Fibonacci Heap

2005019 Jarin Tasneem
2005027 Swastika Pandit
2005030 Fairuz Mubashwera

Department of CSE
Bangladesh University of Engineering and Technology

February 13, 2024

1. Introduction
2. Motivation
3. Features
4. Operations
5. Advantages and Disadvantages
6. Application

Table of Contents

1. Introduction
2. Motivation
3. Features
4. Operations
   - Insert
   - Union
   - Decrease-Key
   - Extract-Min
5. Advantages and Disadvantages
   - Advantages
   - Disadvantages
6. Application

What is Fibonacci Heap?

Fibonacci heap is a data structure used for implementing priority queues.

What is Fibonacci Heap?

Fibonacci heap is a data structure used for implementing priority queues.
It is an optimized extension of binomial heap.

Table of Contents

1. Introduction
2. Motivation
3. Features
4. Operations
   - Insert
   - Union
   - Decrease-Key
   - Extract-Min
5. Advantages and Disadvantages
   - Advantages
   - Disadvantages
6. Application

Comparison between binary heap, binomial heap and Fibonacci heap

Operation	Binary heap	Binomial heap	Fibonacci heap
Insert	O(log(n))	O(log(n))	O(1)
Decrease key	O(log(n))	O(log(n))	O(1)
Extract min	O(log(n))	O(log(n))	O(log(n))
Delete	O(log(n))	O(log(n))	O(log(n))
Find min	O(1)	O(log(n))	O(1)
Union	O(n)	O(log(n))	O(1)

Table of Contents

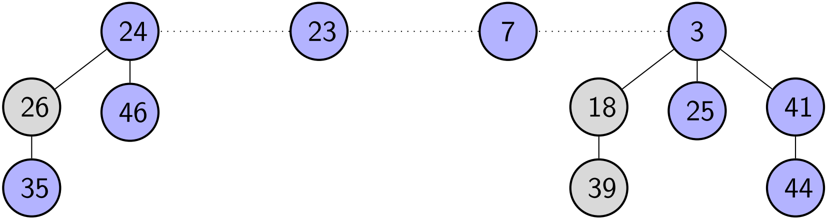
1. Introduction
2. Motivation
3. Features
4. Operations
   - Insert
   - Union
   - Decrease-Key
   - Extract-Min
5. Advantages and Disadvantages
   - Advantages
   - Disadvantages
6. Application

Basic idea

- Structure similar to binomial heap, but more flexible.
- "Lazy" union
  - Defers union until the next extract-min
- Decrease-key and union run in $O(1)$ amortized time

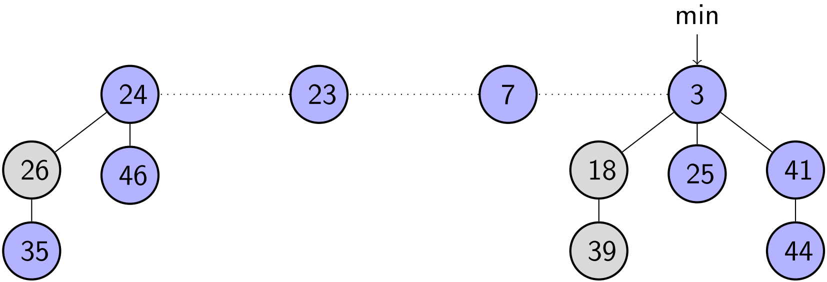
Structure of Fibonacci heap

- Set of heap-ordered trees



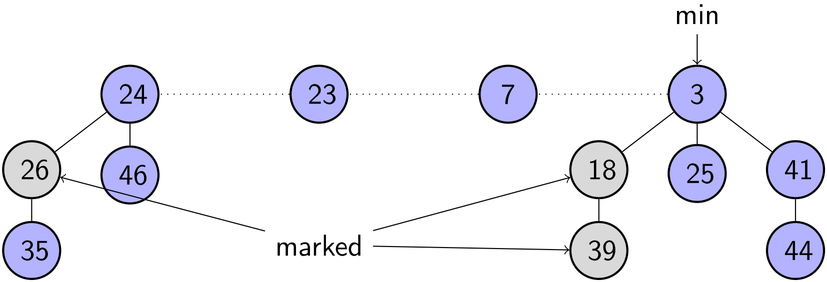
Structure of Fibonacci heap

- Set of heap-ordered trees
- Pointer to the minimum element



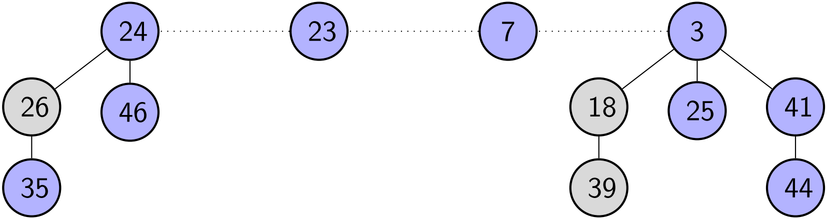
Structure of Fibonacci heap

- Set of heap-ordered trees
- Pointer to the minimum element
- Set of marked nodes



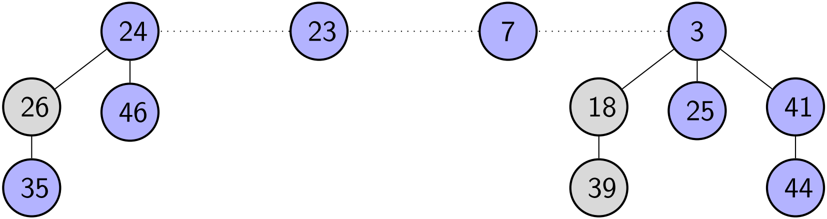
Notations

- n : Number of nodes in a Fibonacci heap H



Notations

- n : Number of nodes in a Fibonacci heap H
- degree(x) : Number of children of a node x



Notations

- n : Number of nodes in a Fibonacci heap H
- degree(x) : Number of children of a node x
- rank(n) : upper bound on the maximum degree of any node of a Fibonacci heap containing n nodes.
  $rank(n) = O(logn)$

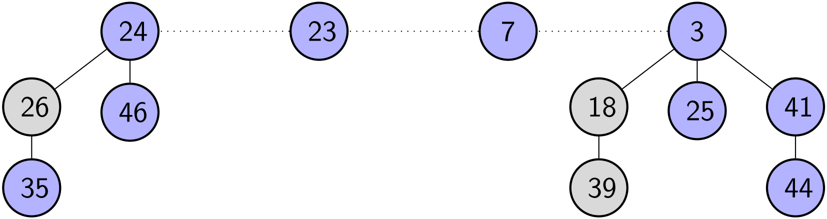


Table of Contents

1. Introduction
2. Motivation
3. Features
4. Operations
   - Insert
   - Union
   - Decrease-Key
   - Extract-Min
5. Advantages and Disadvantages
   - Advantages
   - Disadvantages
6. Application

Fibonacci Heap: Insertion

- Step 1: Make a new tree with the new key

Fibonacci Heap: Insertion

- Step 1: Make a new tree with the new key
- Step 2: Add the new tree to the heap (usually next to the min pointer)

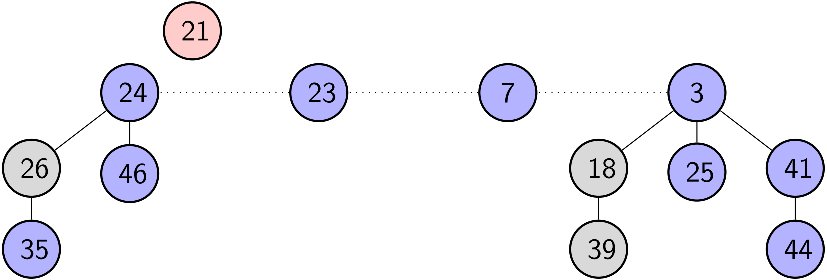
Fibonacci Heap: Insertion

- Step 1: Make a new tree with the new key
- Step 2: Add the new tree to the heap (usually next to the min pointer)
- Step 3: Update the min pointer (if required)

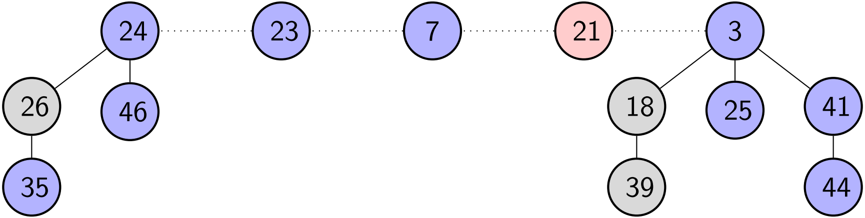
Fibonacci Heap: Insertion

- Let's see a simulation!

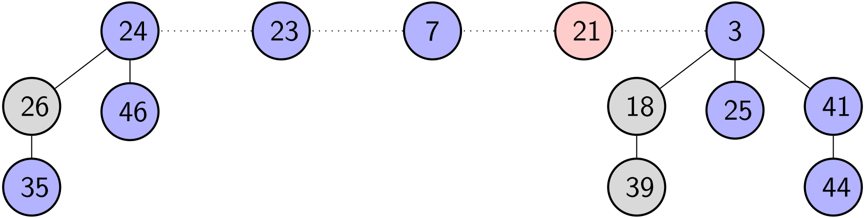
Insert 21



Insert 21



Insert 21



Since 21 > 3, no need to update the min pointer.

Fibonacci Heap: Insertion

Time complexity of insertion = $O(1)$

Fibonacci Heap: Union

Union operation takes two Fibonacci heaps H_1 and H_2 and merges them to one heap H.

- Step 1: Merge the two rootlists (usually rootlists are stored in a doubly-linked list)

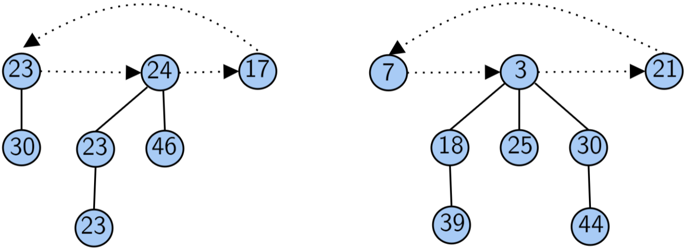
Fibonacci Heap: Union

Union operation takes two Fibonacci heaps H_1 and H_2 and merges them to one heap H.

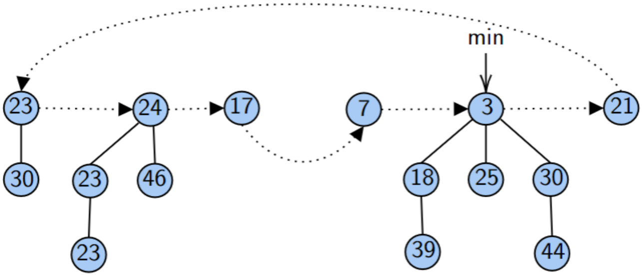
- Step 1: Merge the two rootlists (usually rootlists are stored in a doubly-linked list)
- Step 2: Make the min pointer with smaller value the min pointer of the merged heap.

Fibonacci Heap: Union

Let's see a simulation!



Fibonacci Heap: Union



Fibonacci Heap: Union

Time complexity of union $= O(1)$

Fibonacci Heap: Decrease Key

- Goal: $O(1)$ amortized running time

Fibonacci Heap: Decrease Key

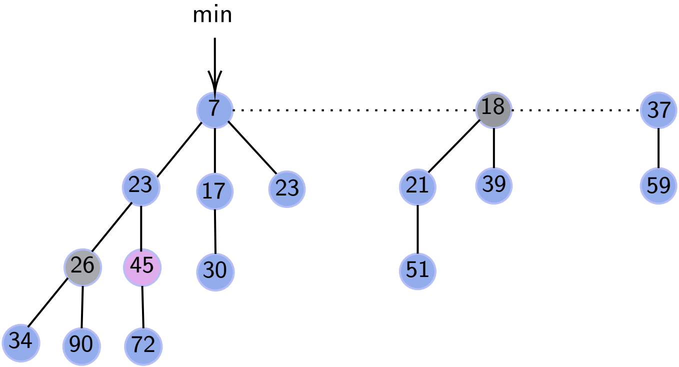
- Goal: O(1) amortized running time
- Obstacles:
  1. Decreasing a key might break the heap-property.
  2. Preserving the heap property needs a heapify operation of O(logn) time in worst case

Fibonacci Heap: Decrease Key

- Goal: O(1) amortized running time
- Obstacles:
  1. Decreasing a key might break the heap-property.
  2. Preserving the heap property needs a heapify operation of O(logn) time in worst case
- Solution:
  1. If the decreased node violates the heap order, cut the sub-tree rooted at the note and add it to the rootlist.
  2. No need to heapify. Problem Solved in O(1) amortized time!

Fibonacci Heap: Decrease Key

Let's see a simulation!

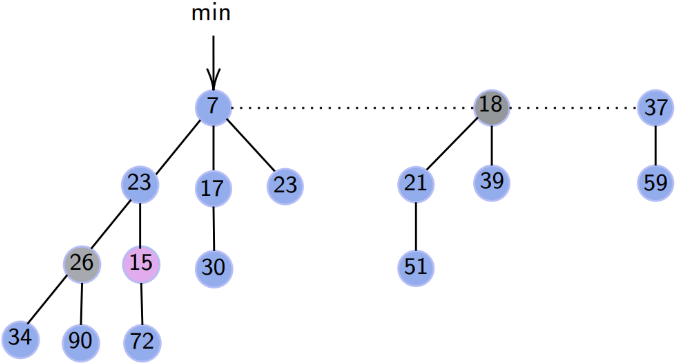


Decrease 46 to 45

No violation of heap-order. Decrease-key complete.

Fibonacci Heap: Decrease Key

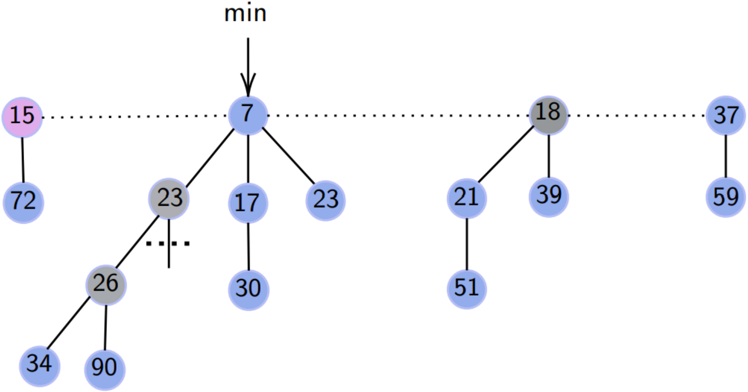
Let's see a simulation



Decrease 45 to 15.
Heap-order violated.

Fibonacci Heap: Decrease Key

Let's see a simulation



Cut the subtree rooted at 15 and add it to the rootlist.
Make the parent of 15 marked

Fibonacci Heap: Decrease-Key

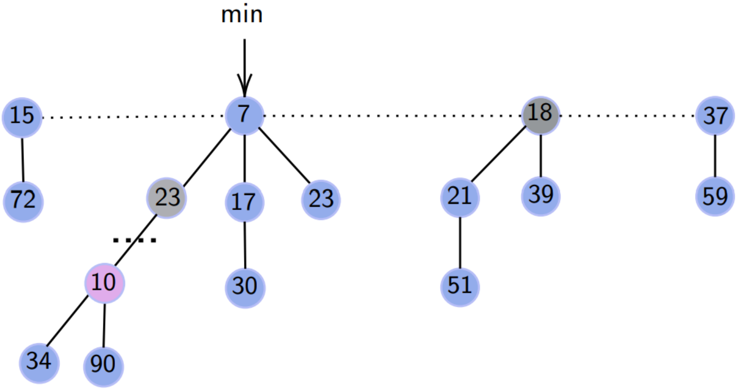
Why do we need to mark the parents?

To keep the maximum degree of a node bounded.

When two children of a node are cut, cut the node itself and do it recursively. This is called Cascading

Fibonacci Heap: Decrease Key

Let's see an example of cascading cut

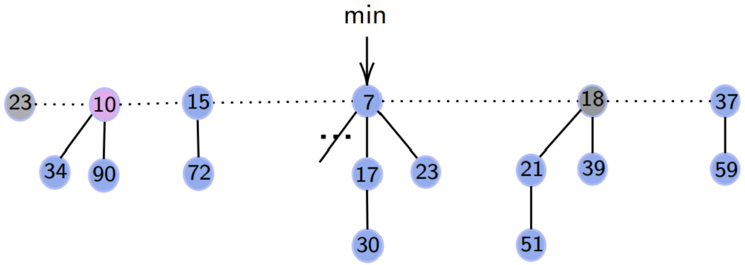


26 decreased to 10

But parent is already marked. So, 23 will be cut as well.

Fibonacci Heap: Decrease Key

Let's see an example of cascading cut



Fibonacci Heap: Decrease key

Amortized time complexity of decrease key = $O(1)$

Fibonacci Heap:Extract-Min

- Goal: O(logn) amortized running time

Fibonacci Heap:Extract-Min

- Goal: O(logn) amortized running time
- Process: Find the minimum node in rootlist

Fibonacci Heap:Extract-Min

- Goal: O(logn) amortized running time
- Process: Find the minimum node in rootlist
- Delete the minimum node

Fibonacci Heap:Extract-Min

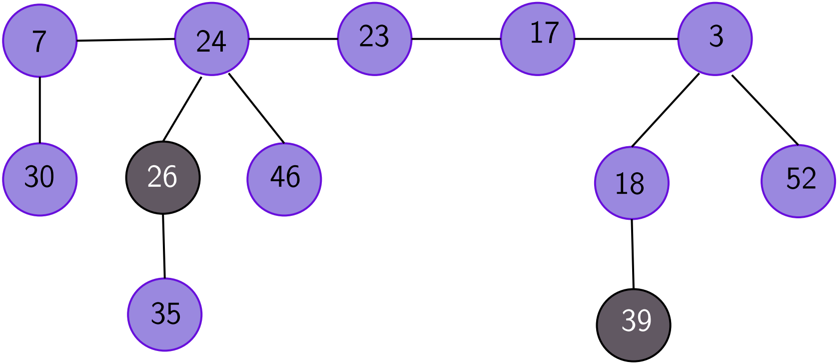
- Goal: O(logn) amortized running time
- Process: Find the minimum node in rootlist
- Delete the minimum node
- Add its children to the rootlist

Fibonacci Heap:Extract-Min

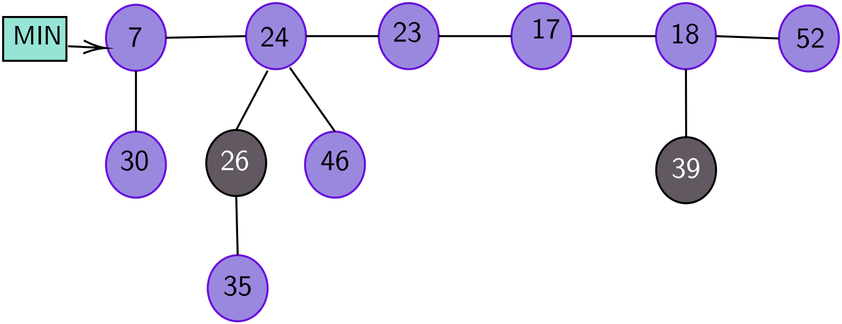
- Goal: O(logn) amortized running time
- Process: Find the minimum node in rootlist
- Delete the minimum node
- Add its children to the rootlist
- Consolidate trees so no two trees have the same degree.

Fibonacci Heap: Extract-min Key

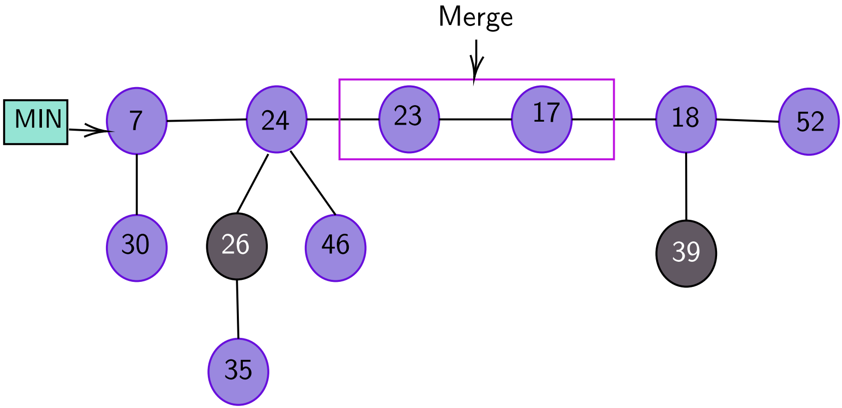
Let's see a simulation!



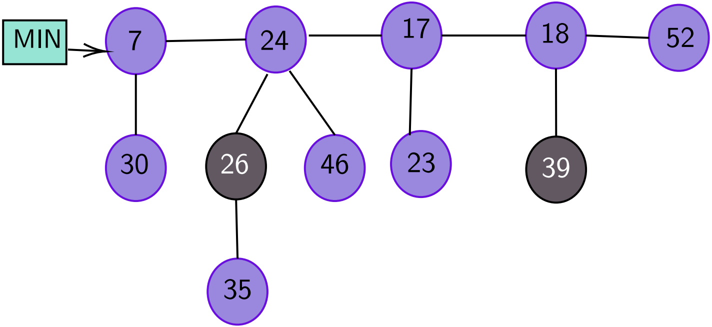
Fibonacci Heap: Extract-min Key



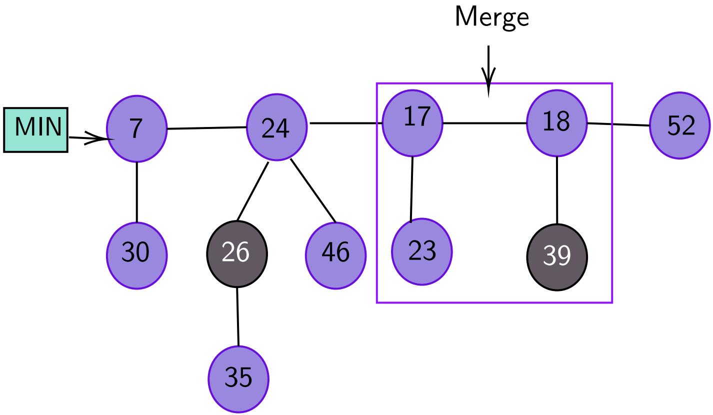
Fibonacci Heap: Extract-min Key



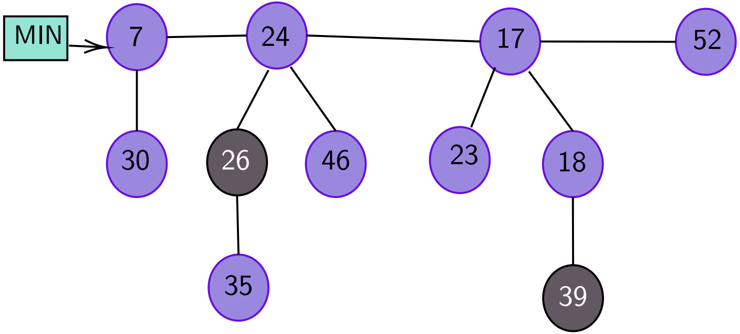
Fibonacci Heap: Extract-min Key



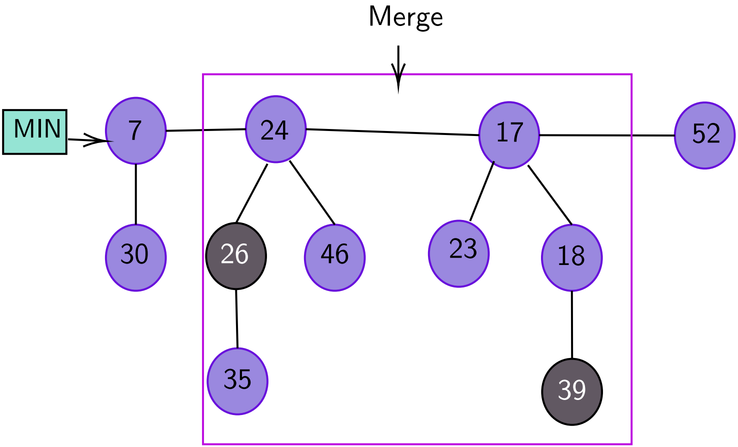
Fibonacci Heap: Extract-min Key



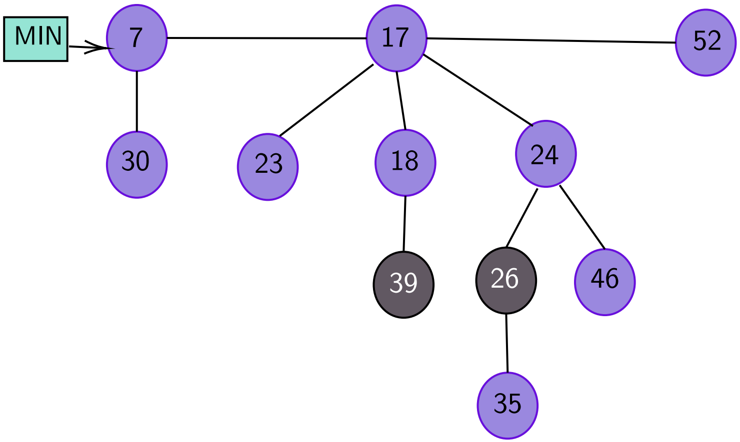
Fibonacci Heap: Extract-min Key



Fibonacci Heap: Extract-min Key



Fibonacci Heap: Extract-min Key



Fibonacci Heap: Extract-Min

Time complexity of Extract-min = O(logn) amortized

Worst case running time=O(n)

Table of Contents

1. Introduction
2. Motivation
3. Features
4. Operations
   - Insert
   - Union
   - Decrease-Key
   - Extract-Min
5. Advantages and Disadvantages
   - Advantages
   - Disadvantages
6. Application

Fibonacci heap:Advantages

- Amortized Time Complexity

Fibonacci heap:Advantages

- Amortized Time Complexity
- Lazy Merging

Fibonacci heap:Advantages

- Amortized Time Complexity
- Lazy Merging
- Efficient Decrease Key Operation

Fibonacci heap:Advantages

- Amortized Time Complexity
- Lazy Merging
- Efficient Decrease Key Operation
- Dynamic Operations:

Fibonacci heap:Disadvantages

- Higher Constant Factors

Fibonacci heap:Disadvantages

- Higher Constant Factors
- Complexity and Implementation Overhead

Fibonacci heap:Disadvantages

- Higher Constant Factors
- Complexity and Implementation Overhead
- Memory Usage

Table of Contents

1. Introduction
2. Motivation
3. Features
4. Operations
   - Insert
   - Union
   - Decrease-Key
   - Extract-Min
5. Advantages and Disadvantages
   - Advantages
   - Disadvantages
6. Application

Fibonacci heap:Application

- Optimizing Dijkstra Algorithm

Fibonacci heap:Application

- Optimizing Dijkstra Algorithm
- Optimizing Prim's Algorithm

Fibonacci heap:Application

- Optimizing Dijkstra Algorithm
- Optimizing Prim's Algorithm
- Network Flow Algorithms

Fibonacci heap:Application

- Optimizing Dijkstra Algorithm
- Optimizing Prim's Algorithm
- Network Flow Algorithms
- Job Scheduling Problem

Thank you