0, 1, 2, 3, 4 (n 이 두 배가 될 때마다 1 증가)
1,000	9
1,000,000	19

완전 이진트리의 높이는 $\lfloor \log_2 n \rfloor$ 이다

일곱 노드에서 여덟 노드로 늘리면

인덱스 7은 새 레벨의 첫 자리가 된다

노드 수	마지막 레벨	높이
$n = 7$	인덱스 3에서 6까지 차 있음	2
$n = 8$	인덱스 7이 처음 채워짐	3

노드 하나가 새 레벨을 시작할 때 높이가 한 단계 증가한다

단말 노드는 배열의 뒤쪽 절반에 있다

근거	내용
마지막 노드	인덱스 $n - 1$
그 부모	인덱스 $\lfloor n/2 \rfloor - 1$
인덱스 $\lfloor n/2 \rfloor$ 부터 $n - 1$	자식 노드가 없는 단말 노드

자식 노드가 있는 노드는 배열의 앞쪽 절반에 모여 있다

다른 구현과 견주면

배열과 힙의 삽입·최솟값 삭제 비용을 비교한다

구현	넣기	최솟값 꺼내기
정렬 안 한 배열	$O(1)$	$O(n)$
정렬된 배열(최솟값을 끝에 둠)	$O(n)$	$O(1)$
힙	$O(\log n)$	$O(\log n)$

힙은 삽입과 최솟값 삭제를 모두 $O(\log n)$ 에 수행한다

힙 성질을 복원하는 연산

상황	복원 연산
값을 넣은 뒤	상향 이동(sift up)
최솟값을 꺼낸 뒤	하향 이동(sift down)
배열 전체를 힙으로 만들 때	heapify

넣기와 꺼내기는 루트와 단말 노드 사이의 한 경로에서만 값을 옮기므로 교환 횟수가 높지 $\lfloor \log_2 n \rfloor$ 를 넘지 않는다

Checkpoint

Height of a Complete Tree

- 완전 이진트리의 높이는 n 과 어떤 관계인가?
- 힙 연산이 $O(\log n)$ 인 이유는 무엇인가?
- 배열에서 단말 노드는 어디에 모여 있는가?

How Do We Maintain Heap Order?

Next Lecture: Heap Operations

- sift up과 sift down
- heapify와 힙 정렬
- Python heapq 사용법