<table>
<thead>
<tr>
<th>Operation</th>
<th>Linked List</th>
<th>Binary Heap</th>
<th>Binomial Heap</th>
<th>Fibonacci Heap</th>
<th>Relaxed Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>make-heap</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>is-empty</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>insert</td>
<td>1</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>delete-min</td>
<td>$n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
</tr>
<tr>
<td>decrease-key</td>
<td>$n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>delete</td>
<td>1</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>merge</td>
<td>1</td>
<td>n</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>meld</td>
<td>$m$</td>
<td>1</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$n = \text{number of elements in priority queue}$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Linked List</th>
<th>Binary Heap</th>
<th>Binomial Heap</th>
<th>Fibonacci Heap</th>
<th>Relaxed Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>make-heap</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>is-empty</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>insert</td>
<td>1</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>delete-min</td>
<td>$n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
</tr>
<tr>
<td>decrease-key</td>
<td>$n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>delete</td>
<td>1</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>merge</td>
<td>1</td>
<td>n</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>meld</td>
<td>$m$</td>
<td>1</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$n = \text{number of elements in priority queue}$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Linked List</th>
<th>Binary Heap</th>
<th>Binomial Heap</th>
<th>Fibonacci Heap</th>
<th>Relaxed Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>make-heap</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>is-empty</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>insert</td>
<td>1</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>delete-min</td>
<td>$n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
</tr>
<tr>
<td>decrease-key</td>
<td>$n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>delete</td>
<td>1</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>merge</td>
<td>1</td>
<td>n</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>meld</td>
<td>$m$</td>
<td>1</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$n = \text{number of elements in priority queue}$

**Fibonacci heap.**
- Set of heap-ordered trees.
- Maintain pointer to minimum element.
- Set of marked nodes.

**Cascading cuts & Successive linking**
- delete-min
- use to keep heaps flat (stay tuned)
- marked

**Fibonacci heap.**
- Set of heap-ordered trees.
- Maintain pointer to minimum element.
- Set of marked nodes.
Cascading cuts & Successive linking

- **delete-min**
  - Execute a delete-min operation.
  - Find the minimum element in the heap.
  - Remove the minimum element from the heap.
  - Perform successive linking if necessary.

- **Successive linking**
  - Perform successive linking on each level of the tree.

### Fibonacci Heaps: Decrease Key

- **Case 1. [heap order not violated]**
  - Decrease key of node x.
  - Change heap min pointer (if necessary).

- **Case 2a. [heap order violated]**
  - Decrease key of node x.
  - Cut tree rooted at x, meld into root list, and unmark.
  - If parent p of x is unmarked (hasn't yet lost a child), mark it; otherwise, cut p, meld into root list, and unmark (and do so recursively for all ancestors that lose a second child).

### Amortized Analysis

- **Amortized O(rank(H))**

---

### Fibonacci Heaps: Decrease Key

- **Case 1. [heap order not violated]**
  - Decrease key of node x.
  - Change heap min pointer (if necessary).
Case 2a. [heap order violated]
- Decrease key of $x$.
- Cut tree rooted at $x$, meld into root list, and unmark.
- If parent $p$ of $x$ is unmarked (hasn't yet lost a child), mark it;
  Otherwise, cut $p$, meld into root list, and unmark
  (and do so recursively for all ancestors that lose a second child).

Case 2b. [heap order violated]
- Decrease key of $x$.
- Cut tree rooted at $x$, meld into root list, and unmark.
- If parent $p$ of $x$ is unmarked (hasn't yet lost a child), mark it;
  Otherwise, cut $p$, meld into root list, and unmark
  (and do so recursively for all ancestors that lose a second child).
Fibonacci Heaps: Decrease Key

- Case 2b. [heap order violated]
  - Decrease key of \( x \).
  - Cut tree rooted at \( x \), meld into root list, and unmark.
  - If parent \( p \) of \( x \) is unmarked (hasn’t yet lost a child), mark it;
    Otherwise, cut \( p \), meld into root list, and unmark
    (and do so recursively for all ancestors that lose a second child).

\[ \text{decrease key of } x \text{ from } 35 \text{ to } 5 \]
שאלה משבוע שעבר
כיצד נبحر בעץ מדרגה \(k\) שאותו "מחט גרג"\?
- נبحر בעץ מדרגה \(k\) \(k\-rank star\) \(\rightarrow\) \(k\-rank star\)
- \(k=1\)
- \(k=2\)
- \(k=8\)

שאלה משבוע שעבר
כיצד מגדיר עץ מדרגה \(k\) שהוא "ממש גרוע"?
- נגדיר עץ מדרגה \(k\) "\(k\-rank star\"
- \(K=\sqrt{n}\)
- \(K=2\)
- \(K=8\)

שאלה משבוע עבר
כיצד מגדיר \(k\-rank star\) ולהבין \(k\-rank star\) \(\Rightarrow\) \(\sqrt{n}\) \(\Rightarrow\) \(\leq 1\)
- \(\leq \sqrt{n}\) \(\Rightarrow\) \(\leq \sqrt{n}\) \(\Rightarrow\) \(\leq 1\)
- \(\leq \sqrt{n}\) \(\Rightarrow\) \(\leq \sqrt{n}\) \(\Rightarrow\) \(\leq 1\)

תרגיל 2 – ערמות פייבונאצי
איך נבחר ערמת פייבונאצי, בהמשפט \(\Rightarrow\)\
- \(\leq 1\) \(\Rightarrow\) \(\leq \sqrt{n}\) \(\Rightarrow\) \(\leq 1\)
- \(\leq 1\) \(\Rightarrow\) \(\leq \sqrt{n}\) \(\Rightarrow\) \(\leq 1\)
- \(\leq 1\) \(\Rightarrow\) \(\leq \sqrt{n}\) \(\Rightarrow\) \(\leq 1\)
תרגיל 2 – ערמות פיבונאצי

אם ניתן לבנות ערמה פיבונאצי'י והאנו קבע, התא לא בעומק h? זה מתרחש אם h = k+1 ו
key[з] ≤ key[y] ≤ key[x] < min[H]
¬ rode, z’, x’, y’, z, x, y מה להתרצה
משתמש בתוכנית הפקה

תרגיל 2 – ערמות פיבונאצי

אם ניתן לבנות ערמה פיבונאצי'י והאנו קבע, התא לא בעומק h? זה מתרחש אם h = k+1 ו
key[з] ≤ key[y] ≤ key[x] < min[H]
¬ rode, z’, x’, y’, z, x, y מה להתרצה
משתמש בתוכנית הפקה

תרגיל 3

***Answer***
- rank(y) ≥ i-3
- Since y had the same rank as x when it became a child of x
- x must have had at least i-1 children at that time, so y had at least i-1 rank.
- It could have lost at most two children since then, therefore rank at least i-3
הסוף