CS4311
Design and Analysis of Algorithms

Lecture 19: Fibonacci Heap II
About this lecture

• Decrease-Key & Delete in Fibonacci Heap
  ➔ Based on cutting some node from its parent, and a simple rule which decides if further cuts are needed

• Bounding $\text{MaxDeg}(n)$
Rule of Further Cuts

- Let $x$ be a node with a parent node $y$, such that at some time, $x$ was a root and was then linked to $y$

The rule is as follows:

After the above linking event, as soon as $x$ has lost its second child, we cut $x$ from $y$, making it a new root.
Rule of Further Cuts

- To help us keep track of the status of each node, we use **marking** in a node !!!
  - **we mark** a non-root node $x$ if it has lost the first child
    - If a non-root marked node loses a child, it is **cut** from its parent
  - **we unmark** a node $x$ if
    - (i) it becomes a new root [after a cut], or
    - (ii) it receives a parent [after Extract-Min]
Decrease-Key

- **Decrease-Key**($H, x, k$):
  
  Report error if $k >$ key of $x$;
  Update the key of $x$ to $k$;

  /* fix if min-heap property is violated */

  if ($x \neq$ root and $x$’s key < its parent’s key)
  { Cut $x$ from its parent;
    Perform further cuts (recursively); }

  Update min[$H$] if needed
Decrease-Key (Example)

Before Decrease-Key

Marked node

min(H)

decrease key to 18
The node with key 32 has its key decreased to 18.
Decrease-Key (Example)

The node with key 21 has its key decreased to 3

Marked node

Decrease key to 14
Decrease-Key (Example)

Decrease Key to 14 (Step 1)

Marked node

$\text{min}(H)$

$H$
Decrease-Key (Example)

Decrease Key to 14 (Step 2)

Marked node

min(\(H\))
Decrease-Key (Example)

Decrease Key to 14 (Step 3)

Marked node
Decrease-Key

- We see that if Decrease-Key decides to cut a node $x$ from its parent, it may create a series of further cuts
  - we call this cascading cuts

![Cascading waterfall](image1)  ![Cascading fountain](image2)
Amortized Cost

• Let $H'$ denote the heap just before the Decrease-Key operation
• Let $c = \#$cascading cuts

$\Rightarrow$ actual cost = $O(c+1)$

potential before: $t(H') + 2m(H')$

potential after: at most $(t(H') + c + 1) + 2(m(H') - c + 1)$

$\Rightarrow$ amortized cost $\leq O(c+1) + 3 - c = O(1)$
Delete

• \( \text{Delete}(H, x) \):

  \( \text{Decrease-Key}(H, x, -\infty) ; \)

  \( \text{Extract-Min}(H) ; \)

\[ \Rightarrow \text{Amortized cost} = O(1) + O(\log n) = O(\log n) \]
Bounding $\text{MaxDeg}(n)$

- Recall that \#trees and height of a tree in a Fibonacci heap is unbounded
  - Can you obtain a component tree in Fibonacci heap whose height = $\Theta(n)$?

- In contrast, we shall show that the degree of a node is bounded by $O(\log n)$
  - We denote this bound by $\text{MaxDeg}(n)$
Bounding $\text{MaxDeg}(n)$

- For any node $x$, we let
  
  $\text{size}(x) = \# \text{nodes in the subtree rooted at } x, \text{ including itself}$
  
  $\text{deg}(x) = \# \text{children of } x$

- Our idea is to show that $\text{size}(x)$ is exponential in $\text{deg}(x)$
A Useful Lemma

Lemma: Let $x$ be a node in the Fibonacci heap, and suppose that $\deg(x) = k$

Let $y_1, y_2, \ldots, y_k$ be the children of $x$, ordered by the time they are linked to $x$

Then, we have:

$\deg(y_1) \geq 0$, and

$\deg(y_j) \geq j - 2$ for $j = 2, 3, \ldots, k$
Proof

• $\text{deg}(y_1) \geq 0$ is trivial

• By the time $y_j$ is linked to $x$, the nodes $y_1, y_2, \ldots, y_{j-1}$ were already linked to $x$.
  $\Rightarrow$ $x$ has at least $j-1$ children
  $\Rightarrow$ $\text{deg}(y_j)$ at that time
  $= \text{deg}(x)$ at that time $\geq j-1$

• Since then, $y_j$ loses at most 1 child (why??), so $\text{deg}(y_j) \geq j-2$
Fibonacci Number

• We are about to see why Fibonacci heap has “Fibonacci” in its name

• Define the $k^{th}$ Fibonacci Number, $F_k$, by:
  
  • $F_0 = 0$, $F_1 = 1$
  
  • For $k \geq 2$, $F_k = F_{k-2} + F_{k-1}$

• For example, the first few Fibonacci numbers are: 0, 1, 1, 2, 3, 5, 8, 13, 21 …
Two Lemmas on $F_k$

Lemma: For all integers $k \geq 0$,

$$F_{k+2} = 1 + F_0 + F_1 + F_2 + \ldots + F_k$$

Lemma: For all integers $k \geq 0$,

$$F_{k+2} \geq \varphi^k,$$

where $\varphi = (1+\sqrt{5})/2 = 1.61803\ldots$

How to prove? (By induction)
A Key Result

Combining previous lemmas, we can show:

Lemma: Let $x$ be a node in the Fibonacci heap, and suppose that $\deg(x) = k$.

Then, we have:

$$\text{size}(x) \geq F_{k+2} \geq \varphi^k$$

Proof: Let $s_k = \min$ possible size of a node whose degree is $k$.

$\Rightarrow \text{size}(x) \geq s_k$
Proof of Key Result

• We shall show by induction that:
  \[ s_k \geq F_{k+2} \]
  If it is true, our proof completes

• Base Case: \( s_0 = 1 = F_2 \) and \( s_1 = 2 = F_3 \)

• Inductive Case:
  Assume \( s_j \geq F_{j+2} \) for all \( j = 0, 1, \ldots, k-1 \)
Proof of Key Result

Consider any deg-$k$ node whose size = $s_k$

By our lemma, we see that its children, say $y_1, y_2, \ldots, y_k$ have degrees satisfying:

$$\text{deg}(y_1) \geq 0, \text{ and } \text{deg}(y_j) \geq j - 2 \text{ for } j \geq 2$$

$$s_k = 1 + \text{size}(y_1) + \text{size}(y_2) + \ldots + \text{size}(y_k)$$

$$\geq 1 + s_{\text{deg}(y_1)} + s_{\text{deg}(y_2)} + \ldots + s_{\text{deg}(y_k)}$$

$$\geq 1 + s_0 + s_0 + \ldots + s_{k-2} \geq 1 + F_2 + F_2 + \ldots + F_k$$

$$= 1 + (F_0 + F_1) + F_2 + \ldots + F_k = F_{k+2}$$
Bounding $\text{MaxDeg}(n)$

Immediately, we have:

**Theorem:** $\text{MaxDeg}(n) \leq \log_\varphi n = O(\log n)$

**Proof:** For any node $x$ with $\text{deg } k$, we have:

$$n \geq \text{size}(x) \geq \varphi^k$$

$\implies$ degree of any node $\leq \log_\varphi n$

$\implies$ the theorem thus follows

**Remark:** Since $\text{MaxDeg}(n)$ must be an integer, we can show a tighter bound: $\text{MaxDeg}(n) \leq \lfloor \log_\varphi n \rfloor$