Fibonacci Heaps

Fibonacci heap history. Fredman and Tarjan (1986)
- Ingenious data structure and analysis.
- Original motivation: $O(m + n \log n)$ shortest path algorithm.
  - also led to faster algorithms for MST, weighted bipartite matching
- Still ahead of its time.

Fibonacci heap intuition.
- Similar to binomial heaps, but less structured.
- Decrease-key and union run in $O(1)$ time.
- "Lazy" unions.

Fibonacci Heaps: Structure

Fibonacci heap.
- Set of min-heap ordered trees.
**Fibonacci Heaps: Implementation**

Implementation.
- Represent trees using left-child, right sibling pointers and circular, doubly linked list.
  - can quickly splice off subtrees
- Roots of trees connected with circular doubly linked list.
  - fast union
- Pointer to root of tree with min element.
  - fast find-min

**Fibonacci Heaps: Potential Function**

Key quantities.
- Degree[x] = degree of node x.
- Mark[x] = mark of node x (black or gray).
- t(H) = # trees.
- m(H) = # marked nodes.
- \( \Phi(H) = t(H) + 2m(H) \) = potential function.

\[ t(H) = 5, \quad m(H) = 3 \]

\[ \Phi(H) = 11 \]

**Fibonacci Heaps: Insert**

Insert.
- Create a new singleton tree.
- Add to left of min pointer.
- Update min pointer.

**Insert 21**
Fibonacci Heaps: Insert

- **Insert.**
  - Create a new singleton tree.
  - Add to left of min pointer.
  - Update min pointer.

**Running time.** $O(1)$ amortized
- Actual cost = $O(1)$.
- Change in potential = +1.
- Amortized cost = $O(1)$.

Fibonacci Heaps: Union

- **Union.**
  - Concatenate two Fibonacci heaps.
  - Root lists are circular, doubly linked lists.

**Running time.** $O(1)$ amortized
- Actual cost = $O(1)$.
- Change in potential = 0.
- Amortized cost = $O(1)$.

Fibonacci Heaps: Delete Min

- **Delete min.**
  - Delete min and concatenate its children into root list.
  - Consolidate trees so that no two roots have same degree.
Delete min.
- Delete min and concatenate its children into root list.
- Consolidate trees so that no two roots have same degree.
Fibonacci Heaps: Delete Min

Delete min.
- Delete min and concatenate its children into root list.
- Consolidate trees so that no two roots have same degree.

Merge 17 and 23 trees.

Merge 7 and 17 trees.

Merge 7 and 24 trees.
Fibonacci Heaps: Delete Min

Delete min.
- Delete min and concatenate its children into root list.
- Consolidate trees so that no two roots have same degree.

- Merge 41 and 18 trees.
Fibonacci Heaps: Delete Min

Delete min.
- Delete min and concatenate its children into root list.
- Consolidate trees so that no two roots have same degree.

Fibonacci Heaps: Delete Min Analysis

Notation.
- \( D(n) \) = max degree of any node in Fibonacci heap with \( n \) nodes.
- \( t(H) \) = # trees in heap \( H \).
- \( \phi(H) = t(H) + 2m(H) \).

Actual cost. \( O(D(n) + t(H)) \)
- \( O(D(n)) \) work adding min’s children into root list and updating min.
  - at most \( D(n) \) children of min node
- \( O(D(n) + t(H)) \) work consolidating trees.
  - work is proportional to size of root list since number of roots decreases by one after each merging
  - \( \leq D(n) + t(H) - 1 \) root nodes at beginning of consolidation

Amortized cost. \( O(D(n)) \)
- \( t(H') \leq D(n) + 1 \) since no two trees have same degree.
- \( \Delta \phi(H) \leq D(n) + 1 - t(H) \).

Is amortized cost of \( O(D(n)) \) good?
- Yes, if only Insert, Delete-min, and Union operations supported.
  - in this case, Fibonacci heap contains only binomial trees since we only merge trees of equal root degree
  - this implies \( D(n) \leq \lceil \log_2 N \rceil \)
- Yes, if we support Decrease-key in clever way.
  - we’ll show that \( D(n) \leq \lceil \log_3 N \rceil \), where \( \phi \) is golden ratio
  - \( \phi^2 = 1 + \phi \)
  - \( \phi = (1 + \sqrt{5}) / 2 = 1.618... \)
  - limiting ratio between successive Fibonacci numbers!
Fibonacci Heaps: Decrease Key

Decrease key of element $x$ to $k$.

- **Case 0**: min-heap property not violated.
  - decrease key of $x$ to $k$
  - change heap min pointer if necessary

- **Case 1**: parent of $x$ is unmarked.
  - decrease key of $x$ to $k$
  - cut off link between $x$ and its parent
  - mark parent
  - add tree rooted at $x$ to root list, updating heap min pointer
Fibonacci Heaps: Decrease Key

Decrease key of element \( x \) to \( k \).

- Case 2: parent of \( x \) is marked.
  - decrease key of \( x \) to \( k \)
  - cut off link between \( x \) and its parent \( p[x] \), and add \( x \) to root list
  - cut off link between \( p[x] \) and \( p[p[x]] \), add \( p[x] \) to root list
    - If \( p[p[x]] \) unmarked, then mark it.
    - If \( p[p[x]] \) marked, cut off \( p[p[x]] \), unmark, and repeat.

Decrease 35 to 5.
Fibonacci Heaps: Decrease Key Analysis

Notation.
- \( t(H) \) = # trees in heap \( H \).
- \( m(H) \) = # marked nodes in heap \( H \).
- \( \Phi(H) = t(H) + 2m(H) \).

Actual cost. \( O(c) \)
- \( O(1) \) time for decrease key.
- \( O(1) \) time for each of \( c \) cascading cuts, plus reinserting in root list.

Amortized cost. \( O(1) \)
- \( t(H') = t(H) + c \)
- \( m(H') \leq m(H) - c + 2 \)
  - each cascading cut unmarks a node
  - last cascading cut could potentially mark a node
- \( \Delta \Phi \leq c + 2(-c + 2) = 4 - c \).

Fibonacci Heaps: Delete

Delete node \( x \).
- Decrease key of \( x \) to \( -\infty \).
- Delete min element in heap.

Amortized cost. \( O(D(n)) \)
- \( O(1) \) for decrease-key.
- \( O(D(n)) \) for delete-min.
- \( D(n) = \text{max degree of any node in Fibonacci heap} \).

Fibonacci Heaps: Bounding Max Degree

Definition. \( D(N) = \text{max degree in Fibonacci heap with } N \text{ nodes} \).

Key lemma. \( D(N) \leq \log_\phi N \), where \( \phi = (1 + \sqrt{5}) / 2 \).

Corollary. Delete and Delete-min take \( O(\log N) \) amortized time.

Lemma. Let \( x \) be a node with degree \( k \), and let \( y_1, \ldots, y_k \) denote the children of \( x \) in the order in which they were linked to \( x \). Then:

\[
\text{degree}(y_i) \geq \begin{cases} 
0 & \text{if } i = 1 \\
 i - 2 & \text{if } i \geq 1 
\end{cases}
\]

Proof.
- When \( y_i \) is linked to \( x \), \( y_1, \ldots, y_{i-1} \) already linked to \( x \),
  \( \Rightarrow \text{degree}(x) = i - 1 \)
  \( \Rightarrow \text{degree}(y_i) = i - 1 \) since we only link nodes of equal degree
- Since then, \( y_i \) has lost at most one child
  - otherwise it would have been cut from \( x \)
- Thus, \( \text{degree}(y_i) = i - 1 \) or \( i - 2 \)

Fibonacci Heaps: Bounding Max Degree

Key lemma. In a Fibonacci heap with \( N \) nodes, the maximum degree of any node is at most \( \log_\phi N \), where \( \phi = (1 + \sqrt{5}) / 2 \).

Proof of key lemma.
- For any node \( x \), we show that \( \text{size}(x) \geq \phi^{\text{degree}(x)} \).
  - \( \text{size}(x) = \text{# node in subtree rooted at } x \)
  - taking base \( \phi \) logs, \( \text{degree}(x) \leq \log_\phi \text{size}(x) \leq \log_\phi N \).
- Let \( s_k \) be min size of tree rooted at any degree \( k \) node.
  - trivial to see that \( s_0 = 1, s_1 = 2 \)
  - \( s_k \) monotonically increases with \( k \)
- Let \( x^* \) be a degree \( k \) node of size \( s_k \), and let \( y_1, \ldots, y_k \) be children in order that they were linked to \( x^* \).

Assume \( k \geq 2 \)

\[
\begin{align*}
s_k &= \text{size} \left( x^* \right) \\
&= 2 + \sum_{i=2}^{k} \text{size}(y_i) \\
&\geq 2 + \sum_{i=2}^{k} s_{\text{deg}(y_i)} \\
&\geq 2 + \sum_{i=2}^{k} s_{i-2} \\
&= 2 + \sum_{i=0}^{k-2} s_i
\end{align*}
\]
Fibonacci Facts

**Definition.** The Fibonacci sequence is:

- $F_k = \begin{cases} 1 & \text{if } k = 0 \\ 2 & \text{if } k = 1 \\ F_{k-1} + F_{k-2} & \text{if } k \geq 2 \end{cases}$
- Slightly nonstandard definition.

**Fact F1.** $F_k \geq \phi^k$, where $\phi = (1 + \sqrt{5}) / 2 = 1.618...$

**Fact F2.** For $k \geq 2$, $F_k = 2 + \sum_{i=0}^{k-2} F_i$

**Consequence.** $s_k \geq F_k \geq \phi^k$.
- This implies that $\text{size}(x) \geq \phi^{\text{degree}(x)}$ for all nodes $x$.

Golden Ratio

**Definition.** The Fibonacci sequence is: 1, 2, 3, 5, 8, 13, 21, ...  

**Definition.** The golden ratio $\phi = (1 + \sqrt{5}) / 2 = 1.618...$

- Divide a rectangle into a square and smaller rectangle such that the smaller rectangle has the same ratio as original one.

Parthenon, Athens Greece

Fibonacci Numbers and Nature

Pinecone

Cauliflower
Fibonacci Proofs

Fact F1.  $F_k \geq \phi^k$.
Proof. (by induction on $k$)
- Base cases:
  - $F_0 = 1$, $F_1 = 2 \geq \phi$.
- Inductive hypotheses:
  - $F_k \geq \phi^k$ and $F_{k+1} \geq \phi^{k+1}$

\[
F_{k+2} = F_k + F_{k+1} \\
\geq \phi^k + \phi^{k+1} \\
= \phi^k (1 + \phi) \\
= \phi^k (\phi^2) \\
= \phi^{k+2}
\]

Fact F2. For $k \geq 2$, $F_k = 2 + \sum_{i=0}^{k-2} F_i$
Proof. (by induction on $k$)
- Base cases:
  - $F_2 = 3$, $F_3 = 5$
- Inductive hypotheses:
  - $F_k = 2 + \sum_{i=0}^{k-2} F_i$

\[
F_{k+2} = F_k + F_{k+1} \\
= 2 + \sum_{i=0}^{k-2} F_i + F_{k+1} \\
= 2 + \sum_{i=0}^{k} F_i
\]

On Complicated Algorithms

"Once you succeed in writing the programs for [these] complicated algorithms, they usually run extremely fast. The computer doesn’t need to understand the algorithm, its task is only to run the programs."

R. E. Tarjan