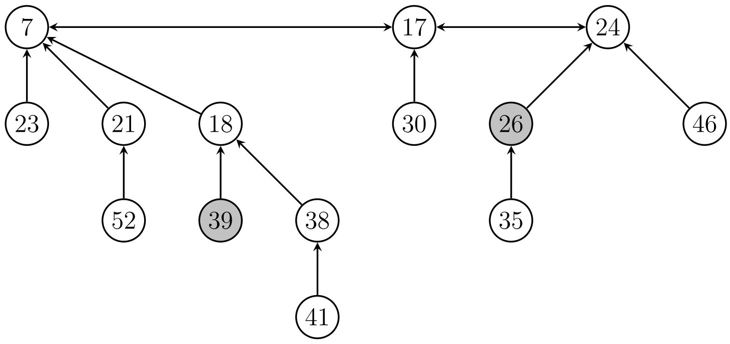
Exercise 6

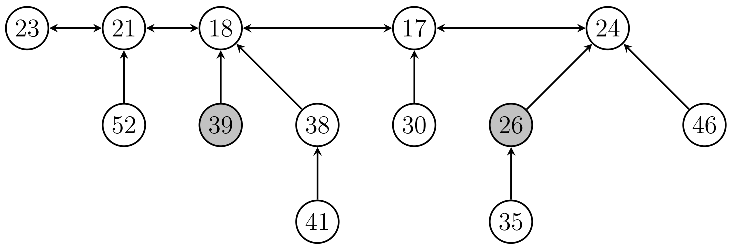
Task 1
Given the following Fibonacci heap:



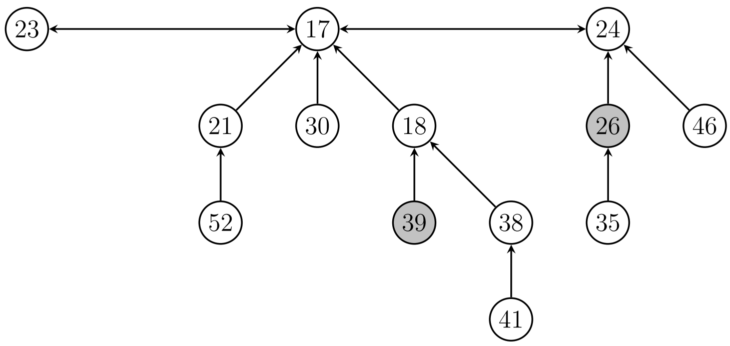
Perform the following operations in that order:
delete-min, **decrease-key**(“52”, 9), **decrease-key**(“46”, 3), **insert**(42), **delete-min**, **decrease-key**(“35”, 7)

Solution

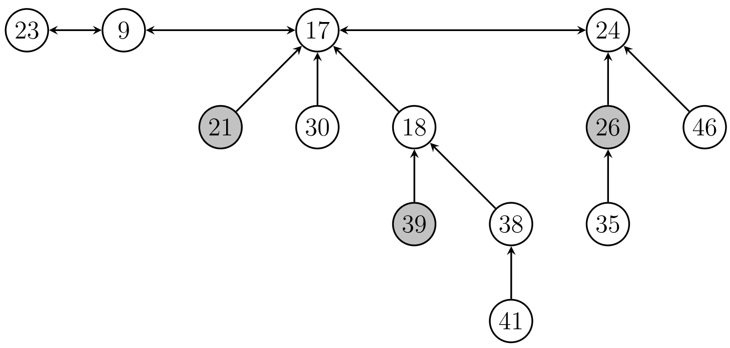
1. **delete-min**: The node with key 7 gets deleted.



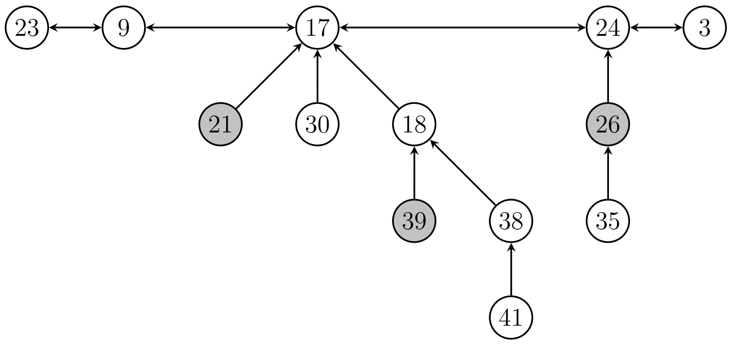
And we tidy the forest a bit.



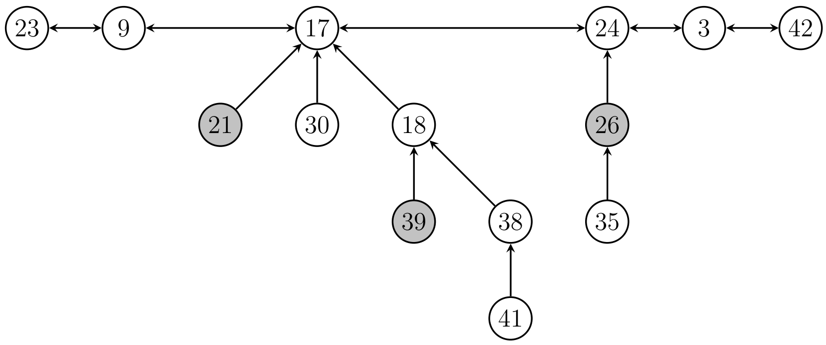
2. **decrease-key**(“52”, 9): 9 moves up, 21 gets marked.



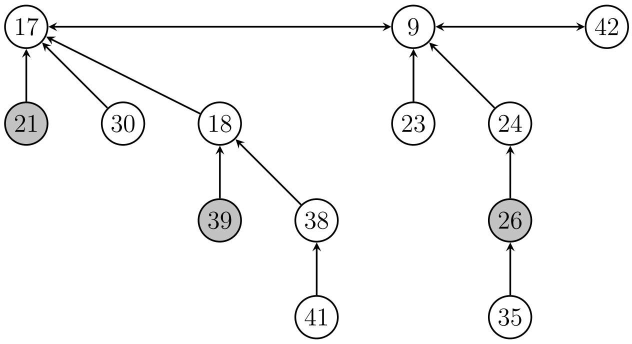
3. **decrease-key**(“46”, 3): 3 moves up, 24 cannot be marked.



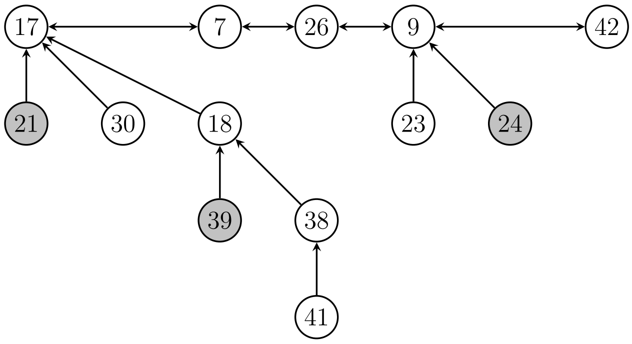
4. **insert**(42): Inserting 42 as a new tree.



5. **delete-min**: Node with key 3 gets deleted and we tidy the forest.



6. **decrease-key**(“35”, 7): 7 moves up and 26 as well, since it is marked (but loses its mark). 24 gets marked.



Task 2

Show Theorem 17 from the lecture: For all $k \geq 0$ we have

$$F_k = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{k+1} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{k+1}.$$

On the last sheet we used $F_0 = 0$ and $F_1 = 1$, but here we use $F_0 = F_1 = 1$!

Solution

Let $x^2 = x + 1$. The two solutions to this equation are $r := \frac{1+\sqrt{5}}{2}$ and $s := \frac{1-\sqrt{5}}{2}$, so we know that $r^2 = r + 1$ and $s^2 = s + 1$.

For $k = 0$ we have

$$\frac{1}{\sqrt{5}}r^1 - \frac{1}{\sqrt{5}}s^1 = \frac{1}{\sqrt{5}}(r - s) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2}\right) = 1 = F_0$$

For $k = 1$ we have

$$\frac{1}{\sqrt{5}}r^2 - \frac{1}{\sqrt{5}}s^2 = \frac{1}{\sqrt{5}}(r^2 - s^2) = \frac{1}{\sqrt{5}}\left((r+1) - (s+1)\right) = \frac{1}{\sqrt{5}}(r - s) = 1 = F_1$$

Assume the statement is already true for $k+1$. Now we prove it for $k+2$:

$$
\begin{aligned}
F_{n+2} &= F_{n+1} + F_n \\
&= \frac{1}{\sqrt{5}} r^{k+1} - \frac{1}{\sqrt{5}} s^{k+1} + \frac{1}{\sqrt{5}} r^k - \frac{1}{\sqrt{5}} s^k \\
&= \frac{1}{\sqrt{5}} \left(r^{k+1} - s^{k+1} + r^k - s^k \right) \\
&= \frac{1}{\sqrt{5}} \left(r^k (r+1) - s^k (s+1) \right) \\
&= \frac{1}{\sqrt{5}} \left(r^{k+2} - s^{k+2} \right) \\
&= \frac{1}{\sqrt{5}} r^{k+2} - \frac{1}{\sqrt{5}} s^{k+2}
\end{aligned}
$$

The induction still works fine, if we use the convention from Sheet 5 for the Fibonacci numbers. Just the formula changes to

$$
F_k = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^k - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^k .
$$

Task 3
Prove or disprove: The height of a Fibonacci heap of size n is at most $O(\log n)$.

Solution
Wrong: A Fibonacci heap of size n can have height n. In order to prove this, we will fix some notation. A Fibonacci heap is a forest, where the roots of the trees have pointers. For trees $t_1, t_2, \dots, t_l$ we write $[t_1 t_2 \cdots t_l]$ for the corresponding Fibonacci heap. For a tree t we write $a(s_1 \cdots s_j)$, if a is its root and the s_i are the subtrees pointing at root a.
We can get a Fibonacci heap of size 1 with height 1 by a single call to insert. Assume we have a Fibonacci heap of size n with height n, say $[a(t)]$, so t has height $n-1$ and size $n-1$. We add three nodes (three calls to insert) with value $b < a$ (so b is the smallest value in the whole forest). This yields $[bbba(t)]$. Then we do one call to delete-min: This removes one of the three nodes we just added: $[bba(t)]$. It also combines the other two new nodes into a tree of rank 1, since both have rank 0: $[b(b)a(t)]$ (rank = number of children). This tree in turn is combined with the old tree, since it also has rank 1: $[b(a(t)b)]$. Since b is the smallest value, it became the new root node. By deleting the single child node labelled with b (by calling decrease-key on it and then delete-min) we obtain $[b(a(t))]$ which is a tree of size $n+1$ and height $n+1$.

Task 4
Find the optimal order to compute the following product (only the dimensions of the matrices are given):

$$
(2 \times 4) \cdot (4 \times 6) \cdot (6 \times 1) \cdot (1 \times 10) \cdot (10 \times 10)
$$

Solution
We compute the number of multiplications by dynamic programming.

Matrix products of length 2: 48 | 24 | 60 | 100

Matrix products of length 3 ($2+1$ or $1+2$):
$48+12$; $24+8=32$ | $24+40=64$; $60+240$ | $60+600$; $100+60=160$

Matrix products of length 4 ($3+1$ or $2+2$ or $1+3$):
$32+20=52$; $48+60+120$; $64+80$ | $64+400$; $24+100+40=164$; $160+240$

Matrix product of length 5 ($4+1$ or $3+2$ or $2+3$ or $1+4$):
$52+200$; $32+100+20=152$; $48+160+120$; $164+80$

Hence, to compute the product $ABCDE$ it is the best to compute $X=BC$ and $Y=DE$ first, then $Z=AX$ and finally ZY, which takes 152 multiplications.

Task 5
Let $X=(x_1,\dots,x_m)$ and $Y=(y_1,\dots,y_n)$ be two sequences. We say X is a *subsequence* of Y if there are indices $1\le i_1<i_2<\dots<i_m\le n$ such that for all $1\le j\le m$ it holds that $x_j=y_{i_j}$.
Use dynamic programming to implement an algorithm that runs in polynomial time which, given two sequences X and Y, computes the length of the longest common subsequence of X and Y.

Solution
Let $c[i,j]$ be the length of a LCS of $(x_1,\dots,x_i)$ and $(y_1,\dots,y_j)$. We have

$$c[i,j]=\begin{cases}0 & \text{if } i=0 \text{ or } j=0,\\ c[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i=y_j,\\ \max(c[i-1,j],c[i,j-1]) & \text{if } i,j>0 \text{ and } x_i\neq y_j.\end{cases}$$

We iniciate the table with 0 at position $c[i,j]$, where i or j is 0. Wlog. let $n<m$. In the first step, we compute $c[i,1]$ for $i=1,\dots,n$ and $c[1,j]$ for $j=1,\dots,m$. In step k we compute $c[i,k]$ for $i=k,\dots,n$ and $c[k,j]$ for $j=k,\dots,m$. After $\min(n,m)=n$ steps we filled in exactly the whole table and we know the value $c[n,m]$. The algorithm works in time $\mathcal{O}(n\cdot m)\subseteq\mathcal{O}(m^2)$.
Example: $X=(1,2,4), Y=(2,3,4,6)$. The goal is $c[3,4]$.

$i\backslash j$	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	1	1	1	1
3	0	1	1	2	2

Since $3<4$, we can also just fill in the table row by row.