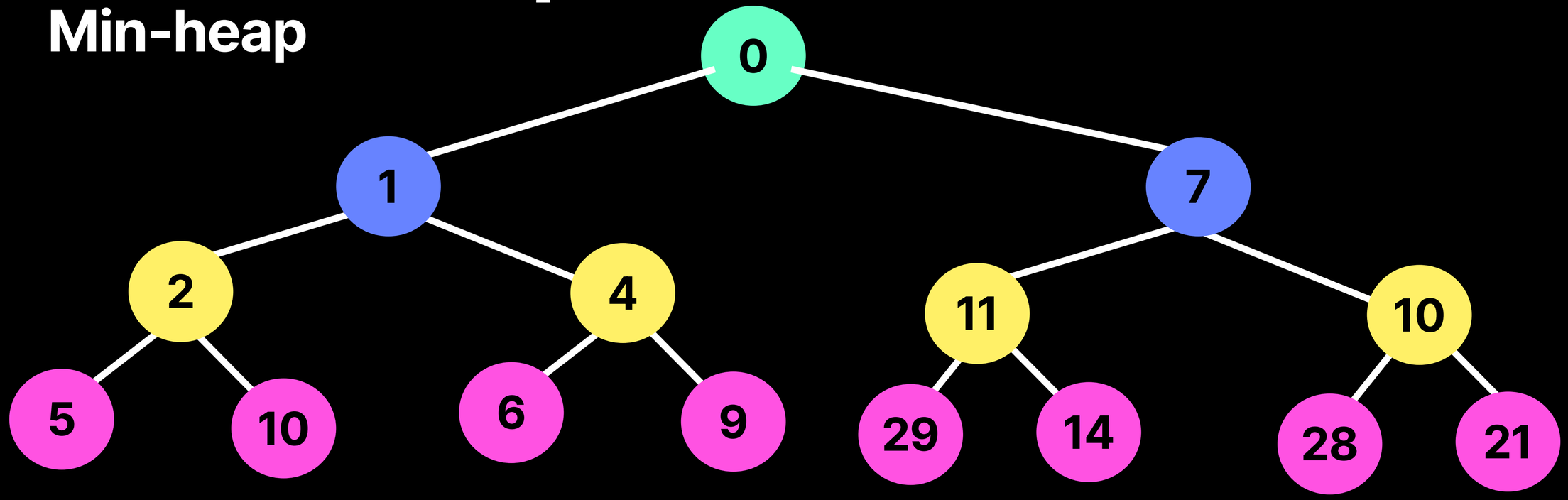
IN2010

- HeapSort
- Huffman
- Rød-svarte trær

Binære heaps

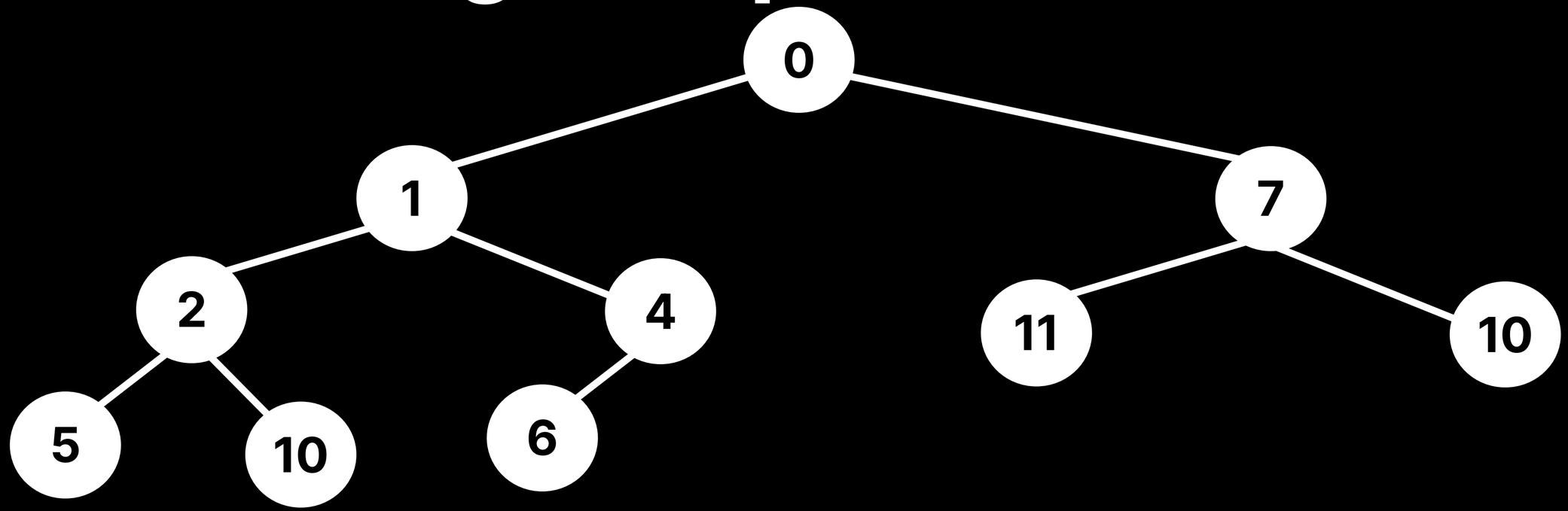
Min-heap



0	1	7	2	4	11	10	5	10	6	9	29	14	28	21

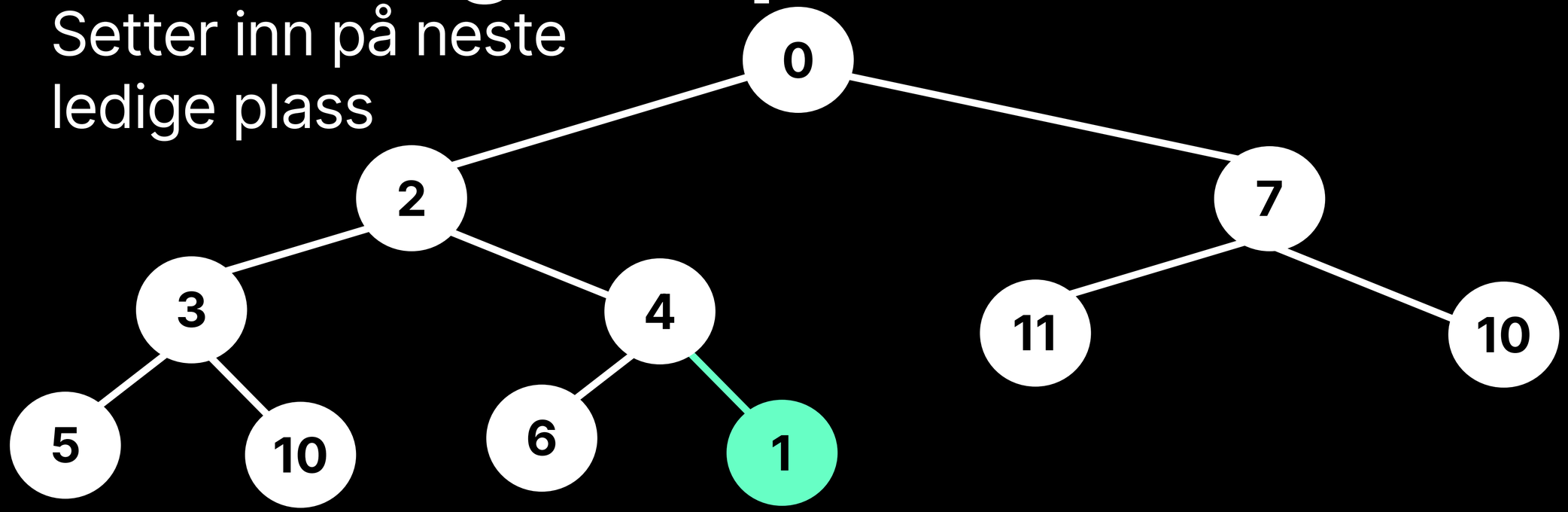
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Innsetting i heap



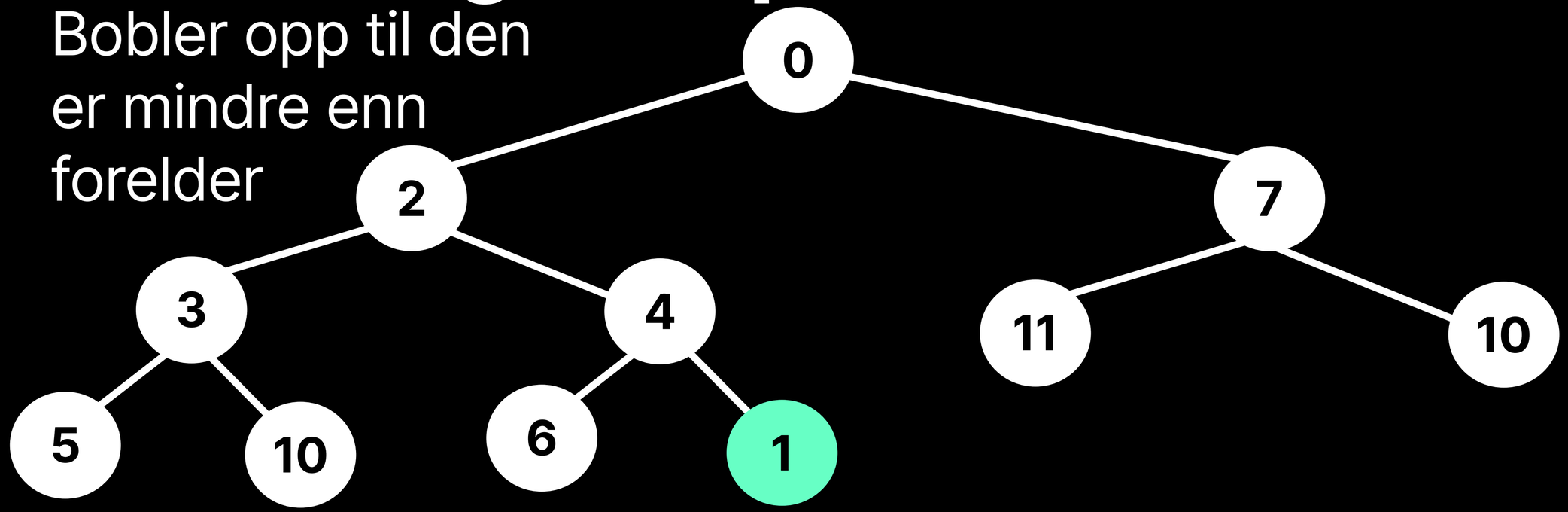
Innsetting i heap

Setter inn på neste ledige plass



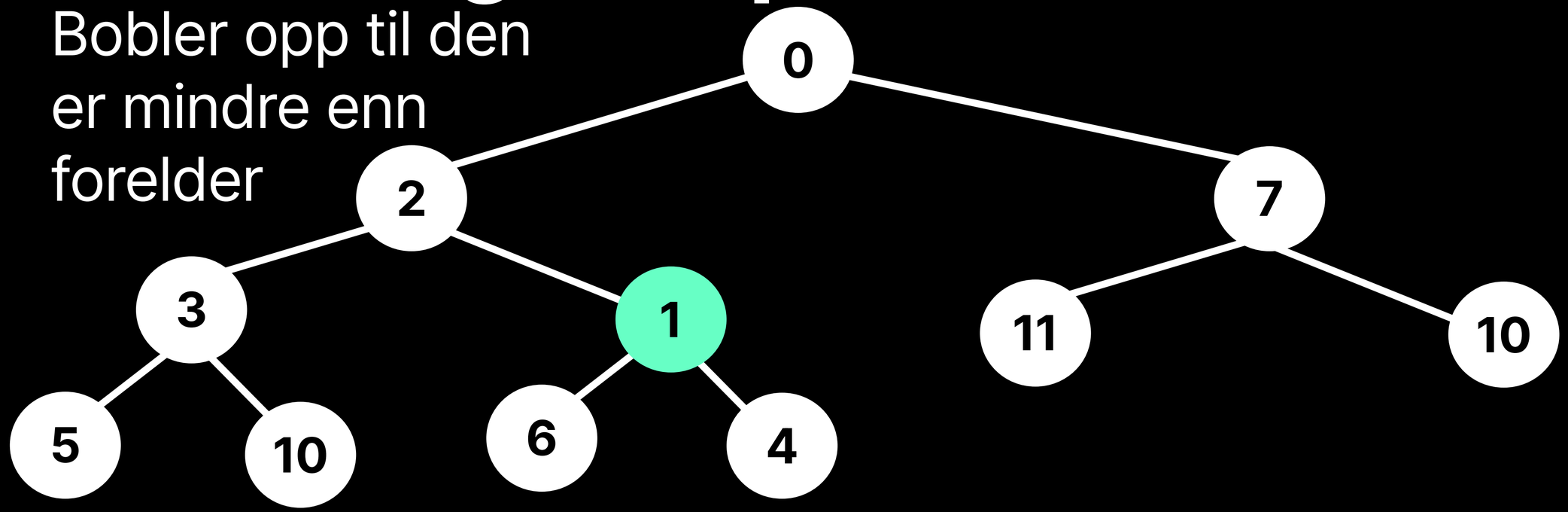
Innsetting i heap

Bobler opp til den er mindre enn forelder



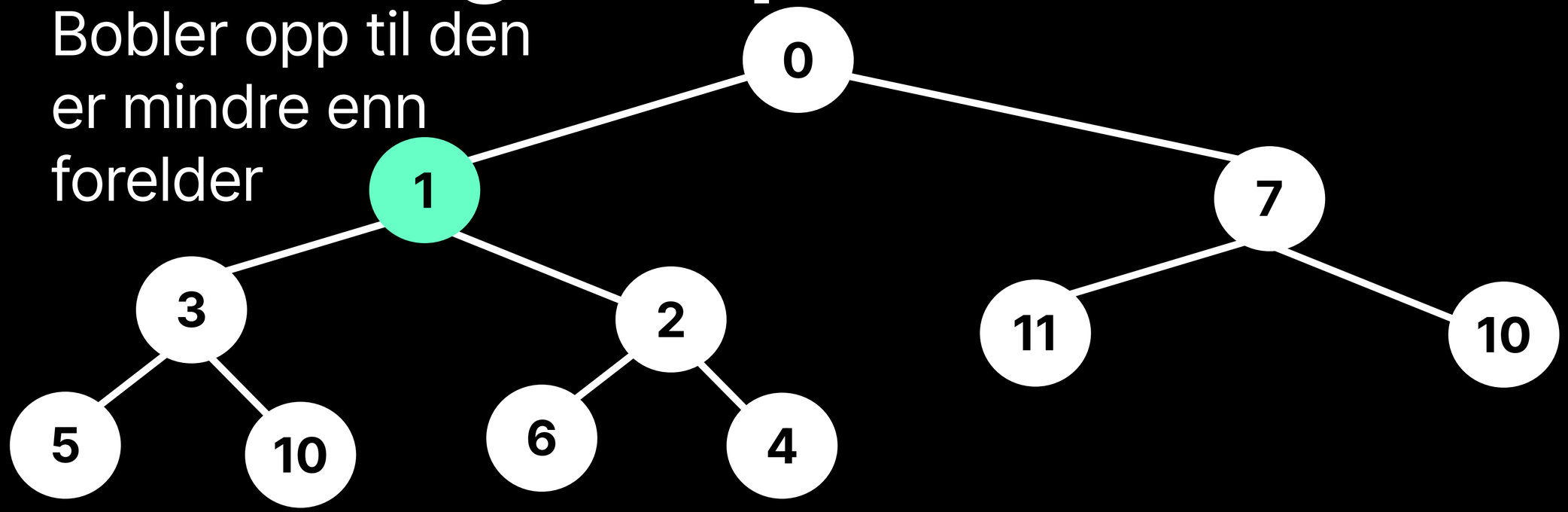
Innsetting i heap

Bobler opp til den er mindre enn forelder



Innsetting i heap

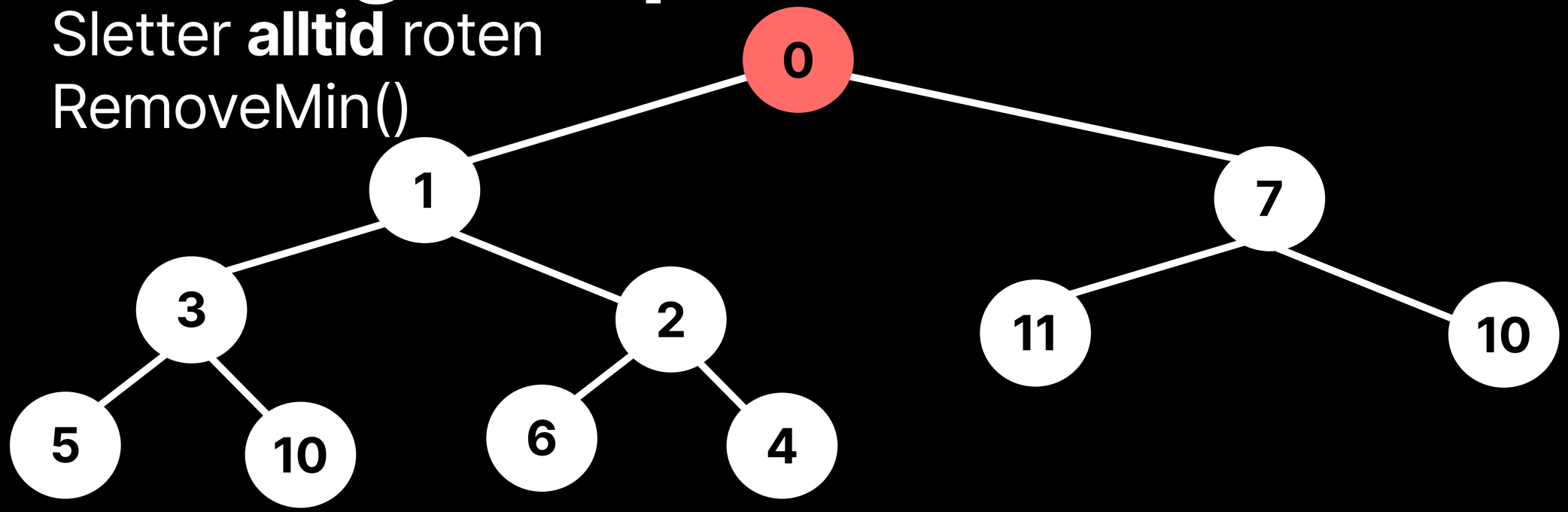
Bobler opp til den er mindre enn forelder



Sletting i heap

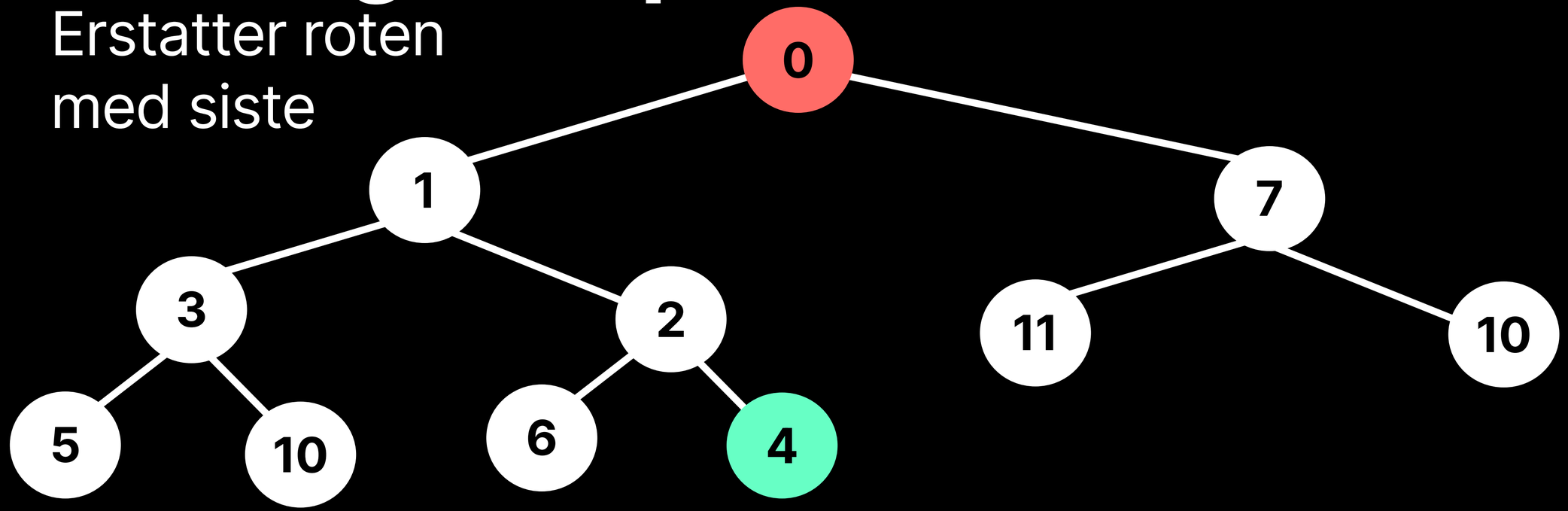
Sletter **alltid** roten

RemoveMin()



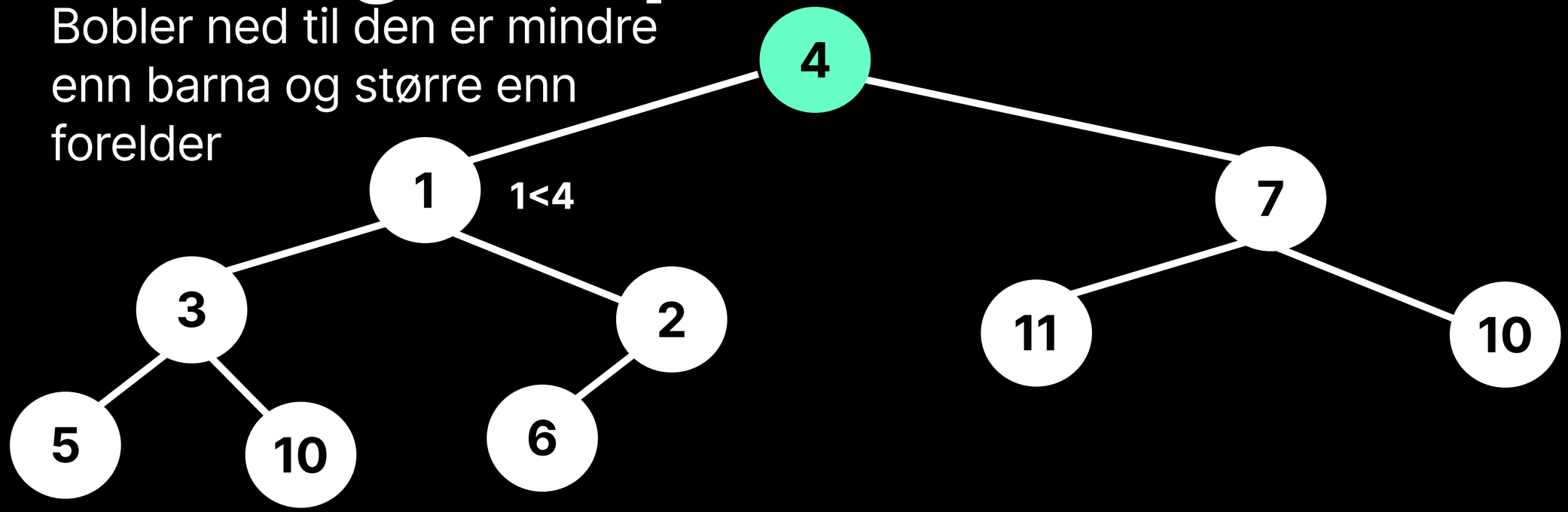
Sletting i heap

Erstatter roten
med siste



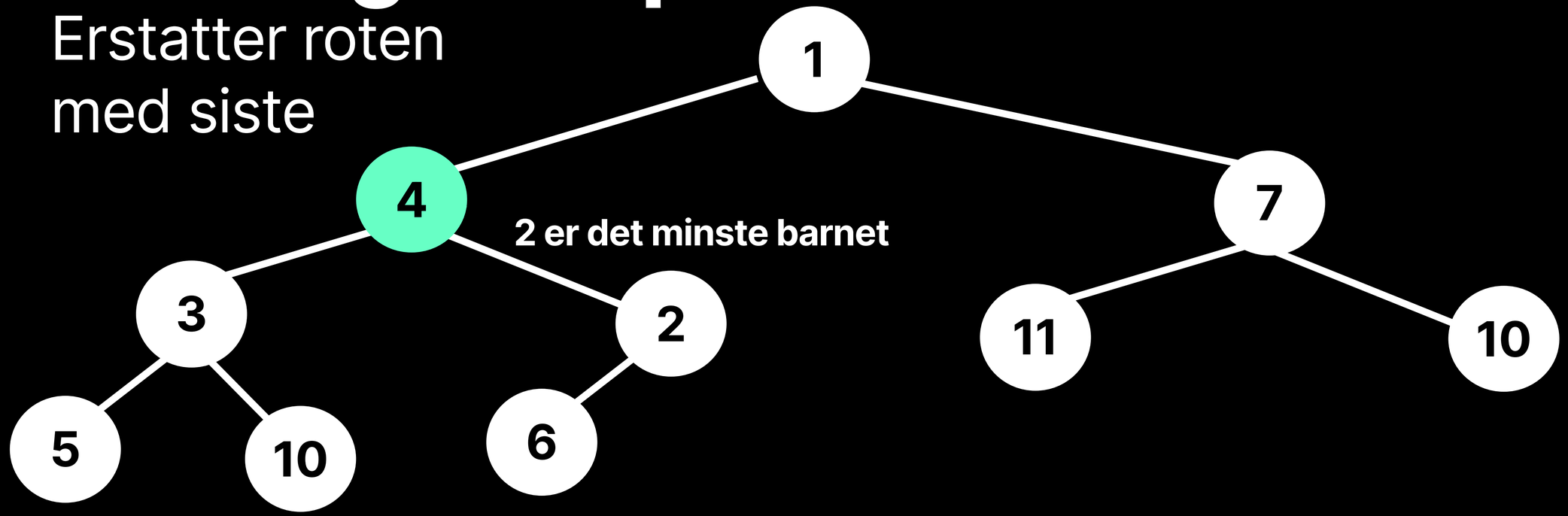
Sletting i heap

Bobler ned til den er mindre enn barna og større enn forelder



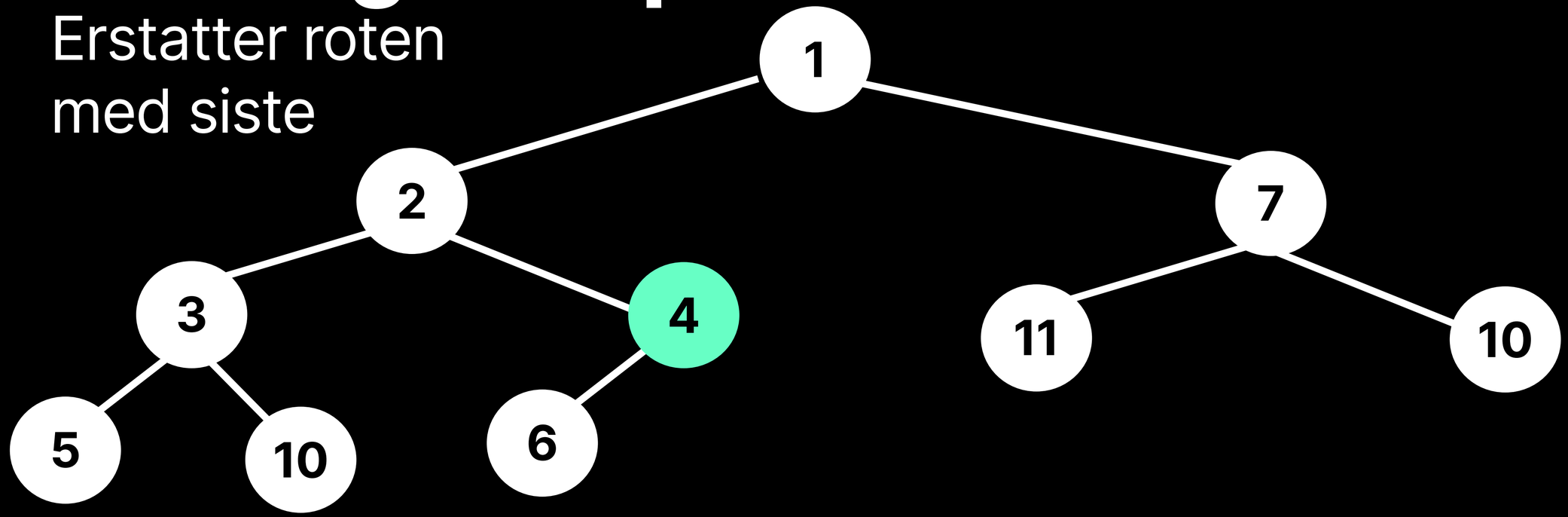
Sletting i heap

Erstatter roten
med siste



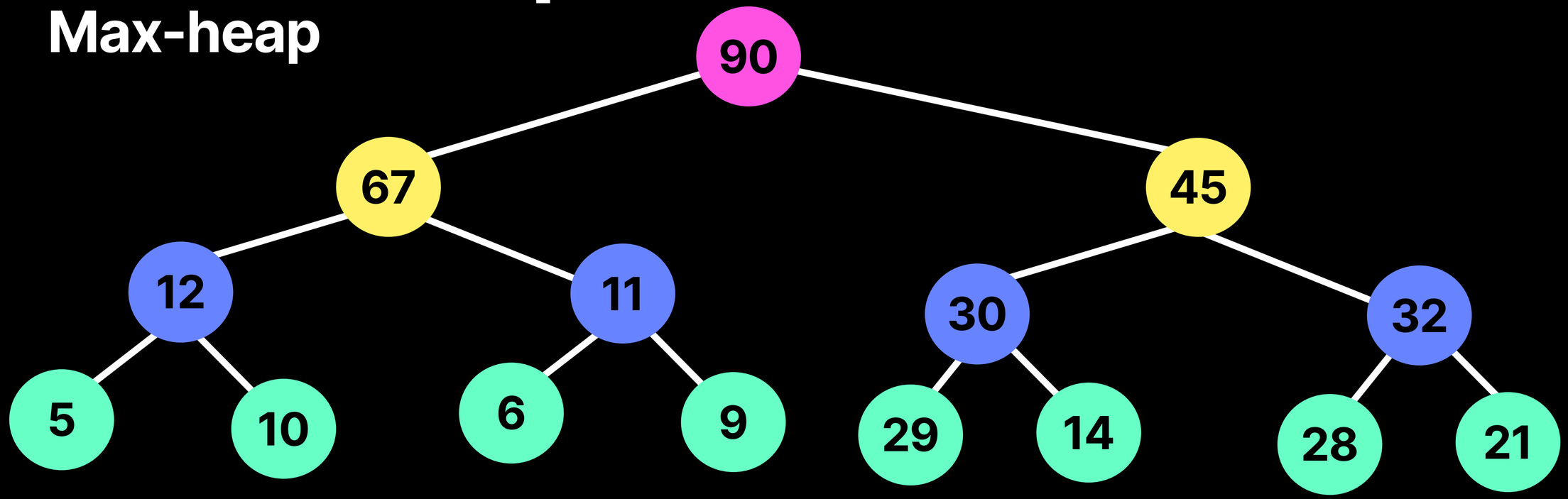
Sletting i heap

Erstatter roten
med siste



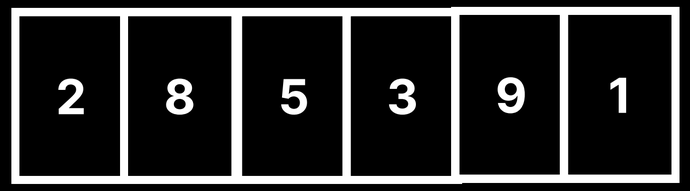
Binære heaps

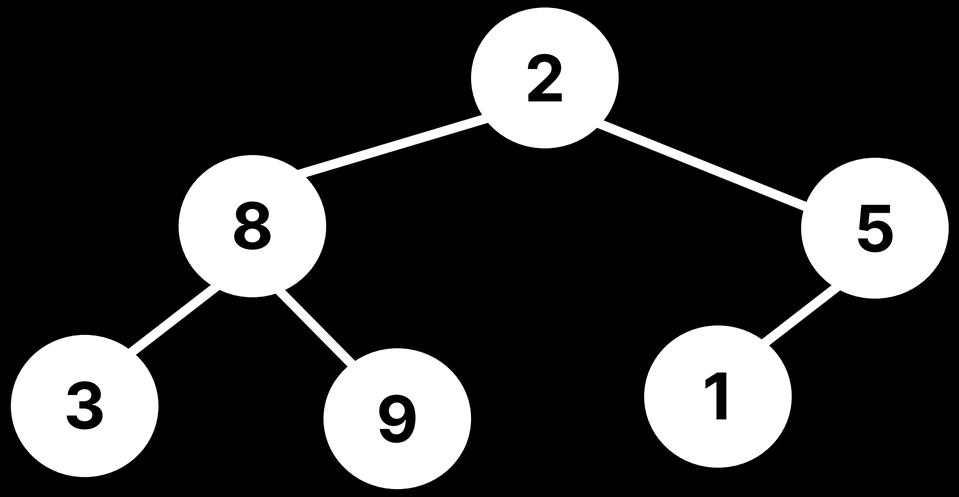
Max-heap



HeapSort

Maks-heap





HeapSort

Maks-heap

1. Lag en maks-heap

2	8	5	3	9	1

ALGORITHM: BYGG EN MAX-HEAP

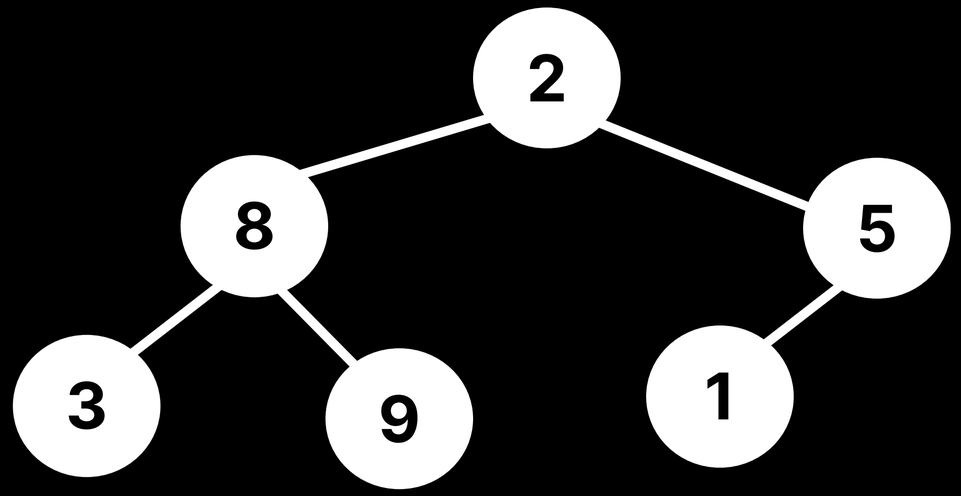
Input: Et array A med n elementer
Output: A som en max-heap

```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

1. Lag en maks-heap

ALGORITHM: BYGG EN MAX-HEAP

Input: Et array A med n elementer
Output: A som en max-heap

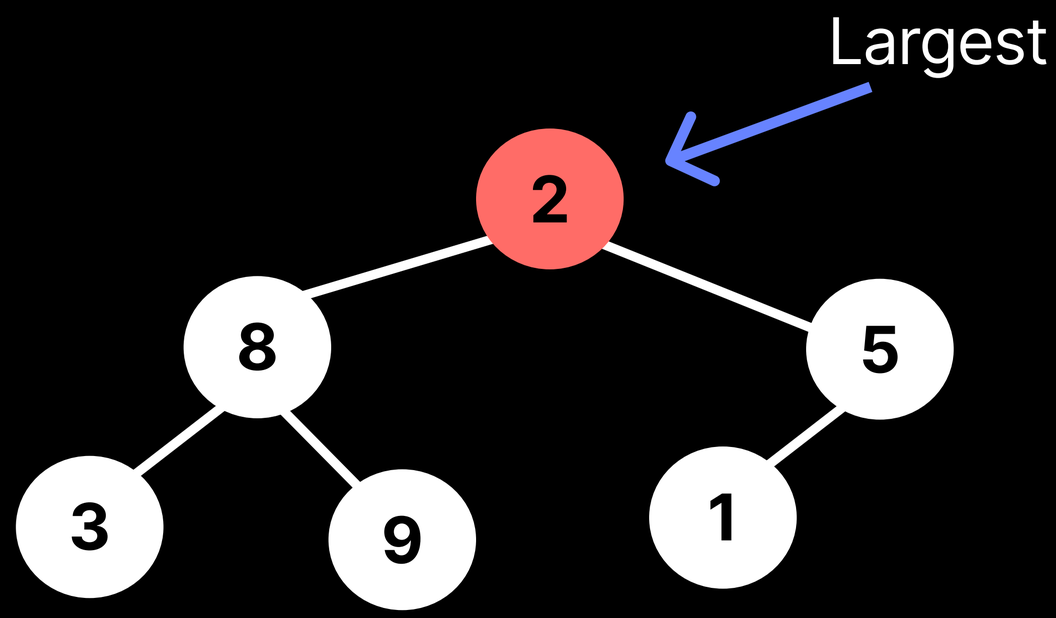
```
1 Procedure BuildMaxHeap(A, n)
2     for i ← ⌊n/2⌋ down to 0 do
3         BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

2	8	5	3	9	1



HeapSort

Maks-heap

1. Lag en maks-heap

2	8	5	3	9	1

ALGORITHM: BYGG EN MAX-HEAP

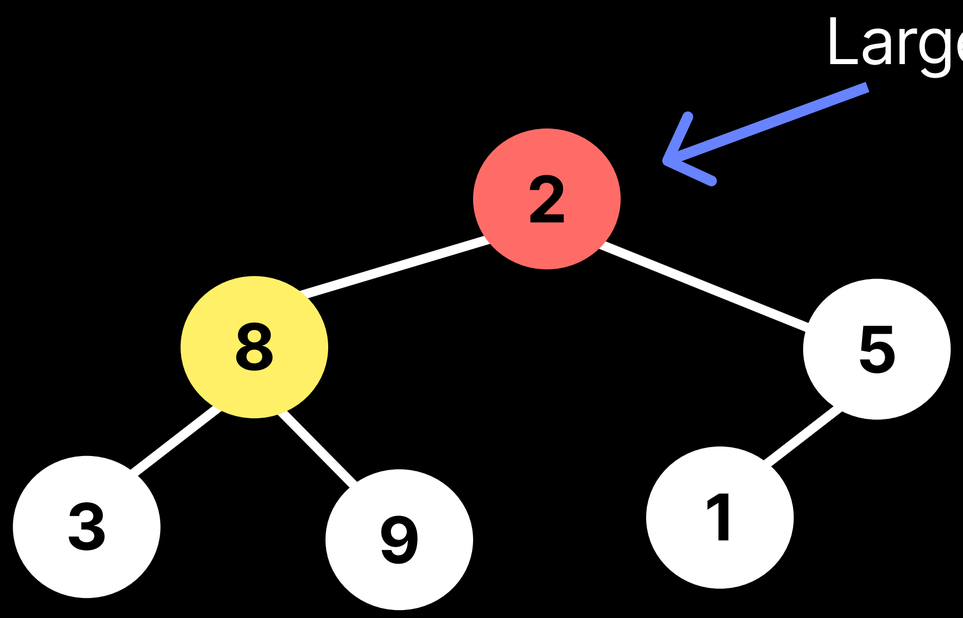
Input: Et array A med n elementer
Output: A som en max-heap

```
1 Procedure BuildMaxHeap(A, n)
2     for i ← ⌊n/2⌋ down to 0 do
3         BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

1. Lag en maks-heap

2	8	5	3	9	1

ALGORITHM: BYGG EN MAX-HEAP

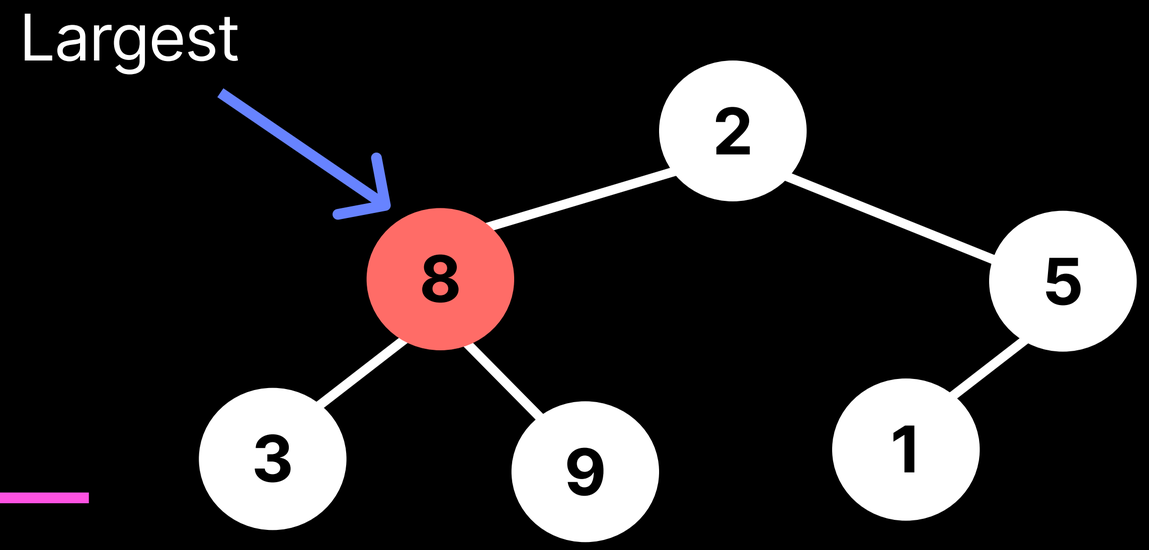
Input: Et array A med n elementer
Output: A som en max-heap

```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

1. Lag en maks-heap

2	8	5	3	9	1

ALGORITHM: BYGG EN MAX-HEAP

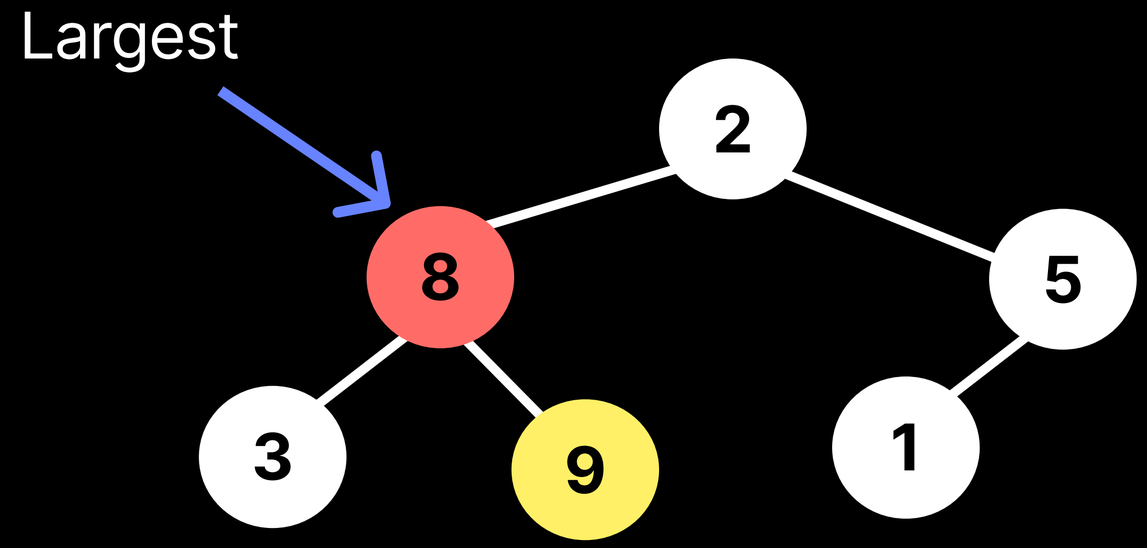
Input: Et array A med n elementer
Output: A som en max-heap

```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

1. Lag en maks-heap

2	8	5	3	9	1

ALGORITHM: BYGG EN MAX-HEAP

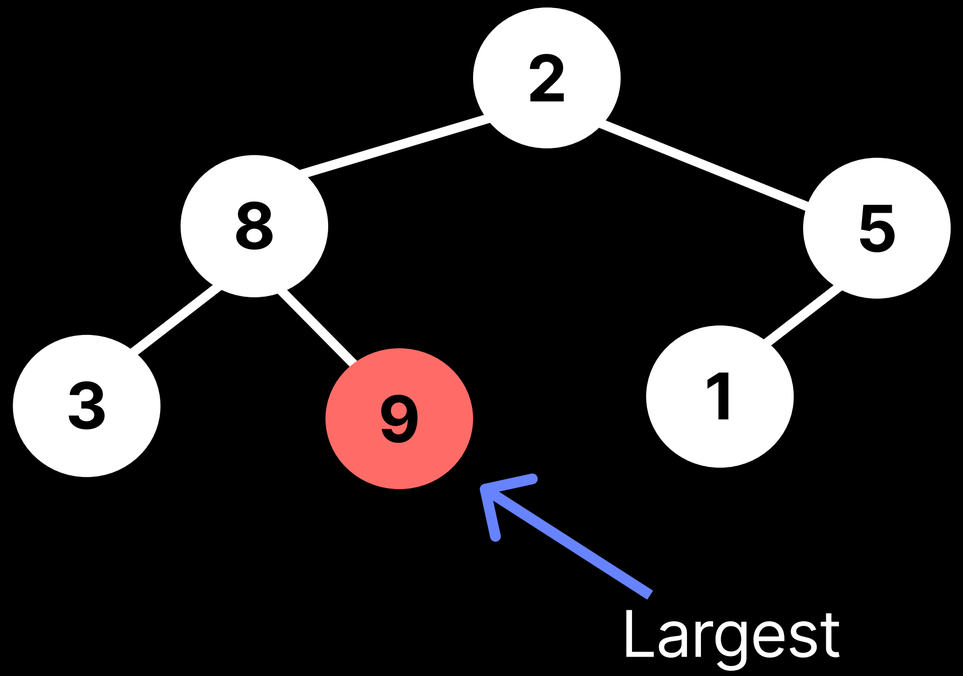
Input: Et array A med n elementer
Output: A som en max-heap

```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

1. Lag en maks-heap

2	8	5	3	9	1

ALGORITHM: BYGG EN MAX-HEAP

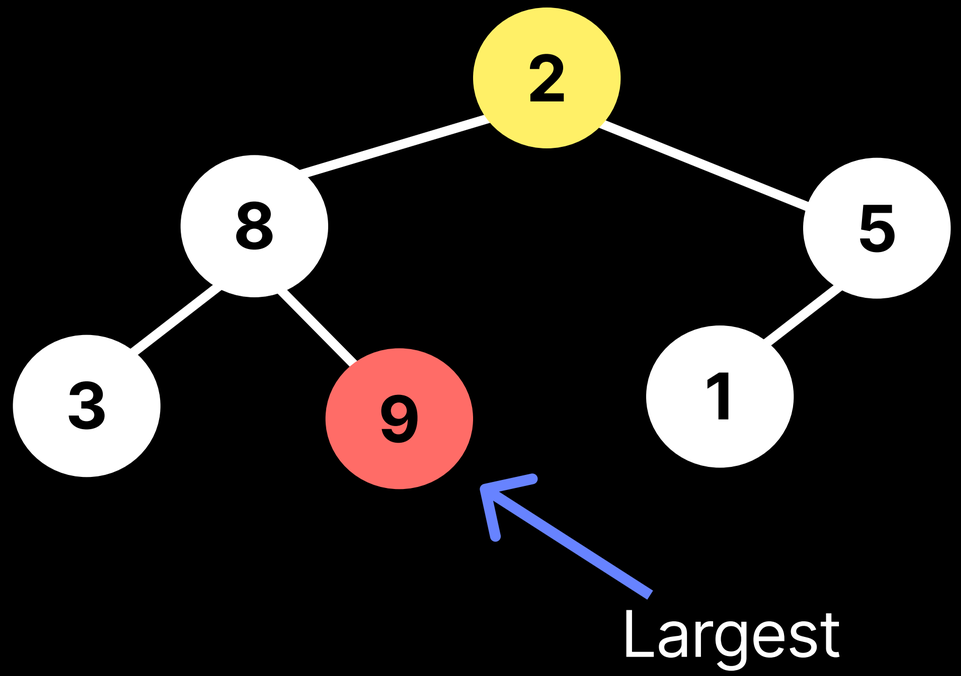
Input: Et array A med n elementer
Output: A som en max-heap

```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

1. Lag en maks-heap

ALGORITHM: BYGG EN MAX-HEAP

Input: Et array A med n elementer
Output: A som en max-heap

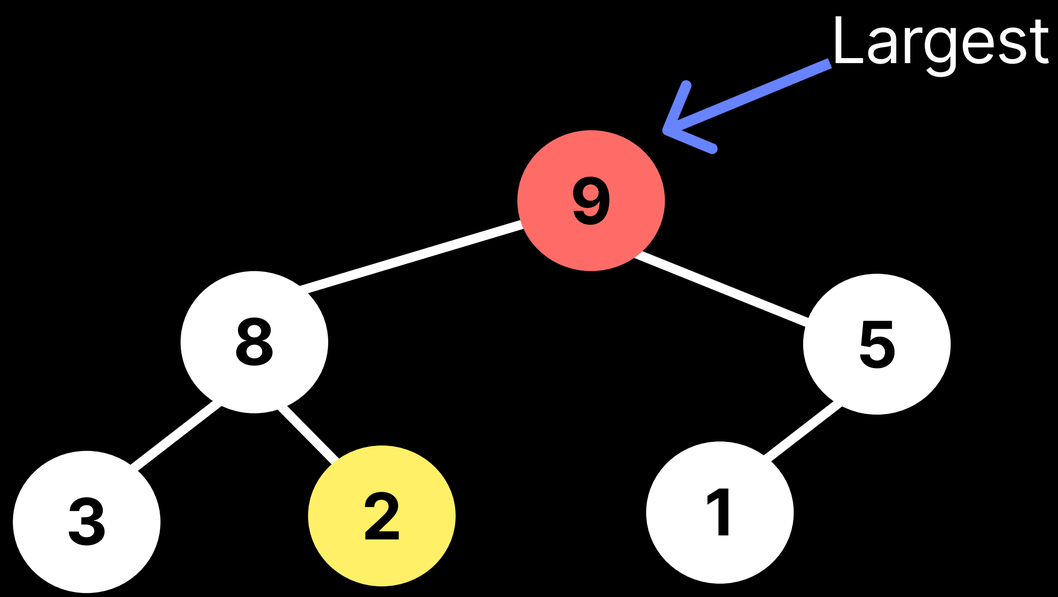
```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

9	8	5	3	2	1



HeapSort

Maks-heap

1. Lag en maks-heap

ALGORITHM: BYGG EN MAX-HEAP

Input: Et array A med n elementer
Output: A som en max-heap

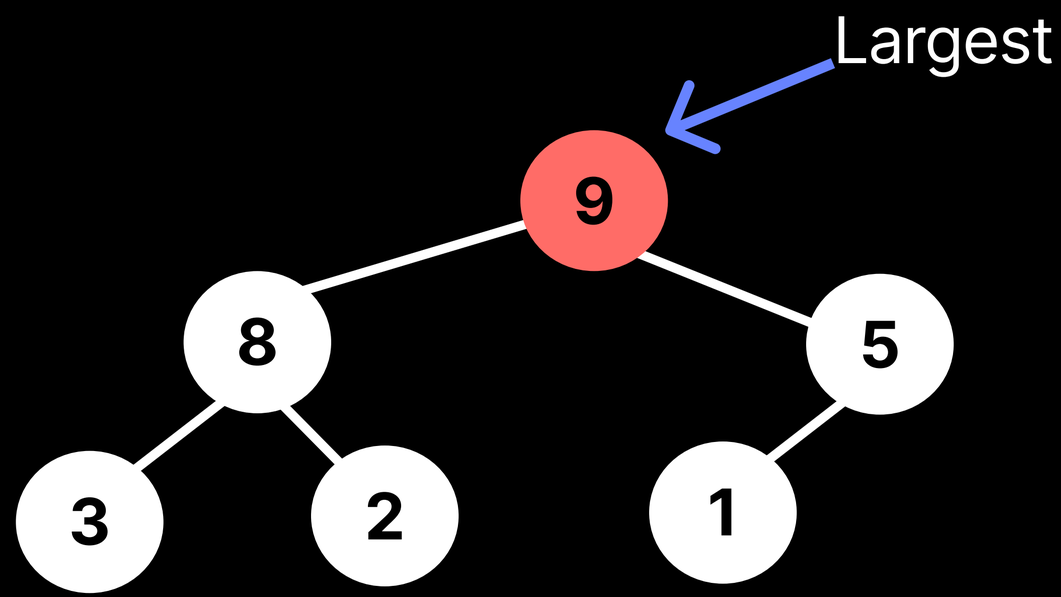
```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

9	8	5	3	2	1



HeapSort

Maks-heap

1. Lag en maks-heap

ALGORITHM: BYGG EN MAX-HEAP

Input: Et array A med n elementer
Output: A som en max-heap

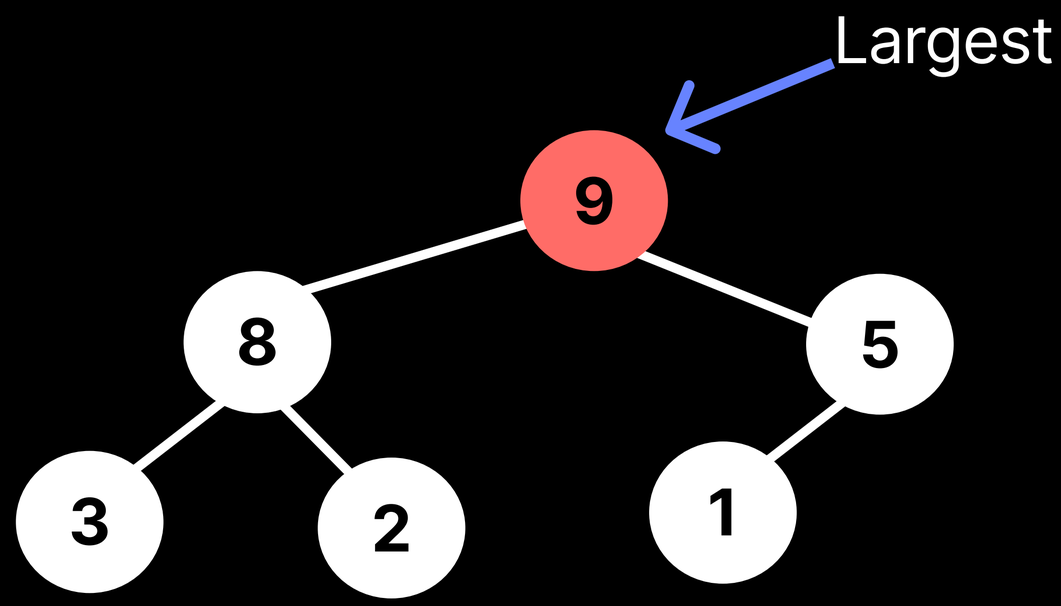
```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

9	8	5	3	2	1



HeapSort

Maks-heap

1. Lag en maks-heap ✅

9	8	5	3	2	1

ALGORITHM: BYGG EN MAX-HEAP

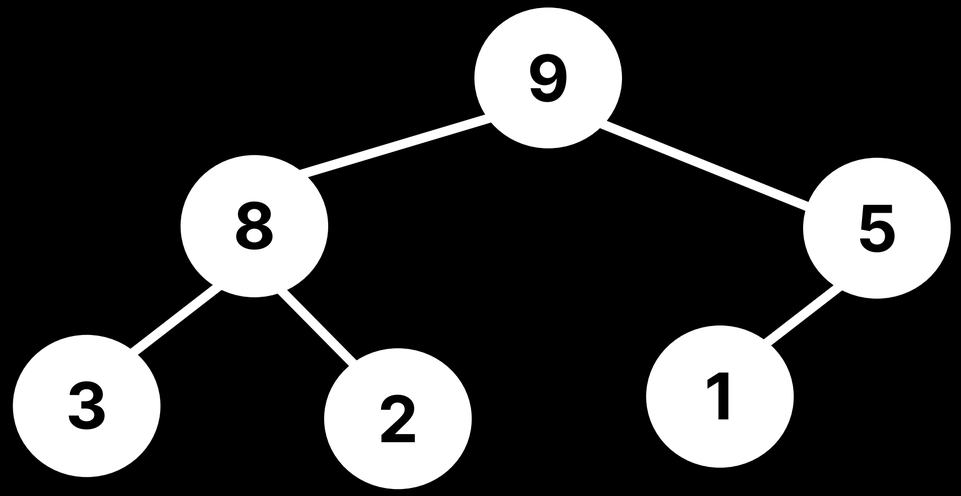
Input: Et array A med n elementer
Output: A som en max-heap

```
1  Procedure BuildMaxHeap(A, n)
2      for i ← ⌊n/2⌋ down to 0 do
3          BubbleDown(A, i, n)
```

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

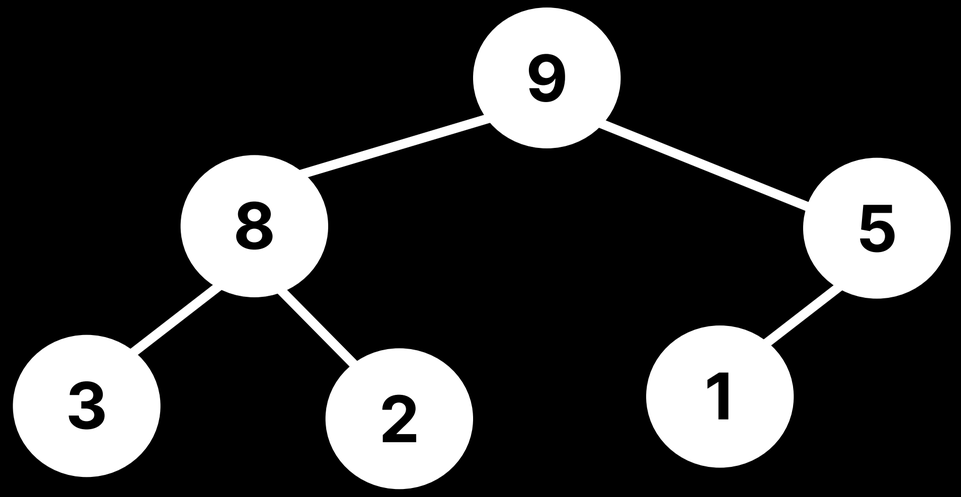
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down

9	8	5	3	2	1

ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

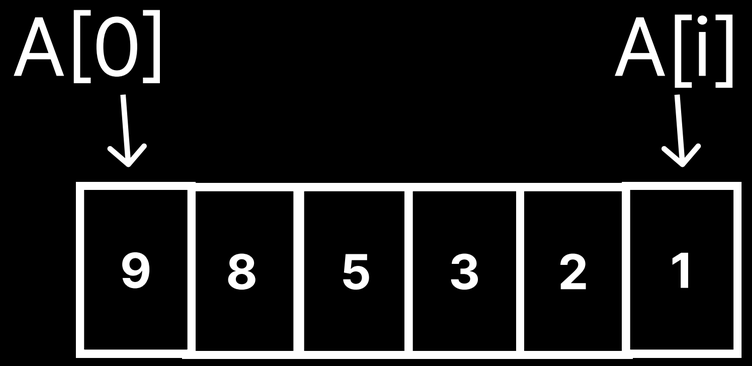
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```

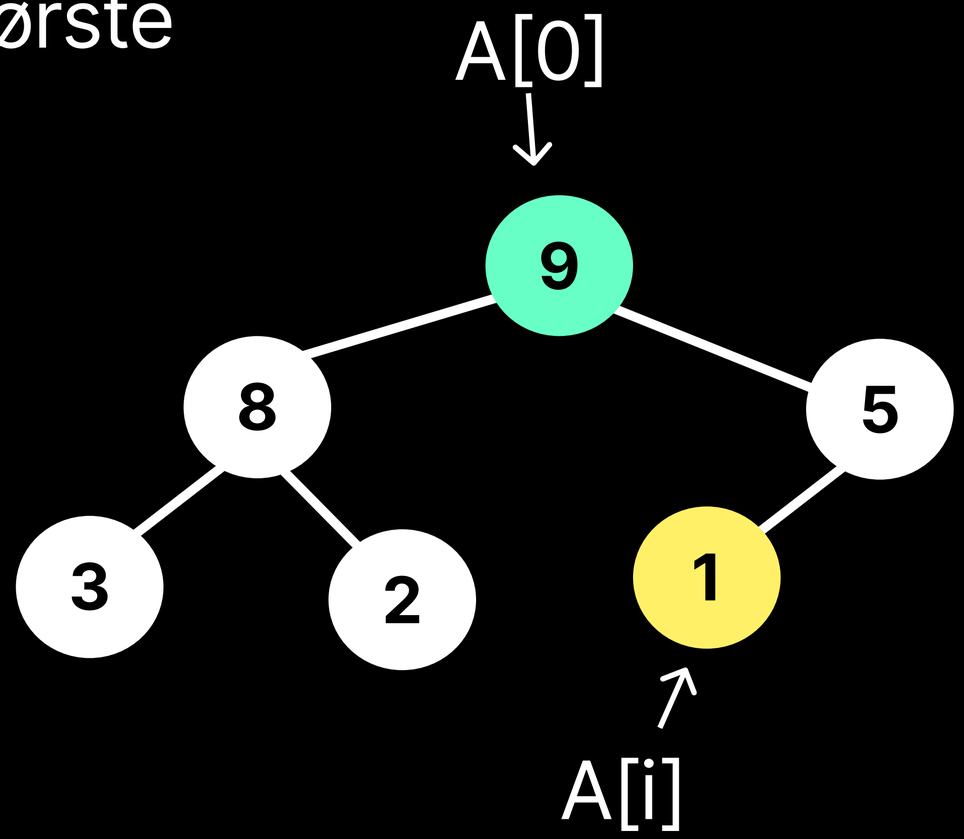


HeapSort

Maks-heap

1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down





ALGORITHM: HEAPSORT

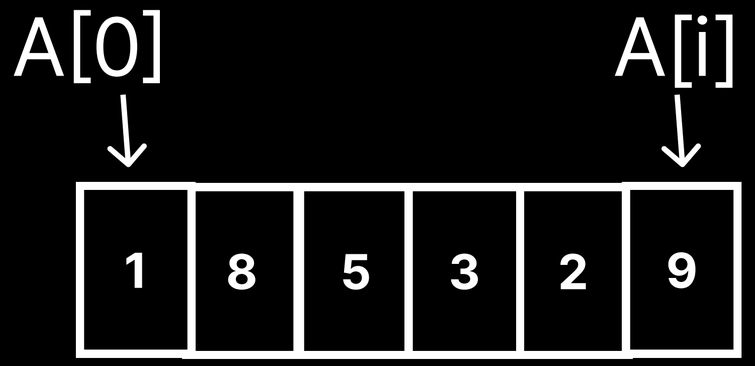
Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

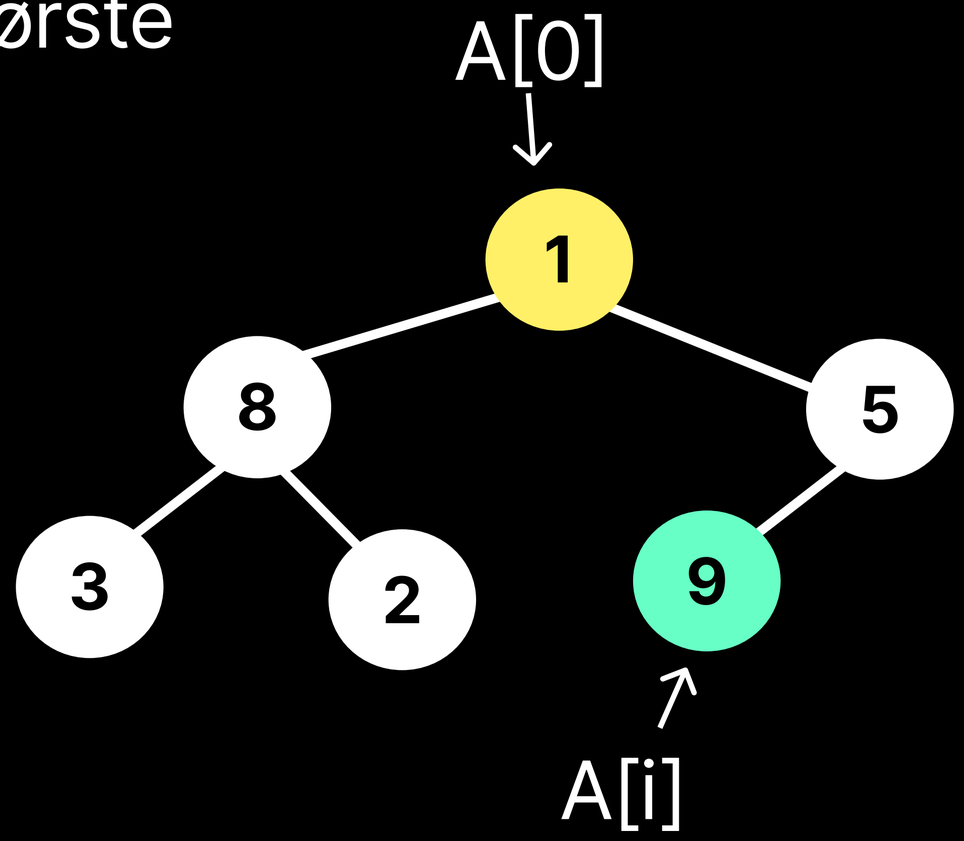
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```

HeapSort

Maks-heap

1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down





ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```

HeapSort

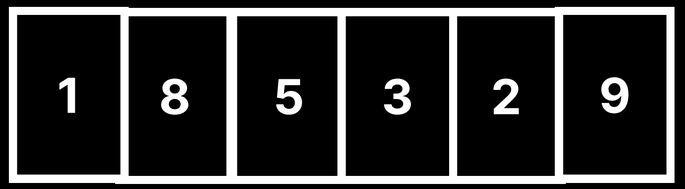
Maks-heap

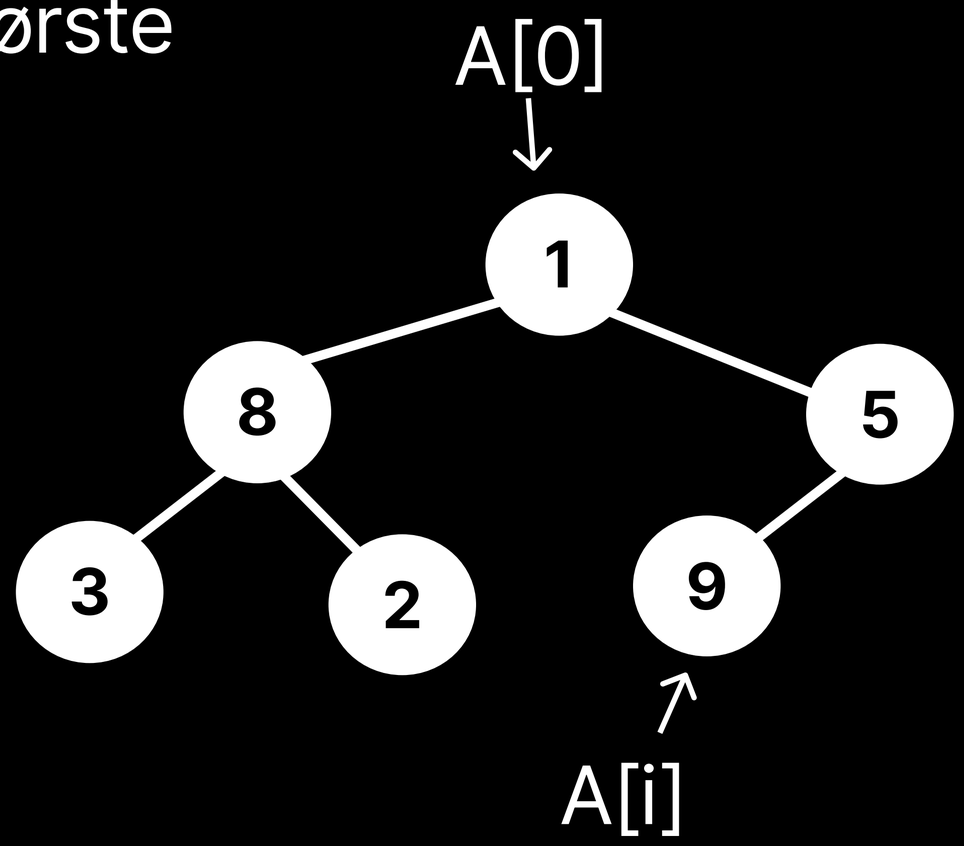
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down

ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```





HeapSort

Maks-heap

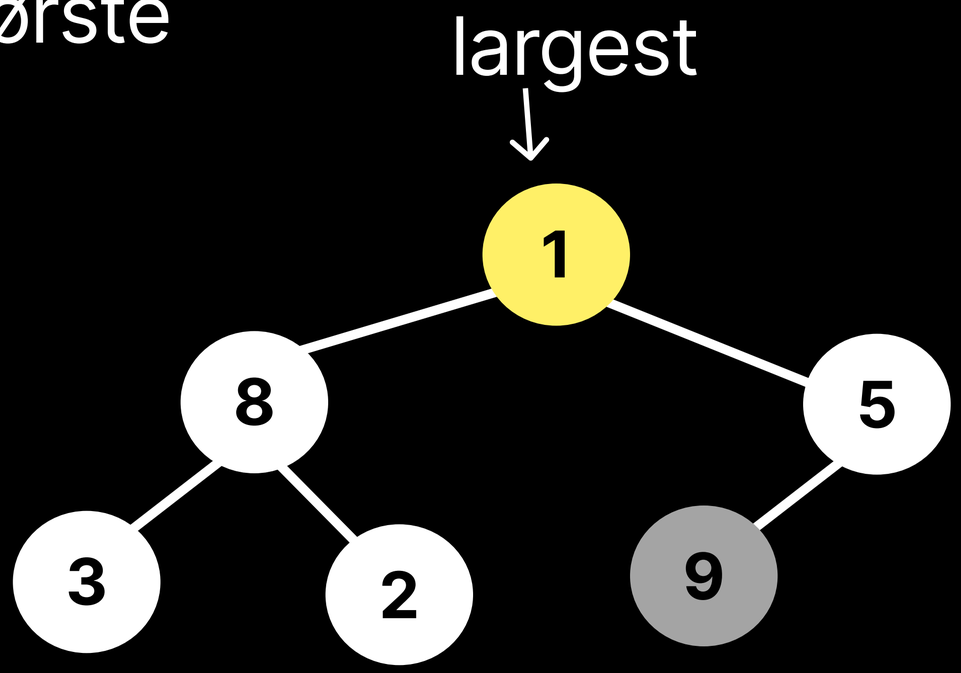
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down

ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

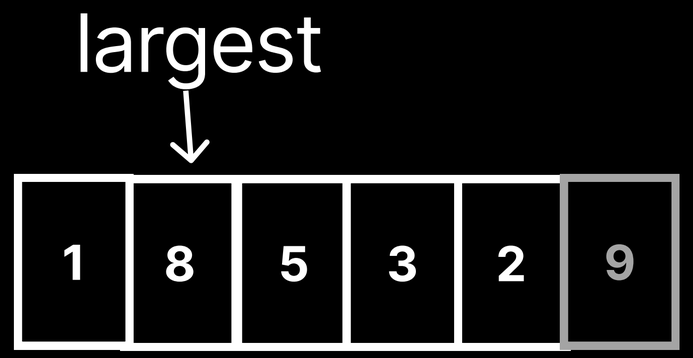




HeapSort

Maks-heap

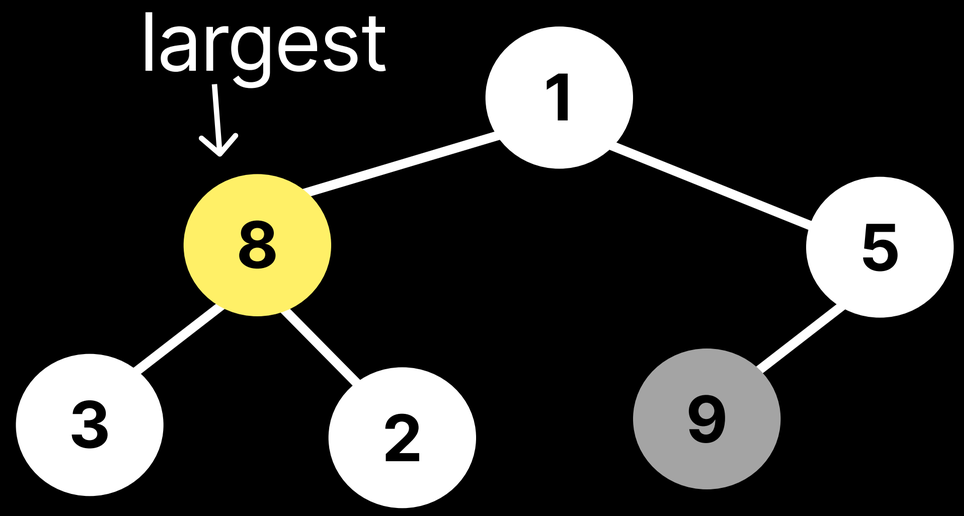
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

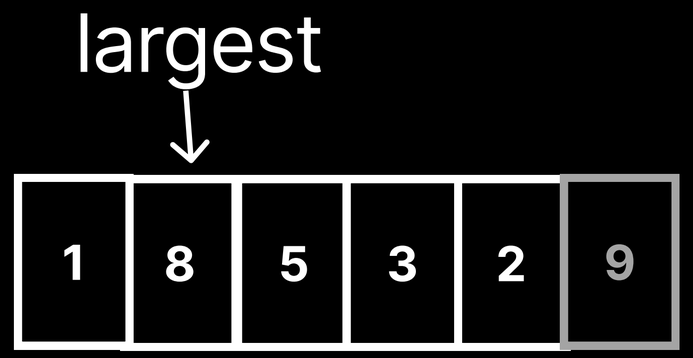
```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

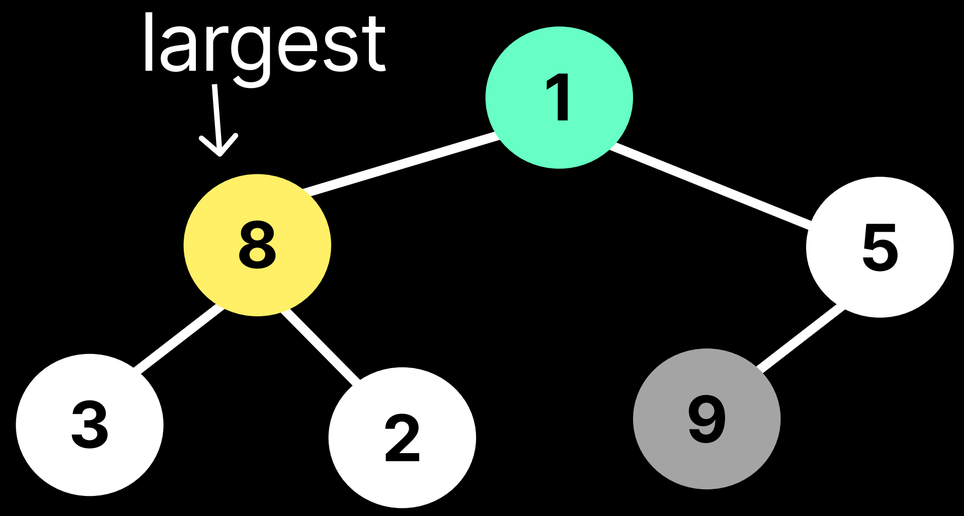


HeapSort

Maks-heap

1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down





ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

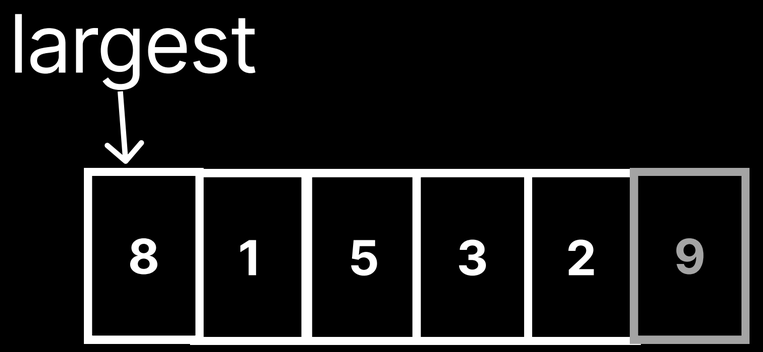
Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

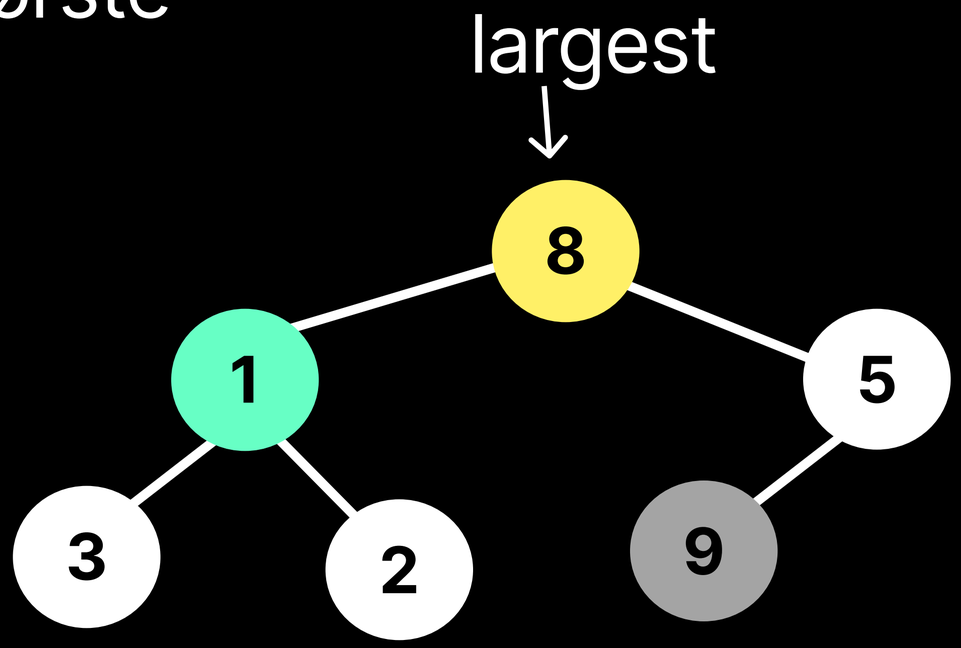
```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

HeapSort

Maks-heap

1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down





ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

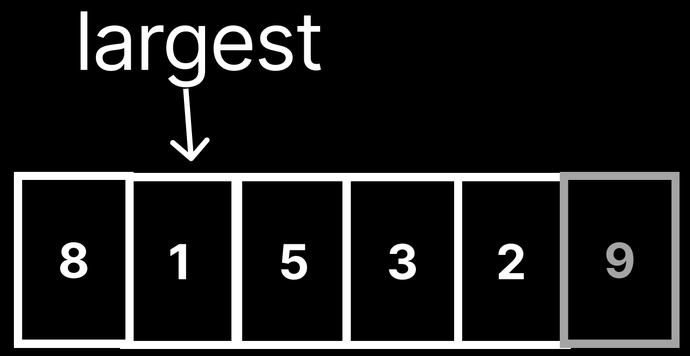
Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

HeapSort

Maks-heap

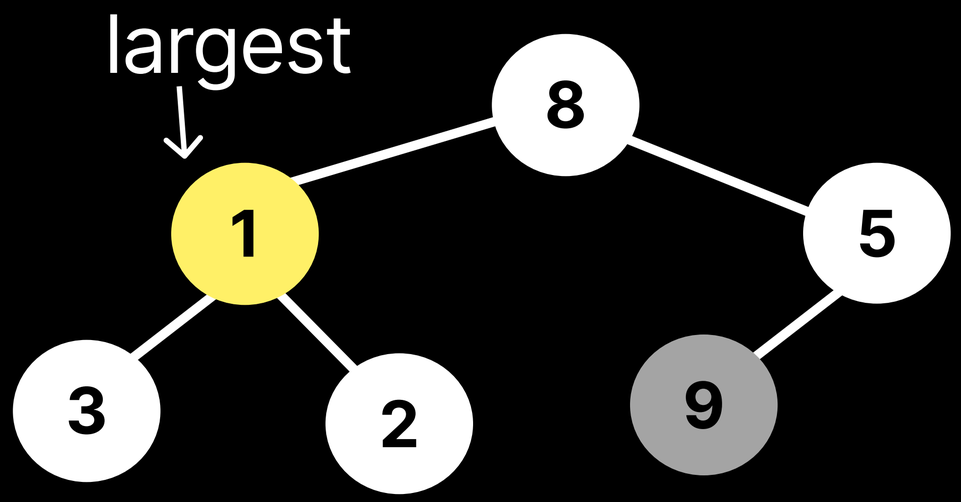
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

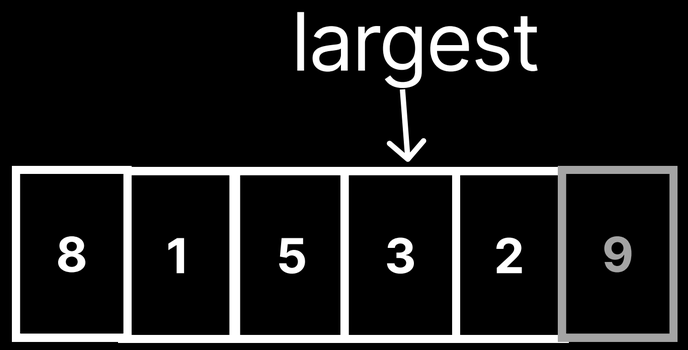
```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

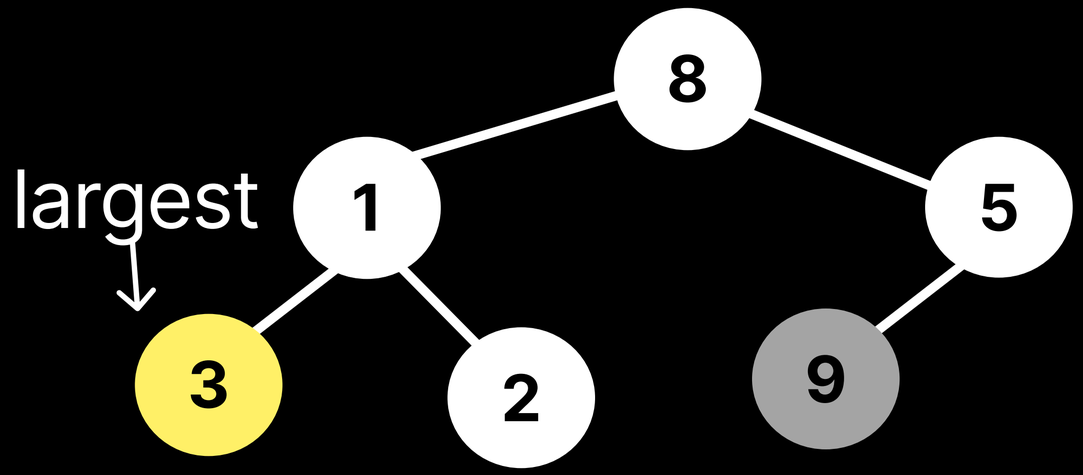
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

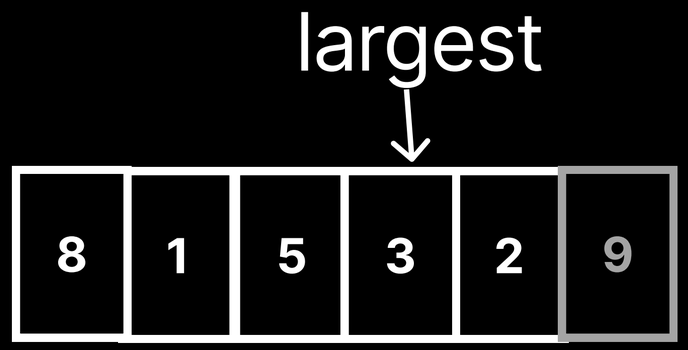
```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

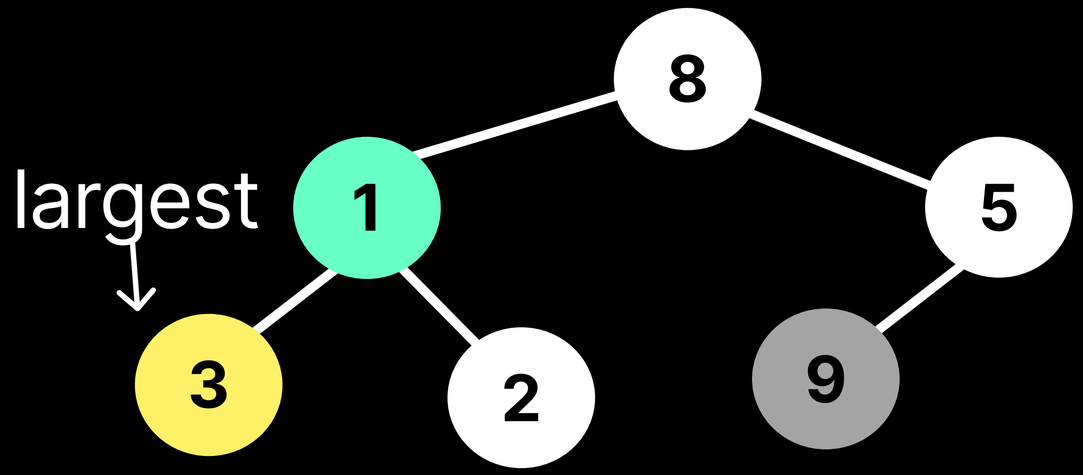
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

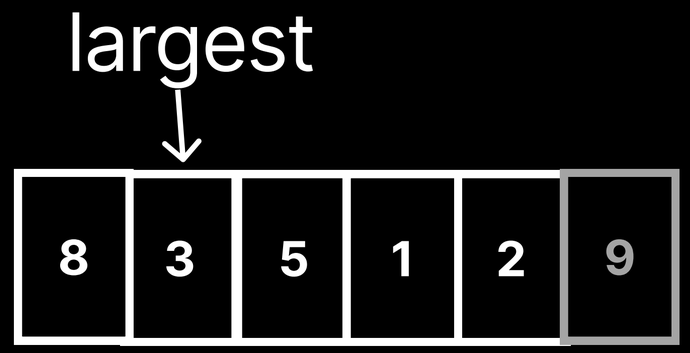
```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```



HeapSort

Maks-heap

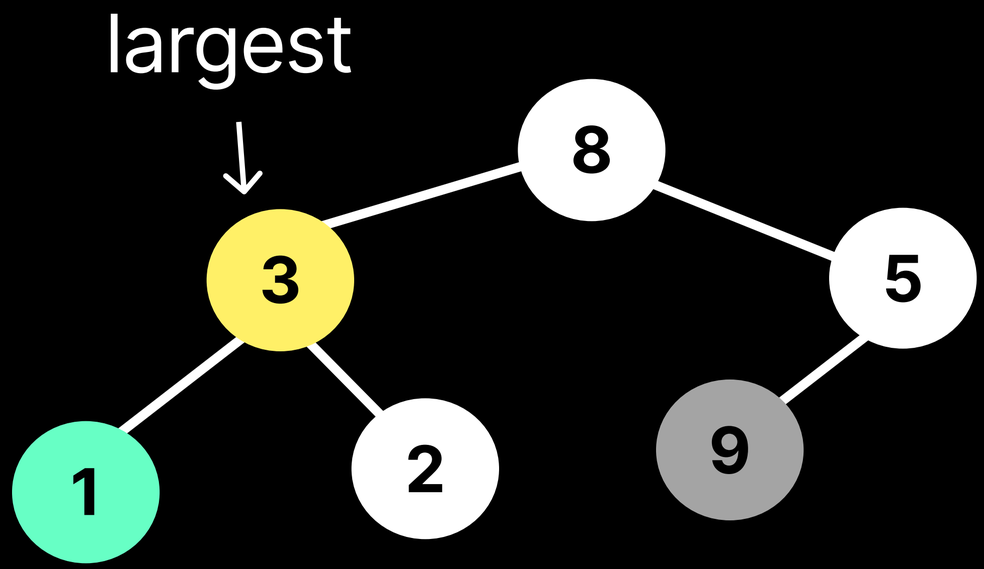
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HJELPEPROSEDYRE FOR Å BYGGE EN MAX-HEAP

Input: En (uferdig) heap A med n elementer der i er roten
Output: En mindre uferdig heap

```
1  Procedure BubbleDown(A, i, n)
2      largest ← i
3      left ← 2i + 1
4      right ← 2i + 2
5
6      if left < n and A[largest] < A[left] then
7          largest ← left
8
9      if right < n and A[largest] < A[right] then
10         largest ← right
11
12     if i ≠ largest then
13         A[i], A[largest] ← A[largest], A[i]
14         BubbleDown(A, largest, n)
```

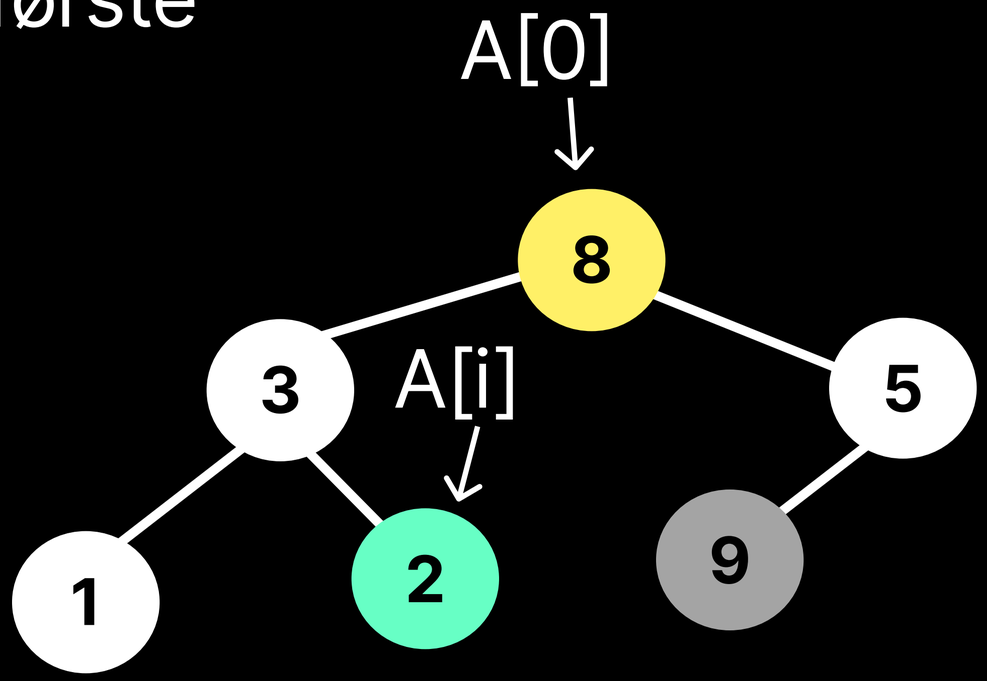


HeapSort

Maks-heap

1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down





ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

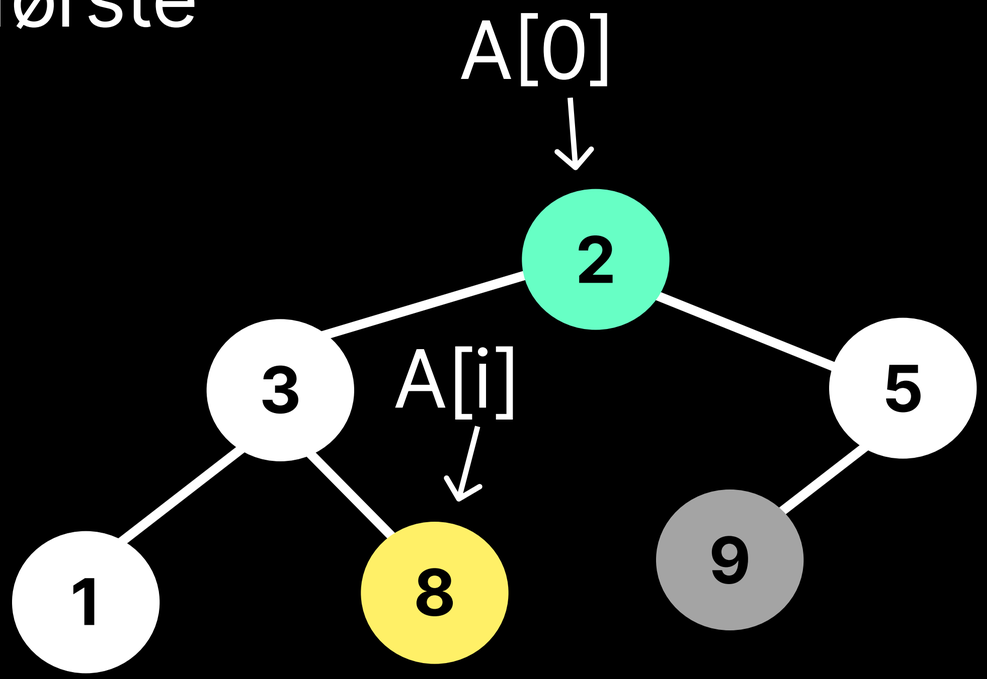
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```

HeapSort

Maks-heap

1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down





ALGORITHM: HEAPSORT

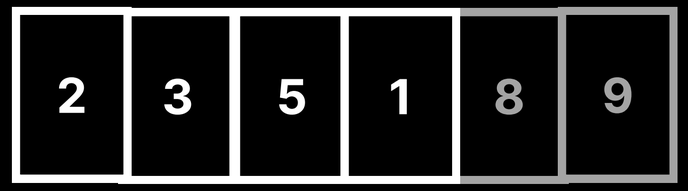
Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```

HeapSort

Maks-heap

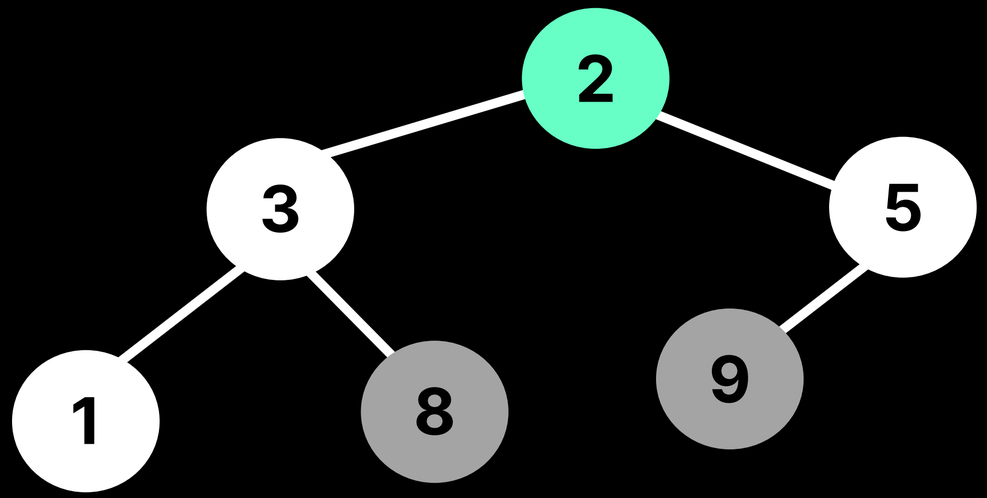
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

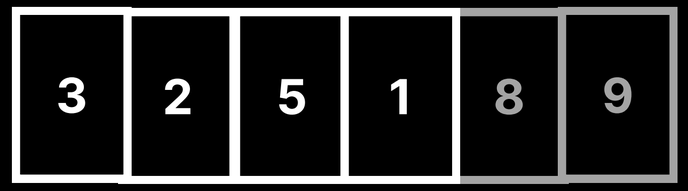
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



HeapSort

Maks-heap

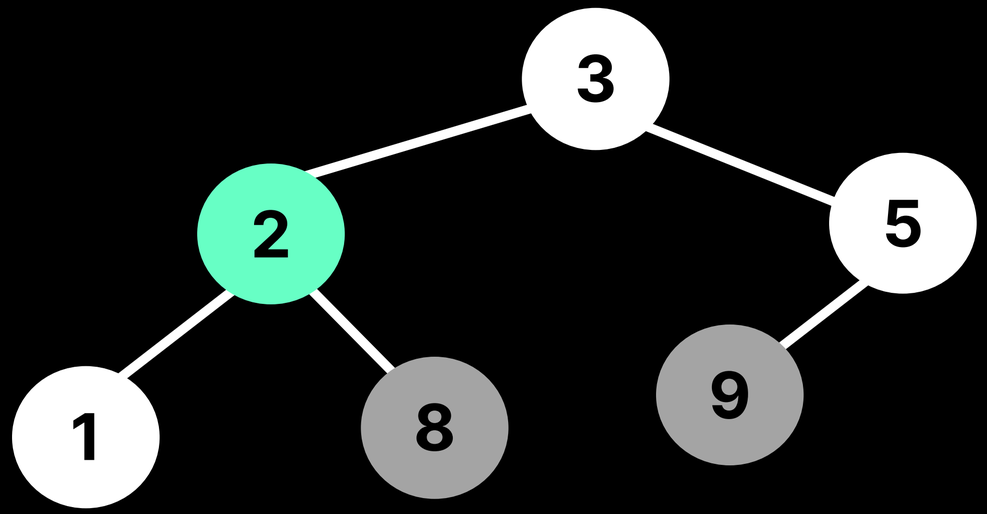
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

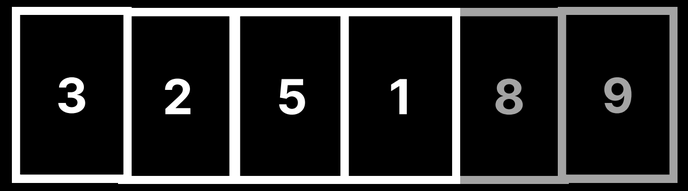
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



HeapSort

Maks-heap

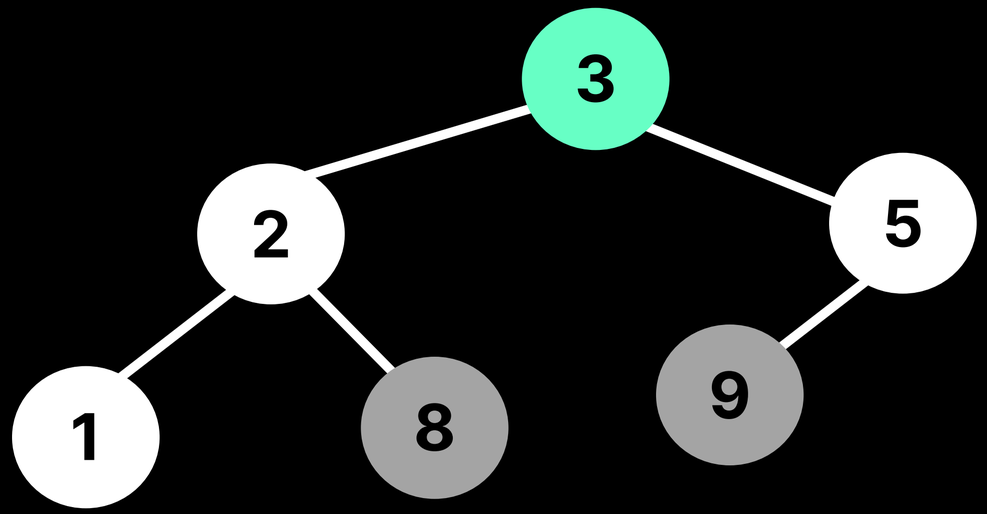
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

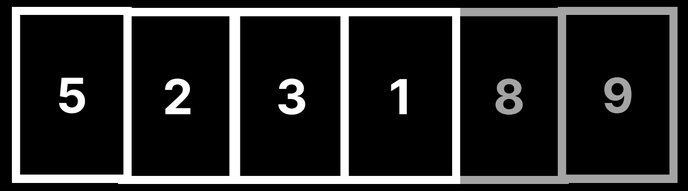
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



HeapSort

Maks-heap

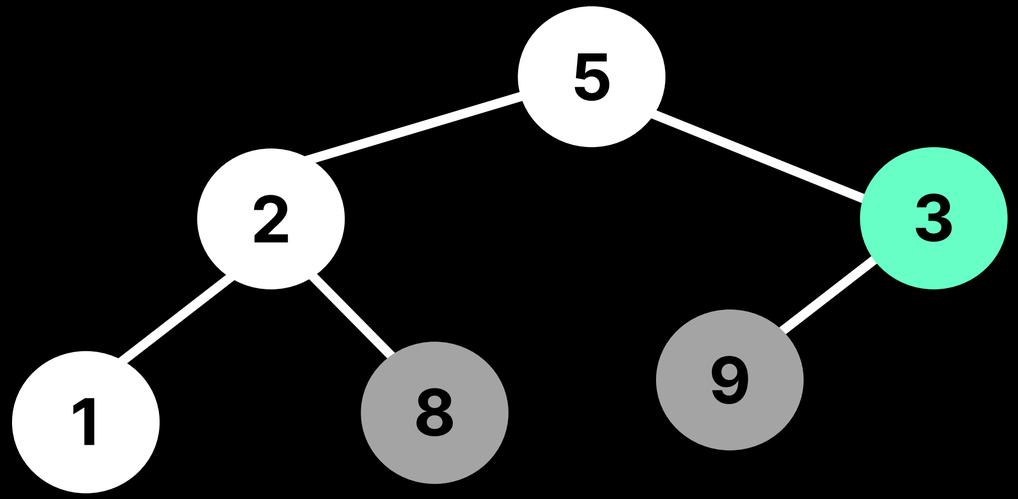
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

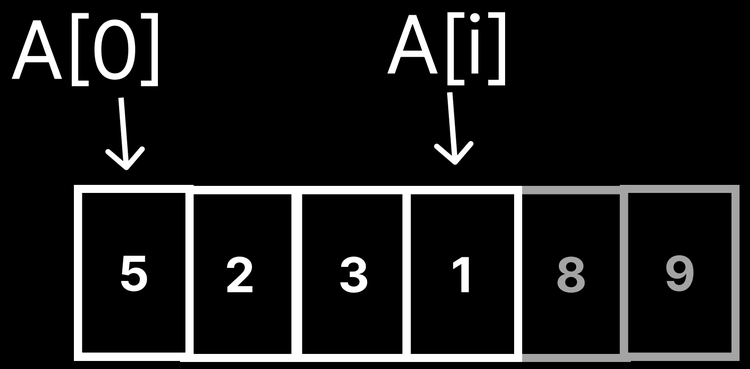
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



HeapSort

Maks-heap

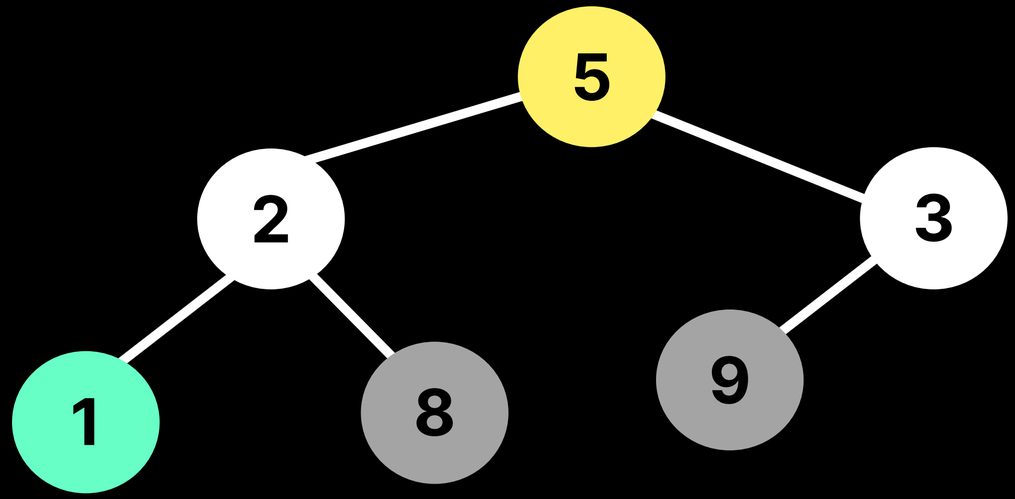
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

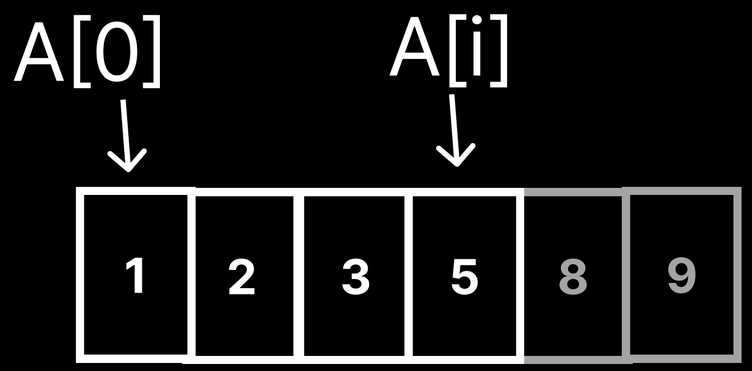
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



HeapSort

Maks-heap

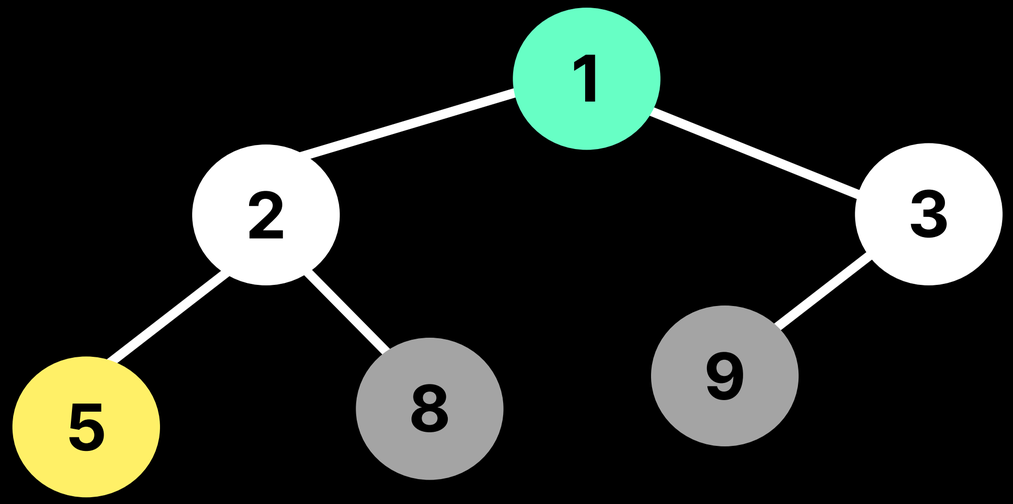
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

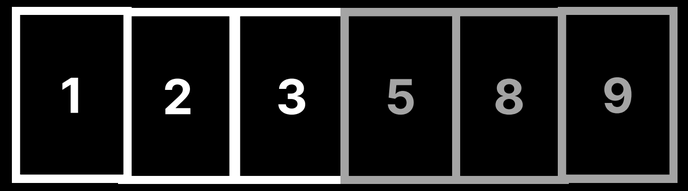
```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



HeapSort

Maks-heap

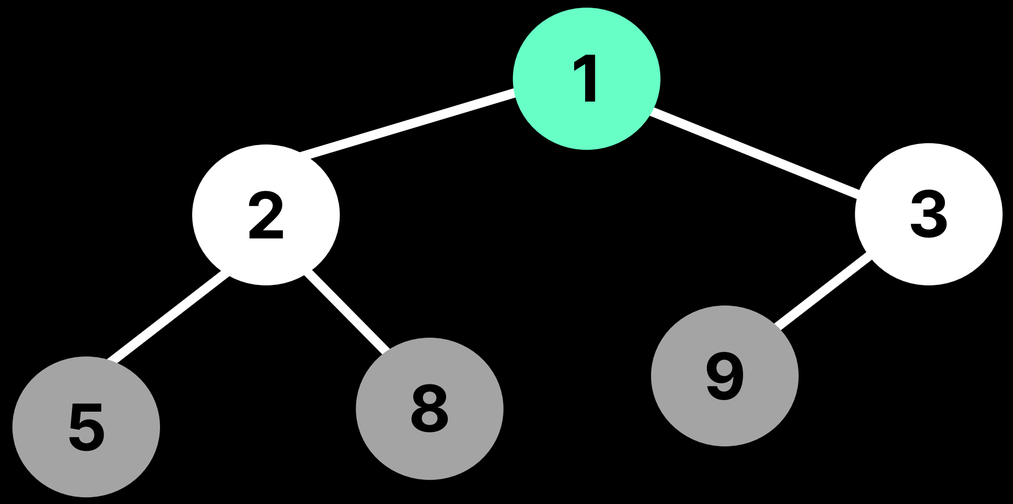
1. Lag en maks-heap ✅
2. Bytt plass mellom i og første
3. Bubble down



ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



HeapSort ✅

Maks-heap

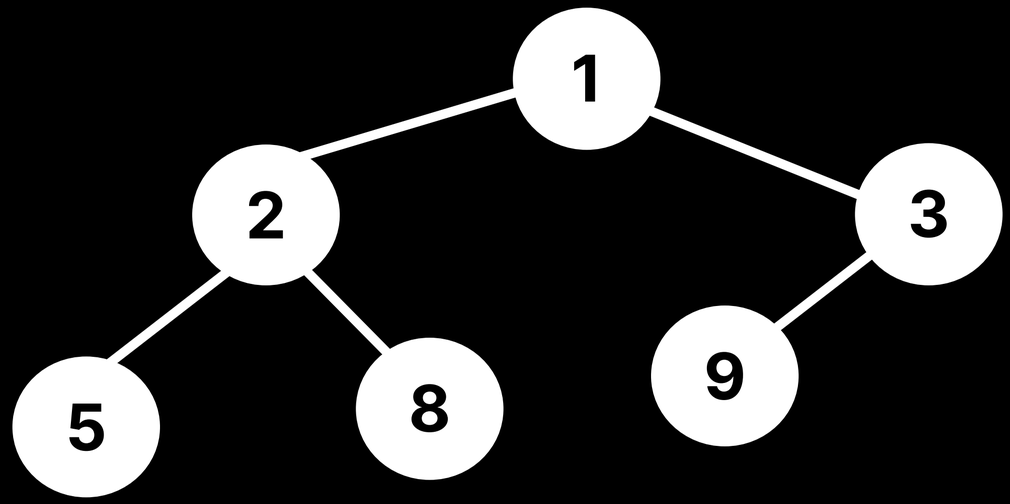
1. Lag en maks-heap
2. Bytt plass mellom i og første
3. Bubble down

1	2	3	5	8	9

ALGORITHM: HEAPSORT

Input: Et array A med n elementer
Output: Et *sortert* array med de samme n elementene

```
1 Procedure HeapSort(A)
2     BuildMaxHeap(A, n)
3     for i ← n − 1 down to 0 do
4         A[0], A[i] ← A[i], A[0]
5         BubbleDown(A, 0, i)
6     return A
```



Huffman-koding

1. Ta en setning
2. Lag en frekvenstabell
3. Bygg et Huffman-tre
    a. Høyest frekvens øverst i treet
    b. Lavest frekens nederst

Huffman-koding

1. Ta en setning

Lina er verdens beste gruppelærer

Huffman-koding

2. Lag en frekvenstabell

lina er verdens beste gruppelærer!

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

lina er verdens beste gruppelærer!

Huffman-koding

2. Bygg et Huffman-tre

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

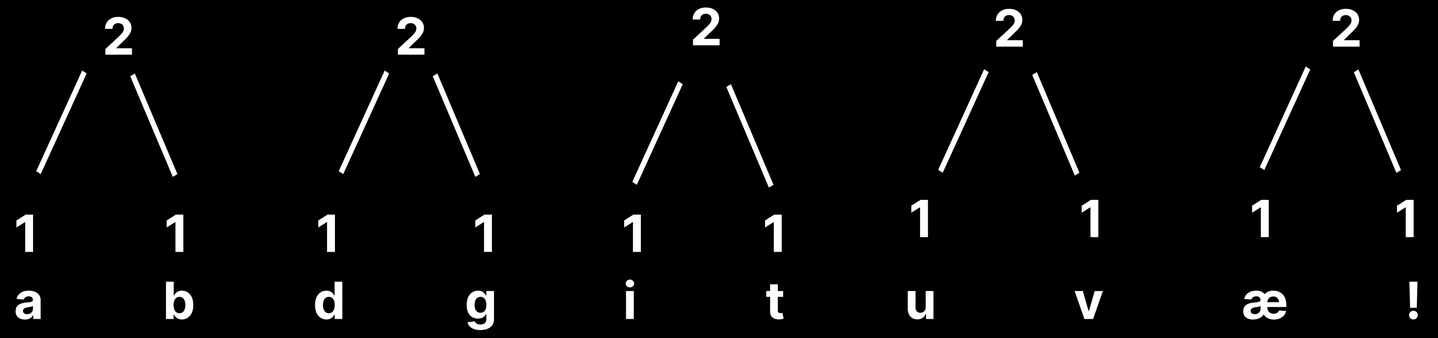
1	1	1	1	1	1	1	1	1	1
a	b	d	g	i	t	u	v	æ	!

lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

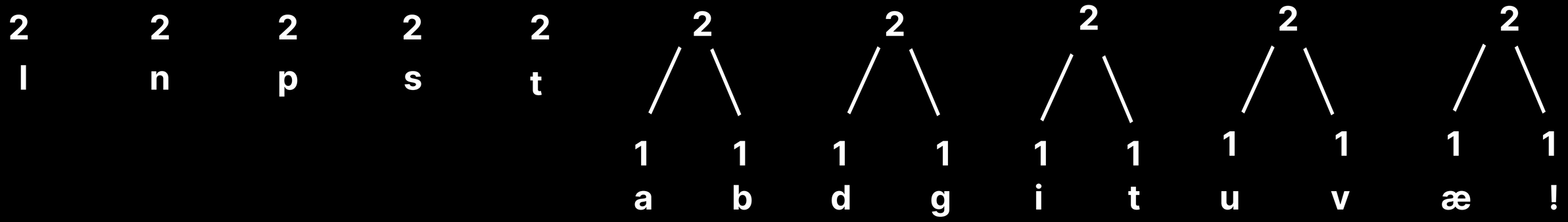


lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

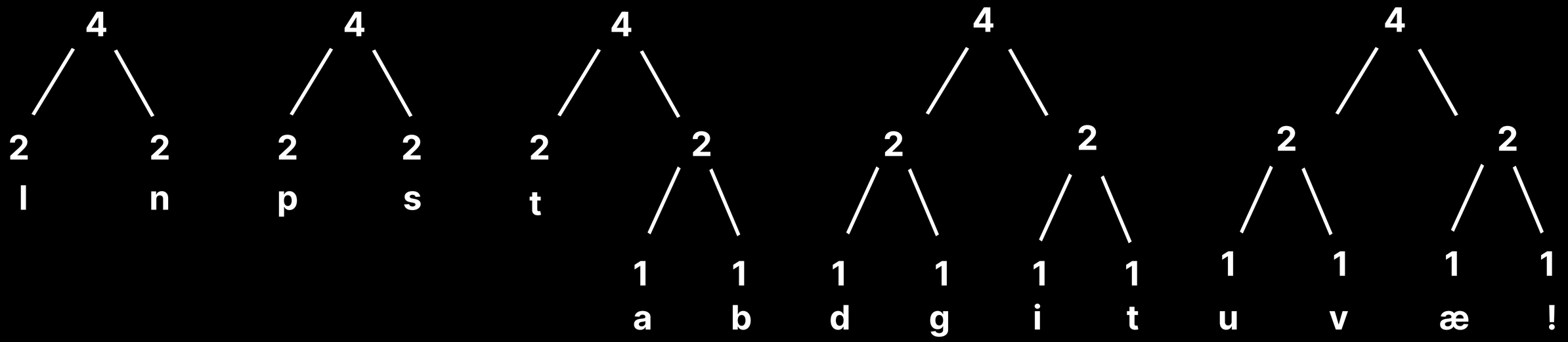


lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

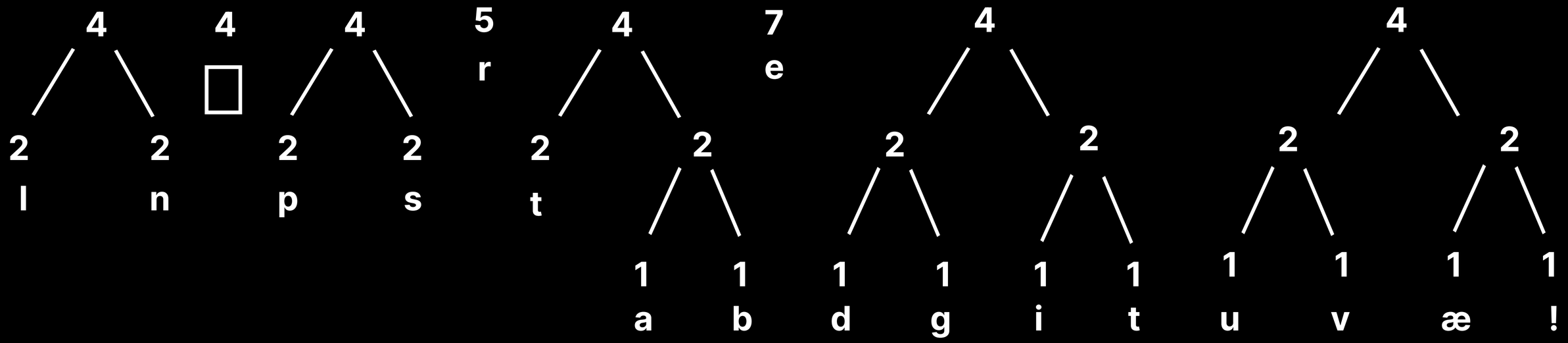


lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

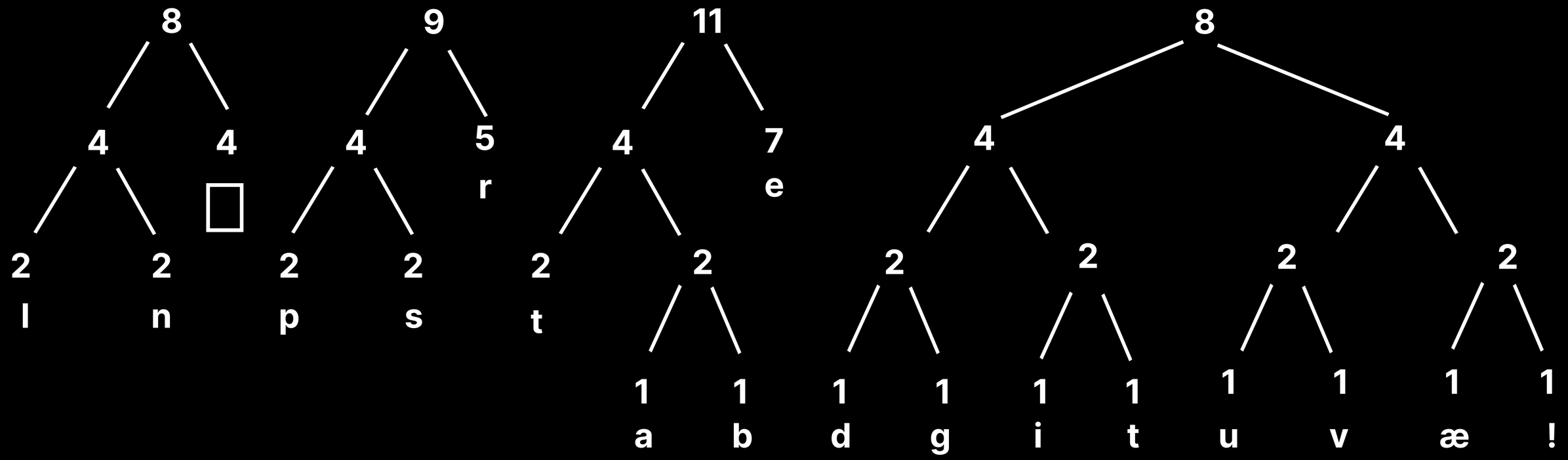


lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

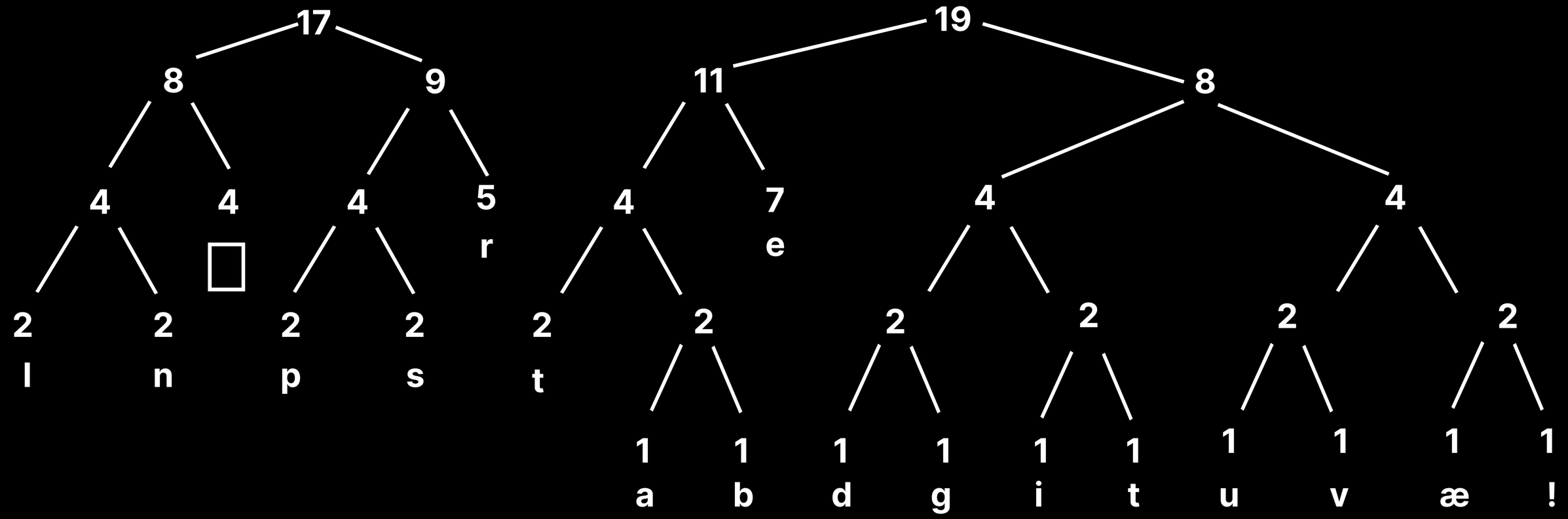


lina er verdens beste gruppelærer!

Huffman-koding

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1

2. Bygg et Huffman-tre
    a. start nederst - lavest frekvenst

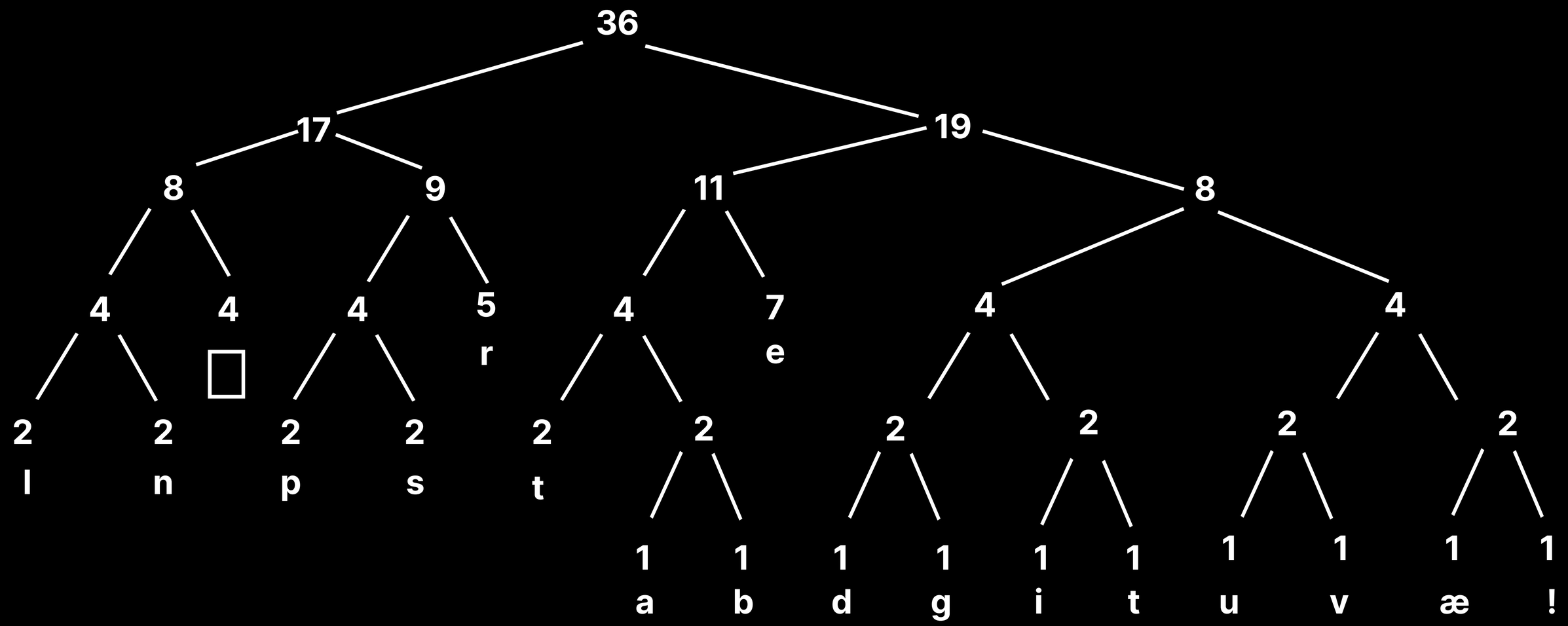


lina er verdens beste gruppelærer!

Huffman-koding

2. Bygg et Huffman-tre

Symbol		a	b	d	e	g	i	l	n	p	r	s	t	u	v	æ	!
Frekvens	4	1	1	1	7	1	1	2	2	2	5	2	1	1	1	1	1



Rød-svarte trær

Rød-svarte trær

Balanserte binære søketrær

Rød-svarte trær

Balanserte binære søketrær

Hver node er rød eller svart

Rød-svarte trær

Balanserte binære søketrær

Hver node er rød eller svart

Roten er svart

Rød-svarte trær

Balanserte binære søketrær

Hver node er rød eller svart

Roten er svart

Røde noder kan ikke ha røde barn

Rød-svarte trær

Balanserte binære søketrær

Hver node er rød eller svart

Roten er svart

Røde noder kan ikke ha røde barn

Hver gren fra rot til tomt tre har like mange svarte noder

Rød-svarte trær

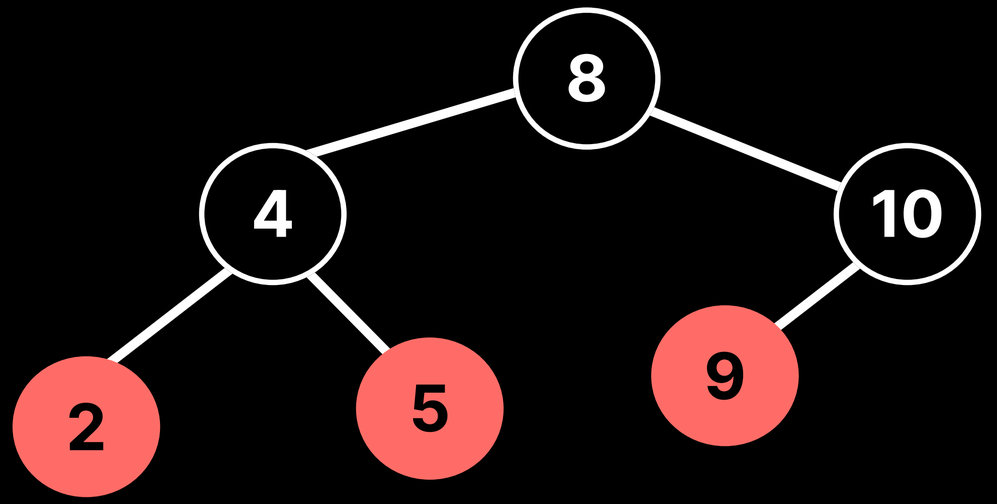
Balanserte binære søketrær

Hver node er rød eller svart

Roten er svart

Røde noder kan ikke ha røde barn

Hver gren fra rot til tomt tre har like mange svarte noder



Er dette et rød-svart tre?

Rød-svarte trær

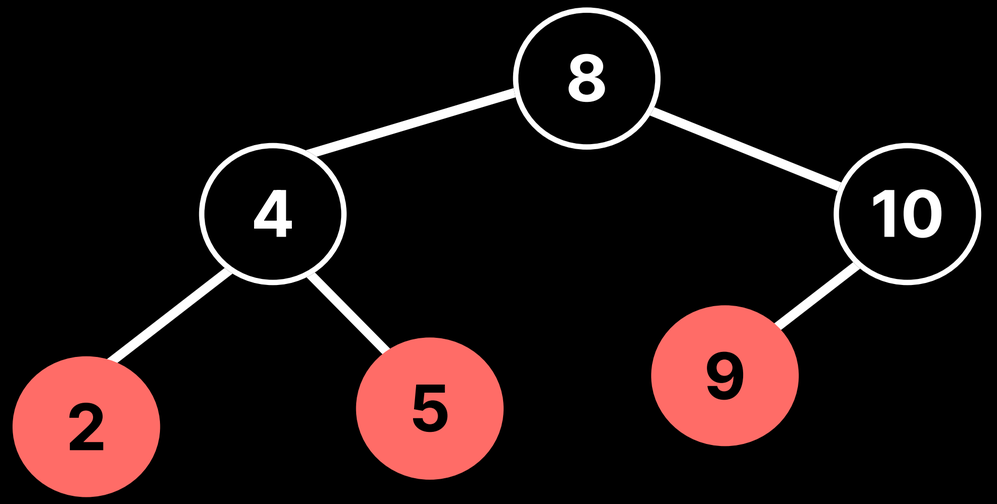
Balanserte binære søketrær

Hver node er rød eller svart

Roten er svart

Røde noder kan ikke ha røde barn

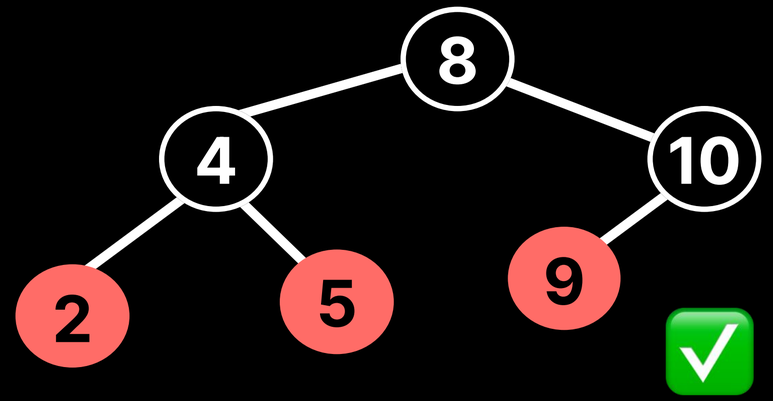
Hver gren fra rot til tomt tre har like mange svarte noder



Er dette et rød-svart tre?

Ja! ✅

Rød-svarte trær



Balanserte binære søketrær

Hver node er rød eller svart

Roten er svart

Røde noder kan ikke ha røde barn

Hver gren fra rot til tomt tre har like mange svarte noder

Rød-svarte trær

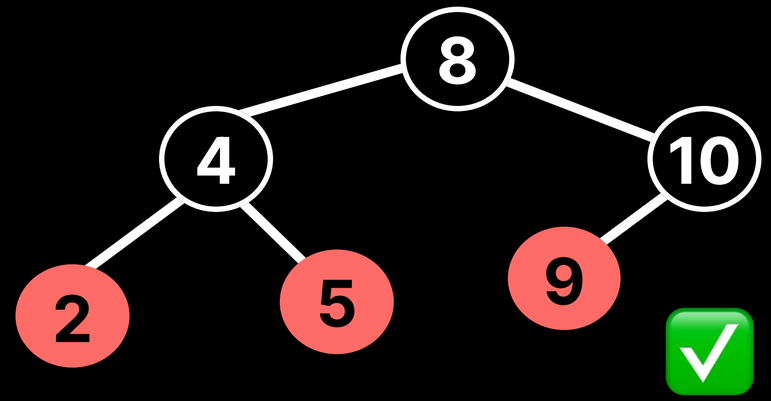
Balanserte binære søketrær

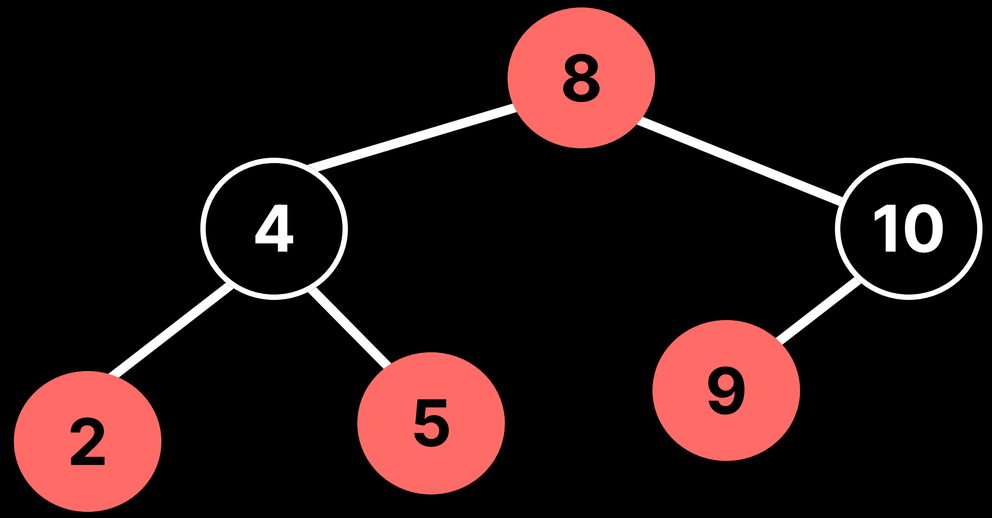
Hver node er rød eller svart

Roten er svart

Røde noder kan ikke ha røde barn

Hver gren fra rot til tomt tre har like mange svarte noder





Er dette et rød-svart tre?

Rød-svarte trær

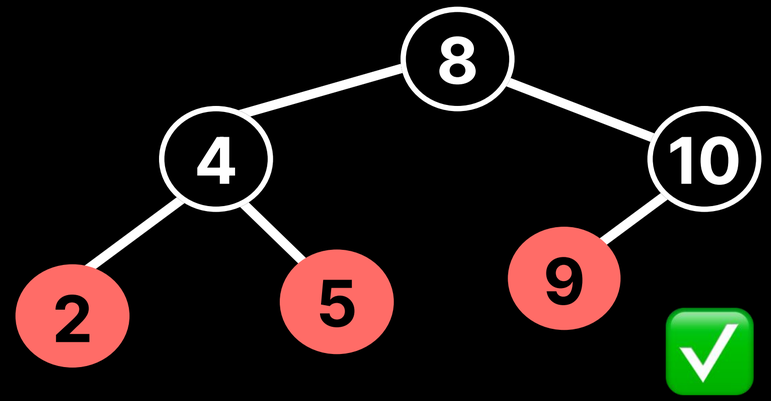
Balanserte binære søketrær

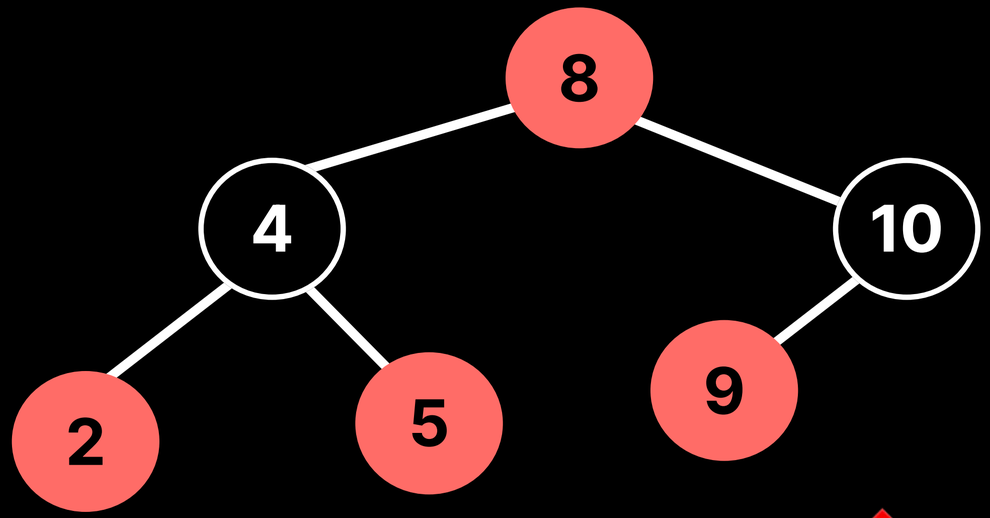
Hver node er rød eller svart

Roten er svart

Røde noder kan ikke ha røde barn

Hver gren fra rot til tomt tre har like mange svarte noder





Er dette et rød-svart tre?

Roten er ikke svart

Rød-svarte trær

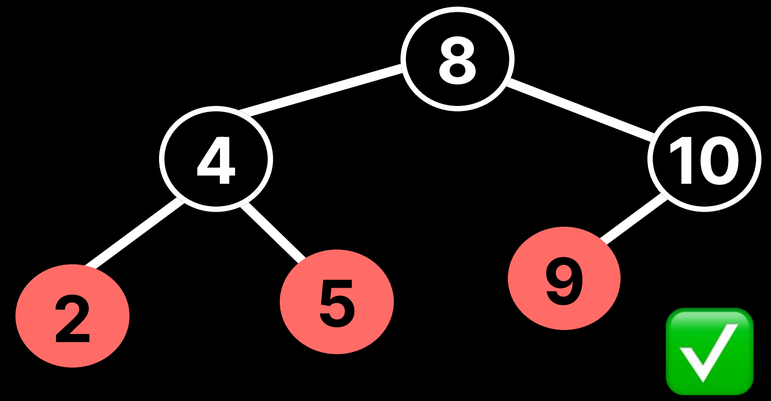
Balanserte binære søketrær

Hver node er rød eller svart

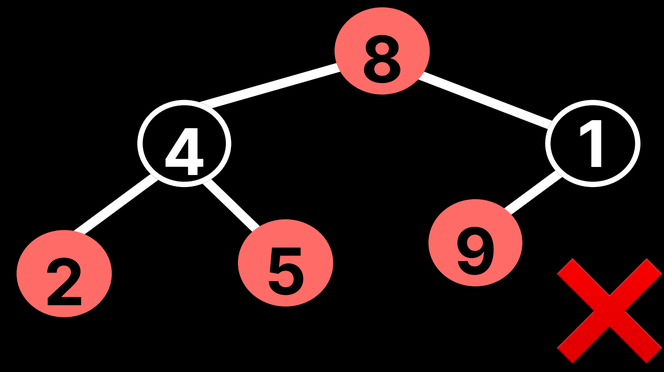
Roten er svart

Røde noder kan ikke ha røde barn

Hver gren fra rot til tomt tre har like mange svarte noder



✅



❌

Rød-svarte trær

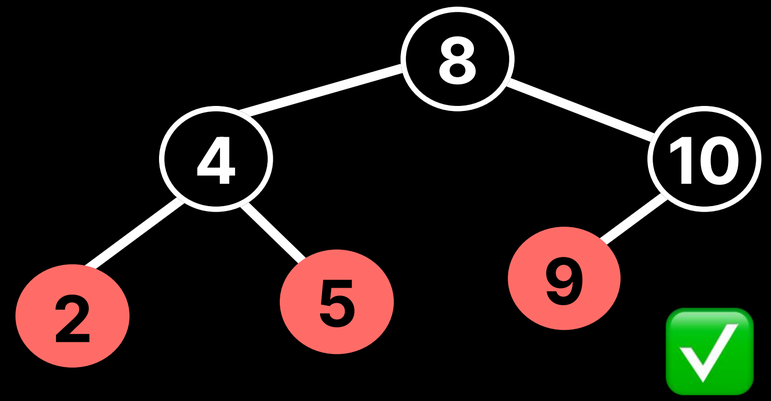
Balanserte binære søketrær

Hver node er rød eller svart

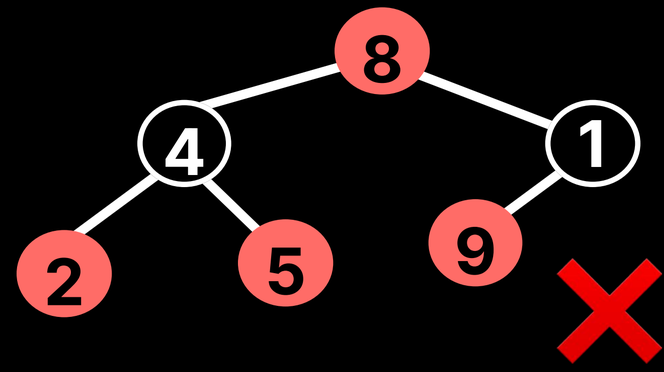
Roten er svart

Røde noder kan ikke ha røde barn

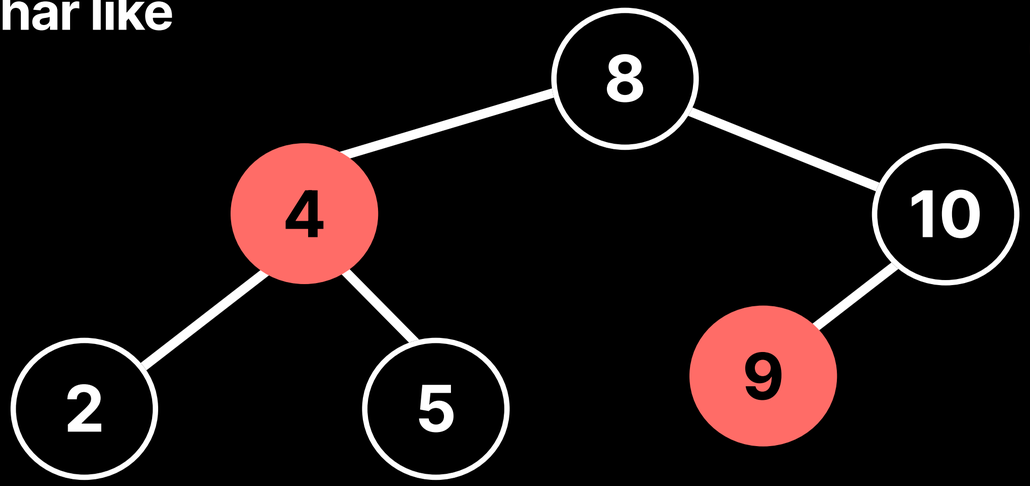
Hver gren fra rot til tomt tre har like mange svarte noder



✅



❌



Er dette et rød-svart tre?

Rød-svarte trær

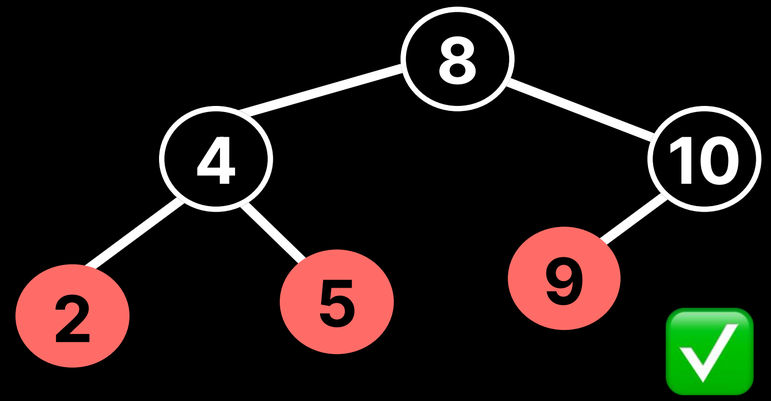
Balanserte binære søketrær

Hver node er rød eller svart

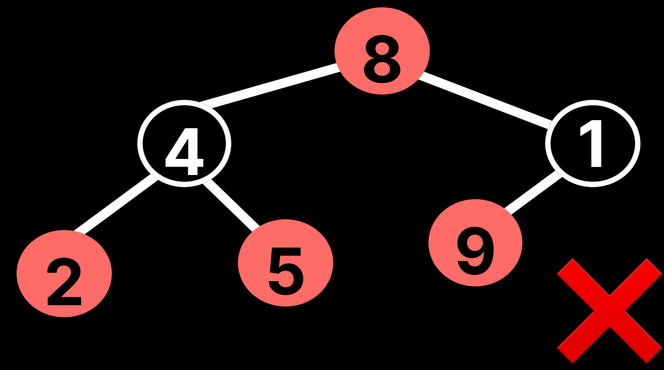
Roten er svart

Røde noder kan ikke ha røde barn

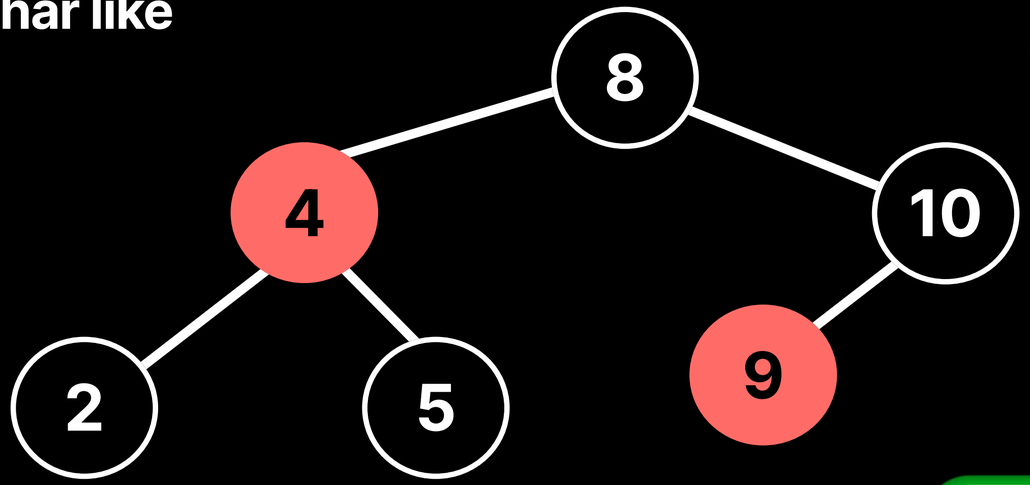
Hver gren fra rot til tomt tre har like mange svarte noder



✅



❌



Er dette et rød-svart tre?

Ja!

✅