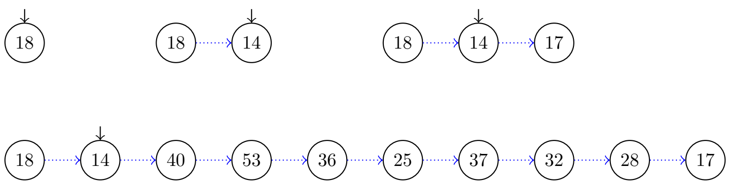
# Algorithms Theory, Solution for Assignment 5

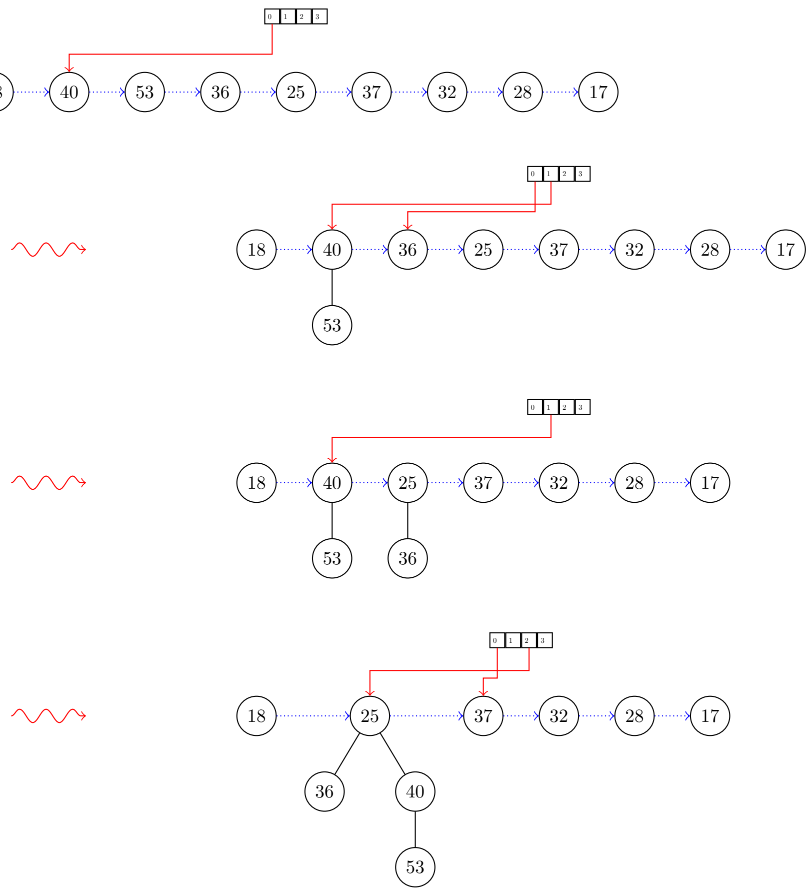
http://lak.informatik.uni-freiburg.de/lak_teaching/ws09_10/algo0910.php

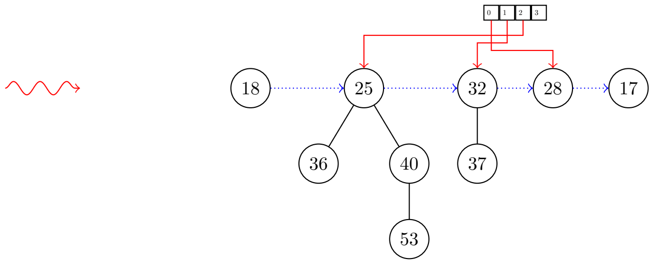
## Exercise 5.1 - Fibonacci Heaps

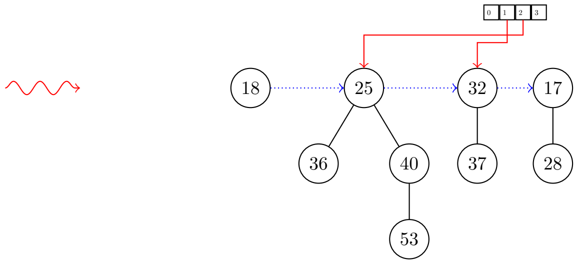
- Inserts

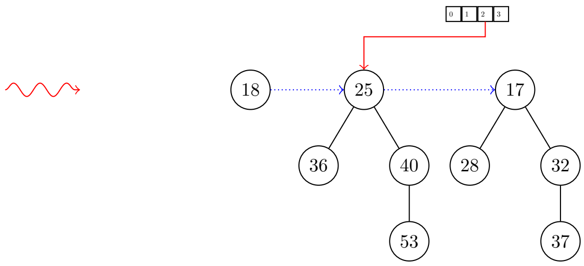


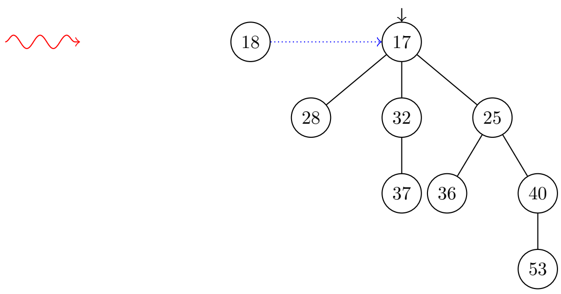

- *deleteMin*()



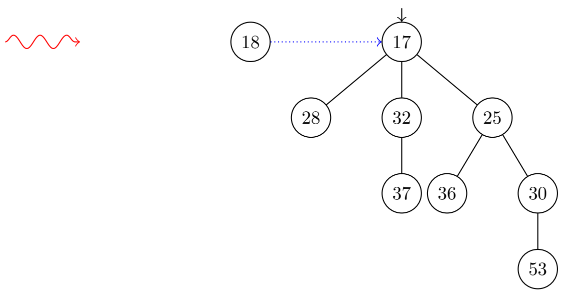


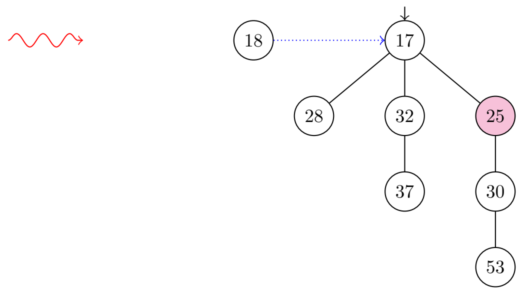



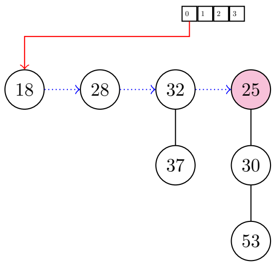

- $decreaseKey(40, 30)$

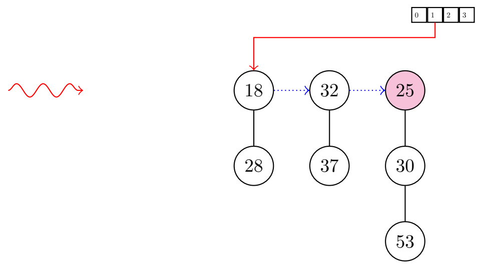


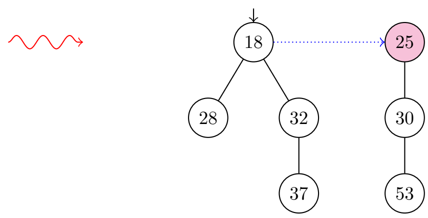

- $delete(36)$


18
17
28
32
25
37
30
53

- $deleteMin()$


0 1 2 3
18
28
32
25
37
30
53


0 1 2 3
18
32
25
28
37
30
53


18
25
28
32
30
37
53

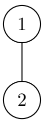
## Exercise 5.2 - Fibonacci Heaps

Suppose that the nodes have key values in $\mathbb{N}_0$.

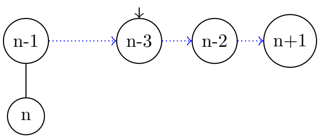
```
insert(n);
if n > 1 then begin
  insert(n-1);
  insert(n-2);
  deletemin();
  for i = 3 to n do begin
    insert(n-i);
    insert(n-(i-1));
    insert(n + 1);
    deletemin();
    decreasekey(n+1,0);
    deletemin();
  end;
end;
```

1. If $n = 1$, only one node with key 1 is inserted and we have a chain of $n = 1$ nodes. 

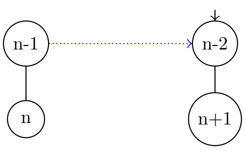

2. Consider the case $n = 2$. Initially, three nodes with keys $2, 1$ and $0$ are inserted as singletons. The following *deletemin* operation deletes the node 0 and, during the consolidation step, links node 2 and node 1. The result is a chain of two nodes 2 and 1, rooted at 1:



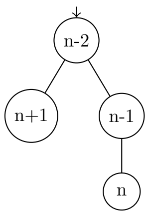

3. Consider the case $n > 2$. When executing the for loop for $i = 3$, nodes $n-3, n-2$ and $n+1$ are inserted as singletons.



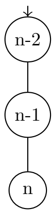

The subsequent *deletemin* operation deletes node $n - 3$. The new minimum is $n - 2$ and in the following consolidation step node $n + 1$ is linked to node $n - 2$.




This yields a tree of rank 1, and hence it has to be united with the existing chain rooted at $n-1$. Since $n-2$ is the current minimum, the existing chain is linked to the new root $n-2$. This root has now two children: $n + 1$ and $n - 1$.




The operation *decreasekey(n+1,0)* and *deletemin* now cause node $n+1$ to be deleted. This does not have any further effects on the rest of the tree. Thus, the resulting tree is again a chain consisting of the consecutive nodes $n-2, n-1, n$.




The next execution of the loop will cause the existing chain to be linked to the new root $n-3$. Continuing up to $i = n$ finally yields a chain from 1 to $n$.

## Exercise 5.3 - Disjoint-set forests

- We need to find a sequence $m$ of operations on $n$ elements that take $\Omega(m \ \lg \ n)$ time. First perform $n$ $MakeSet$ operations. Then we have the following singleton sets: $\{x_1\}, \{x_2\}, \ldots, \{x_n\}$. Now perform the $n-1$ $Union$ operations bellow, to create a single set whose tree has depth $\lg n$

| | |
|---|---|
| $Union(x_1, x_2)$ | $n/2$ of these op. |
| $Union(x_3, x_4)$ | |
| $Union(x_5, x_6)$ | |
| $\vdots$ | |
| $Union(x_{n-1}, x_n)$ | |
| $Union(x_2, x_4)$ | $n/4$ of these op. |
| $Union(x_6, x_8)$ | |
| $Union(x_{10}, x_{12})$ | |
| $\vdots$ | |
| $Union(x_{n-2}, x_n)$ | |
| $Union(x_4, x_8)$ | $n/8$ of these op. |
| $Union(x_{12}, x_{16})$ | |
| $Union(x_{20}, x_{24})$ | |
| $\vdots$ | |
| $Union(x_{n-4}, x_n)$ | |
| $\vdots$ | |
| $Union(x_{n/2}, x_n)$ | 1 of these op. |

  Finally, perform $m - 2n + 1$ $findSet$ operations on the deepest element on the tree. Each of these operations take $\Omega(\lg n)$ time. Letting $m \geq 3n$ we have more than $\frac{m}{3}$ $findSet$ operations. Therefore, the total cost is $\Omega(m \lg n)$.

- We use a stack S.

```
function find-set(x) begin
    S.init()
    top = x
    while top ≠ top.parent do
        S.push(top)
        top = top.parent
    end
    while ¬ (S.isEmpty) do
        tmp = S.pop()
        tmp.parent = top
    end
    return top
```

## Exercise 5.4 - Ackerman Function

### 5.4.1 - Definition of Ackerman Function

The lecture introduces a modified version of the Ackerman Function which is defined as:

$$\begin{aligned} A(0,j) &= j+1 \\ A(k,j) &= A^{(j+1)}(k-1,j) && \text{for } k \geq 1 \\ \text{where } A^{(i+1)}(k,j) &:= A(k, A^{(i)}(k,j)) && \text{for } i \in \mathbb{N} \end{aligned}$$

### 5.4.2 - Prove

We prove monotony of the Ackerman Function in both of its components. We prove

$$A(k, j+1) \geq A(k,j) \tag{M:j}$$
$$A(k+1, j) \geq A(k,j) \tag{M:k}$$

for all $k, j \in \mathbb{N}$.

#### 5.4.2.1 - Lemma[1]

It holds

$$A(1,j) = 2j+1 \tag{1}$$

**Prove** We prove $A^{(i)}(0,j) \stackrel{!}{=} j+i$, which implies (1), since $A(1,j) = A^{(j+1)}(0,j)$. We continue by induction over $i$:

**Induction Start:** $i = 0$

$$A^{(0)}(0,j) = A(0,j) = j+0 = j+i$$

**Induction Step:** $i-1 \to i$. Assume $A^{(i-1)}(0,j) = j + (i-1)$

$$\begin{aligned} A^{(i)}(0,j) &= A(0, A^{(i-1)}(0,j)) \\ &= A^{(i-1)}(0,j) + 1 \\ &= j + i - 1 + 1 \\ &= j + i \end{aligned}$$

□

[1] known from the lecture

#### 5.4.2.2 - Induction over $k$

**Induction Start:** $k = 0$. We have to show that (M:j) and (M:k) for all $j$. It holds that $A(0, j + 1) = j + 1 + 1 \geq j + 1 = A(0, j)$. Due to (1) we have $A(1, j) = 2j + 1 \geq j + 1 = A(0, j)$.

**Induction Step:** $k' \to k+1$ for all $k' \leq k$. We assume (M:j) and (M:k) holds for $k'$ and for all $j$:

$$A(k', j+1) \geq A(k', j) \tag{M:j'}$$
$$A(k'+1, j) \geq A(k', j) \tag{M:k'}$$

We have to prove:

$$A(k+1, j+1) \overset{!}{\geq} A(k+1, j) \tag{2}$$
$$A(k+2, j) \overset{!}{\geq} A(k+1, j) \tag{3}$$

We start proving (2):

$$\begin{aligned} A(k+1, j+1) &= A^{(j+2)}(k, j+1) \\ &= A(k, A^{(j+1)}(k, j+1)) \end{aligned}$$

Since the first parameter is $k$, we can apply (M:j') (which reduces j+1 by one):

$$\geq A(k, A^{(j+1)}(k, j))$$

Now we use (M:k') $k$ times to reduce $k$ to 0:

$$\begin{aligned} &\geq A(0, A^{(j+1)}(k, j)) \\ &= A^{(j+1)}(k, j) + 1 \\ &\geq A^{(j+1)}(k, j) \\ &= A(k+1, j) \end{aligned}$$

Next we prove (3).

$$\begin{aligned} A(k+2, j) &= A^{(j+1)}(k+1, j) \\ &\geq A^{(j+1)}(k, j) && \text{apply (M:k') } j+1 \text{ times} \\ &= A(k+1, j) \end{aligned}$$

Hence (2) and (3) are shown. □