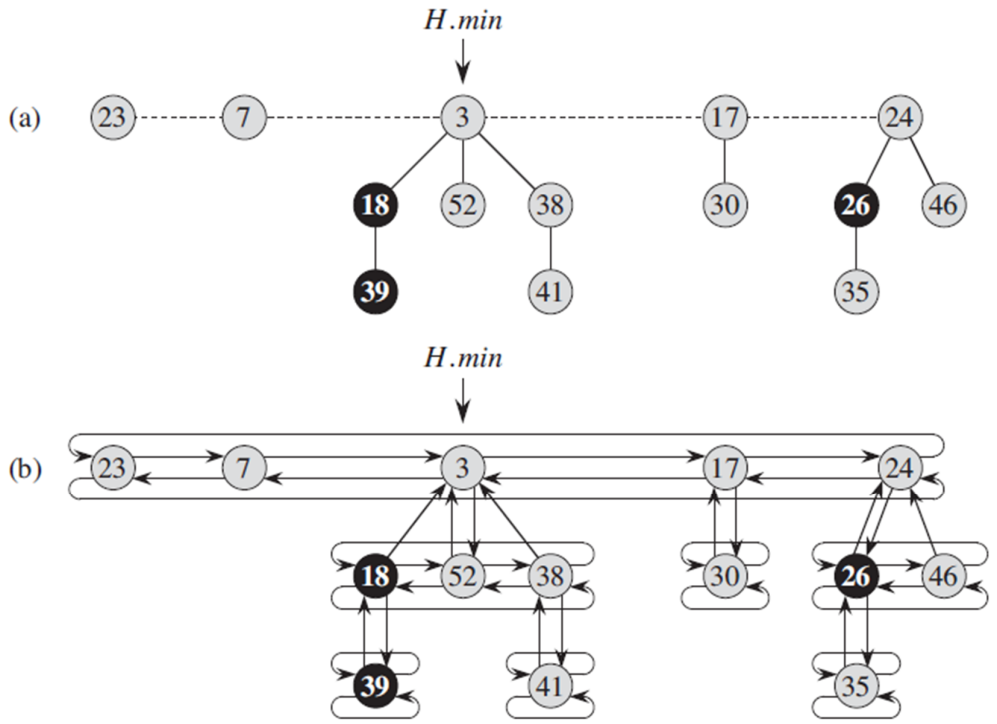
- Mergeable Min-heap

Procedure	Binary heap (worst-case)	Binomial heap (worst-case)	Fibonacci heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\lg n)$	$O(\lg n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$O(\lg n)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$
UNION	$\Theta(n)$	$O(\lg n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
DELETE	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$

Fibonacci Heap

Un Fibonacci heap H es un conjunto árboles que cumple propiedad de min-heap, la clave de cualquier nodo del árbol es a lo sumo el valor de la clave del nodo padre.



Fibonacci Heap

H.min apunta al nodo raíz cuya clave es mínima entre todas las raíces

H.n cantidad de nodos en H distribuido en los árboles de H

Cada nodo tiene los siguientes campos

- key clave
- degree #hijos
- mark flag indicador si haya perdido un hijo después de ser un nodo no raíz (la última vez y sigue vigente)
- p puntero al padre
- child puntero a un hijo
- left puntero al hermano izquierdo /raíz del árbol izquierdo
- right puntero al hermano derecho /raíz del árbol derecho

$$\Phi(H) = t(H) + 2\,m(H)$$

Donde t(H) es la cantidad de árboles de H y m es la cantidad de nodos donde mark es true.
Llamamos D(n) una cota superior de grado (degree) máximo de cualquier nodo en un Fibonacci heap de n nodos. Más adelante vamos a ver que D(n) = O(log n)

Fibonacci Heap

Make-heap()

H.min=nil; H.n=0

$$\phi(H) = t(H) + 2m(H) = 0 + 2\times 0 = 0$$

Insert(H,x)

```
1   x.degree = 0
2   x.p = NIL
3   x.child = NIL
4   x.mark = FALSE
5   if H.min == NIL
6        create a root list for H containing just x
7        H.min = x
8   else insert x into H's root list
9        if x.key < H.min.key
10            H.min = x
11  H.n = H.n + 1
```

$\hat{C} = O(1) + (t(H) + 1) + 2m(H) - (t(H) + 2m(H)) = O(1) + 1 = O(1)$

Fibonacci Heap

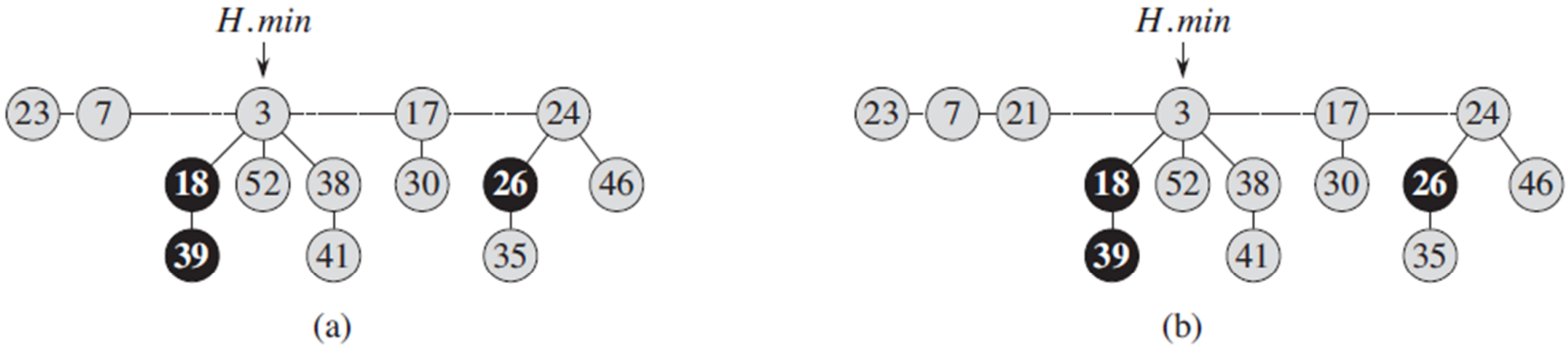


Figure 19.3 Inserting a node into a Fibonacci heap. **(a)** A Fibonacci heap H. **(b)** Fibonacci heap H after inserting the node with key 21. The node becomes its own min-heap-ordered tree and is then added to the root list, becoming the left sibling of the root.

Minimum(H)

Return H.min $\hat{C} = C = O(1)$

Fibonacci Heap

Union(H_1,H_2)

2 $H.min = H_1.min$
3 concatenate the root list of H_2 with the root list of H
4 **if** ($H_1.min$ == NIL) or ($H_2.min \neq$ NIL and $H_2.min.key < H_1.min.key$)
5 $\quad H.min = H_2.min$
6 $H.n = H_1.n + H_2.n$
7 **return** H

$$\begin{aligned}\Phi(H) - (\Phi(H_1) + \Phi(H_2)) &= (t(H) + 2m(H)) - ((t(H_1) + 2m(H_1)) + (t(H_2) + 2m(H_2))) \\ &= 0\,,\end{aligned}$$

because $t(H) = t(H_1) + t(H_2)$ and $m(H) = m(H_1) + m(H_2)$. The amortized cost of FIB-HEAP-UNION is therefore equal to its $O(1)$ actual cost.

Fibonacci Heap

Extract-min(H)

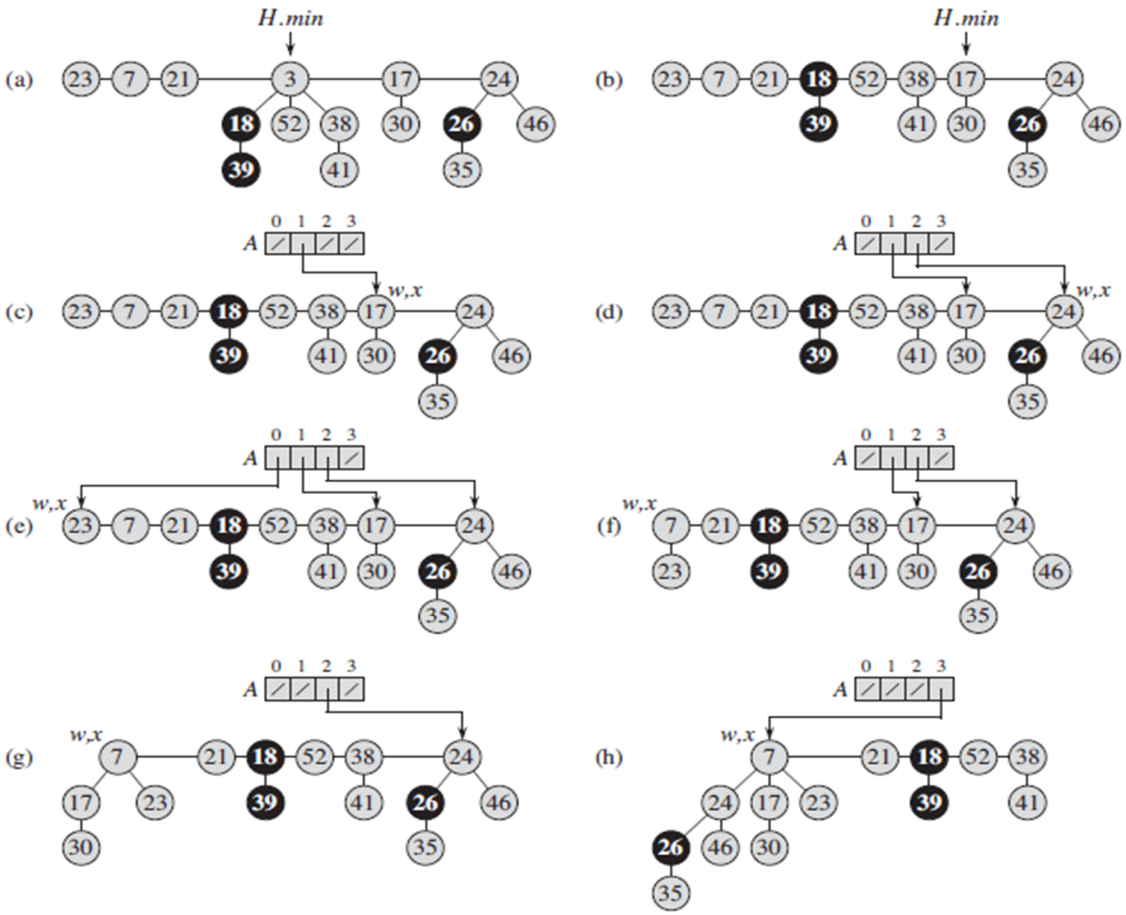
```
1   z = H.min
2   if z ≠ NIL
3       for each child x of z
4           add x to the root list of H
5           x.p = NIL
6       remove z from the root list of H
7       if z == z.right
8           H.min = NIL
9       else H.min = z.right
10          CONSOLIDATE(H)
11      H.n = H.n − 1
12  return z
```

```
FIB-HEAP-LINK(H, y, x)
1  remove y from the root list of H
2  make y a child of x, incrementing x.degree
3  y.mark = FALSE
```

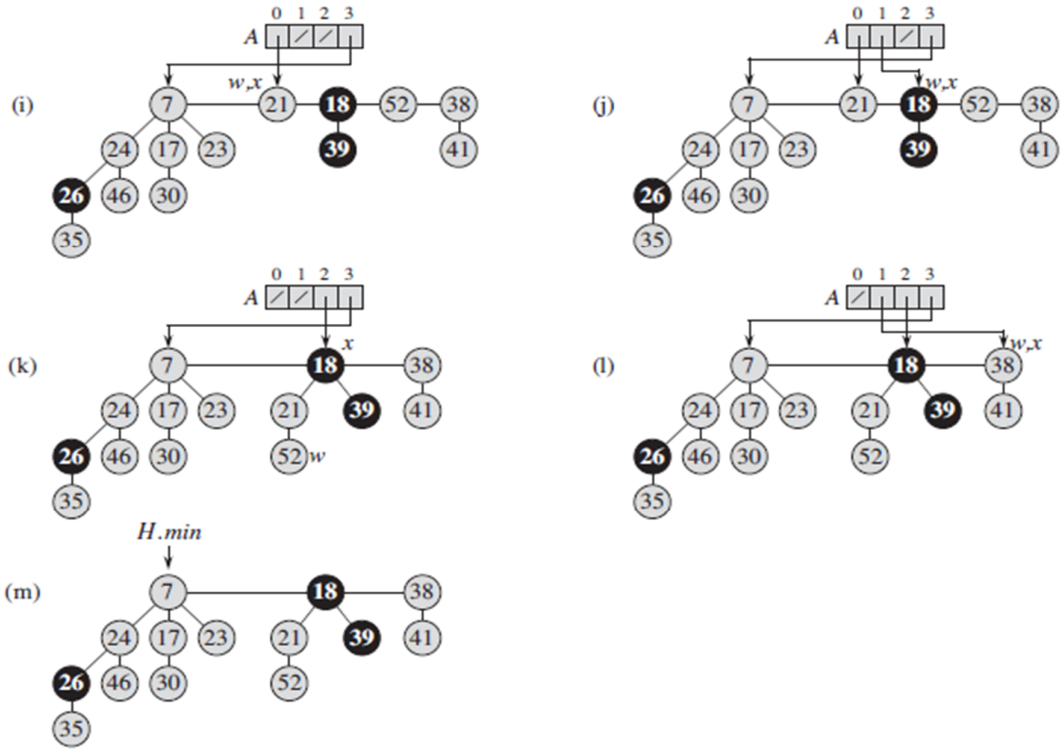
Fibonacci Heap

```
CONSOLIDATE(H)
 1  let A[0..D(H.n)] be a new array
 2  for i = 0 to D(H.n)
 3      A[i] = NIL
 4  for each node w in the root list of H
 5      x = w
 6      d = x.degree
 7      while A[d] ≠ NIL
 8          y = A[d]          // another node with the same degree as x
 9          if x.key > y.key
10              exchange x with y
11          FIB-HEAP-LINK(H, y, x)
12          A[d] = NIL
13          d = d + 1
14      A[d] = x
15  H.min = NIL
16  for i = 0 to D(H.n)
17      if A[i] ≠ NIL
18          if H.min == NIL
19              create a root list for H containing just A[i]
20              H.min = A[i]
21          else insert A[i] into H's root list
22              if A[i].key < H.min.key
23                  H.min = A[i]
```

Fibonacci Heap



Fibonacci Heap



$$
\begin{aligned}
& O(D(n) + t(H)) + ((D(n) + 1) + 2m(H)) - (t(H) + 2m(H)) \\
&\quad = \quad O(D(n)) + O(t(H)) - t(H) \\
&\quad = \quad O(D(n)) \,,
\end{aligned}
$$

Fibonacci Heap

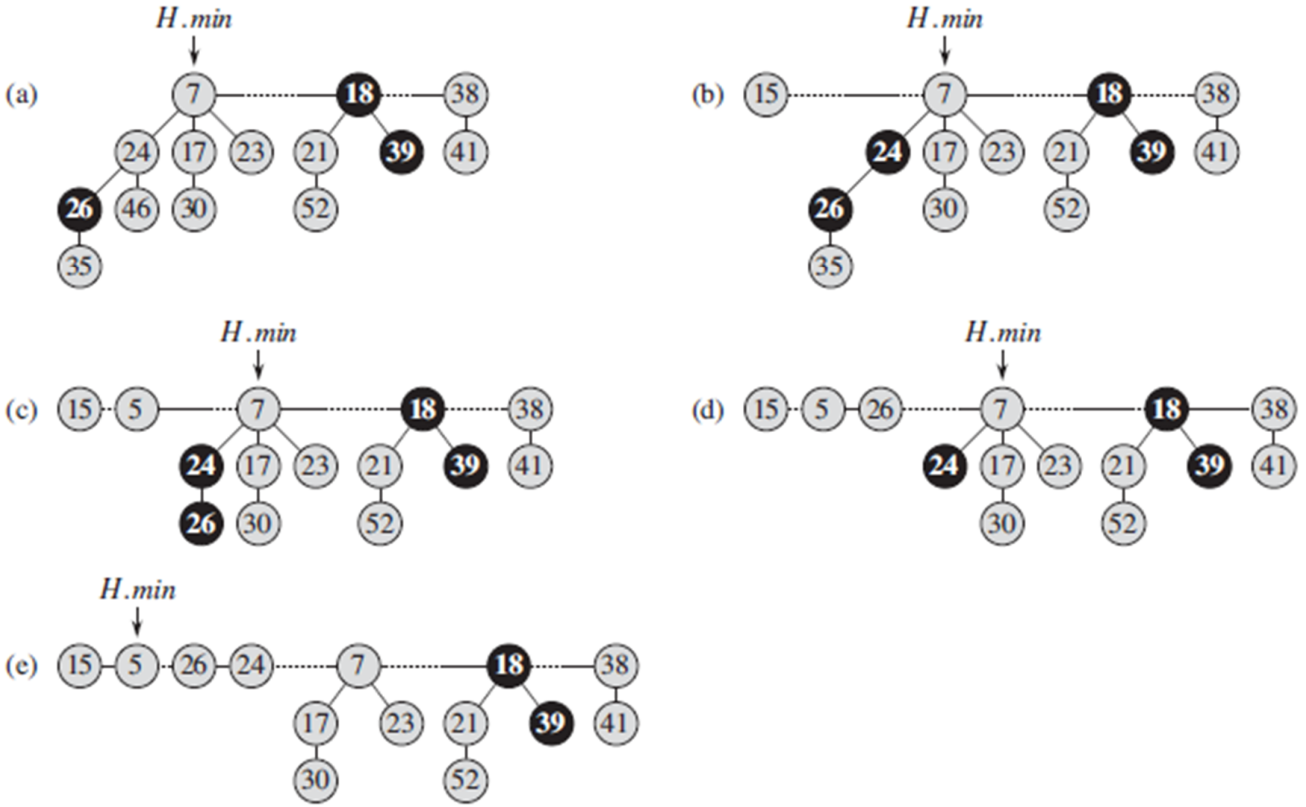
Decrease-key(H,x,k)

```
1  if k > x.key
2      error "new key is greater than current key"
3  x.key = k
4  y = x.p
5  if y ≠ NIL and x.key < y.key
6      CUT(H, x, y)
7      CASCADING-CUT(H, y)
8  if x.key < H.min.key
9      H.min = x
```

```
CUT(H, x, y)
1  remove x from the child list of y, decrementing y.degree
2  add x to the root list of H
3  x.p = NIL
4  x.mark = FALSE
```

```
CASCADING-CUT(H, y)
1  z = y.p
2  if z ≠ NIL
3      if y.mark == FALSE
4          y.mark = TRUE
5      else CUT(H, y, z)
6          CASCADING-CUT(H, z)
```

Fibonacci Heap



46 se redujo a 15

Fibonacci Heap

Variación de ϕ

$$((t(H)+c)+2(m(H)-c+2))-(t(H)+2m(H))=4-c$$

Siendo c la cantidad de llamadas recursivas de CASCADING-CUT, se crearon c – 1 árboles en las primeras c – 1 llamadas y uno por x. La cantidad marca true bajo por esas c-1 llamadas pero la última aumentó.

Ĉ es

$$O(c)+4-c=O(1)$$

Lemma 19.1

Let x be any node in a Fibonacci heap, and suppose that $x.degree = k$. Let $y_1, y_2, \ldots, y_k$ denote the children of x in the order in which they were linked to x, from the earliest to the latest. Then, $y_1.degree \geq 0$ and $y_i.degree \geq i - 2$ for $i = 2, 3, \ldots, k$.

Proof Obviously, $y_1.degree \geq 0$.

For $i \geq 2$, we note that when y_i was linked to x, all of $y_1, y_2, \ldots, y_{i-1}$ were children of x, and so we must have had $x.degree \geq i - 1$. Because node y_i is linked to x (by CONSOLIDATE) only if $x.degree = y_i.degree$, we must have also had $y_i.degree \geq i - 1$ at that time. Since then, node y_i has lost at most one child, since it would have been cut from x (by CASCADING-CUT) if it had lost two children. We conclude that $y_i.degree \geq i - 2$. ■

Fibonacci Heap

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

Lemma 19.2
For all integers $k \geq 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i\,.$$

Proof The proof is by induction on k. When $k = 0$,

$$\begin{aligned} 1 + \sum_{i=0}^{0} F_i &= 1 + F_0 \\ &= 1 + 0 \\ &= F_2\,. \end{aligned}$$

We now assume the inductive hypothesis that $F_{k+1} = 1 + \sum_{i=0}^{k-1} F_i$, and we have

$$\begin{aligned} F_{k+2} &= F_k + F_{k+1} \\ &= F_k + \left(1 + \sum_{i=0}^{k-1} F_i\right) \\ &= 1 + \sum_{i=0}^{k} F_i\,. \end{aligned}$$

■

Fibonacci Heap

Lemma 19.3
For all integers $k \geq 0$, the $(k+2)$nd Fibonacci number satisfies $F_{k+2} \geq \phi^k$.

Proof The proof is by induction on k. The base cases are for $k = 0$ and $k = 1$. When $k = 0$ we have $F_2 = 1 = \phi^0$, and when $k = 1$ we have $F_3 = 2 > 1.619 > \phi^1$. The inductive step is for $k \geq 2$, and we assume that $F_{i+2} > \phi^i$ for $i = 0, 1, \ldots, k-1$. Recall that ϕ is the positive root of equation (3.23), $x^2 = x+1$. Thus, we have

$$\begin{aligned}
F_{k+2} &= F_{k+1} + F_k \\
&\geq \phi^{k-1} + \phi^{k-2} && \text{(by the inductive hypothesis)} \\
&= \phi^{k-2}(\phi + 1) \\
&= \phi^{k-2} \cdot \phi^2 && \text{(by equation (3.23))} \\
&= \phi^k \,.
\end{aligned}$$

■

Fibonacci Heap

Lemma 19.4

Let x be any node in a Fibonacci heap, and let $k = x.degree$. Then $size(x) \geq F_{k+2} \geq \phi^k$, where $\phi = (1 + \sqrt{5})/2$.

Proof Let s_k denote the minimum possible size of any node of degree k in any Fibonacci heap. Trivially, $s_0 = 1$ and $s_1 = 2$. The number s_k is at most $size(x)$ and, because adding children to a node cannot decrease the node's size, the value of s_k increases monotonically with k. Consider some node z, in any Fibonacci heap, such that $z.degree = k$ and $size(z) = s_k$. Because $s_k \leq size(x)$, we compute a lower bound on $size(x)$ by computing a lower bound on s_k. As in Lemma 19.1, let $y_1, y_2, \ldots, y_k$ denote the children of z in the order in which they were linked to z. To bound s_k, we count one for z itself and one for the first child y_1 (for which $size(y_1) \geq 1$), giving

$$\begin{aligned} size(x) &\geq s_k \\ &\geq 2 + \sum_{i=2}^{k} s_{y_i.degree} \\ &\geq 2 + \sum_{i=2}^{k} s_{i-2} , \end{aligned}$$

Fibonacci Heap

where the last line follows from Lemma 19.1 (so that $y_i.degree \geq i - 2$) and the monotonicity of s_k (so that $s_{y_i.degree} \geq s_{i-2}$).

We now show by induction on k that $s_k \geq F_{k+2}$ for all nonnegative integers k. The bases, for $k = 0$ and $k = 1$, are trivial. For the inductive step, we assume that $k \geq 2$ and that $s_i \geq F_{i+2}$ for $i = 0, 1, \ldots, k-1$. We have

$$
\begin{aligned}
s_k &\geq 2 + \sum_{i=2}^{k} s_{i-2} \\
&\geq 2 + \sum_{i=2}^{k} F_i \\
&= 1 + \sum_{i=0}^{k} F_i \\
&= F_{k+2} && \text{(by Lemma 19.2)} \\
&\geq \phi^k && \text{(by Lemma 19.3)} .
\end{aligned}
$$

Thus, we have shown that $\text{size}(x) \geq s_k \geq F_{k+2} \geq \phi^k$. ■