Fibonacci Heap

Konrad Rozpadek

Kalki Sarangan

David Litton

What is a Fibonacci Heap and its uses

- Lazy data structure
- Priority queue researched for dijkstra's algorithm
- Contains operations such as:
  - Insert,
  - Get Min,
  - Extract Min,
  - Decrease Key
- Achieve low time complexity through amortization

How is the Fibonacci Heap Organized

- Starts with a double linked list, called the root list, of trees satisfying min heap property
  - Implemented as either a min or max heap
- Each layer of each tree is also a doubly linked list
- There is always a pointer to the minimum element of the heap
- Tries to balance number of root nodes and children of nodes
- The number of children is referred to as the degree of a node

Operations

Insert

Insert adds the element as a single node tree to the root list.

Time Complexity O(1)

Space Complexity O(1)

Find Min/Max

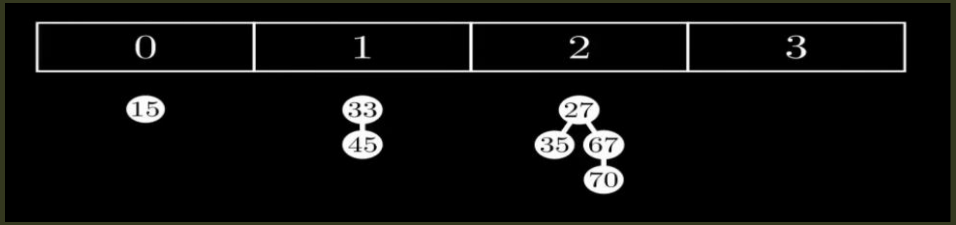
Reads the value that the Min/Max pointer points to and returns it

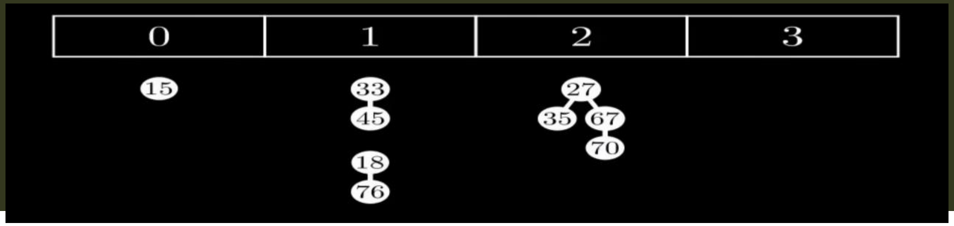
Time Complexity O(1)

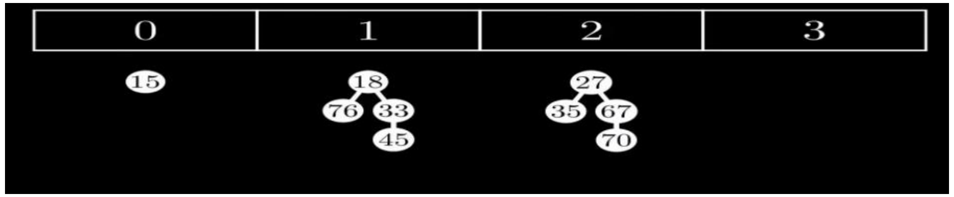
Space Complexity O(1)

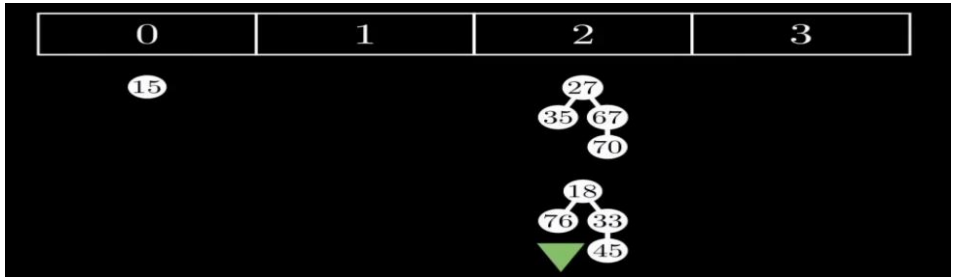
Extract Min

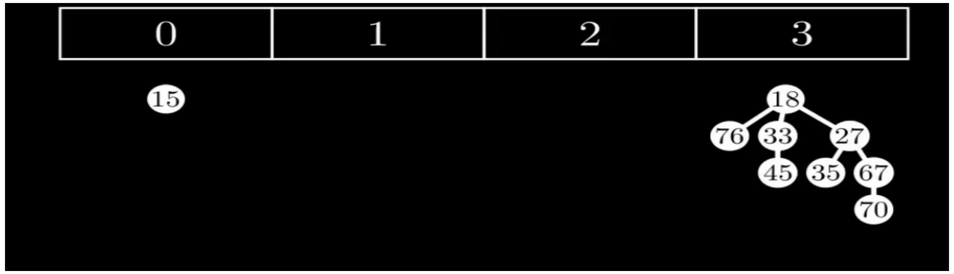
- Removes and returns the minimum element from the heap
- Restructures the heap
    - Loop over the root list, setting them in an array where the index matches the trees degree
    - If there is already one tree of that degree there, merge the trees keeping heap property
    - Update the degree and index.
    - Ultimately this makes the heap have at most d+1 trees (root nodes) where d is the highest degree and there is no more than 1 tree for each degree


0 1 2 3
15
33 45
27 35 67 70


0 1 2 3
15
33 45
27 35 67 70
18 76


0 1 2 3
15
18 76 33 45
27 35 67 70


0 1 2 3
15
27 35 67 70
18 76 33 45


0 1 2 3
15
18 76 33 27 45 35 67 70

Extract Min Time Complexity

Time Complexity

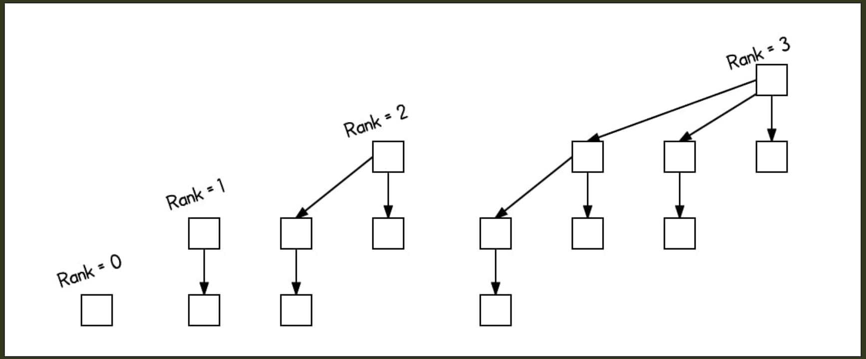
Worst Case : Getting the min is constant. Since you need loop over the trees you have t operations. Since every tree could be the same, at most you could have t merges. This gives t+t. The array must be turned back into the heap which takes d, the max degree, operations since each index is looped over. This results in T(t+t+d) = O(t+d)

Amortization

Since each insert results in one more tree, each insert can be thought of as constantly increasing n by some number. If you attribute the n operations across each insert, insert is still constant time and Extract Min is O(d) where d is the max degree.

Space Complexity O(1) as the number of internal structures stays the same

What is d(max degree)?



- When merging each tree is turned into a binomial tree
- What is a binomial tree?
- A bionomial tree of degree k has 2^k nodes
- Thus d = log(n)
- Time complexity of Extract Min is O(log n) where n is num of elements in heap

Decrease Key

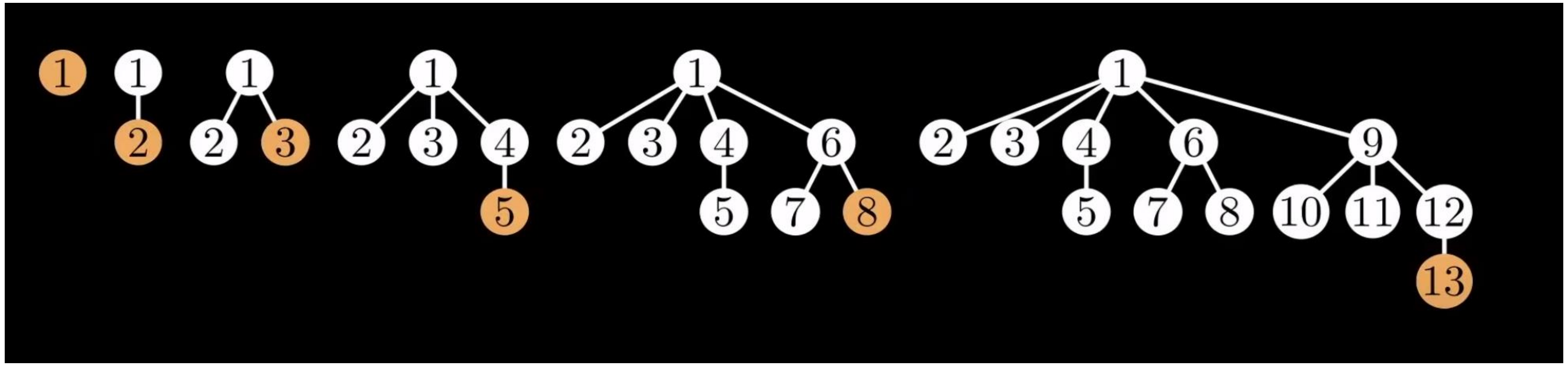
- Decreases the value of a node. Useful for an algorithm such as Dijkstras.
- Change value and maintain heap property
- Bubbling up? O(log n)
- Better approach
    - Remove the node and its children and append it to the root list
    - Time Complexity O(1)
    - Space Complexity O(1) as the number of internal structures stays the same
    - Results in number of trees increasing so it also amortizes the cost of extract min

Problem with the approach

- In order for max degree to be logarithmic n = C^d for some constant C > 1
- Decrease key breaks the binomial tree!
- Solution
  - Allow each node to have at most one child removed in order to be still approximately log n
  - If more than 1 is removed, remove the parent as well
    - Decreases degree of parent by one
- This approach results in the total number of cut nodes to be at most 2k where k is the number of cut nodes.
- This results in cut to be amortized O(1)

Fibonacci!

- When merging trees the latest child is at the end of the children list
- You always merge such that the degree of the node added equals the parents degree
- A node can lose at most one child before being cut
- Thus the ith child of a node has at least degree i-2
- Still exponential so d = log n with some base > 1



Advantages and Disadvantages

- Advantages
  - Constant time for all operations but extract min
  - Efficient Space Wise
- Disadvantages
  - High Overhead
  - Need a very large n to have performance advantage over binary heap
  - Much more complicated to implement to other priority queues such as binary heap or binomial heap

Jupyter Notebook

Thank you

Citations

https://www.geeksforgeeks.org/difference-between-binary-heap-binomial-heap-and-fibonacci-heap/

https://www.youtube.com/watch?v=6JxvKfSV9Ns <- graphics in slides were sources here

https://brilliant.org/wiki/fibonacci-heap/#:~:text=In%20Fibonacci%20heaps%2C%20merging%20is,be%20done%20in%20constant%20time.