ITMO UNIVERSITY

How to Win Coding Competitions: Secrets of Champions

Week 4: Sorting and Search Algorithms 2
Lecture 6: Priority queue and binary heap

Maxim Buzdalov
Saint Petersburg 2016

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Alone in Berlin

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Alone in Berlin
Monster Trucks

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Alone in Berlin
Monster Trucks

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Monster Trucks

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Monster Trucks
The Book of Love

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

The Book of Love

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

The Book of Love
Sleepless

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

The Book of Love
Sleepless

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Sleepless

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Sleepless
The Red Turtle

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Sleepless
The Red Turtle
The Sense of an Ending

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

Sleepless
The Red Turtle
The Sense of an Ending

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

The Red Turtle
The Sense of an Ending

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

The Red Turtle
The Sense of an Ending

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

The Sense of an Ending

- Assume you are a cinema critic, and your job is to watch movies once a day
- You may process requests in the order of their appearance
- Ordinary queue (First In, First Out)

The Sense of an Ending

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017
The Fate of the Furious: 04 Apr 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017
The Fate of the Furious: 04 Apr 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017
Beauty and the Beast: 23 Feb 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017
Beauty and the Beast: 23 Feb 2017
Despicable Me 3: 14 Jun 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017
Beauty and the Beast: 23 Feb 2017
Despicable Me 3: 14 Jun 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017
Despicable Me 3: 14 Jun 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017
Despicable Me 3: 14 Jun 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017

- Assume you are a cinema critic, and your job is to watch movie trailers once a day
- You have to do it before the movie premiers
- Priority queue: First In, Urgent Out

Spider-Man: Homecoming: 28 Jun 2017

Priority queue: Interface and implementation

	Vector, Queue, Deque, ...	Same with sorting
[+]: add an element	$O(1)$	$O(n)$
[?]: query the smallest element	$O(n)$	$O(1)$
[−]: remove the smallest element	$O(n)$	$O(1)$

Priority queue: Interface and implementation

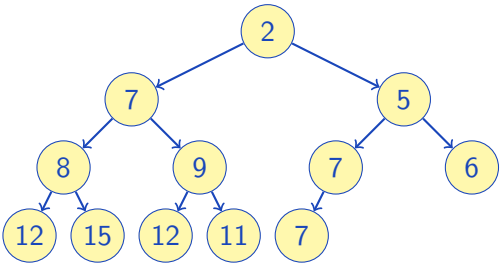
	Vector, Queue, Deque, ...	Same with sorting
[+]: add an element	$O(1)$	$O(n)$
[?]: query the smallest element	$O(n)$	$O(1)$
[−]: remove the smallest element	$O(n)$	$O(1)$

Priority queue: Interface and implementation

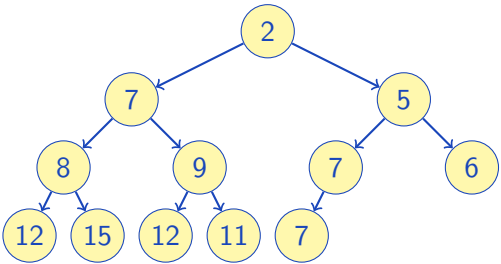
	Vector, Queue, Deque, ...	Same with sorting
[+]: add an element	$O(1)$	$O(n)$
[?]: query the smallest element	$O(n)$	$O(1)$
[−]: remove the smallest element	$O(n)$	$O(1)$

Is this the best we can do?

Binary heap: a tree-like structure



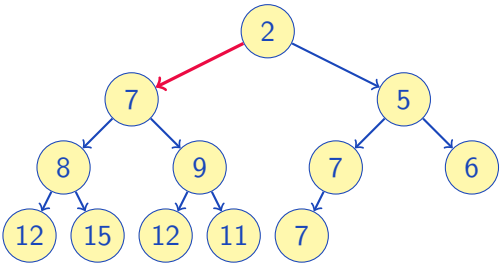
Binary heap: a tree-like structure



Properties of a binary heap:

- A node is not less than its parent

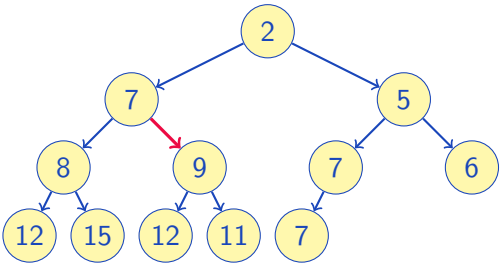
Binary heap: a tree-like structure



Properties of a binary heap:

- A node is not less than its parent

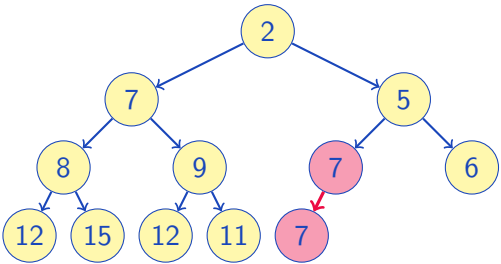
Binary heap: a tree-like structure



Properties of a binary heap:

- A node is not less than its parent

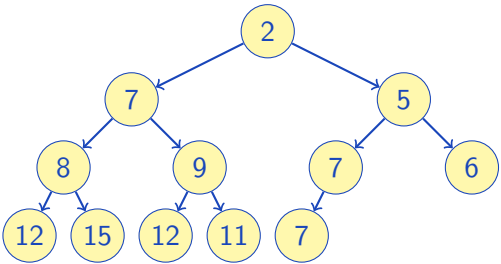
Binary heap: a tree-like structure



Properties of a binary heap:

- A node is not less than its parent

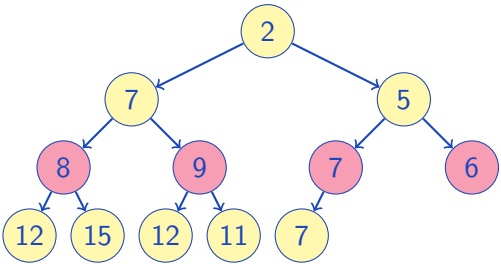
Binary heap: a tree-like structure



Properties of a binary heap:

- A node is not less than its parent
- Every node has at most two children

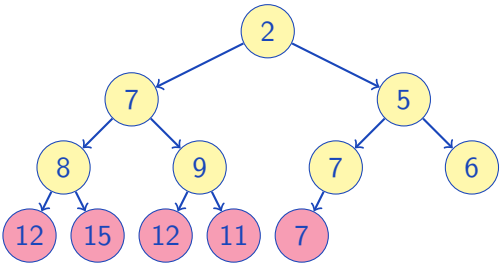
Binary heap: a tree-like structure



Properties of a binary heap:

- A node is not less than its parent
- Every node has at most two children
- All levels but the last one are complete

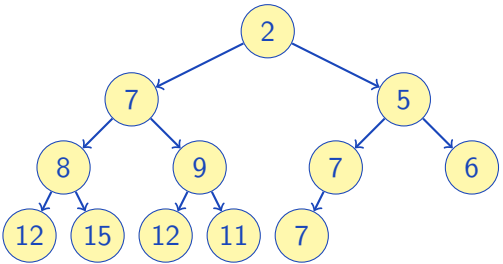
Binary heap: a tree-like structure



Properties of a binary heap:

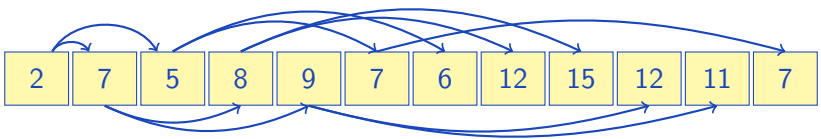
- A node is not less than its parent
- Every node has at most two children
- All levels but the last one are complete
- The last level is filled from the left

Binary heap: a tree-like structure

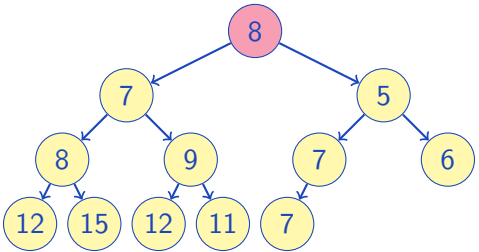


Properties of a binary heap:

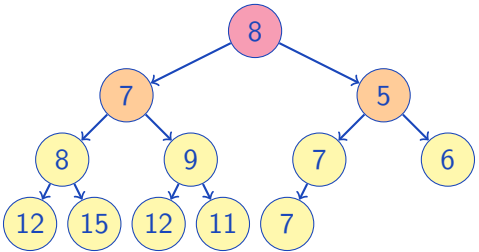
- A node is not less than its parent
- Every node has at most two children
- All levels but the last one are complete
- The last level is filled from the left
- Can be stored in an array or a vector
- Node at index $i \rightarrow$ parent at index $\lfloor \frac{i}{2} \rfloor$



A node has just become greater than one of its children. How to repair the heap?

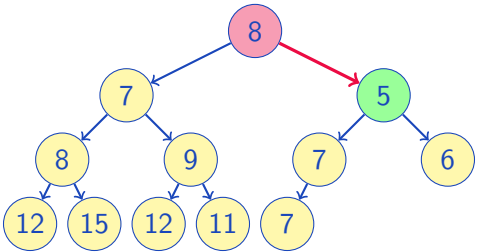


A node has just become greater than one of its children. How to repair the heap?



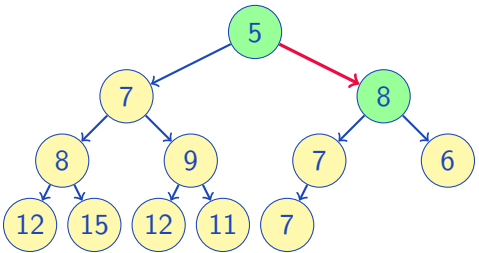
- Look at the children

A node has just become greater than one of its children. How to repair the heap?



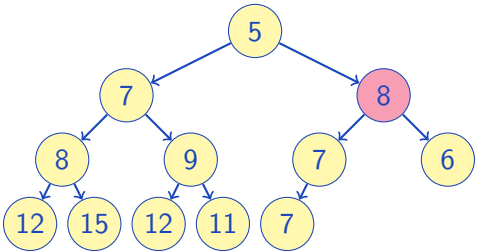
- Look at the children
- Choose the smallest

A node has just become greater than one of its children. How to repair the heap?



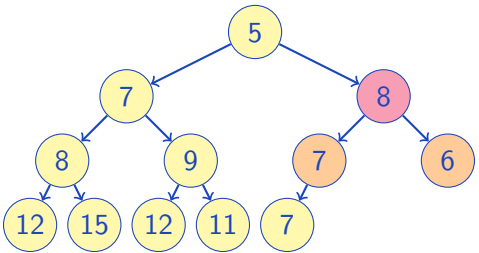
- Look at the children
- Choose the smallest
- Swap with it

A node has just become greater than one of its children. How to repair the heap?



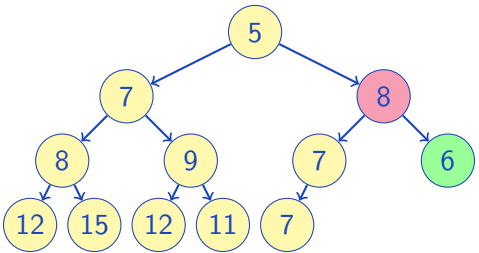
- Look at the children
- Choose the smallest
- Swap with it
- If still too large, continue

A node has just become greater than one of its children. How to repair the heap?



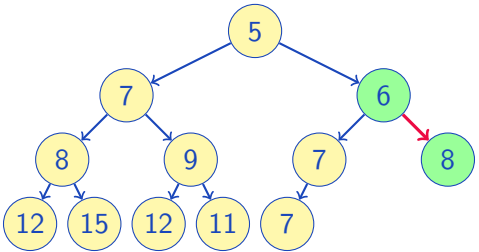
- Look at the children
- Choose the smallest
- Swap with it
- If still too large, continue

A node has just become greater than one of its children. How to repair the heap?



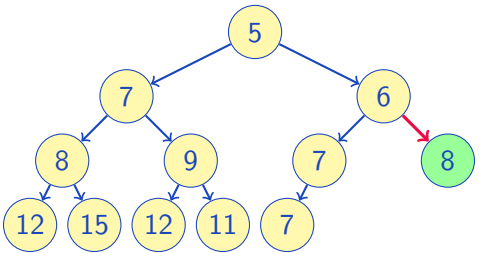
- Look at the children
- Choose the smallest
- Swap with it
- If still too large, continue

A node has just become greater than one of its children. How to repair the heap?



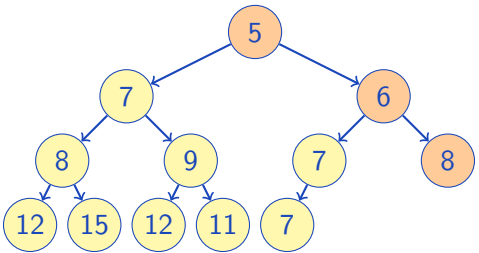
- Look at the children
- Choose the smallest
- Swap with it
- If still too large, continue

A node has just become greater than one of its children. How to repair the heap?



- Look at the children
- Choose the smallest
- Swap with it
- If still too large, continue

A node has just become greater than one of its children. How to repair the heap?

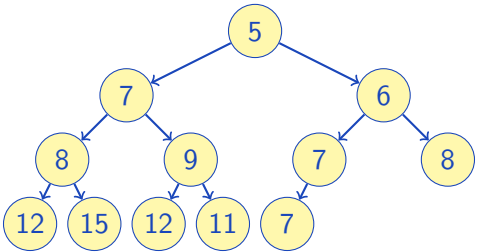


- Look at the children
- Choose the smallest
- Swap with it
- If still too large, continue

Complexity of the operation:

$O(\text{height of a broken node}) = O(\log n)$

A node has just become greater than one of its children. How to repair the heap?



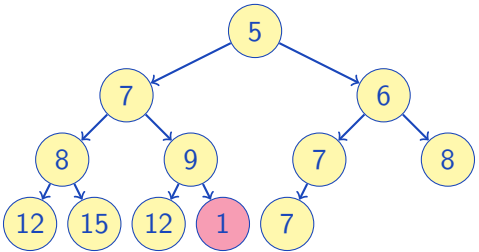
- Look at the children
- Choose the smallest
- Swap with it
- If still too large, continue

Complexity of the operation:

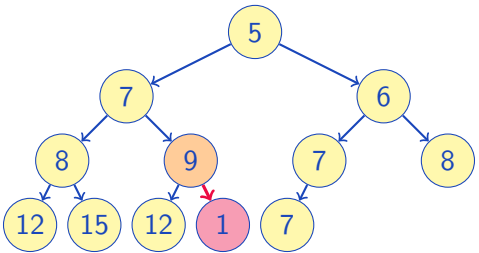
$O(\text{height of a broken node}) = O(\log n)$

We will call this operation "Sift down"

A node has just become smaller. How to repair the heap?

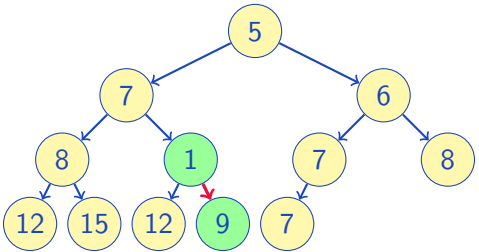


A node has just become smaller. How to repair the heap?



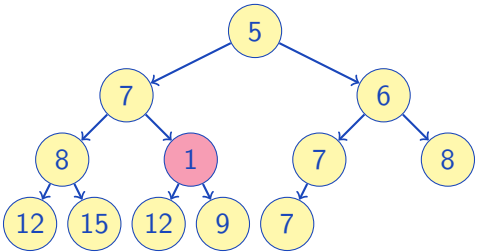
- Look at the parent

A node has just become smaller. How to repair the heap?



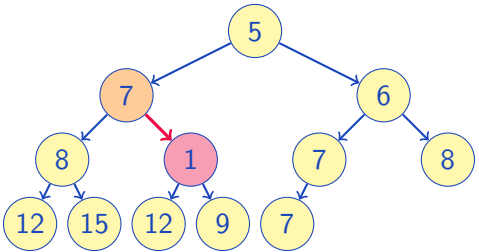
- Look at the parent
- If the parent is greater, swap with it

A node has just become smaller. How to repair the heap?



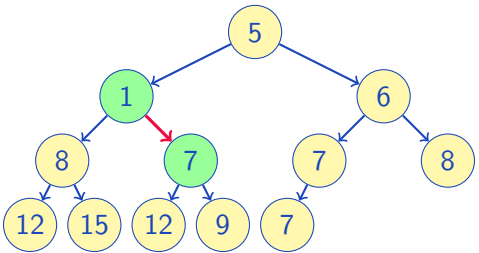
- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

A node has just become smaller. How to repair the heap?



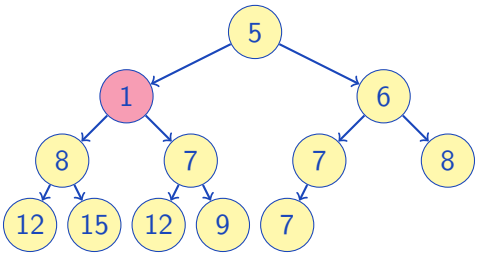
- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

A node has just become smaller. How to repair the heap?



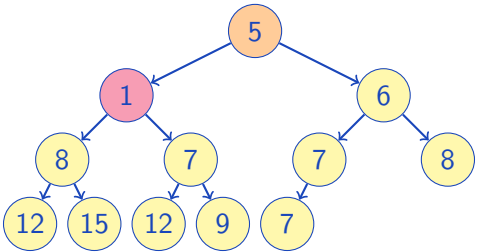
- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

A node has just become smaller. How to repair the heap?



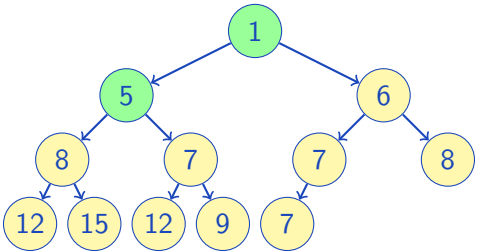
- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

A node has just become smaller. How to repair the heap?



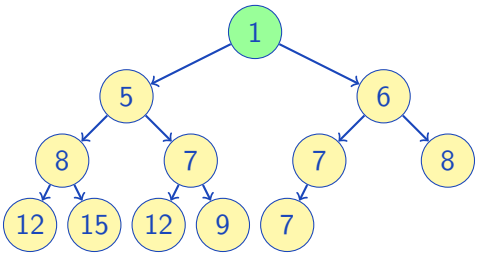
- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

A node has just become smaller. How to repair the heap?



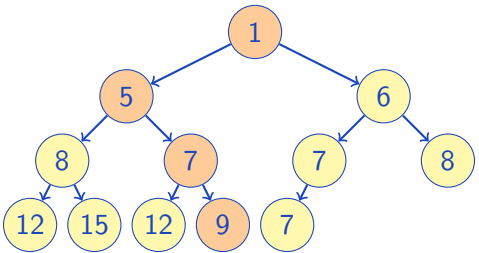
- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

A node has just become smaller. How to repair the heap?



- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

A node has just become smaller. How to repair the heap?

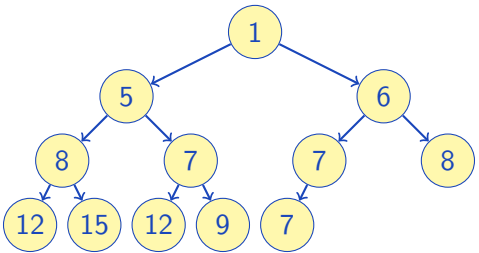


- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

Complexity of the operation:

$O(\text{depth of the broken node}) = O(\log n)$

A node has just become smaller. How to repair the heap?



- Look at the parent
- If the parent is greater, swap with it
- If still too small, continue

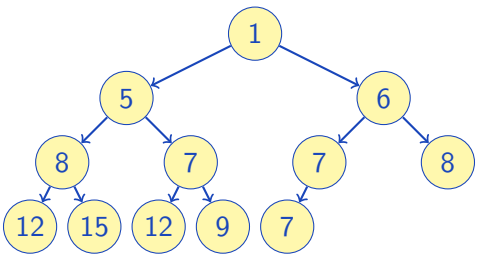
Complexity of the operation:

$O(\text{depth of the broken node}) = O(\log n)$

We will call this operation "Sift up"

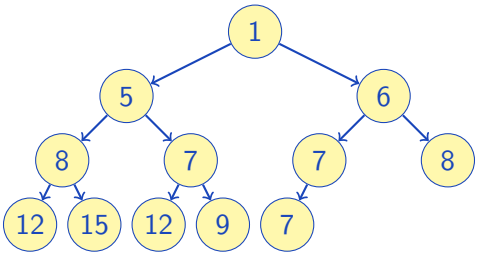
Priority queue: Supported operations

- [+]: add an element
- [?]: query the smallest element
- [−]: remove the smallest element



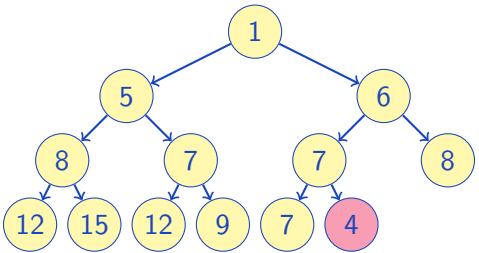
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up
- [?]: query the smallest element
- [−]: remove the smallest element



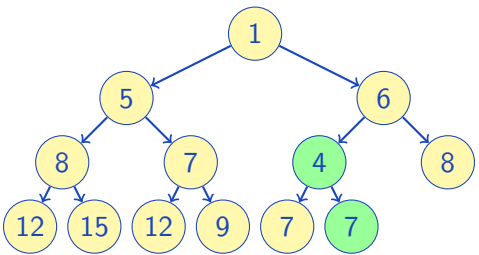
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up
- [?]: query the smallest element
- [−]: remove the smallest element



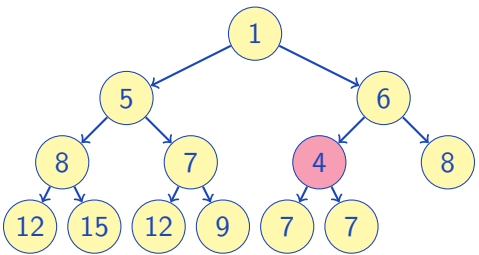
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up
- [?]: query the smallest element
- [−]: remove the smallest element



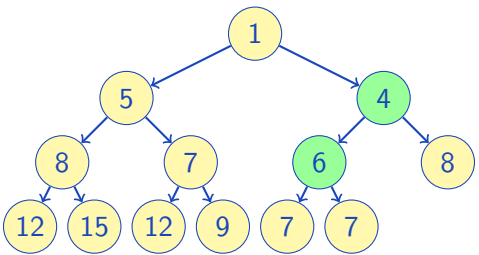
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up
- [?]: query the smallest element
- [−]: remove the smallest element



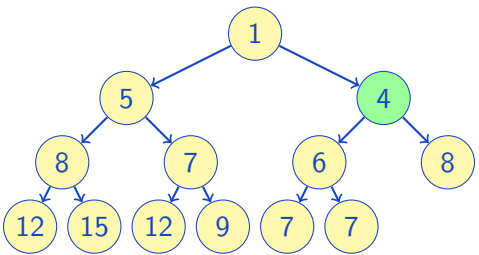
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up
- [?]: query the smallest element
- [−]: remove the smallest element



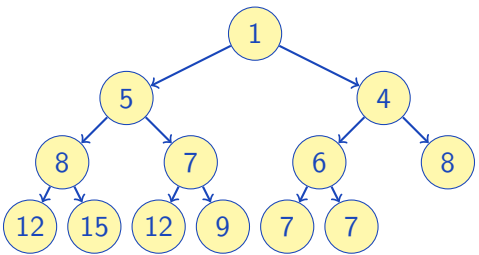
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up
- [?]: query the smallest element
- [−]: remove the smallest element



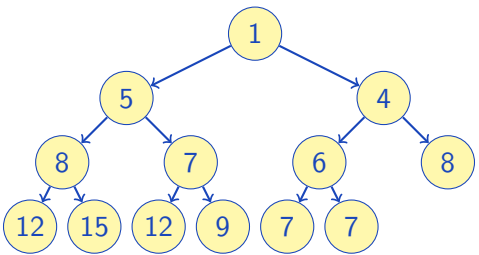
Priority queue: Supported operations

- [+]: add an element $\rightarrow$ add the new node, then sift it up $\rightarrow$ $O(\log n)$
- [?]: query the smallest element
- [−]: remove the smallest element



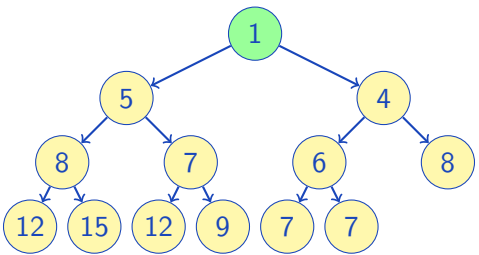
Priority queue: Supported operations

- [+]: add an element $\rightarrow$ add the new node, then sift it up $\rightarrow O(\log n)$
- [?]: query the smallest element
- [−]: remove the smallest element



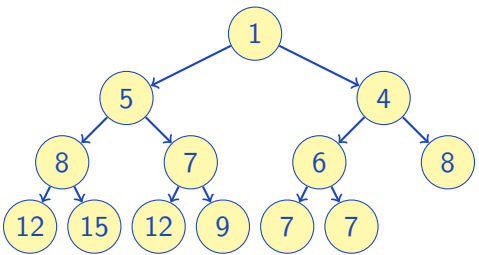
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root
- [−]: remove the smallest element



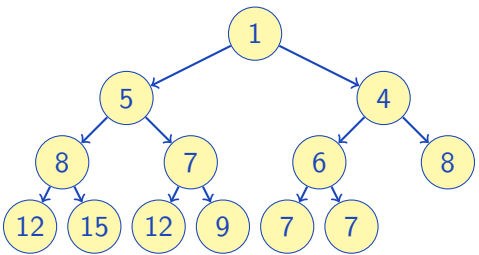
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element



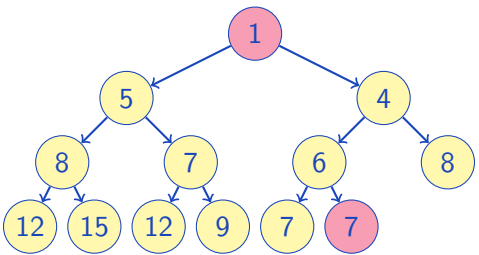
Priority queue: Supported operations

- [+]: add an element $\rightarrow$ add the new node, then sift it up $\rightarrow$ $O(\log n)$
- [?]: query the smallest element $\rightarrow$ query the root $\rightarrow$ $O(1)$
- [-]: remove the smallest element



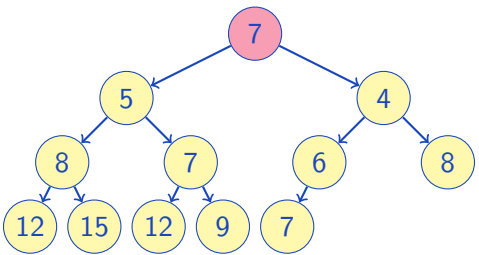
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element
  → move the last node to the root, then sift it down



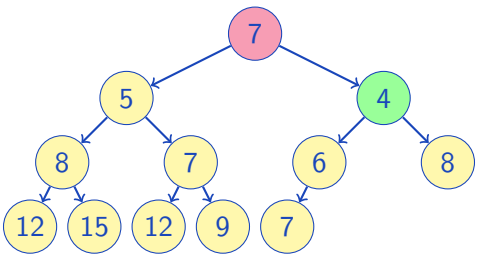
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element
  → move the last node to the root, then sift it down



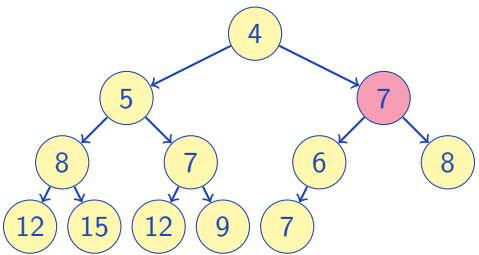
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element
  → move the last node to the root, then sift it down



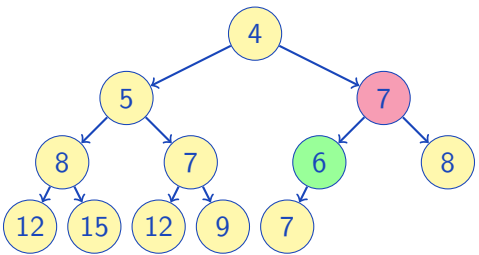
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element
  → move the last node to the root, then sift it down



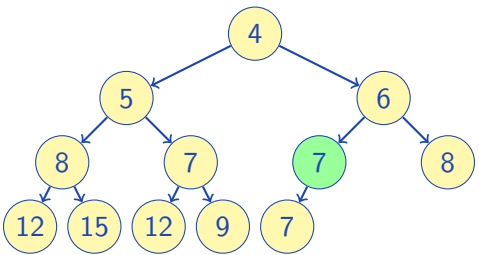
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element
  → move the last node to the root, then sift it down



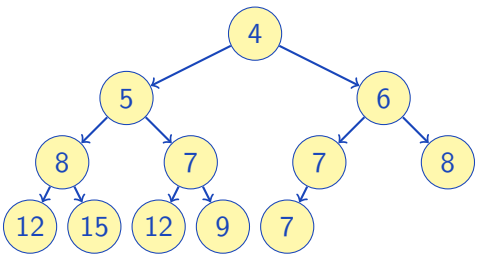
Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element
  → move the last node to the root, then sift it down



Priority queue: Supported operations

- [+]: add an element → add the new node, then sift it up → $O(\log n)$
- [?]: query the smallest element → query the root → $O(1)$
- [−]: remove the smallest element
  → move the last node to the root, then sift it down → $O(\log n)$



Priority queue: Implementations and their performance

	Vector, Queue, Deque, ...	Same with sorting	Binary heap
[+]	$O(1)$	$O(n)$	$O(\log n)$
[?]	$O(n)$	$O(1)$	$O(1)$
[−]	$O(n)$	$O(1)$	$O(\log n)$

- Java
  - `java.util.PriorityQueue<T>`
- C++
  - Warning: C++ assumes MAXIMUM value at root
    - everything is the same, but min $\leftrightarrow$ max, $\leq \leftrightarrow \geq$, etc.
  - High-level data structure from `<queue>`
    - `std::priority_queue<T>`
  - Low-level operations from `<algorithm>`
    - `std::make_heap`
    - `std::push_heap`
    - `std::pop_heap`
    - `std::sort_heap`
- Python
  - The `heapq` module