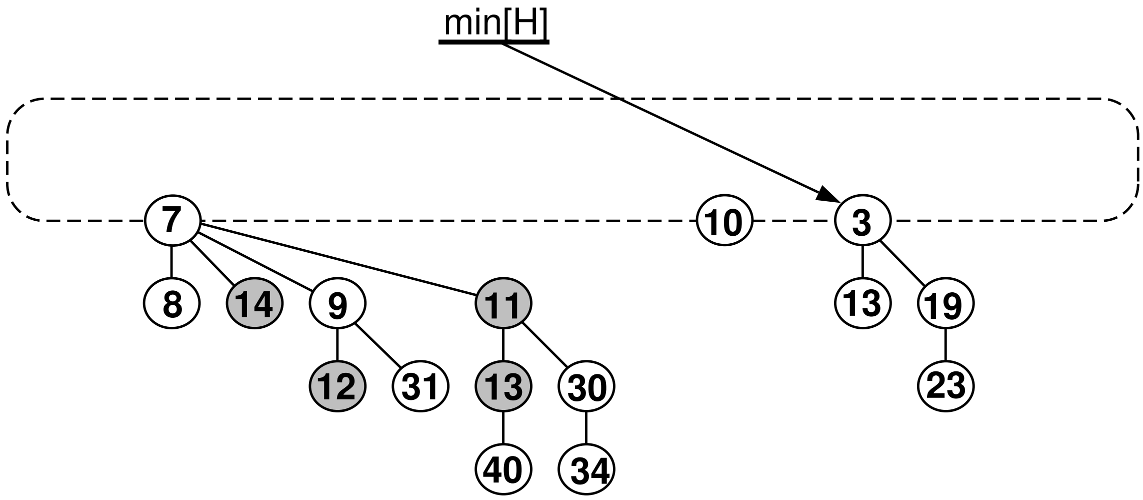
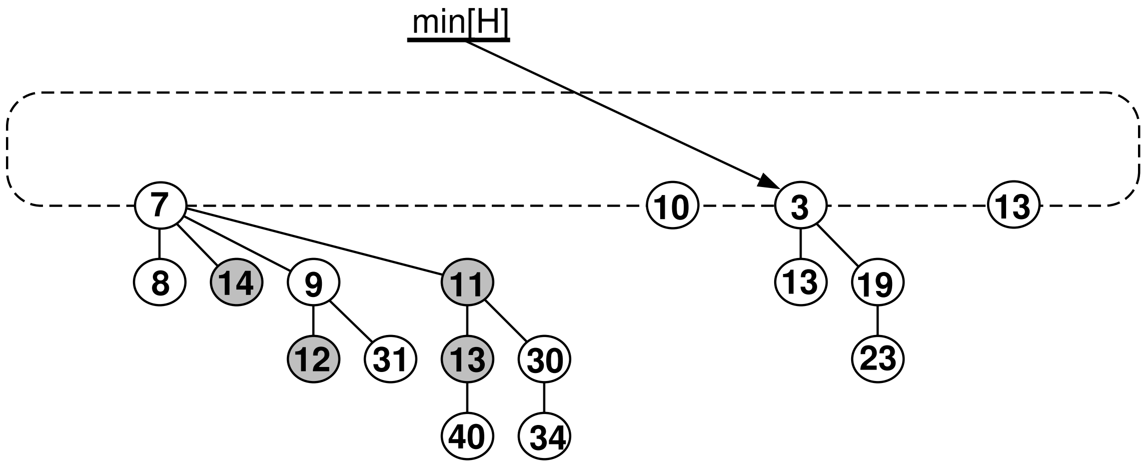
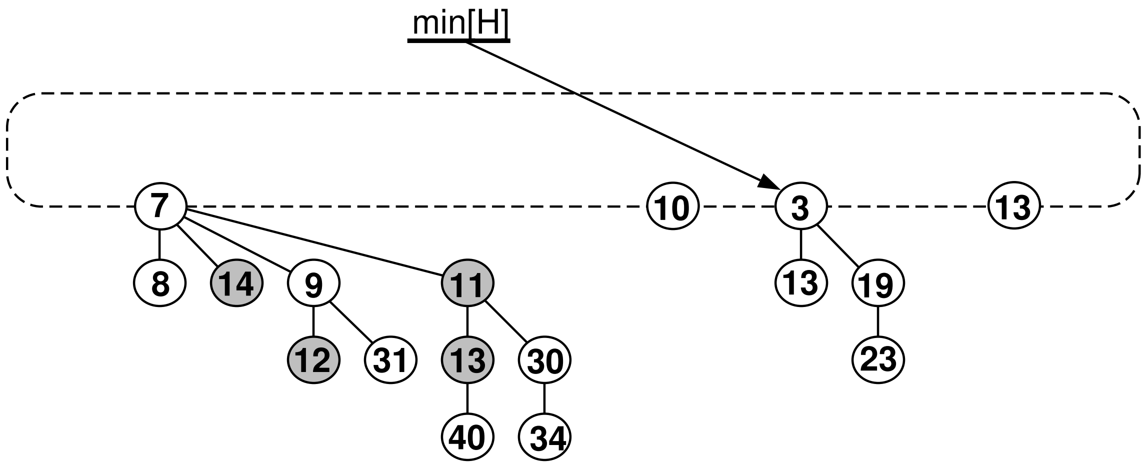


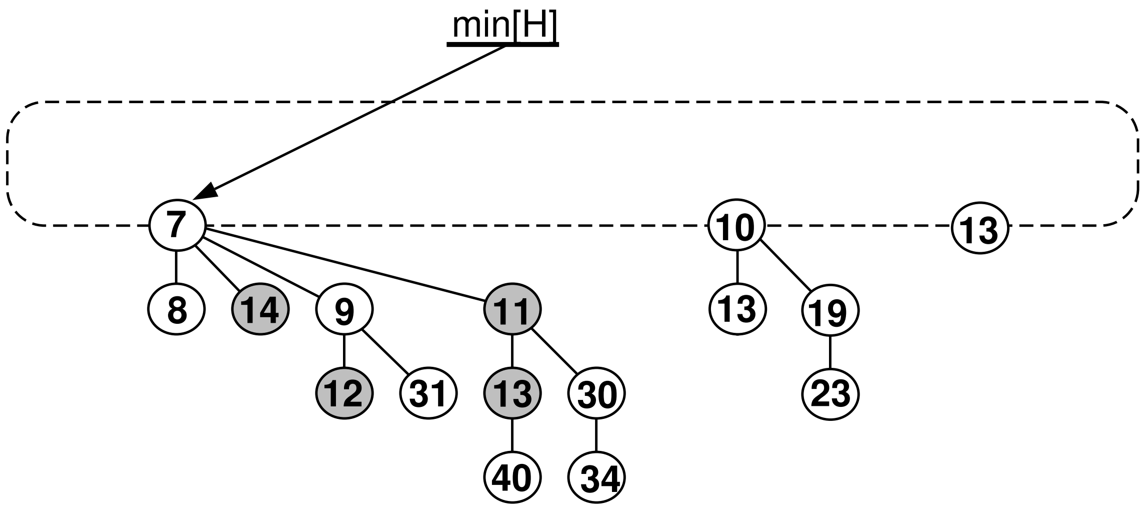
[Die Knoten x mit $\texttt{marked}[x]$ = $\textsc{True}$ sind grau unterlegt.]

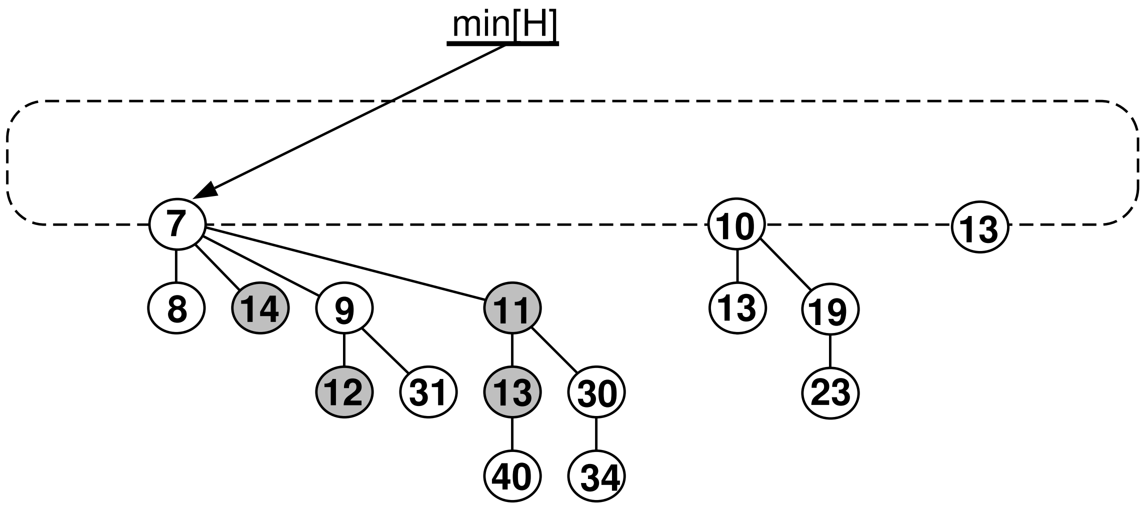
$\underline{\textsc{Insert}(H, 13)}$:



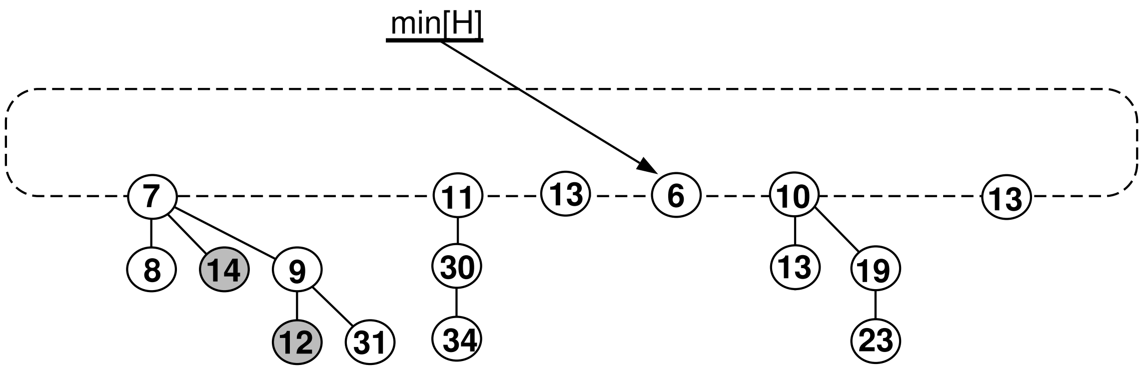


$\textsc{ExtractMin}(H)$:





$\textsc{DecreaseKey}(H,$ <Pointer auf Knoten mit Schlüssel 40>$, 6)$:



```
Extract-Min(H)
──────────────
z := min[H];
if z ≠ nil then
    for each child x of z do
        parent[x] := nil;
        add x to the root list of H;
    remove z from the root list of H;
    if root list of H empty
        then min[H] := nil
        else Consolidate(H);
return z;
```

```
Consolidate(H)
──────────────
for i := 1 to ⌊log_{3/2} n[H]⌋ do A[i] := nil;
repeat
    remove an arbitrary node x from the root list of H;
    while A[rank[x]] ≠ nil do
        y := A[rank[x]]; A[rank[x]] := nil;
        if key[x] > key[y] then exchange x ↔ y;
        make y a child of x, incrementing rank[x];
        marked[x] := FALSE;
    enddo ;
    A[rank[x]] := x;
until root list of H is empty;
min[H] := nil;
for all A[i] ≠ nil do
        add A[i] to the root list of H, updating min[H] if necessary;
```

`Decrease-Key`(H, x, k)

if $k >$ key$[x]$ **then error** ("new key is greater than current key");
key$[x] := k$;
if parent$[x] \neq \mathbf{nil}$ and key$[x] <$ key[parent$[x]$] **then** `Cut`(H, x);

`Cut`(H, x)

if parent$[x] \neq \mathbf{nil}$ **then**
begin
    $y :=$ parent$[x]$;
    parent$[x] := \mathbf{nil}$;
    remove x from the child list of y, decrementing rank$[y]$;
    add x to the root list of H, updating $\min[H]$ if necessary;
    if marked$[y]$ **then** `Cut`(H, y) **else** marked$[y] := \textsc{True}$;
end;