힙 (*Heap*)

연세대 미래캠퍼스 고급 알고리즘 강의 2주차

이혜아

2022년 7월 13일

추상 자료형, 데이터 구조

추상 자료형 (*Abstract Data Type*)

사용자가 자료형에 대해서 할 수 있는 동작을 나열한 것

데이터 구조 (*Data Structure*)

구현자가 데이터의 효율적인 접근을 위해 특정한 형태로 정리한 것

우선순위 큐

우선순위 큐 – 추상 자료형

1. 초기화: 다중집합(*multiset*) S를 ϕ로 초기화
2. 원소 삽입: S에 v를 추가
3. 최솟값: $\min S$를 구함
4. 최솟값 삭제: S에서 $\min S$를 삭제함

- 우선순위 큐는 추상 자료형이기 때문에, 다양한 방법으로 구현할 수 있음

우선순위 큐의 구현 – 배열

1. 초기화: `A = []`
   - 시간복잡도: $\mathcal{O}(1)$
2. 원소 삽입: `A.append(v)`
   - 시간복잡도: $\mathcal{O}(1)$
3. 최솟값: `min(A)`
   - 시간복잡도: $\mathcal{O}(N)$
4. 최솟값 삭제: `A.pop(A.index(min(A)))`
   - 시간복잡도: $\mathcal{O}(N)$

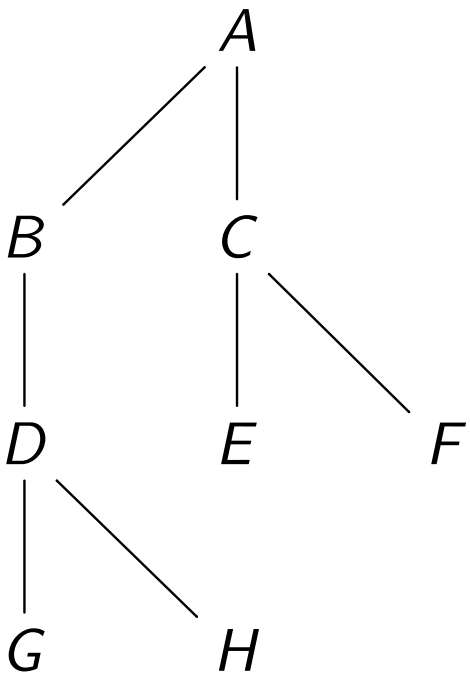
우선순위 큐의 구현 – 정렬된 배열

1. 초기화: `A = []`
   - 시간복잡도: $\mathcal{O}(1)$
2. 원소 삽입: `A.append(v); A.sort(reverse=True)`
   - 시간복잡도: $\mathcal{O}(N \log N)$
   - 삽입정렬과 비슷한 방식으로 구현하면: $\mathcal{O}(N)$도 가능
3. 최솟값: `A[-1]`
   - 시간복잡도: $\mathcal{O}(1)$
4. 최솟값 삭제: `A.pop()`
   - 시간복잡도: $\mathcal{O}(1)$

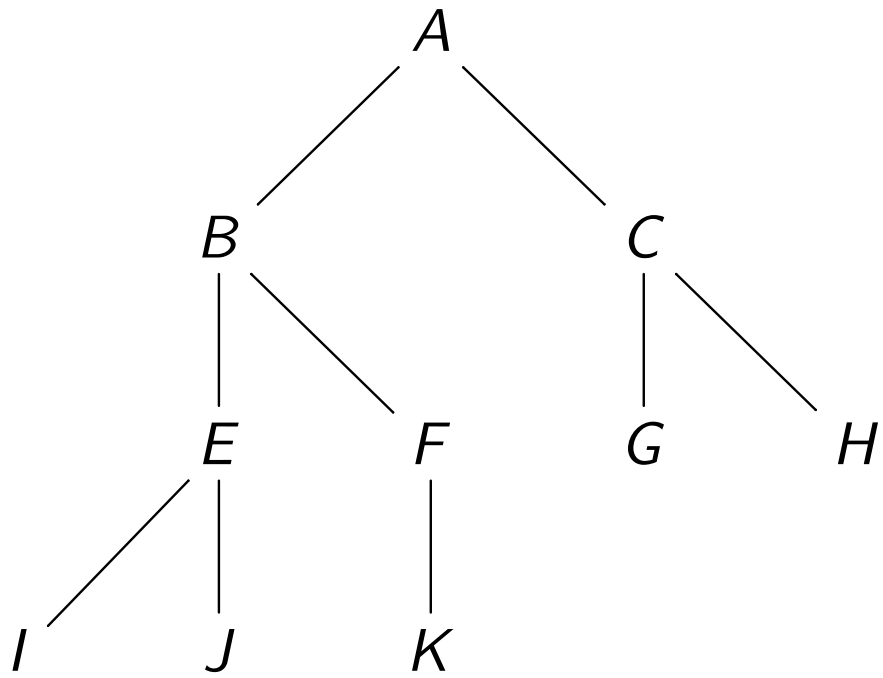
우선순위 큐의 구현 – 힙(*Heap*)

- 두 방식 모두 원소의 삽입과 삭제를 더하면 $\mathcal{O}(N \log N)$ 혹은 $\mathcal{O}(N)$이 걸림
- 좀 더 효율적인 자료구조가 필요함
- 힙이라는 형태의 이진트리를 사용하면 삽입과 삭제 모두 $\mathcal{O}(\log N)$ 시간에 구현 가능

이진트리



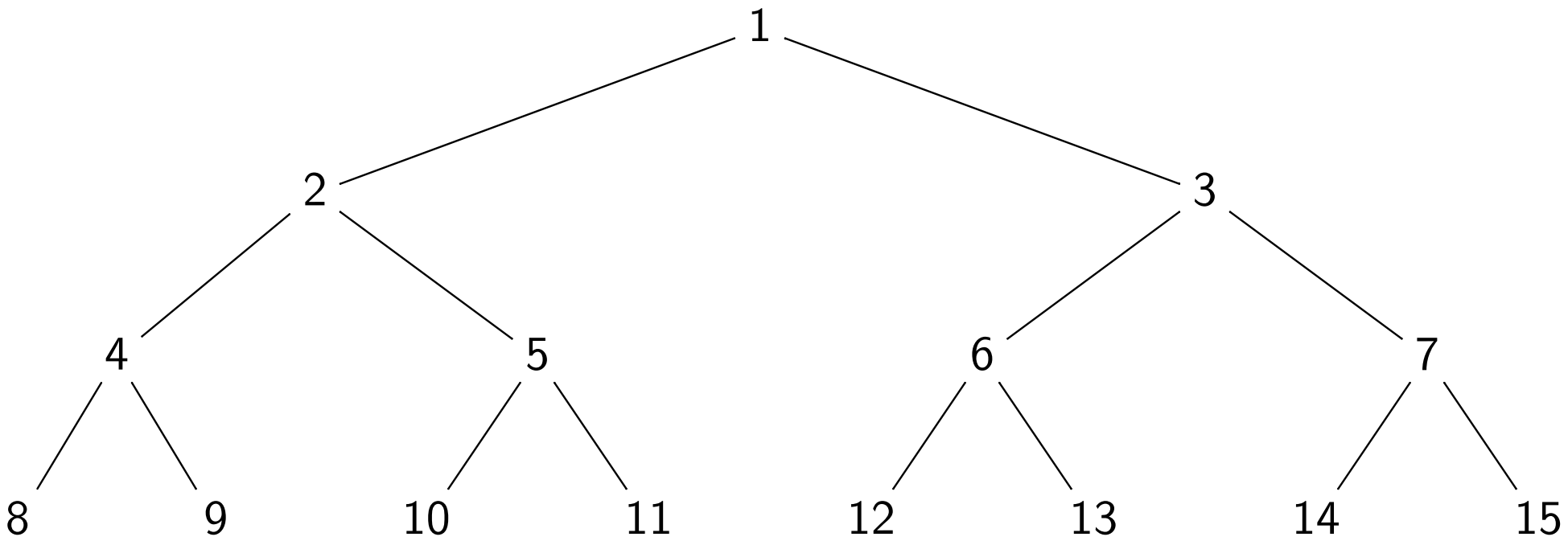
이진 트리



완전 이진트리

- 이진트리: 각 노드의 자식이 2개 이하인 트리
- 완전 이진트리: 높이가 낮은것 부터, 같다면 왼쪽부터 노드를 채운 이진트리
  - 트리의 크기가 정해지면 모양이 정해짐

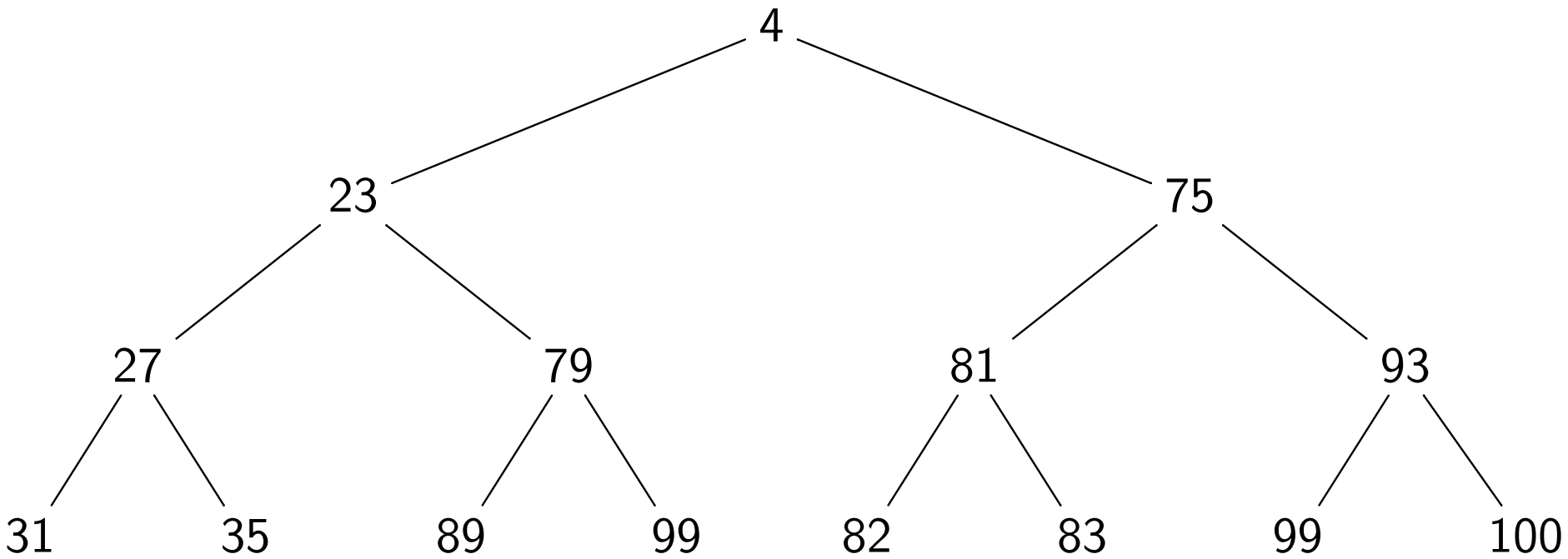
완전이진트리의 표현



- 1부터 차례로 번호를 붙였을 때
  - x의 왼쪽 자식은 $2x$, 오른쪽 자식은 $2x+1$, x의 부모는 $\lfloor \frac{x}{2} \rfloor$
- 포인터를 쓰지 않고 배열로 표현 가능

이진 힙(*Binary Heap*)

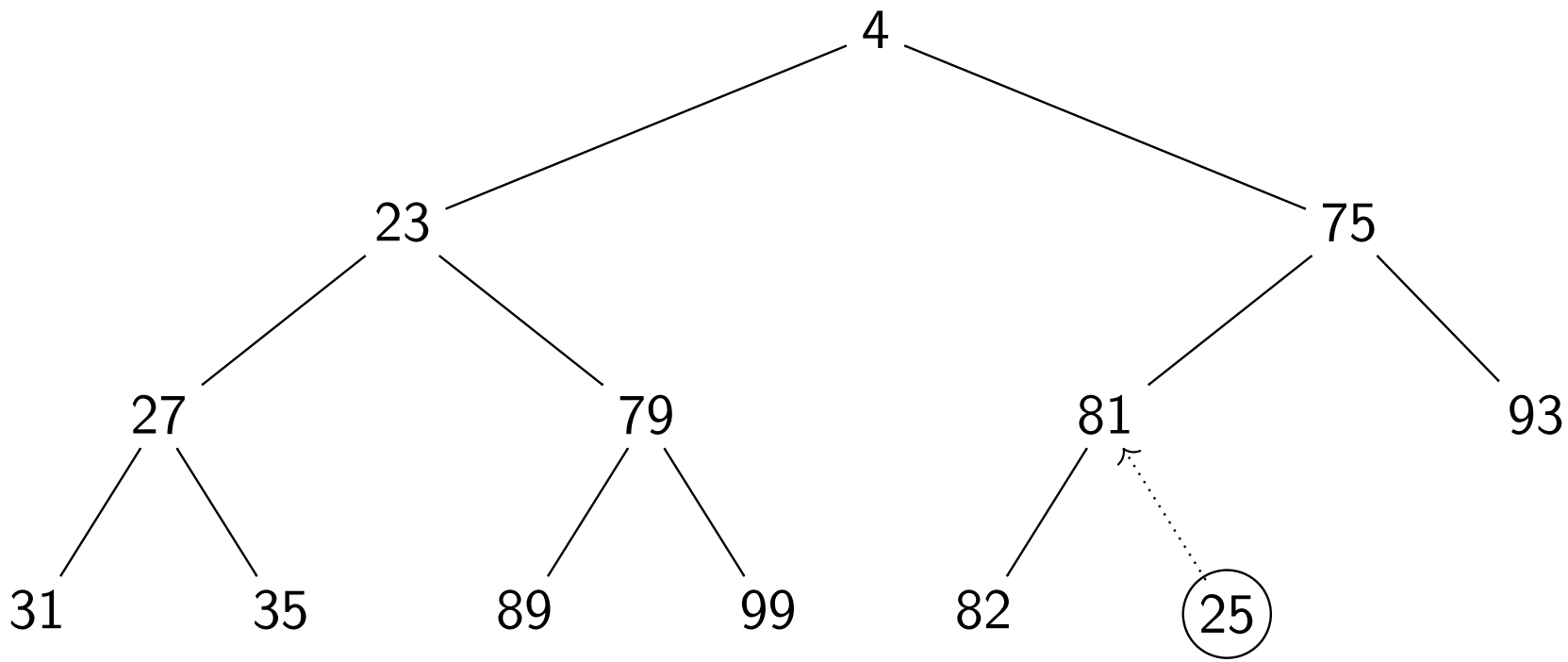
힙 속성



- 힙 속성: A가 B의 부모노드이면, A의 값은 B의 값보다 작거나 같다.
- 최솟값은 항상 루트에 위치하게 됨 – 최솟값을 $O(1)$에 찾을 수 있음

이진 힙(*Binary Heap*)

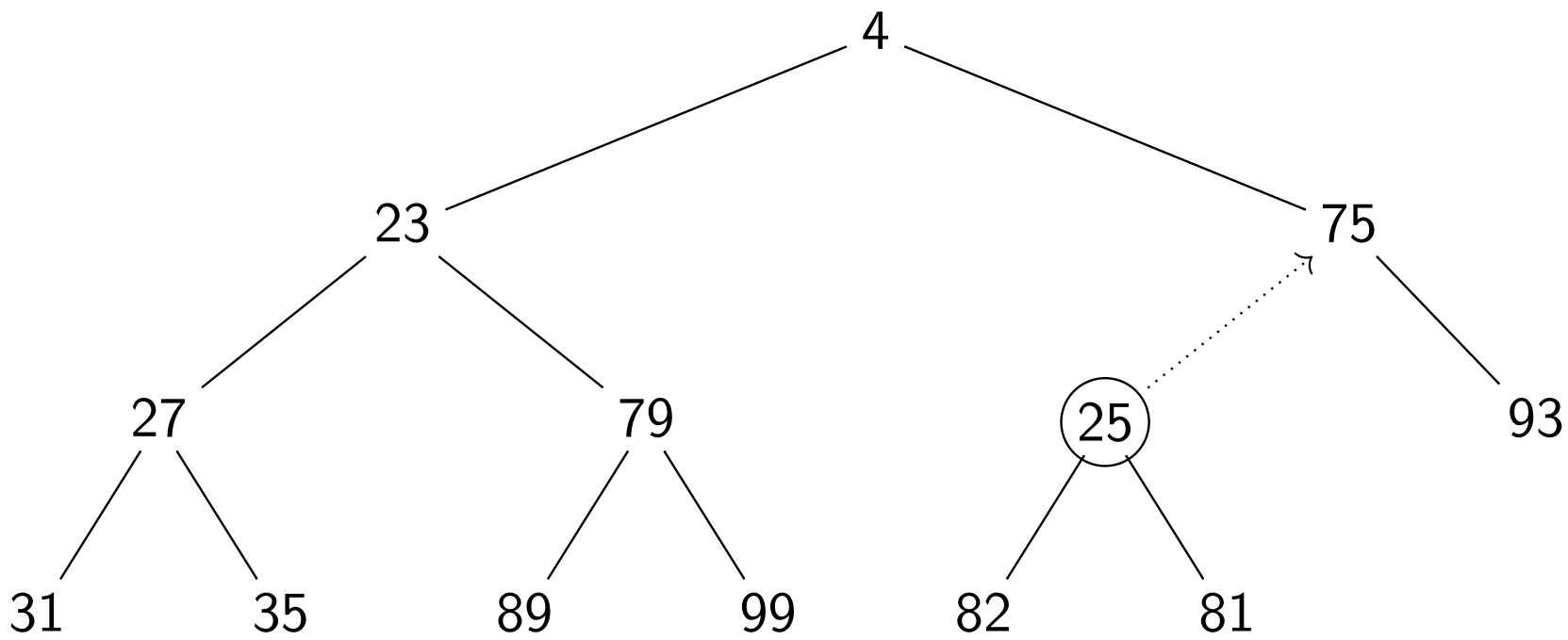
원소 삽입



- 완전 이진트리이기 때문에 삽입하는 곳의 위치가 정해져 있음
- 삽입 후에 힙 속성을 만족하지 않는 것을 고침
- 부모가 자기 자신보다 크면 위치를 바꿈

이진 힙(*Binary Heap*)

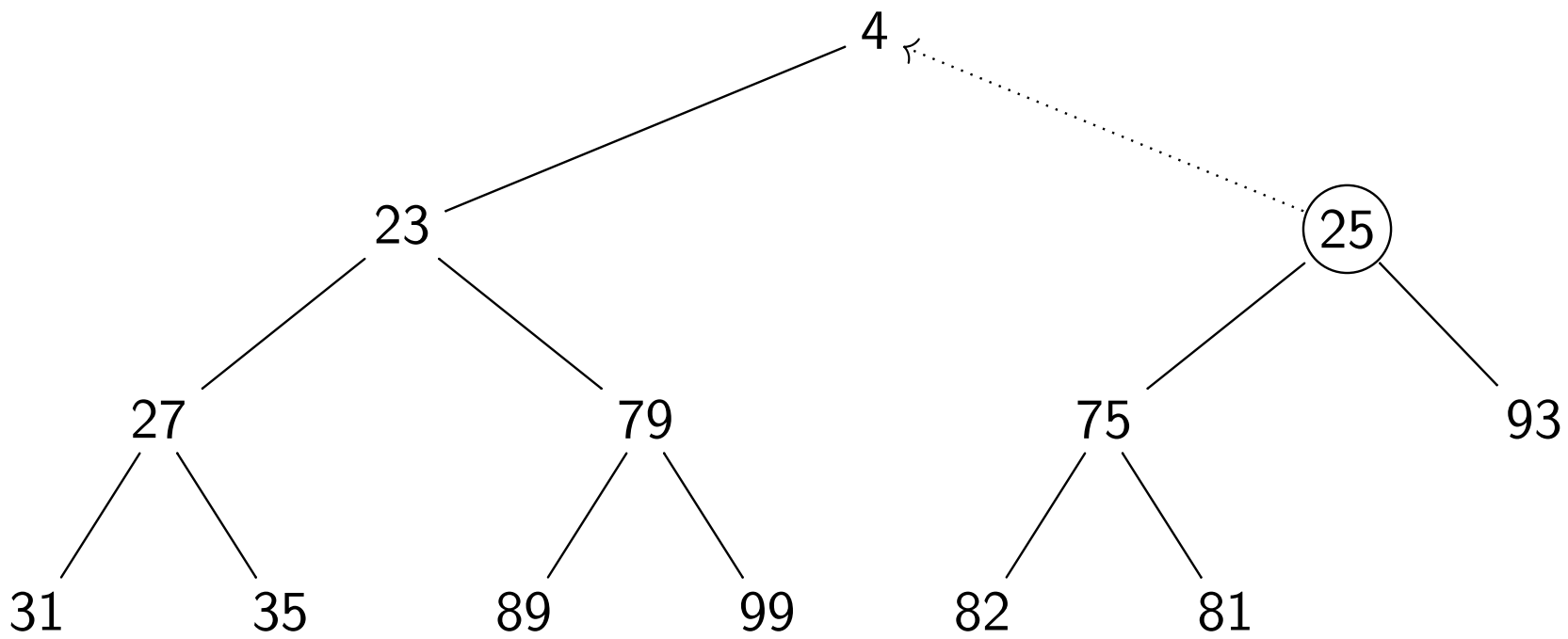
원소 삽입



- 완전 이진트리이기 때문에 삽입하는 곳의 위치가 정해져 있음
- 삽입 후에 힙 속성을 만족하지 않는 것을 고침
- 부모가 자기 자신보다 크면 위치를 바꿈

이진 힙(*Binary Heap*)

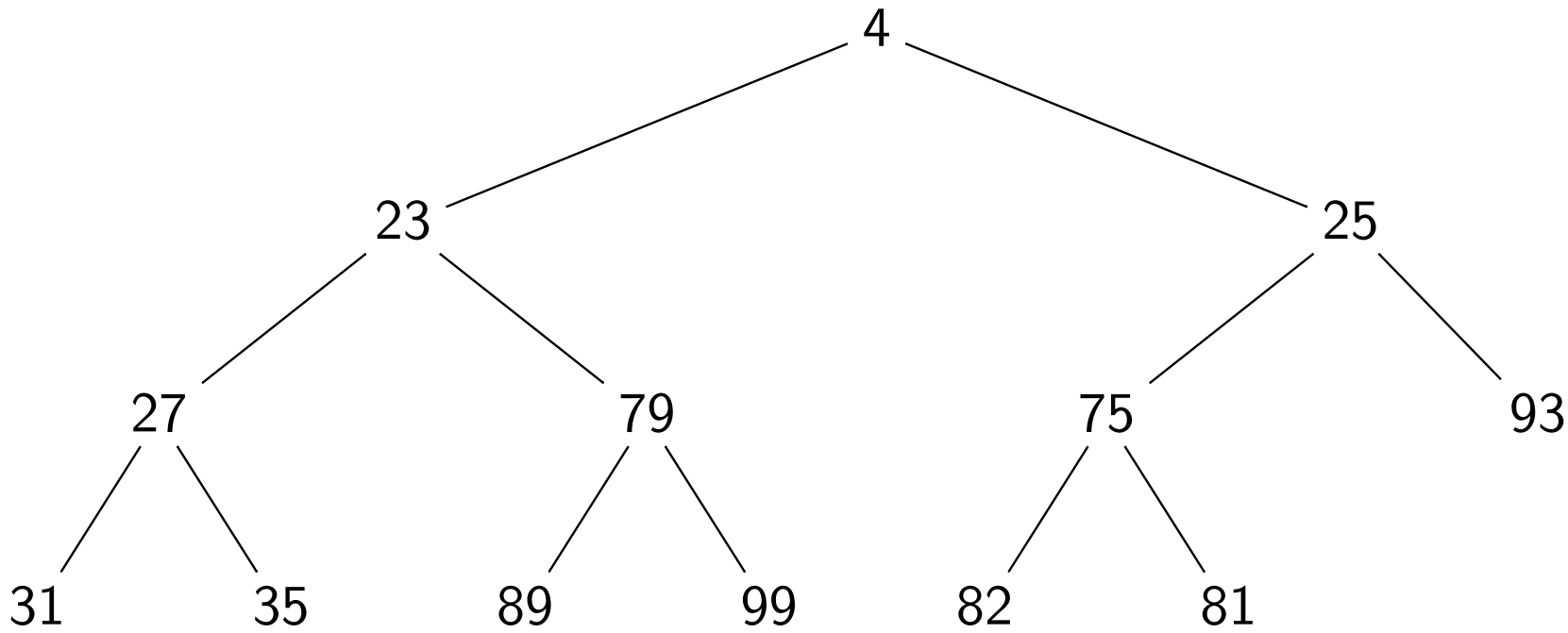
원소 삽입



- 완전 이진트리이기 때문에 삽입하는 곳의 위치가 정해져 있음
- 삽입 후에 힙 속성을 만족하지 않는 것을 고침
- 부모가 자기 자신보다 크면 위치를 바꿈

이진 힙(*Binary Heap*)

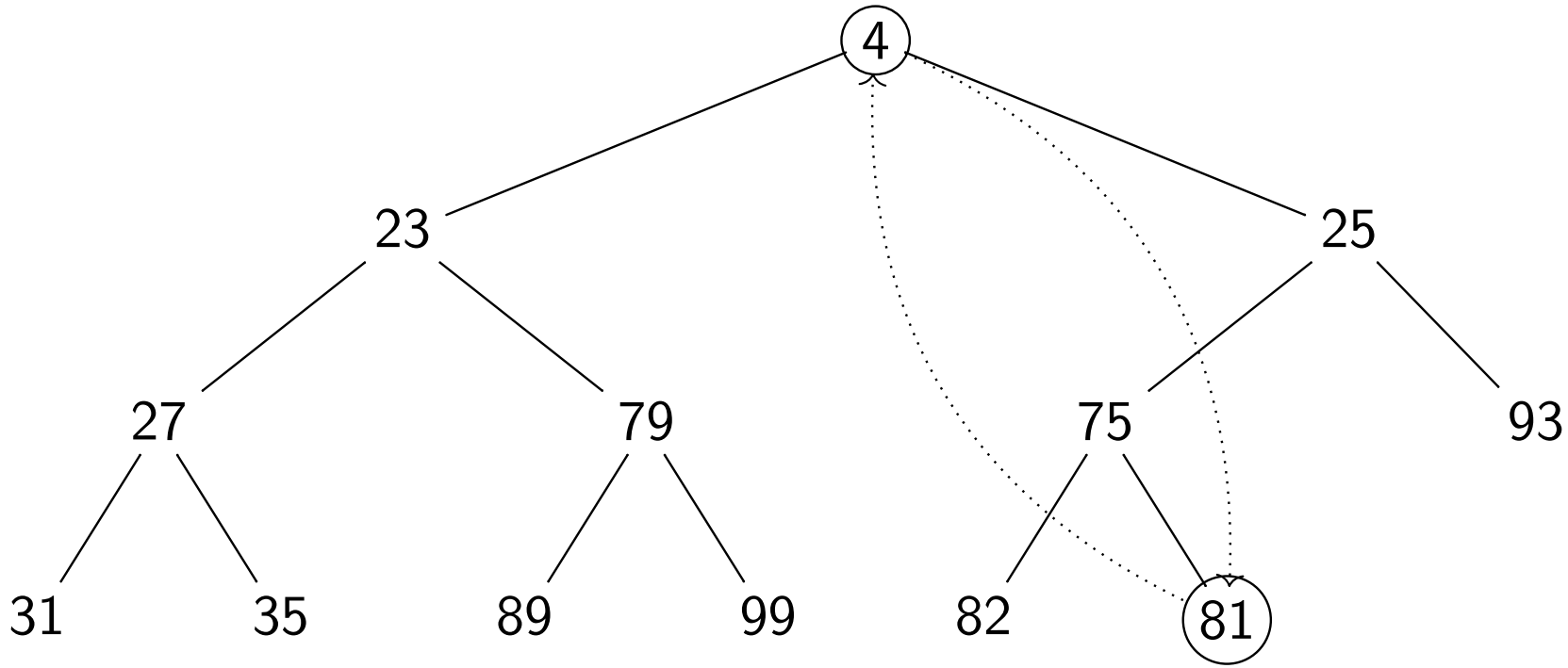
원소 삽입



- 완전 이진트리이기 때문에 삽입하는 곳의 위치가 정해져 있음
- 삽입 후에 힙 속성을 만족하지 않는 것을 고침
- 부모가 자기 자신보다 크면 위치를 바꿈

이진 힙(*Binary Heap*)

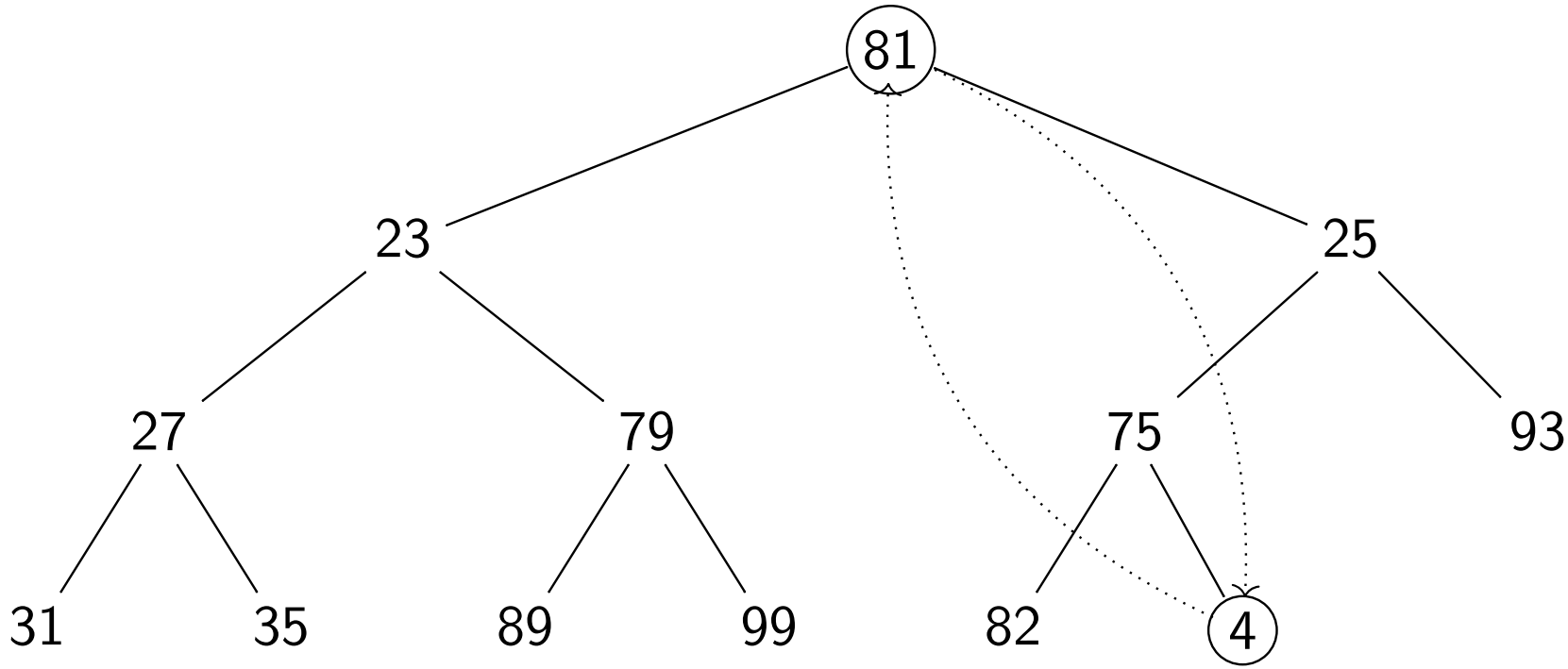
원소 삭제



- 가장 마지막 원소와 위치를 바꾼 후 가장 마지막 원소를 삭제

이진 힙(*Binary Heap*)

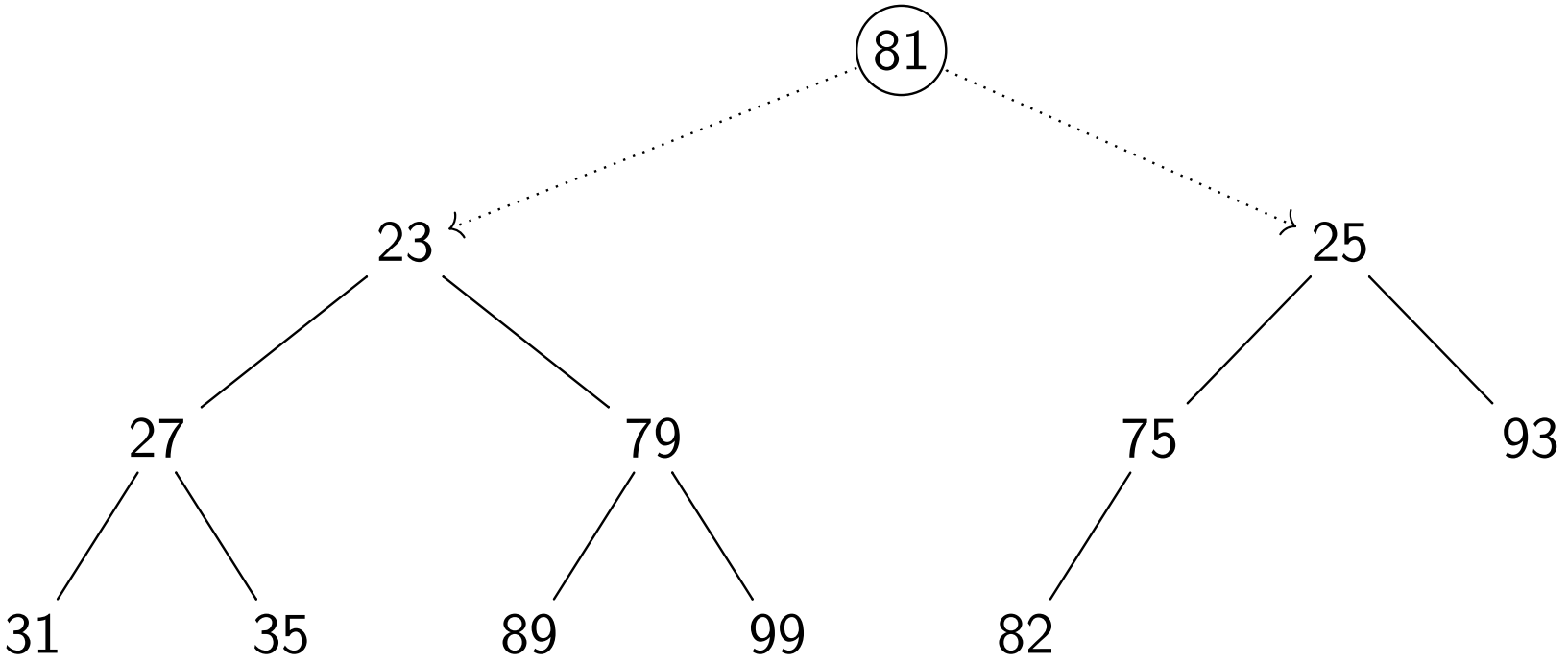
원소 삭제



- 가장 마지막 원소와 위치를 바꾼 후 가장 마지막 원소를 삭제

이진 힙(*Binary Heap*)

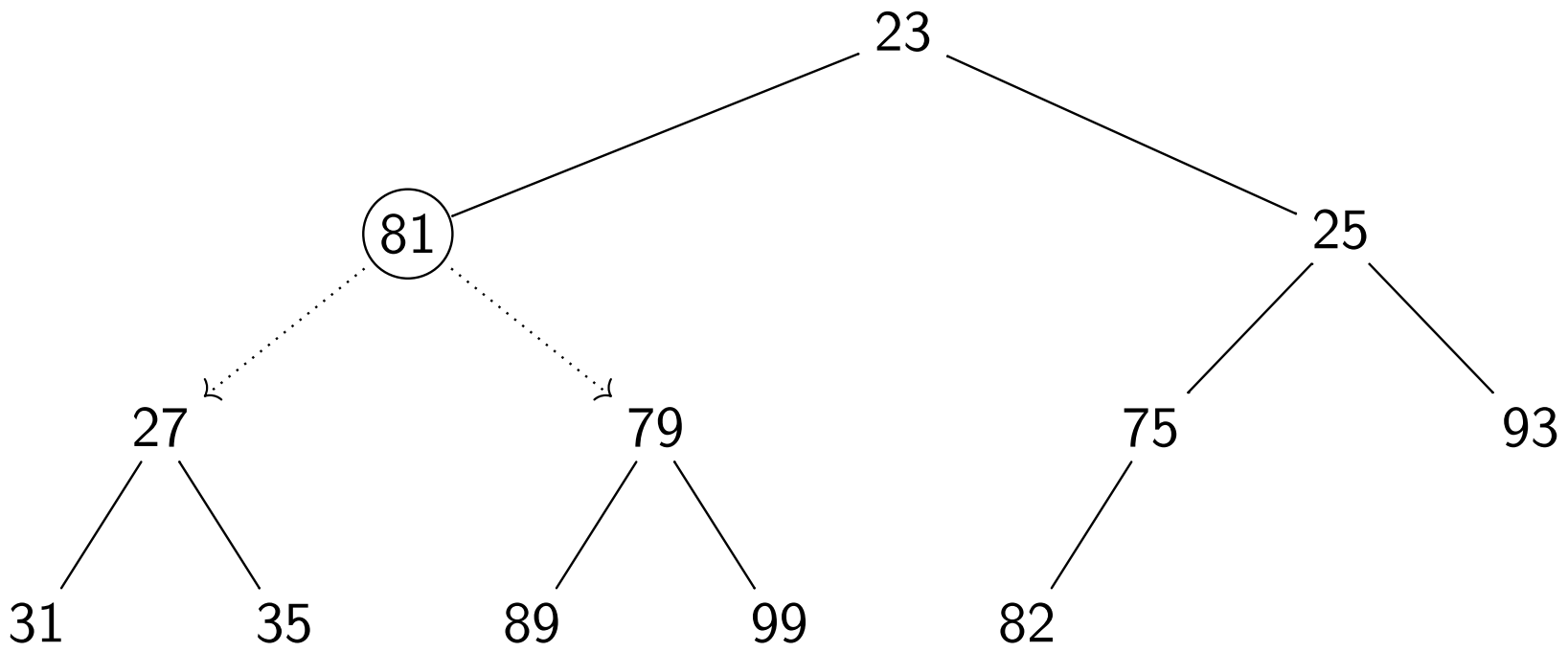
원소 삭제



- 힙 조건을 만족하도록 수를 두 자식 중 작은 수와 바꿈

이진 힙(*Binary Heap*)

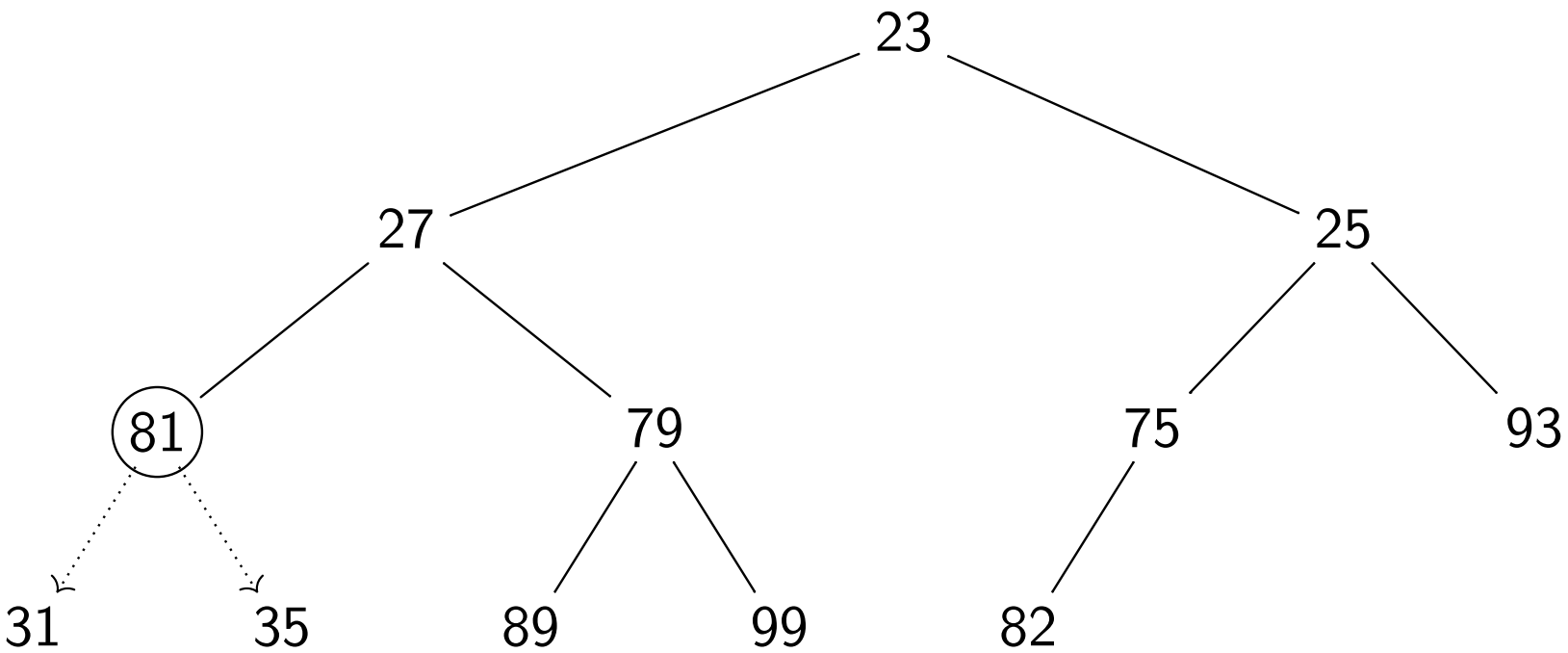
원소 삭제



- 힙 조건을 만족하도록 수를 두 자식 중 작은 수와 바꿈

이진 힙(*Binary Heap*)

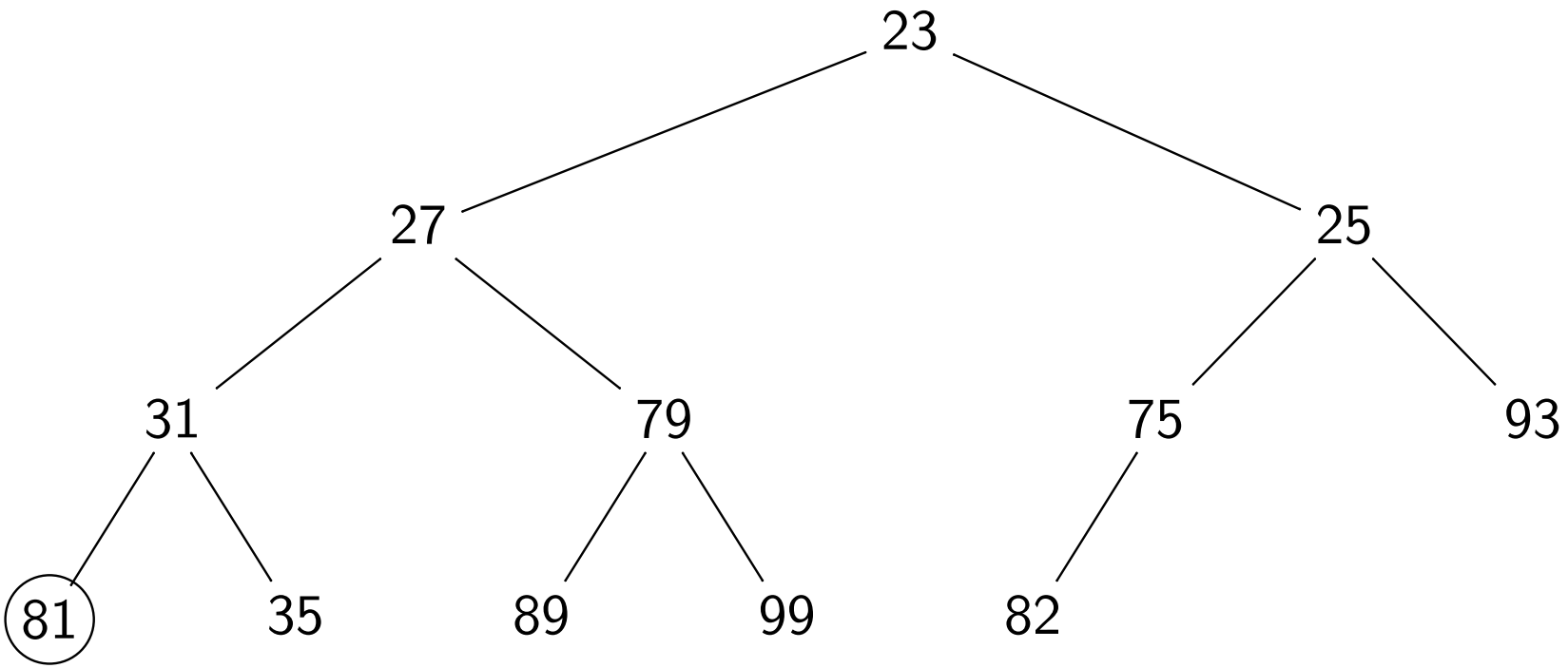
원소 삭제



- 힙 조건을 만족하도록 수를 두 자식 중 작은 수와 바꿈

이진 힙(*Binary Heap*)

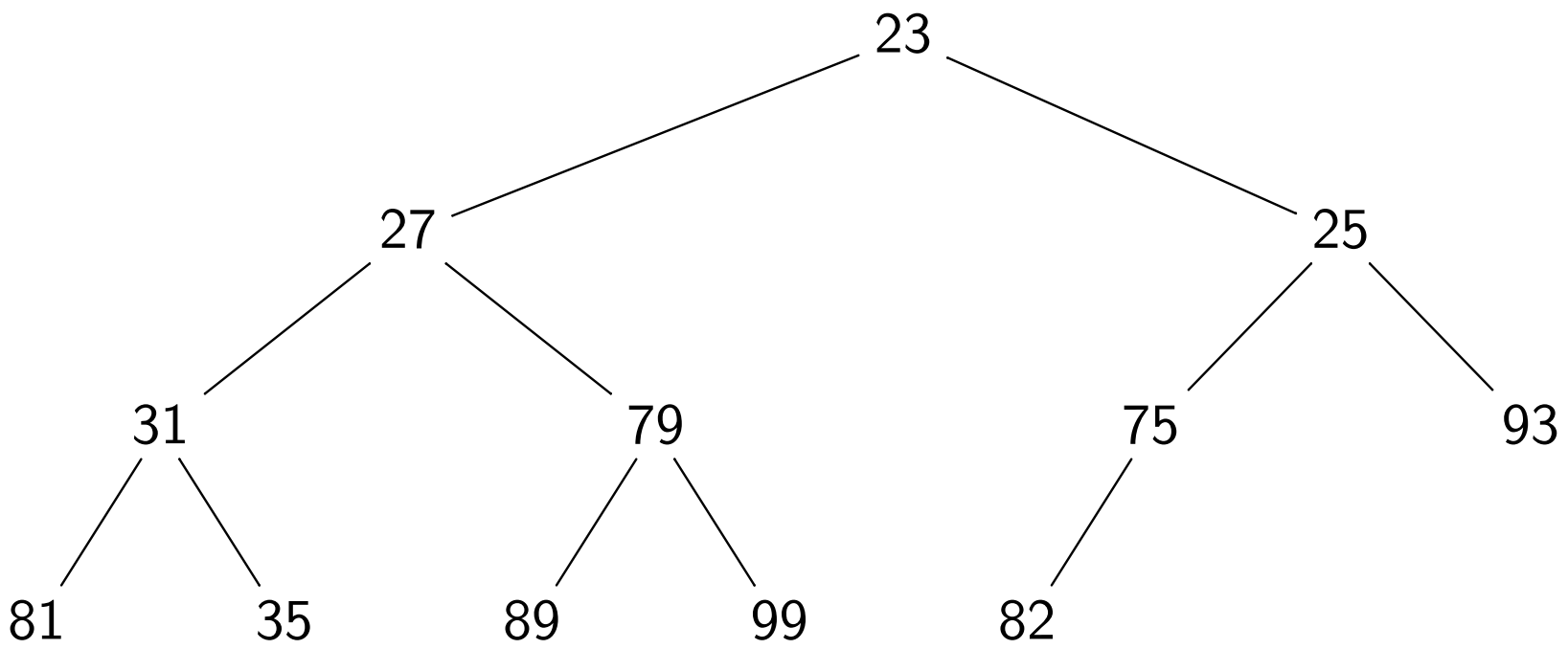
원소 삭제



- 힙 조건을 만족하도록 수를 두 자식 중 작은 수와 바꿈

이진 힙(*Binary Heap*)

원소 삭제



- 힙 조건을 만족하도록 수를 두 자식 중 작은 수와 바꿈

시간복잡도

연습문제

1. *(Easy)* 힙에 최솟값 찾기가 $\mathcal{O}(1)$임을 증명하여라.
2. *(Medium)* 힙의 높이가 $\mathcal{O}(\log N)$임을 증명하여라.
3. *(Easy)* 힙에 새로운 원소 추가가 $\mathcal{O}(\log N)$임을 증명하여라.
4. *(Easy)* 힙의 최솟값 삭제가 $\mathcal{O}(\log N)$임을 증명하여라.
5. 힙 정렬은 힙에 N개의 원소를 차례로 넣고, 차례로 삭제하는 과정을 말한다.
   - *(Easy)* 힙 정렬의 시간 복잡도가 최악의 경우 $\mathcal{O}(N \log N)$임을 증명하여라.
   - *(Very Hard)* 모든 수가 다른 배열을 힙 정렬 할 때, 시간 복잡도가 최선의 경우 $\Omega(N \log N)$임을 증명하여라.

우선순위 큐의 구현 – 이진 힙

1. 초기화: 시간복잡도 $\mathcal{O}(1)$
2. 원소 삽입: $\mathcal{O}(\log N)$
3. 최솟값: $\mathcal{O}(1)$
4. 최솟값 삭제: $\mathcal{O}(1)$

우선순위 큐의 활용

연습문제

1. *(Hard)* 우선순위 큐 두 개를 써서, 임의의 값을 삭제할 수 있는 우선순위 큐를 만들어라.
2. *(Hard)* 우선순위 큐 두 개를 써서, 원소의 삽입과 중앙값을 처리할 수 있는 우선순위 큐를 만들어라.