# F06 - Sortering

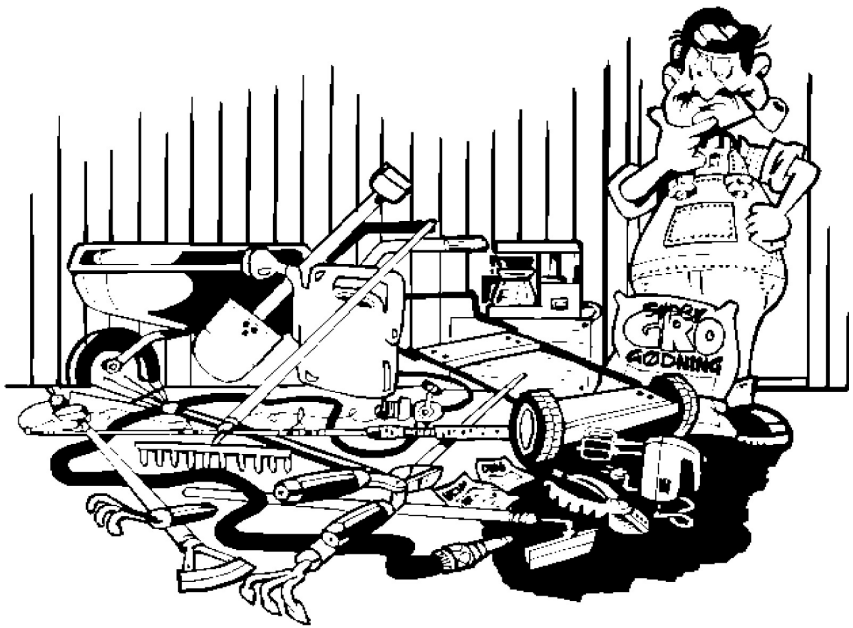
## 5DV149 Datastrukturer och algoritmer
## Kapitel 15

Niclas Börlin
niclas.borlin@cs.umu.se

2020-02-06 Thu

# Sortering

- Snabba upp andra algoritmer.
  - Sökning.
  - Hantera stora datamängder.



# Sortering kontra Sorterad datatyp

- Sortering:
  - förändrar ordningen mellan objekten i en struktur efter en sorteringsordning (fallande, ökande).
- Sorterad datatyp:
  - elementen i datatypen hålls sorterade enligt en sorteringsordning av de strukturförändrande operationerna i gränsytan (insert, delete).

# Saker att beakta

- Absolut komplexitet:
  - Totala komplexiteten i alla implementationssteg
    - Kö (konstruerad som) lista (implementerad som) dubbellänkad lista.
- Passar en viss typ av sortering för en viss typ av implementation?
  - Fält-baserad lista
  - Länkad lista
- Ska man:
  1. sortera listan (tar tid) och sedan söka (går fortare)
  2. behålla listan osorterad (sparar tid) och göra linjär sökning (tar längre tid)?

# Komplexitet

- Kan vara stor skillnad mellan värsta och bästa fallet.
- Fundera på:
  - Hur hanterar algoritmen en redan sorterad lista?
  - Hur hanterar algoritmen en motsatt sorterad lista?

# Stabilitet

- Den inbördes relationen mellan två objekt med samma nyckel bibehålls vid sortering:
  - Lista sorterad efter första elementvärdet:

    (0,J), (2,C), (5,G), (8,A), (10,G)

  - Lista omsorterad efter andra elementvärdet:

    (8,A), (2,C), (5,G), (10,G), (0,J)

- Alla sorteringsalgoritmer går inte att göra stabila.

# Grundprinciper

- Instickssortering:
  - Välj ett godtyckligt element och sätt in det på rätt plats. Ex: *Insertion sort*.
- Urvalssortering:
  - Välj ut det objekt som är på tur att sättas in. Sätt in det sist/först. Ex: *Selection sort*.
- Utbytessortering:
  - Byt plats på objekt som ligger fel inbördes. Ex: *Bubble sort*.
- Samsortering:
  - Sammanslagning av redan sorterade strukturer. Ex. *Merge sort*, *Inplace Quicksort*.
- (Nyckelsortering:)
  - Kräver mer information/kunskap om objektmängden.

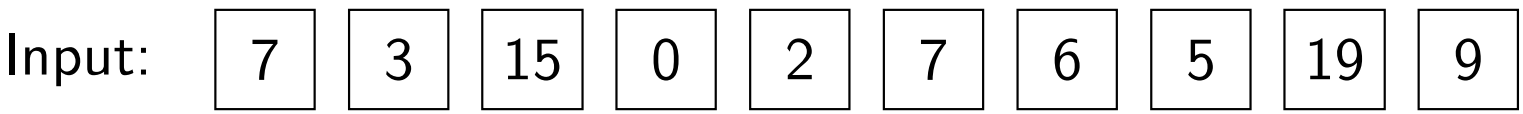
# Utbytessortering

- Utbytessorteringsalgoritmer (*inplace*-) kräver endast $O(1)$ extra minne.
- List-algoritmer kan kräva $O(n)$ extra minne.
- Flera algoritmer finns både som list-versioner och utbytesversioner.

# Sorteringsalgoritmer

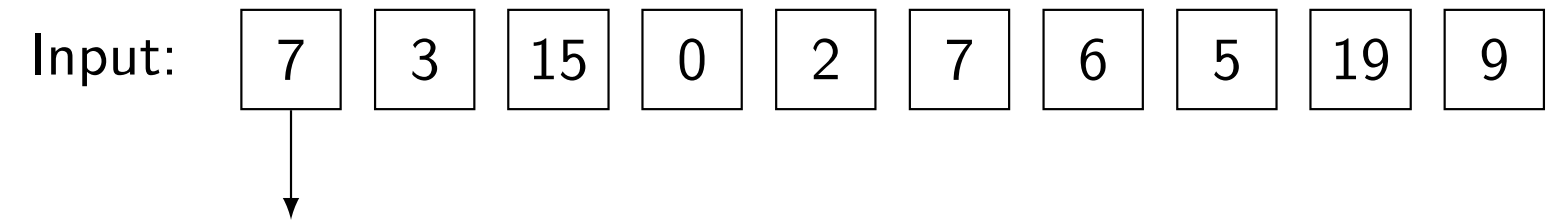
- Idag:
  - Insticksortering (*Insertion Sort*)
  - Urvalssortering (*Selection Sort*)
  - Bubbelsortering (*Bubble Sort*)
  - Mergesort
  - Quicksort
- Senare:
  - Heapsort
  - (Radix Exchange Sort)

# Instickssortering



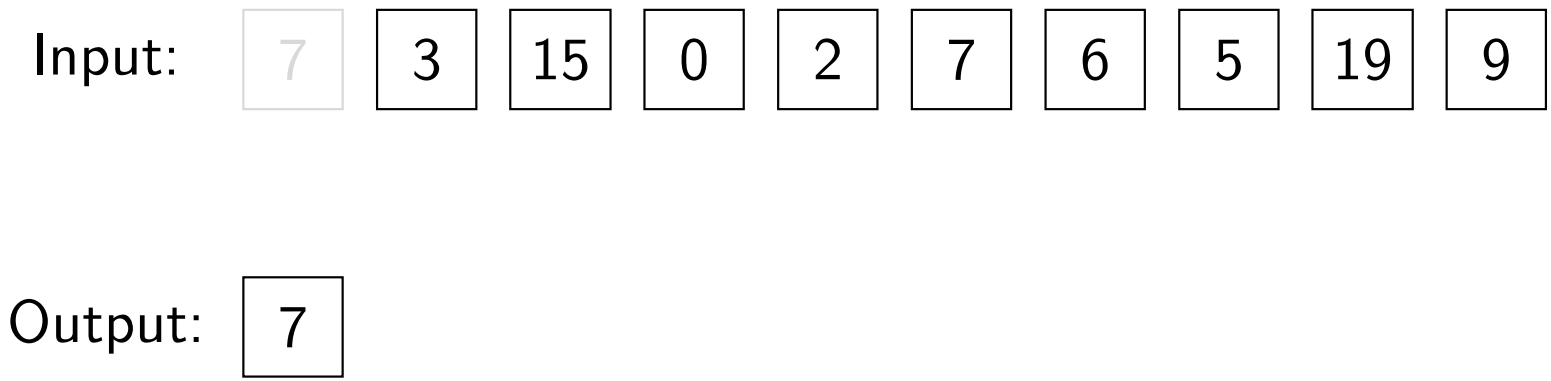

Output:

# Instickssortering

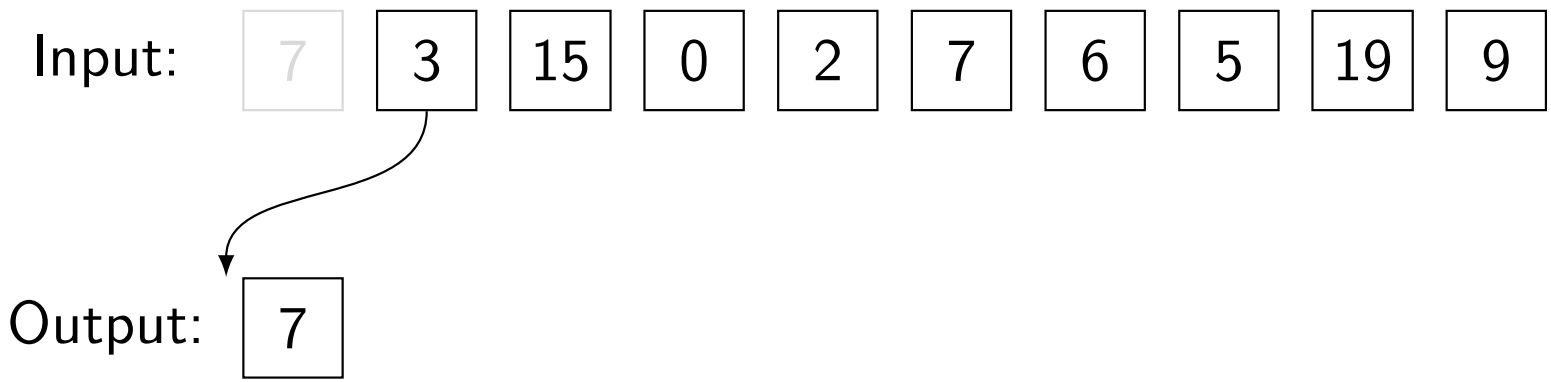



Output:

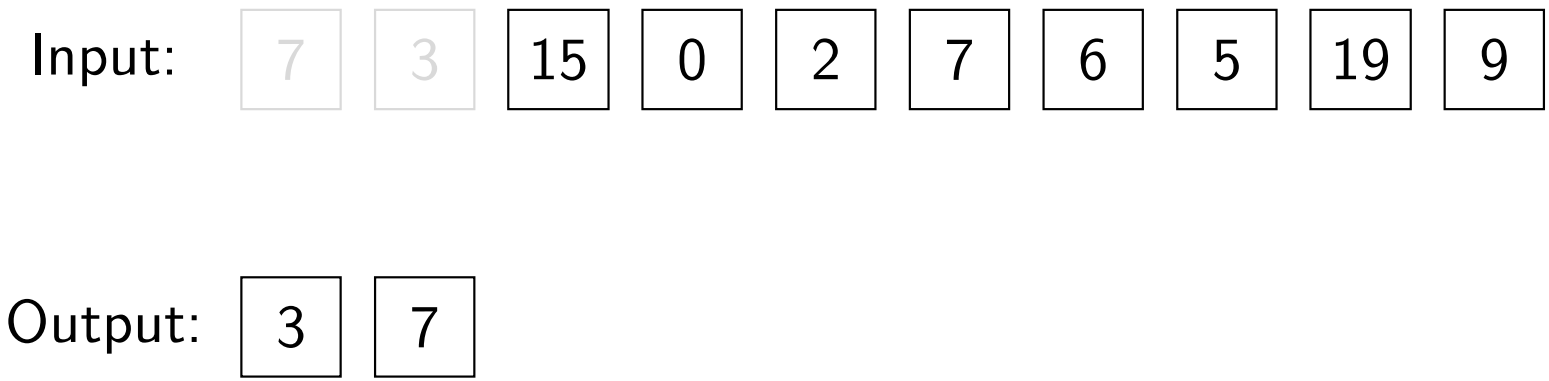
# Instickssortering



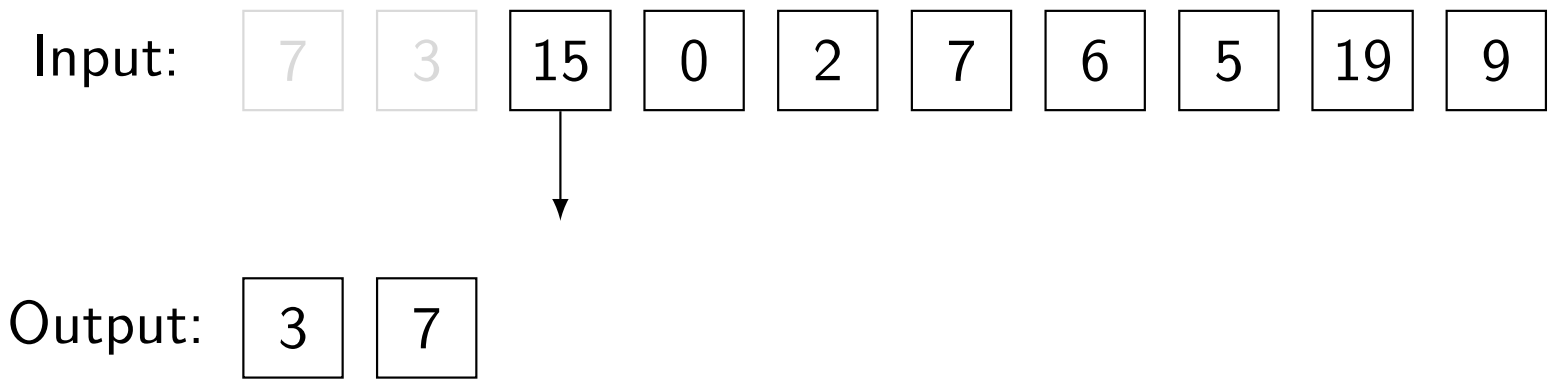
# Insticksortering



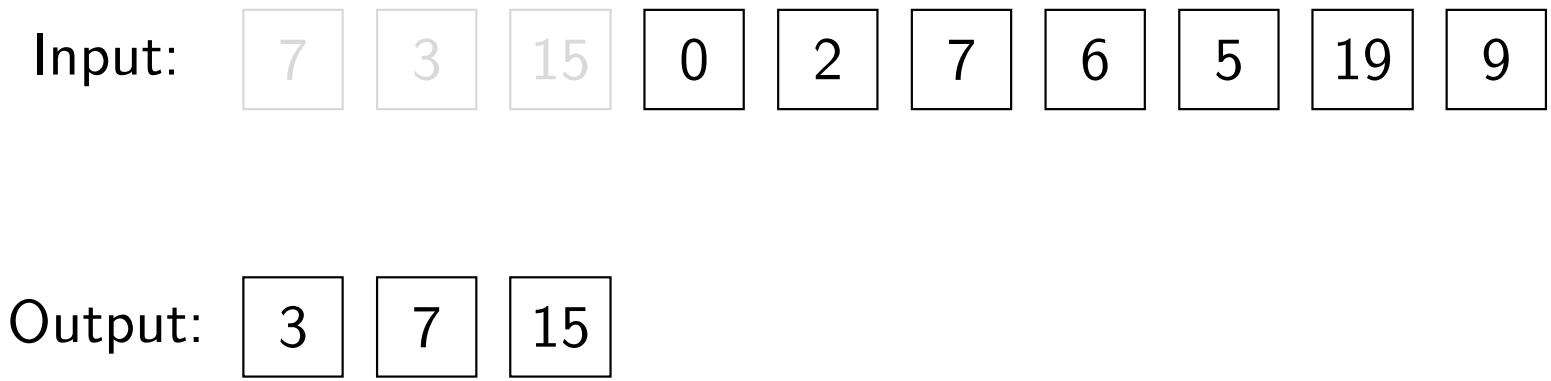
# Insticksortering



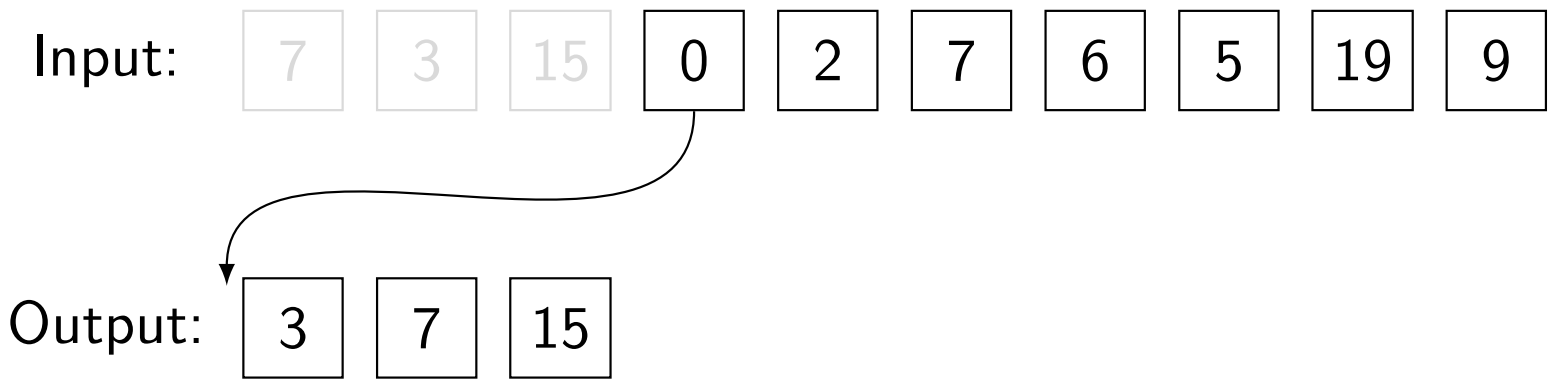
# Instickssortering



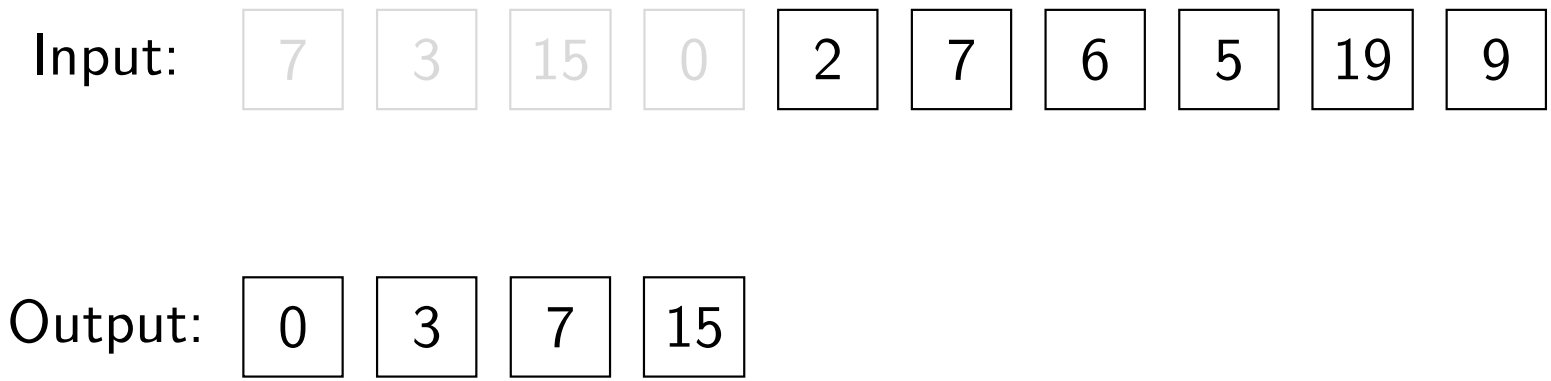
# Instickssortering



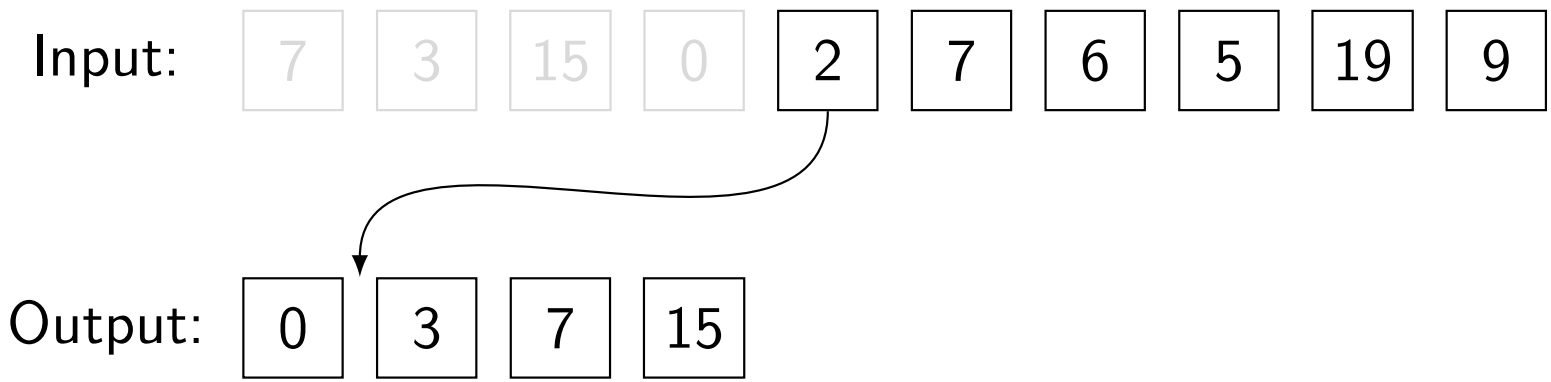
# Insticksortering



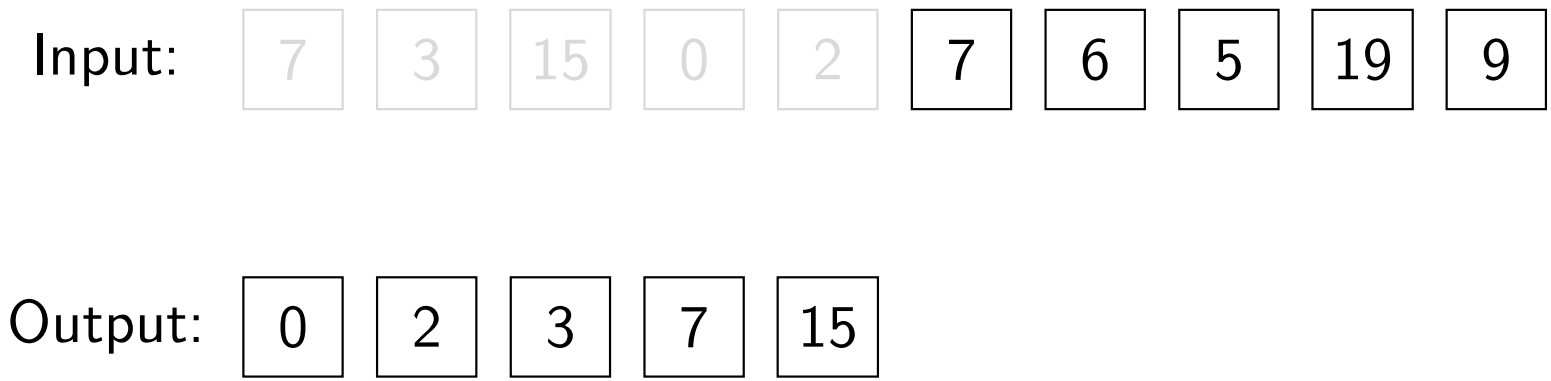
# Insticksortering



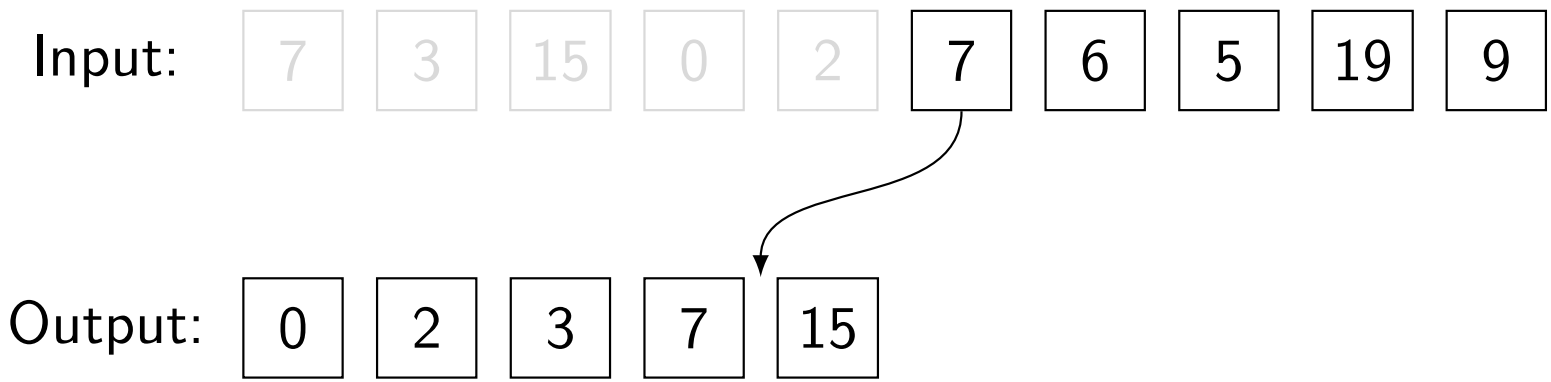
# Insticksortering



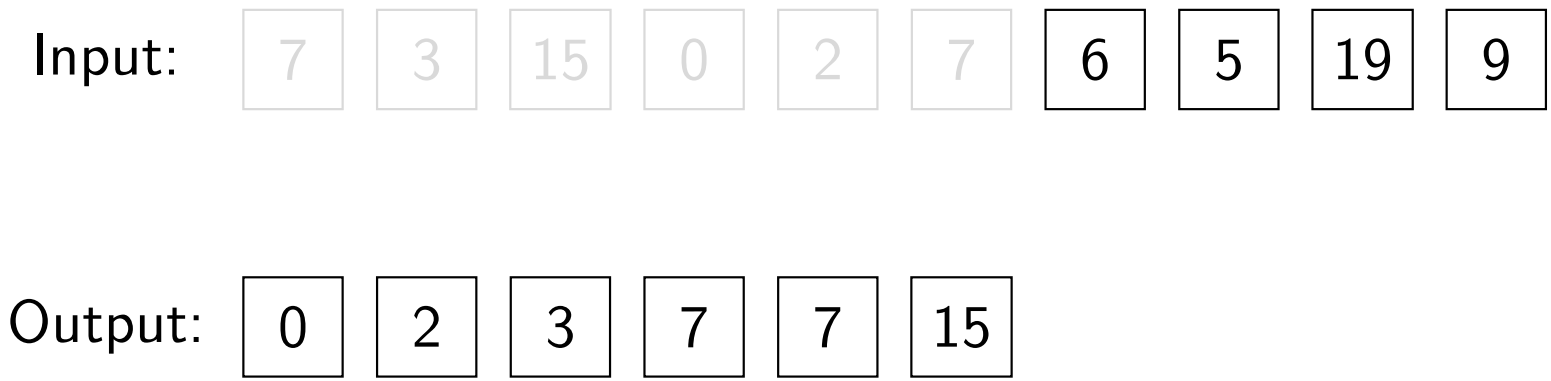
# Instickssortering



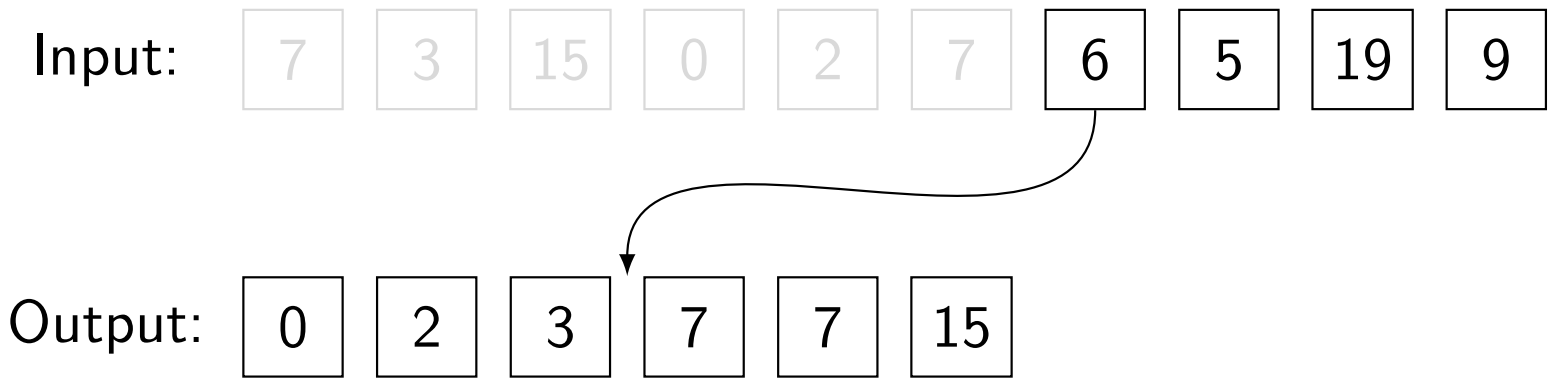
# Instickssortering



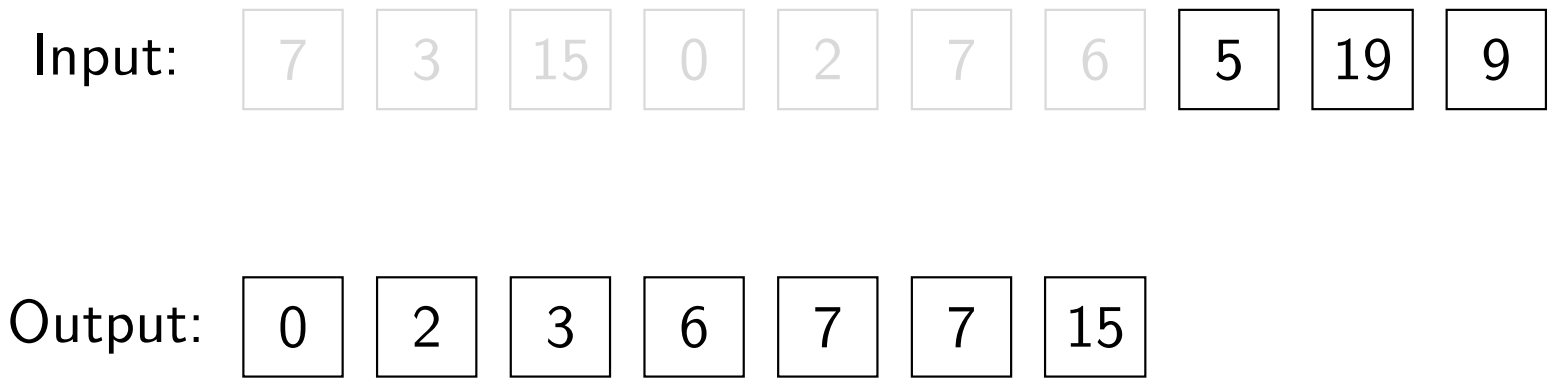
# Instickssortering



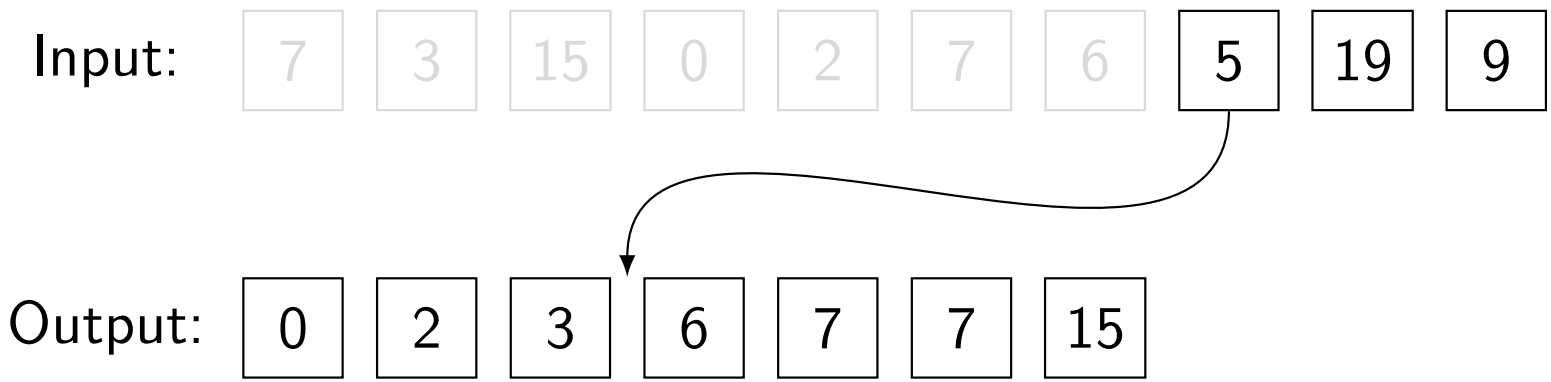
# Insticksortering



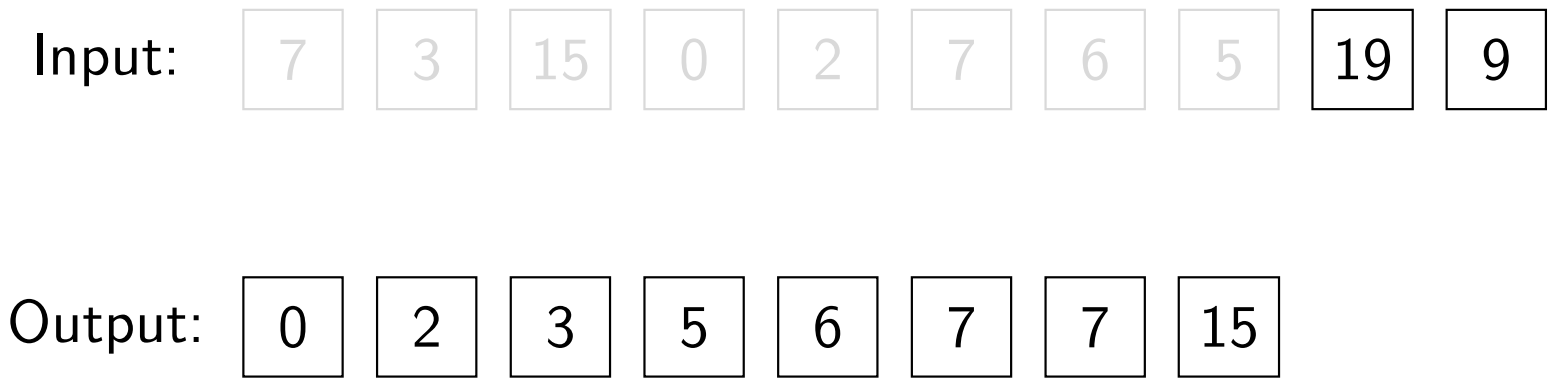
# Instickssortering



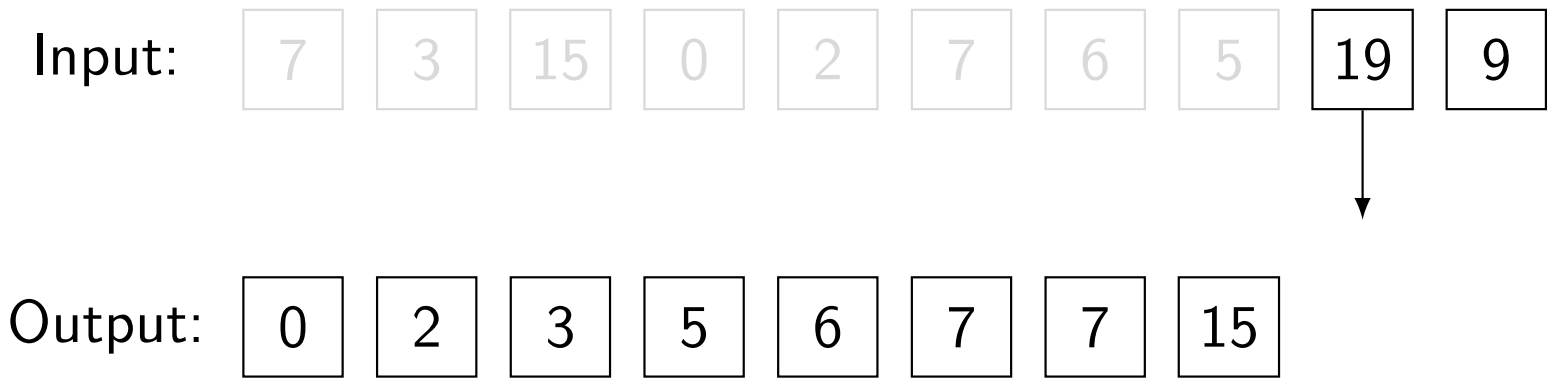
# Instickssortering



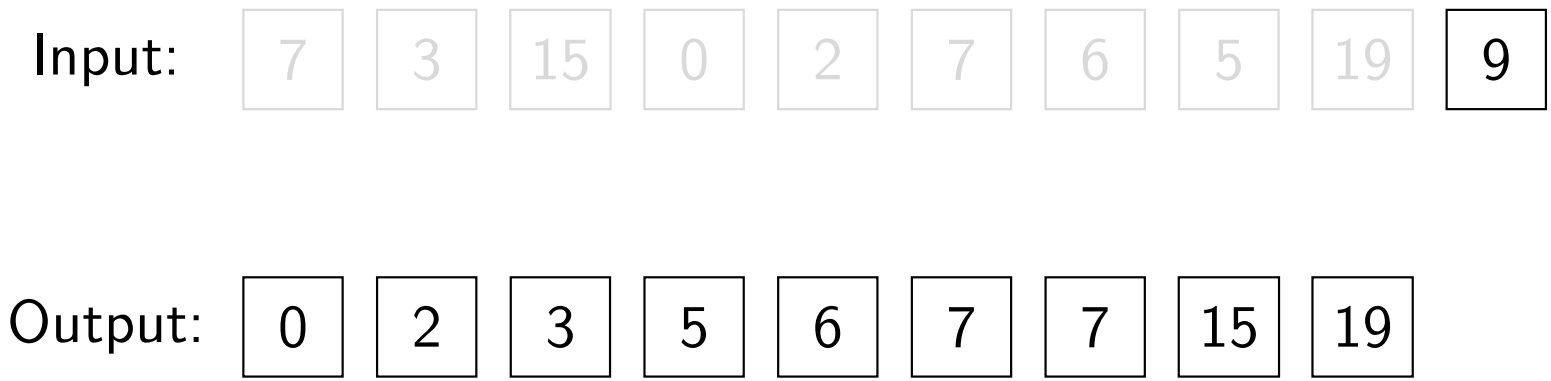
# Instickssortering



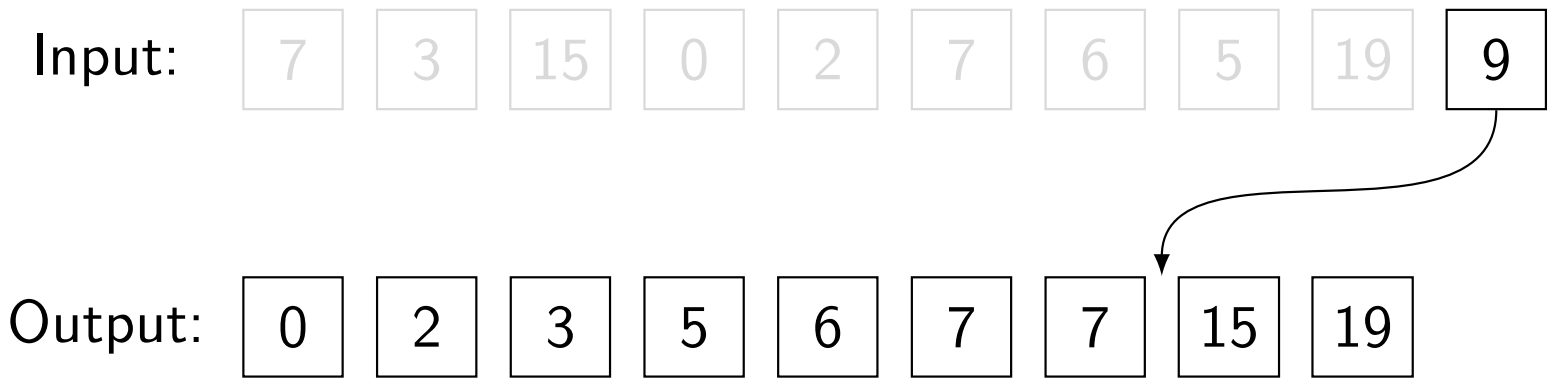
# Insticksortering



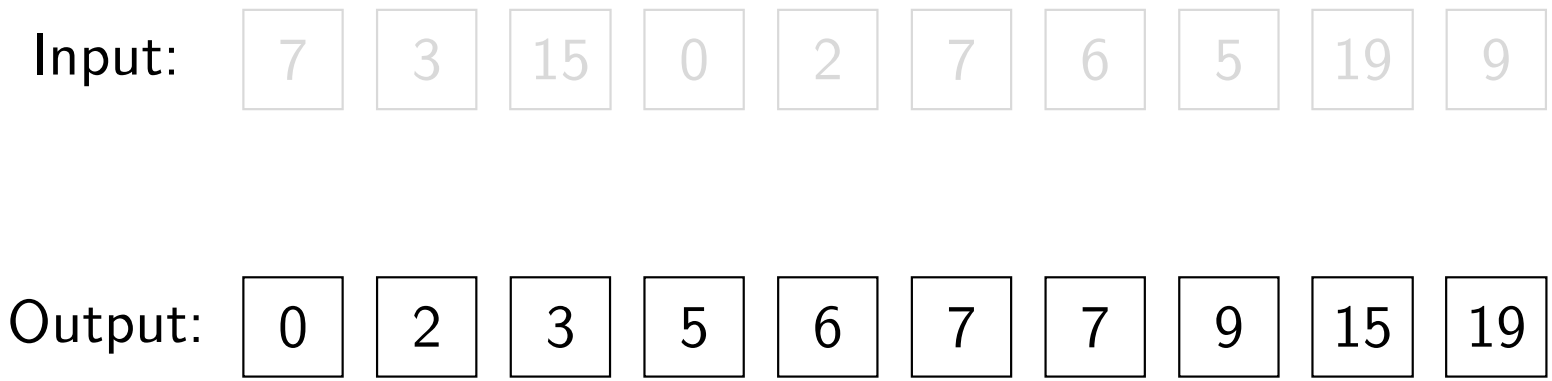
# Instickssortering



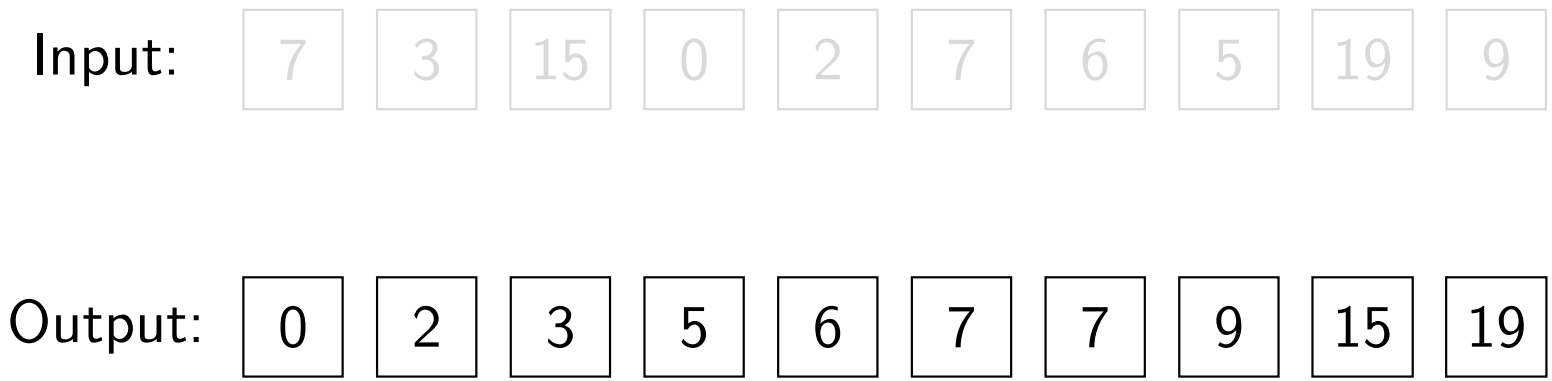
# Instickssortering



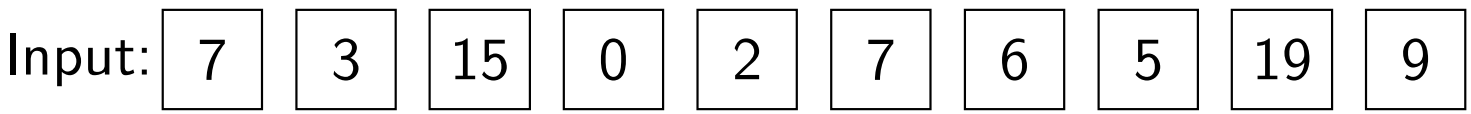
# Instickssortering



# Instickssortering

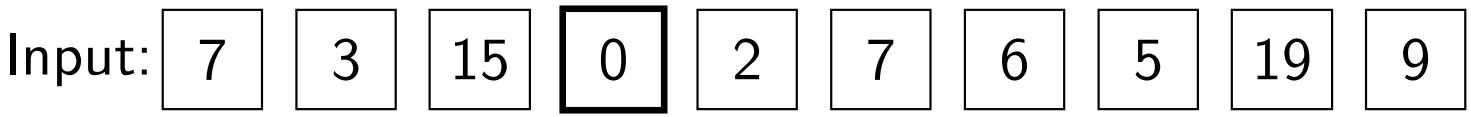


# Urvalssortering



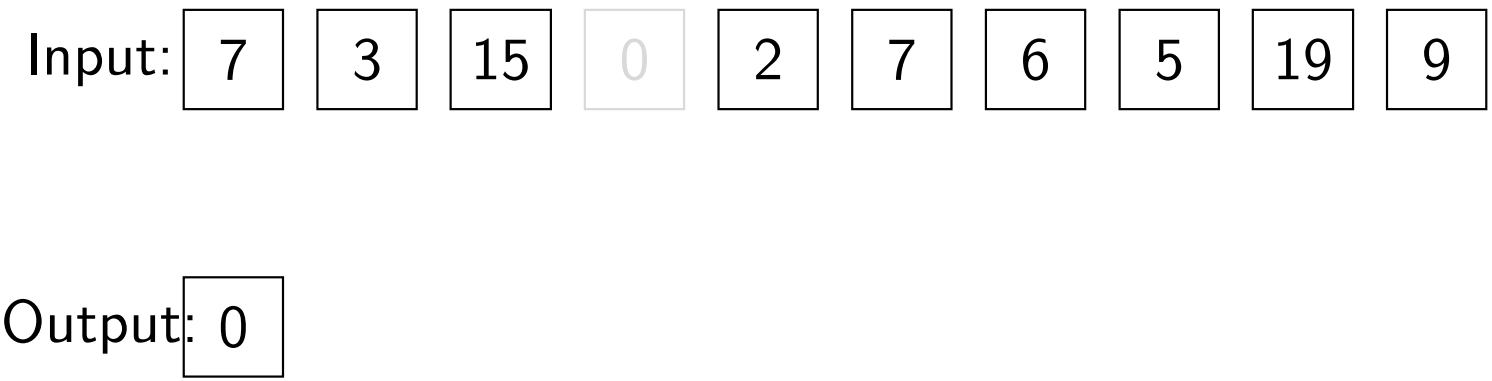

Output:

# Urvalssortering

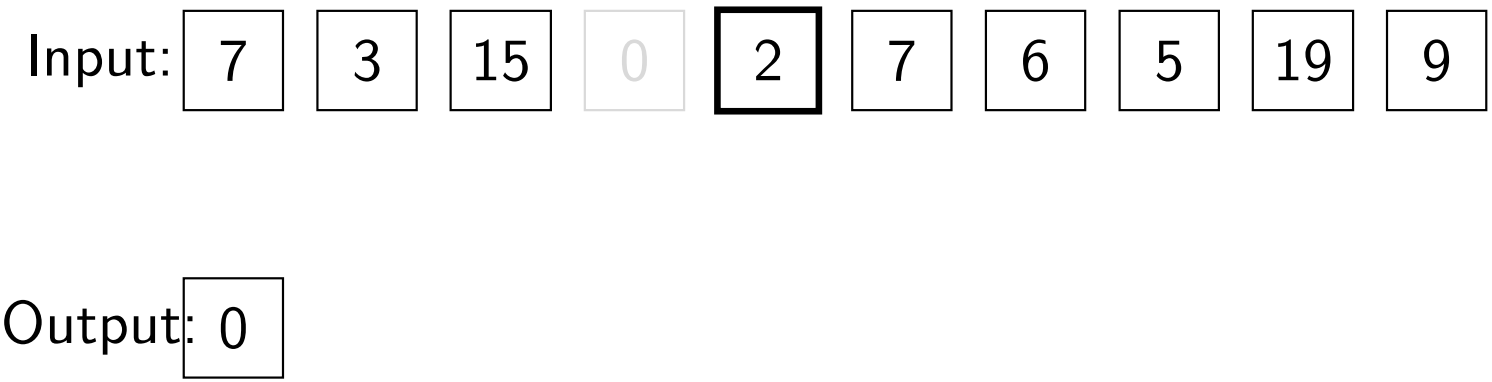



Output:

# Urvalssortering



# Urvalssortering



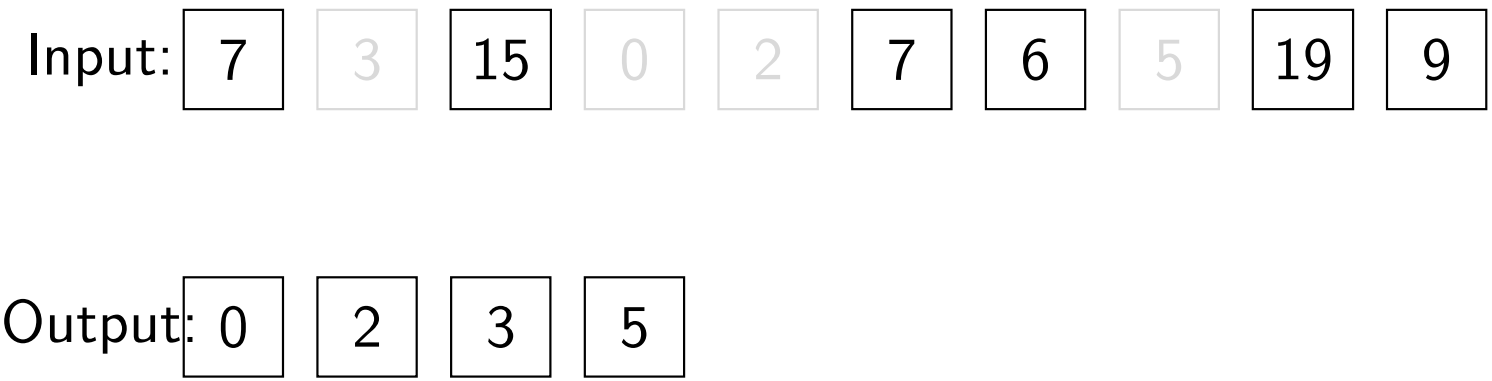
# Urvalssortering

# Urvalssortering

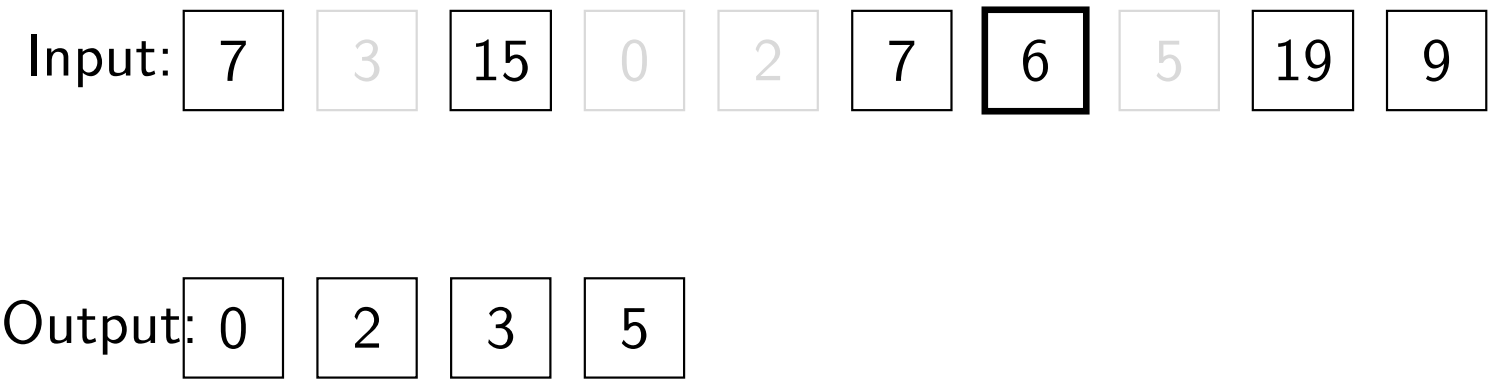
# Urvalssortering

# Urvalssortering

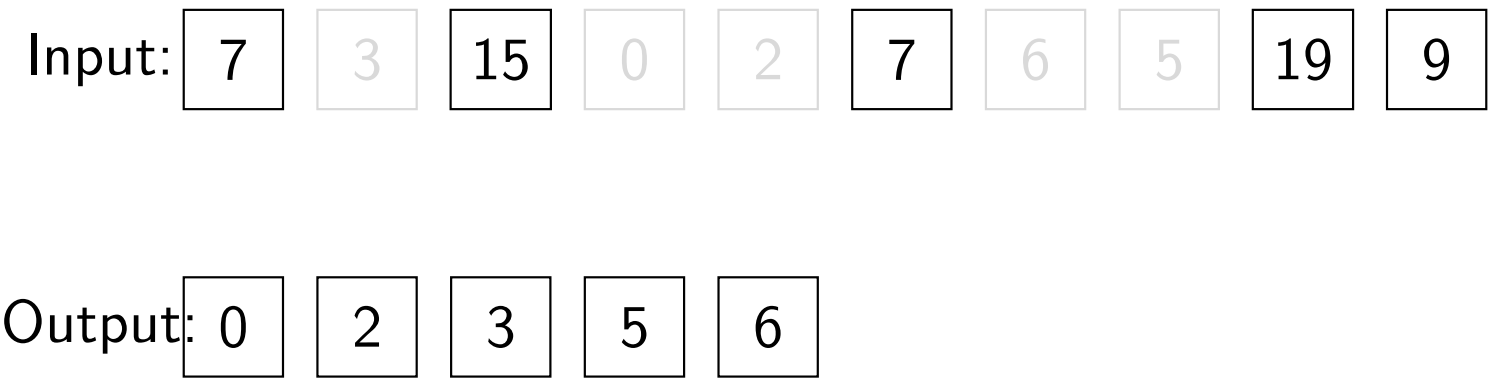
# Urvalssortering



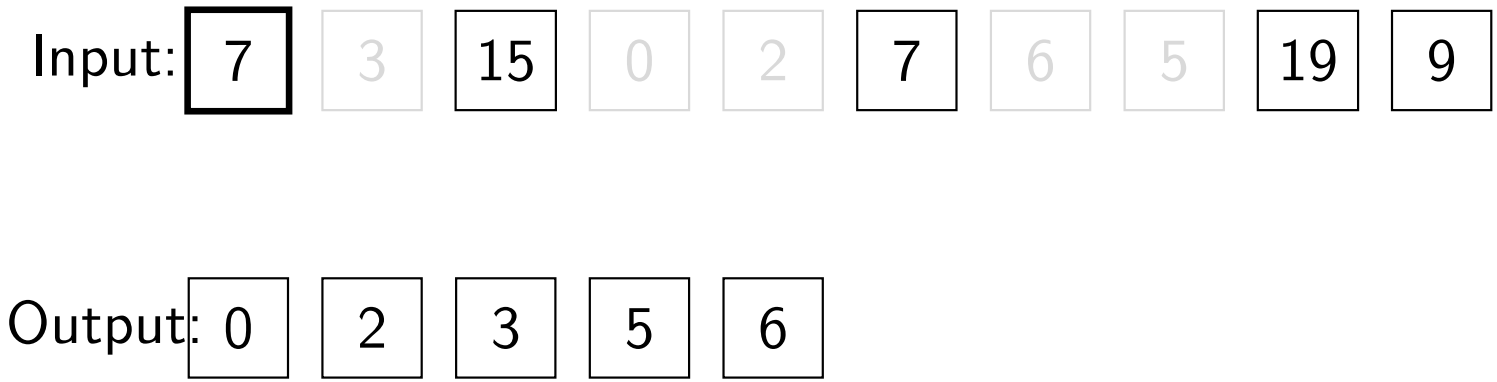
# Urvalssortering



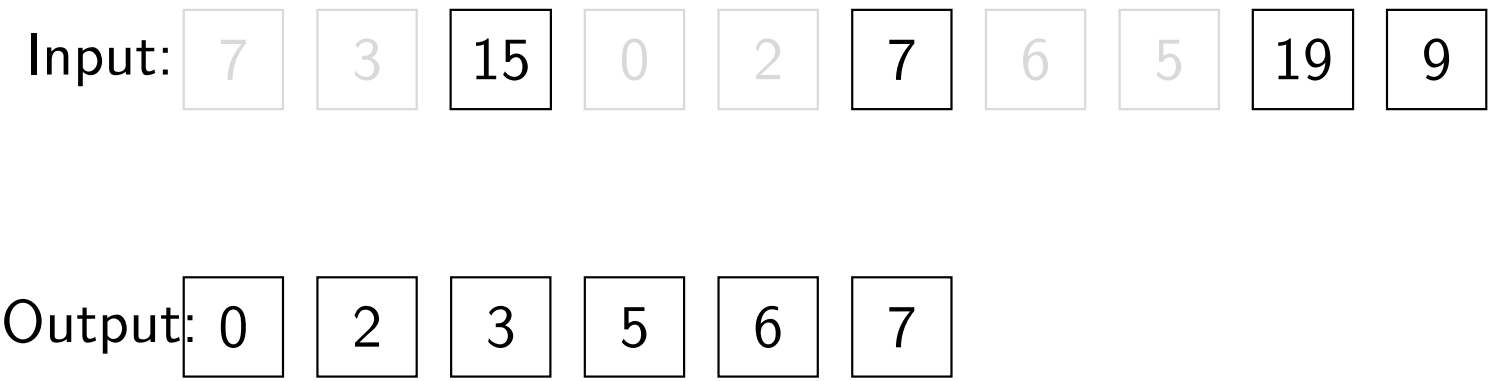
# Urvalssortering



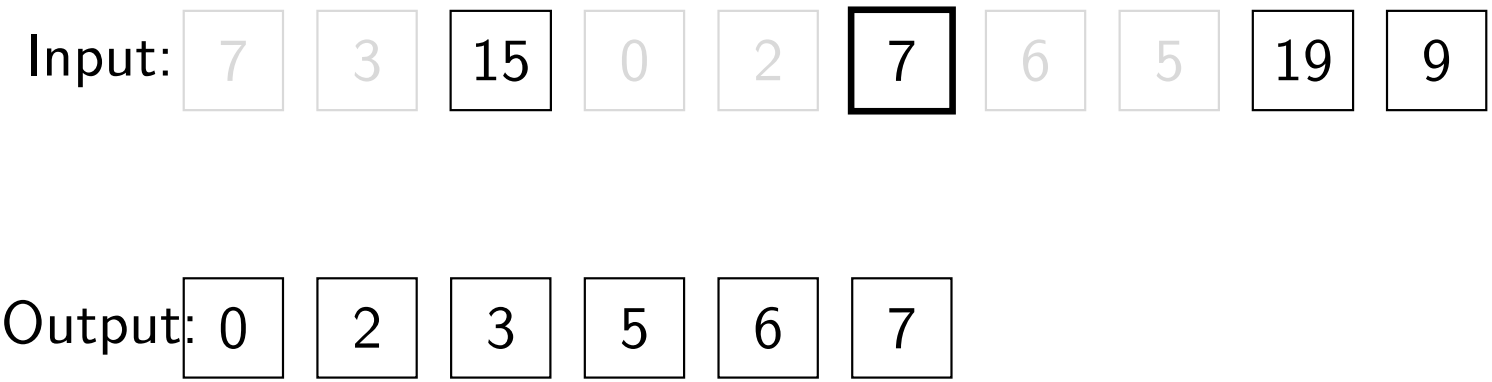
# Urvalssortering



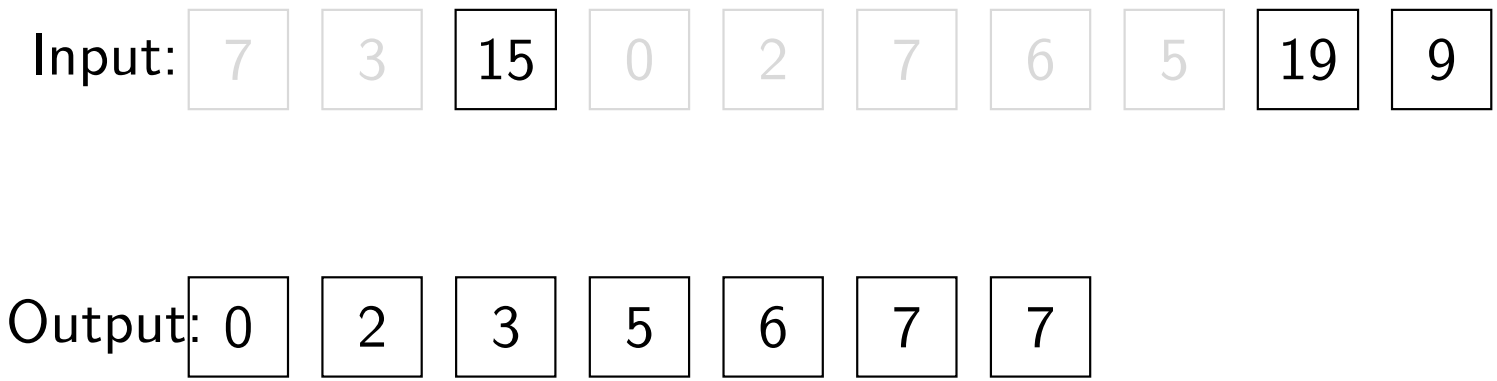
# Urvalssortering



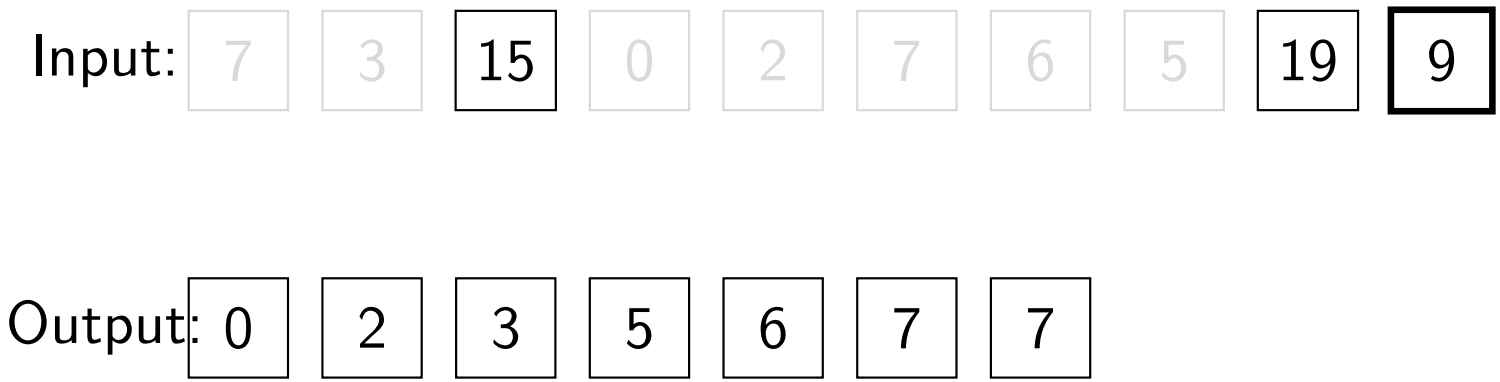
# Urvalssortering



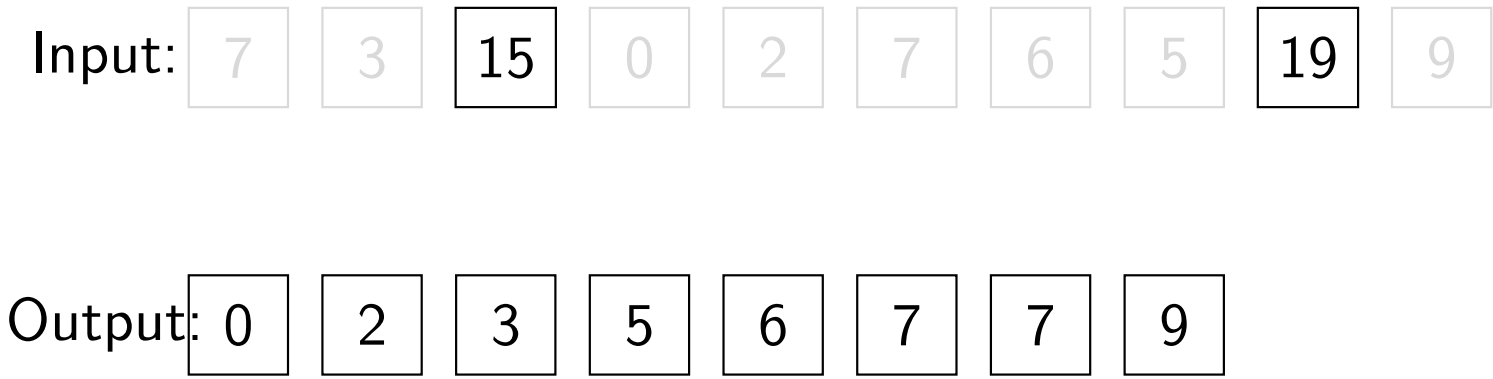
# Urvalssortering



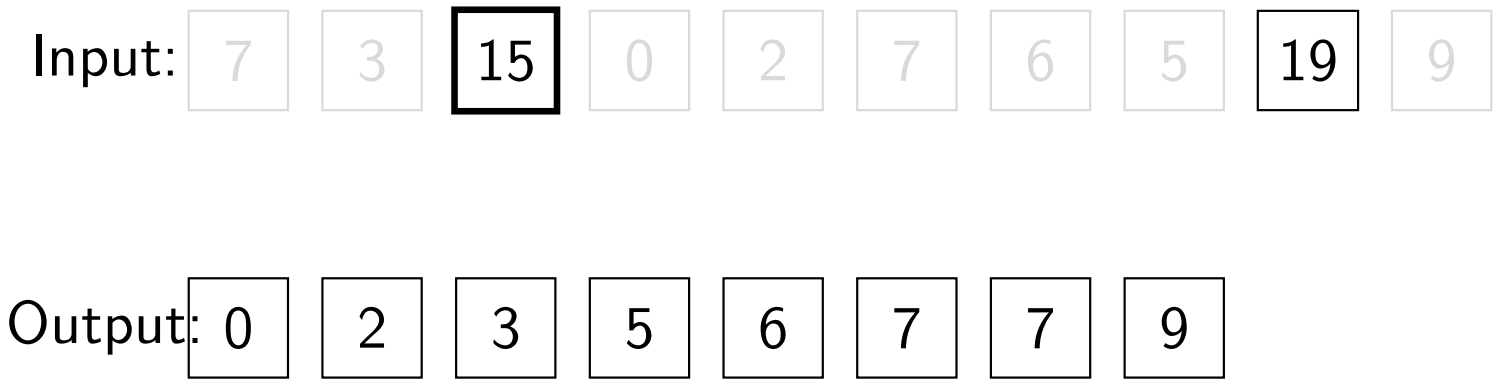
# Urvalssortering



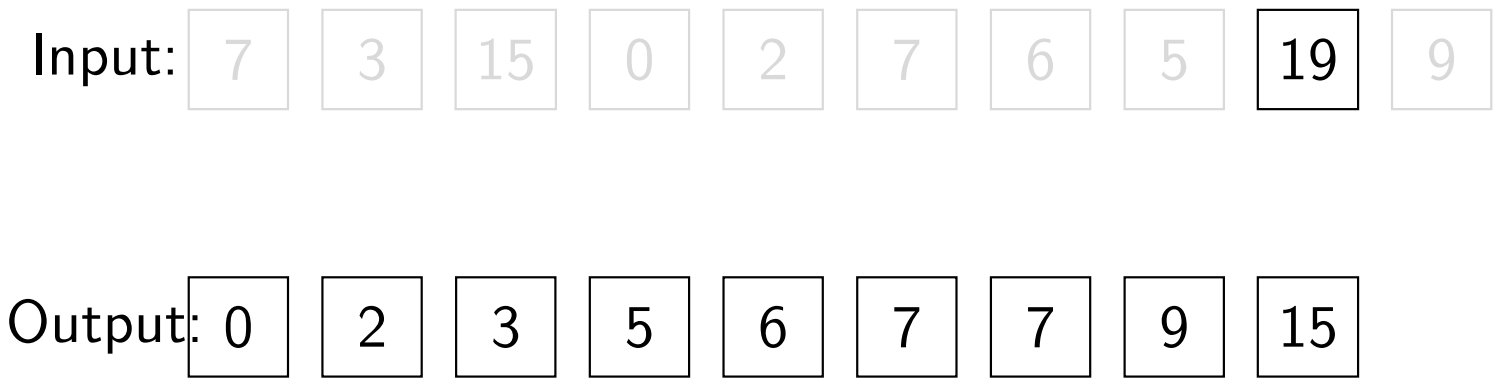
# Urvalssortering



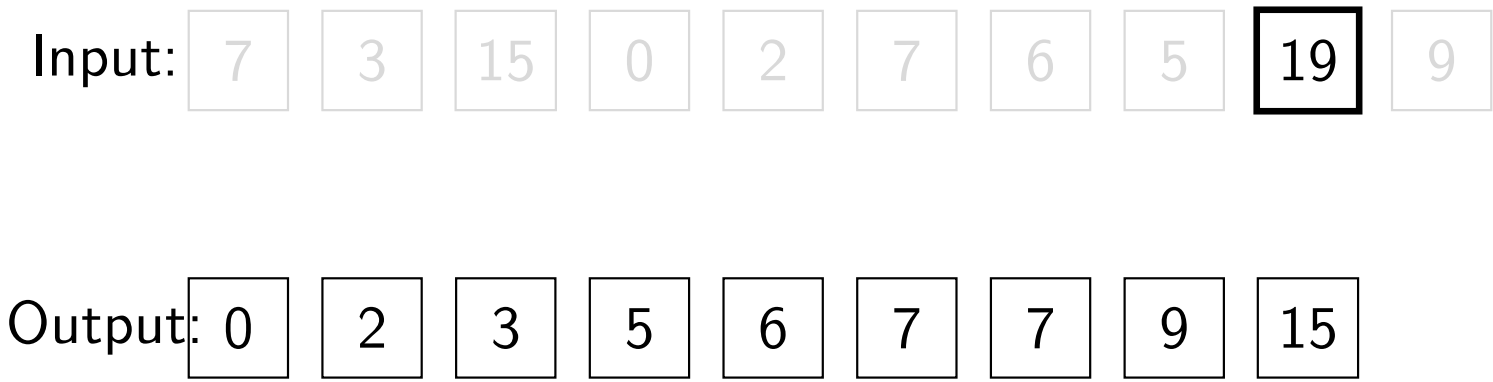
# Urvalssortering



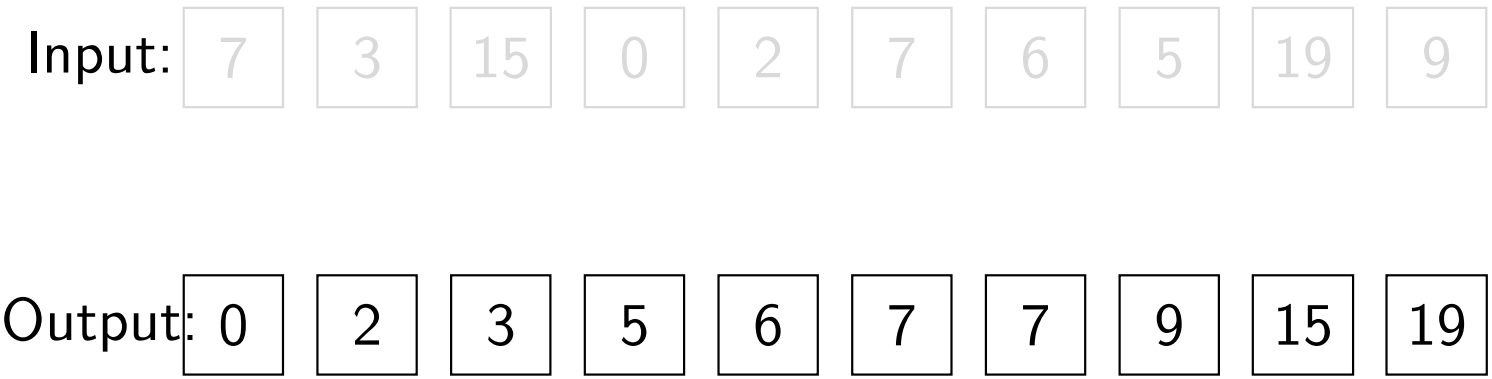
# Urvalssortering



# Urvalssortering



# Urvalssortering



# Bubbelsortering

```
Algorithm Bubblesort(arr)
repeat
    swapped ← false
    for j←low(arr) to high(arr)-1 do
        if arr[j] > arr[j+1] then
            temp ← arr[j]
            arr[j] ← arr[j+1]
            arr[j+1] ← temp
            swapped ← true
until not swapped
```

- Stabil sortering?
- Tidskomplexiteten?
  - $O(n^2)$ för en fältbaserad lista.
  - $O(?)$ för en länkad lista?

# Divide-and-Conquer

- Rekursiv algoritmprincip:
  - Grundidé: Dela upp problemet i mindre och mindre problem tills problemet är trivialt.
  - Lös det triviala basfallet.
  - Slå ihop till en totallösning.
- *Mergesort* och *Quicksort* är av denna typ.
- Komplexitet: $O(n \log n)$.

# Mergesort

- Algoritm för att sortera sekvensen S:
- Om S har bara ett element (trivialt):
  - Sekvensen S redan sorterad. Returnera S.

  annars
  - Divide: Dela S i två lika stora delsekvenser S1 och S2.
    - Sortera sekvenserna S1 och S2 rekursivt.
  - Conquer: Slå samman S1 och S2 till en sorterad sekvens S. Returnera S.

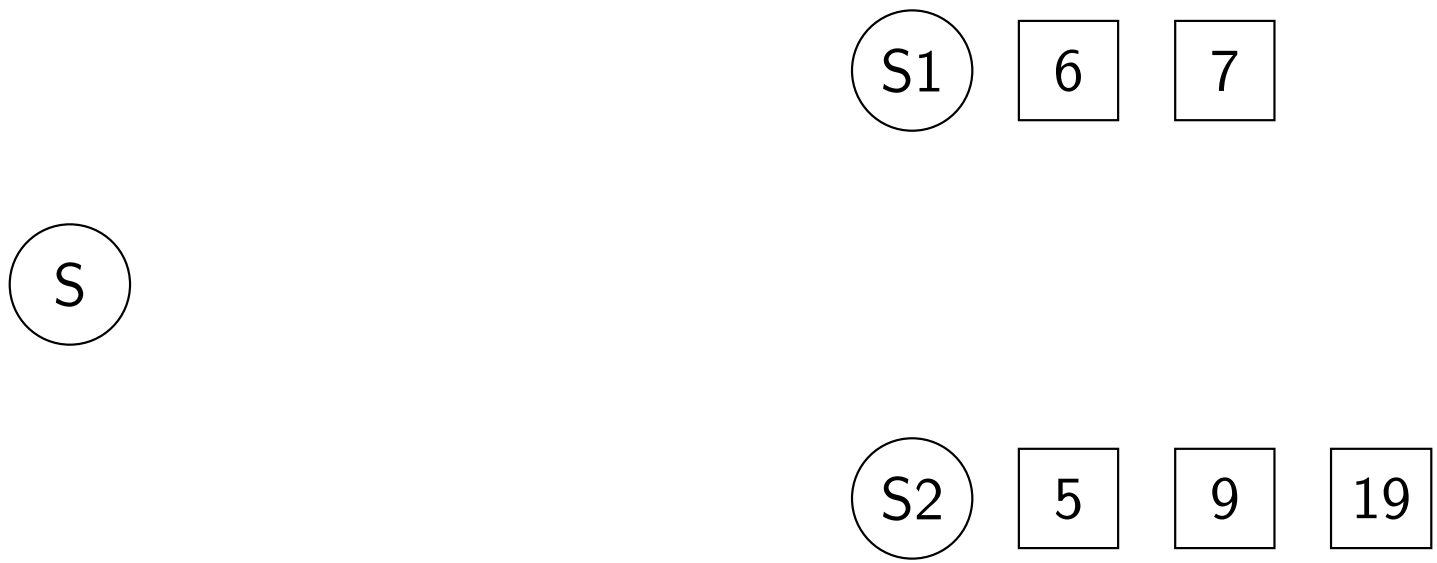
# Algoritm Mergesort (1)

```
Algorithm Mergesort(S)

if length(S) > 1 then
    (S1,S2) ← Split(S)
    S1 ← Mergesort(S1)
    S2 ← Mergesort(S2)
    S ← Merge(S1,S2)
return S
```

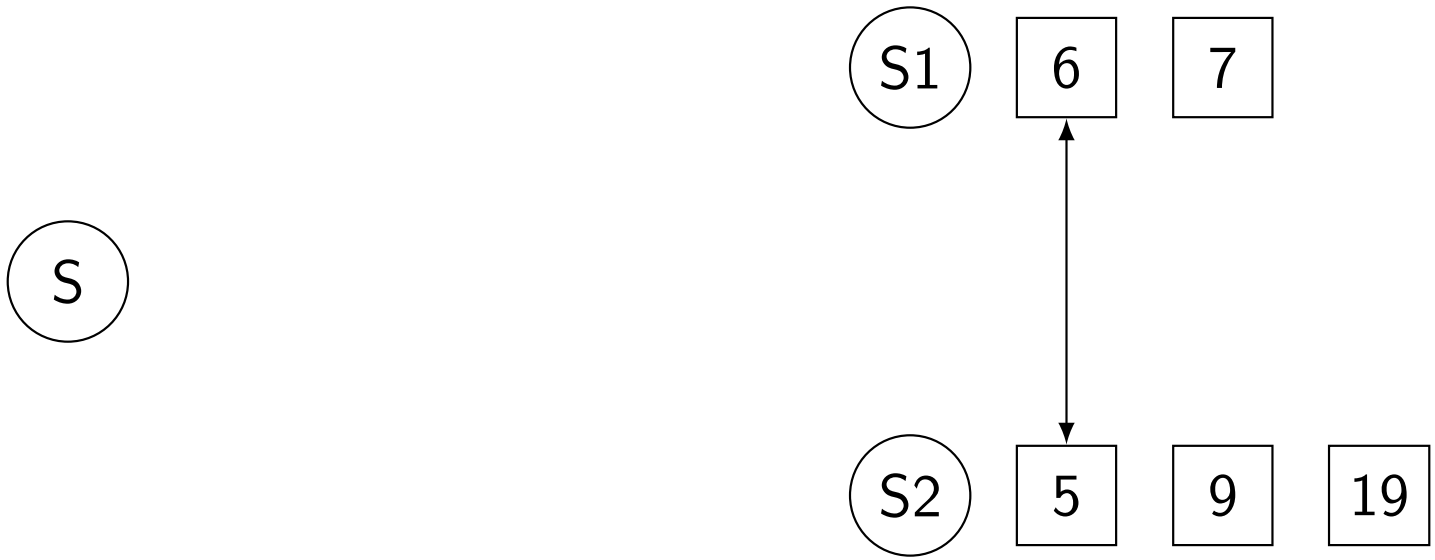
# Algoritm Mergesort (2)

```
Algorithm Merge(S1,S2)
S ← Empty()
while not Isempty(S1) and not Isempty(S2) do
    if Inspect(first(S1),S1) ≤ Inspect(first(S2),S2) then
        Insert(Inspect(first(S1),S1),end(S),S)
        Remove(First(S1),S1)
    else
        Insert(Inspect(first(S2),S2),end(S),S)
        Remove(First(S2),S2)

while not Isempty(S1) do
    Insert(Inspect(First(S1),S1),end(S),S)
    Remove(First(S1),S1)
while not Isempty(S2) do
    Insert(Inspect(First(S2),S2),end(S),S)
    Remove(First(S2),S2)
return S
```

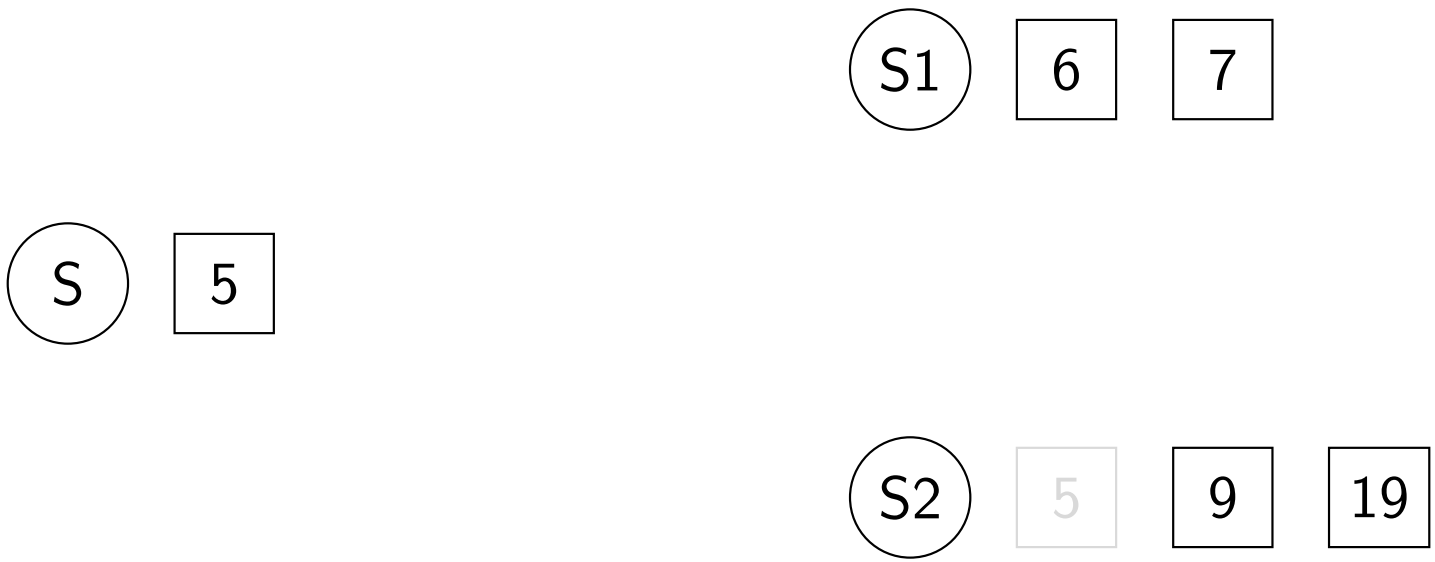
# Exempel Merge



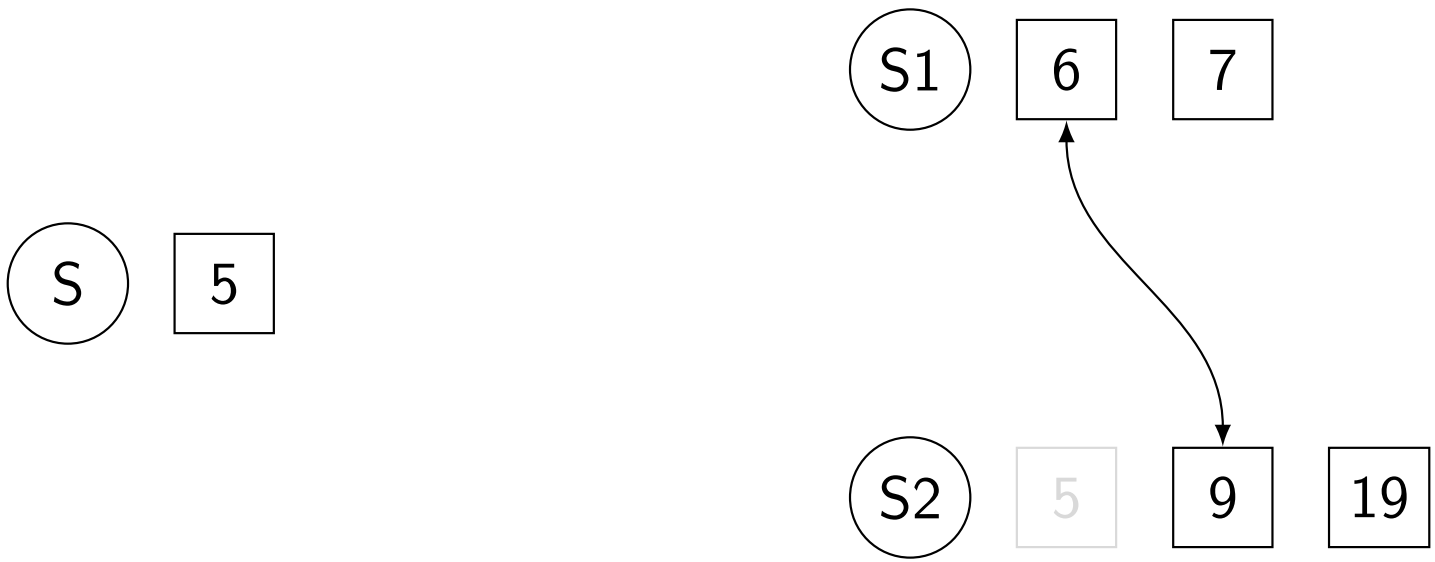
# Exempel Merge



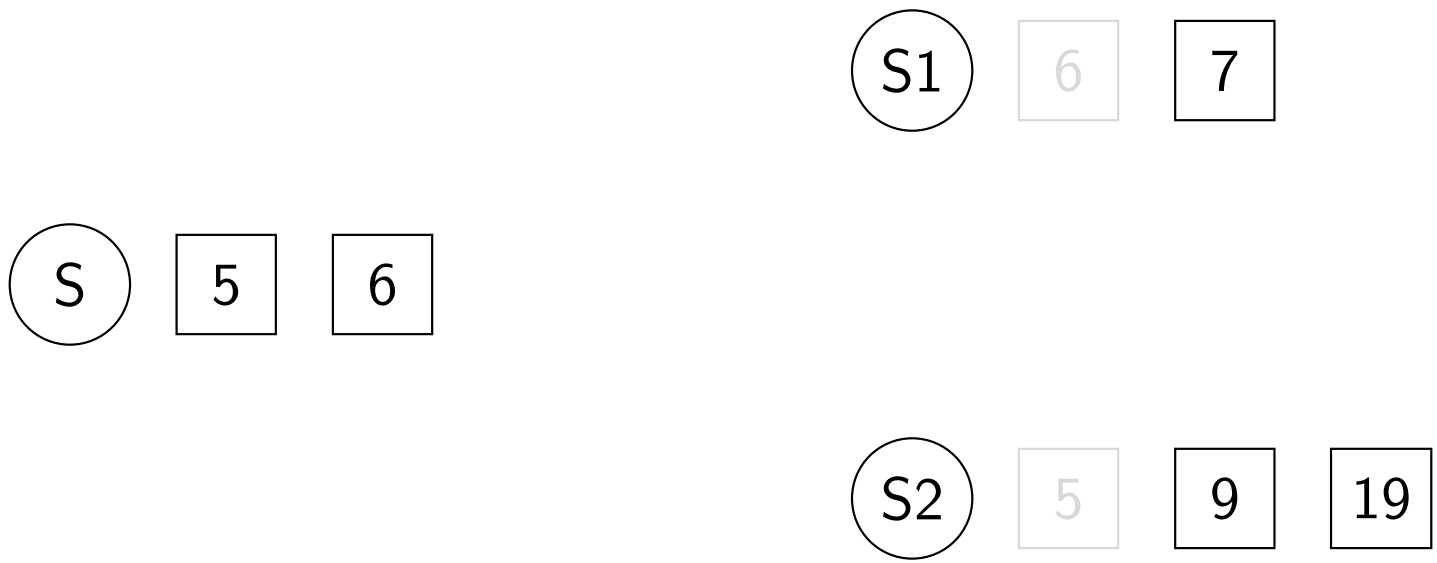
# Exempel Merge



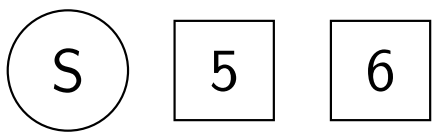
# Exempel Merge

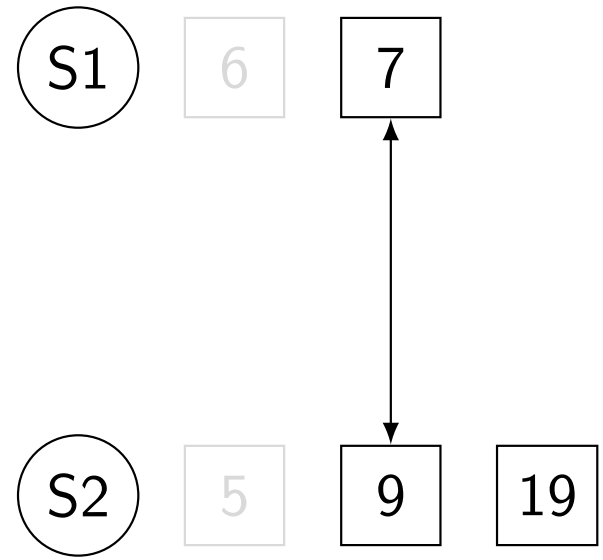


# Exempel Merge

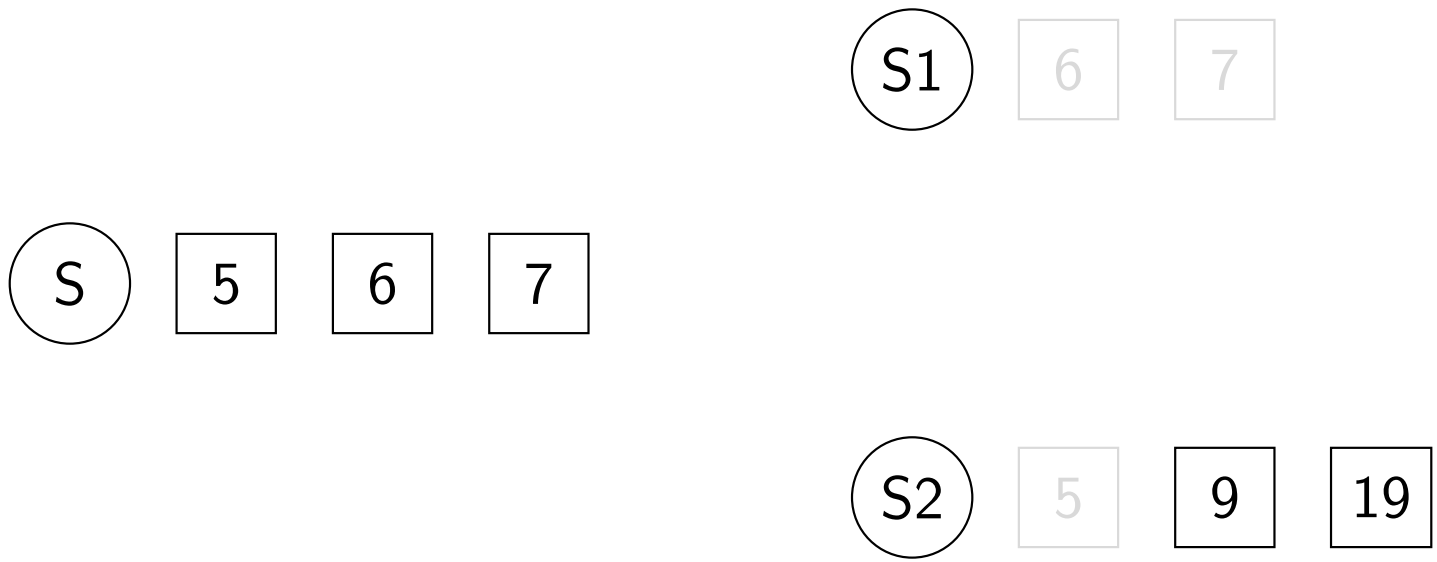


# Exempel Merge

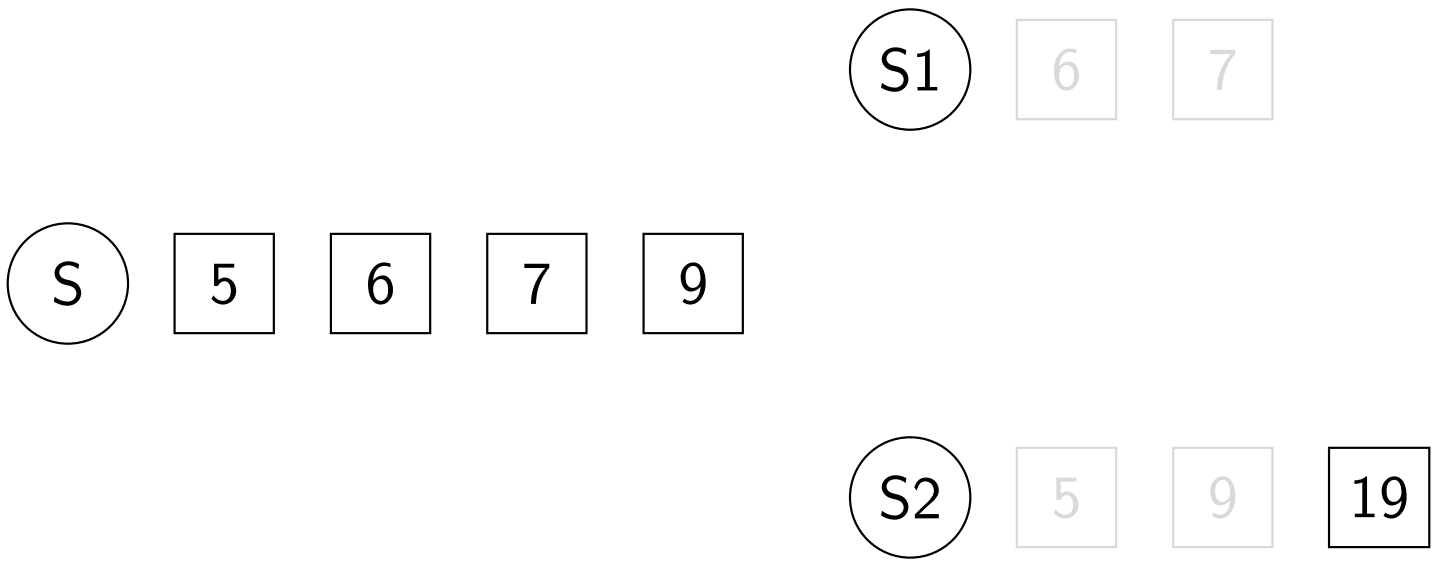


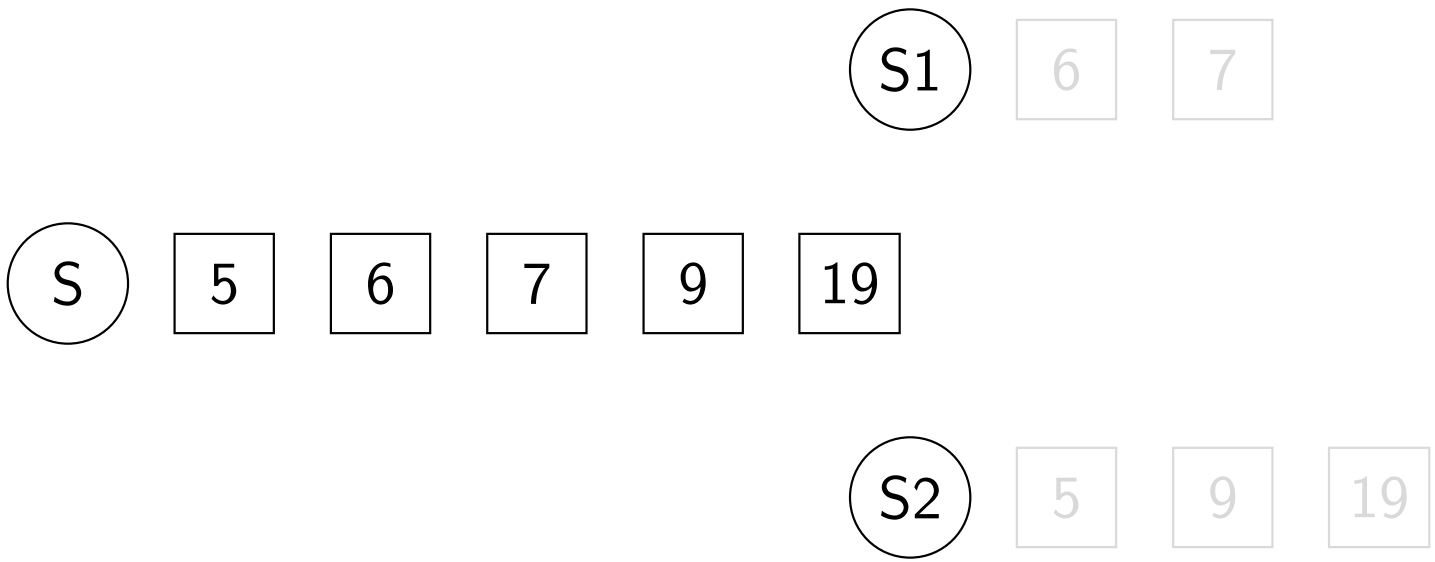
# Exempel Merge



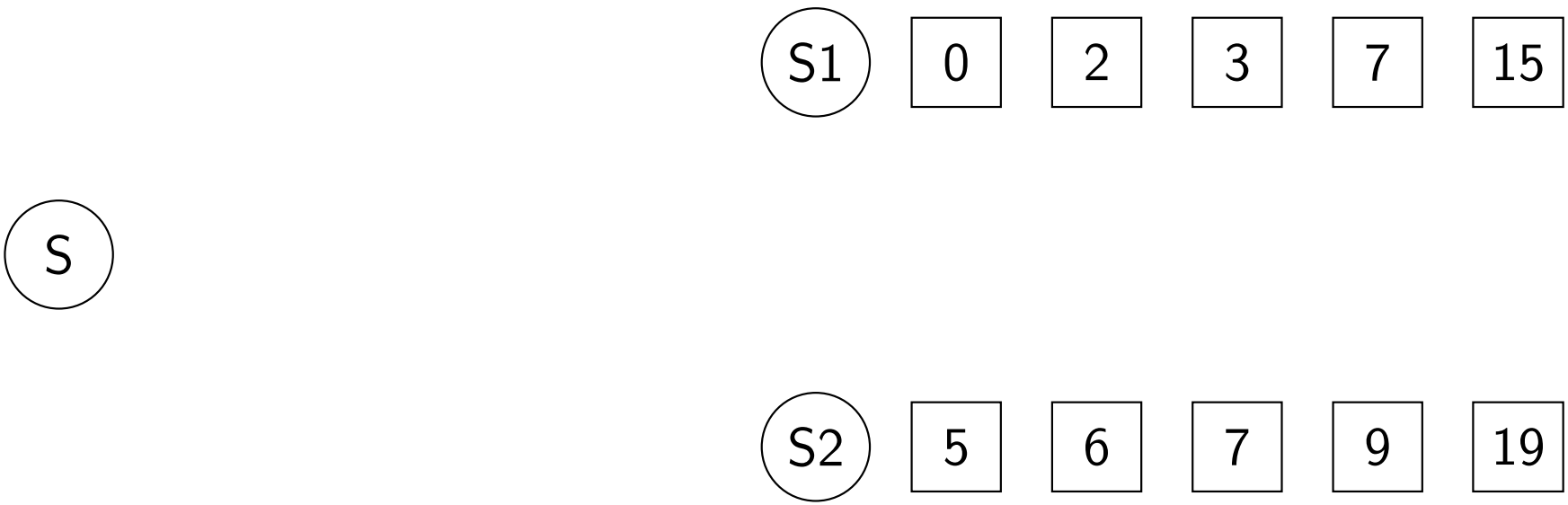
# Exempel Merge



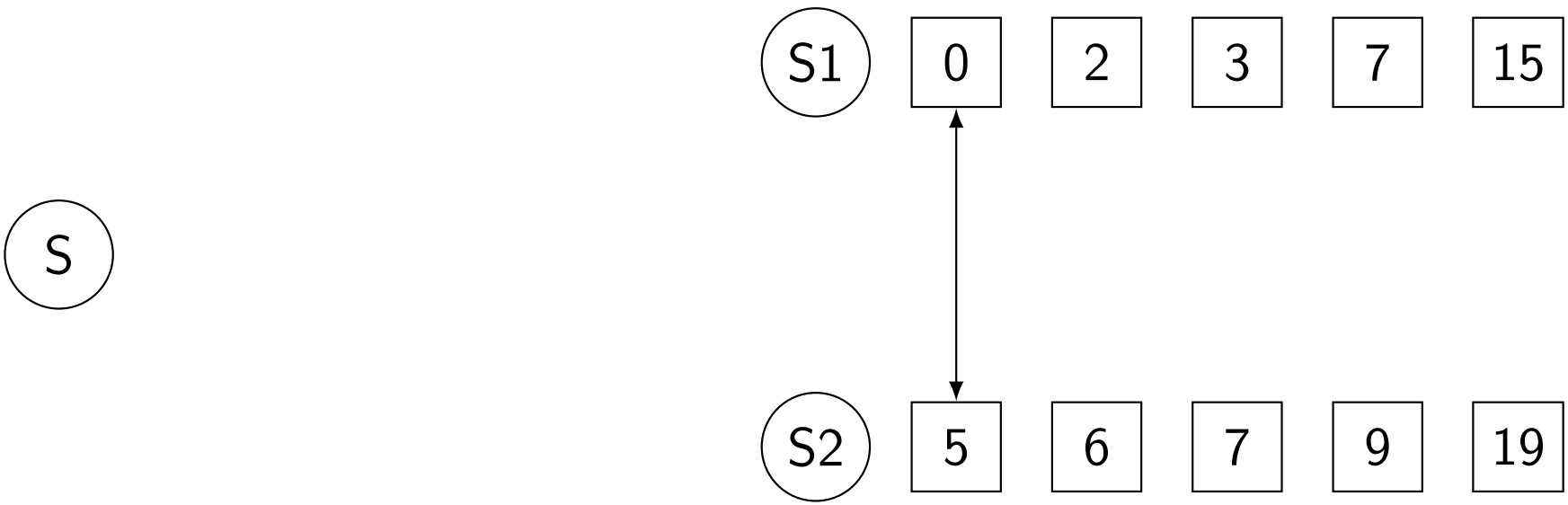
# Exempel Merge



# Exempel Merge

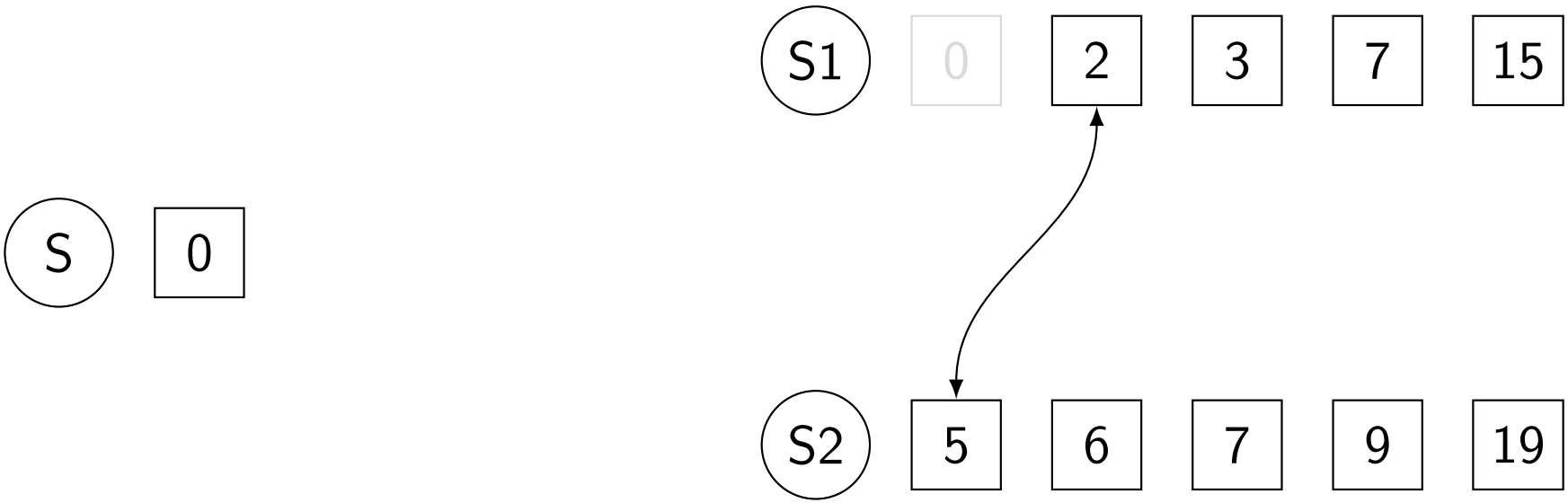


# Exempel Merge



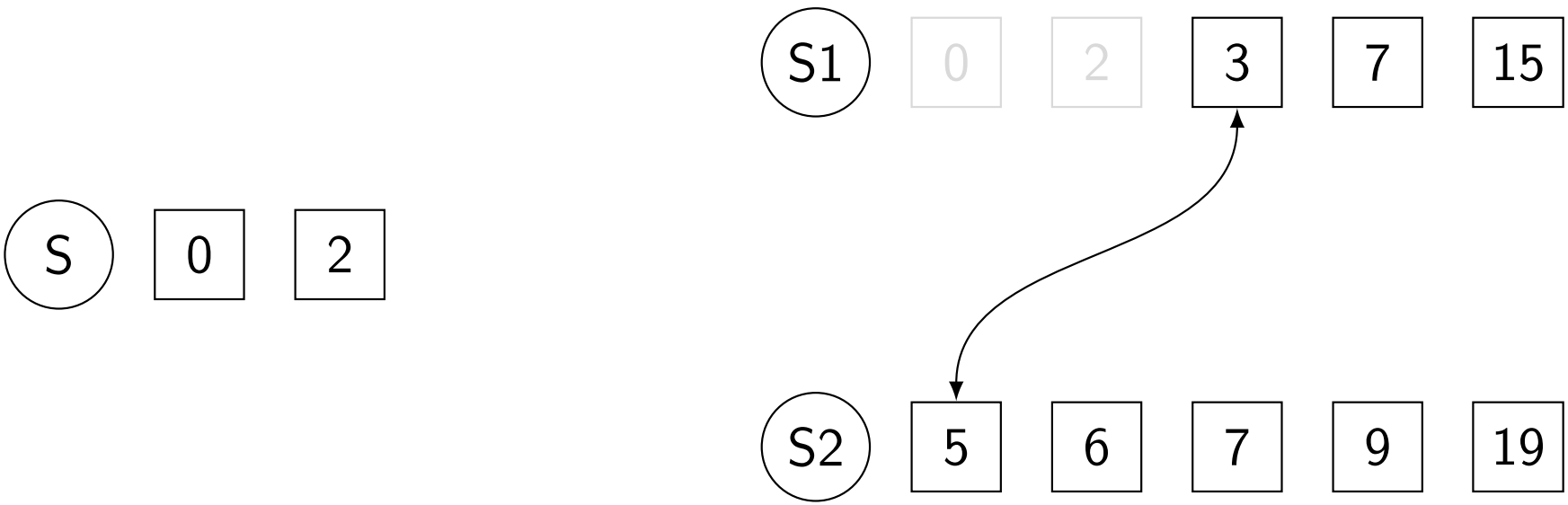
# Exempel Merge

# Exempel Merge



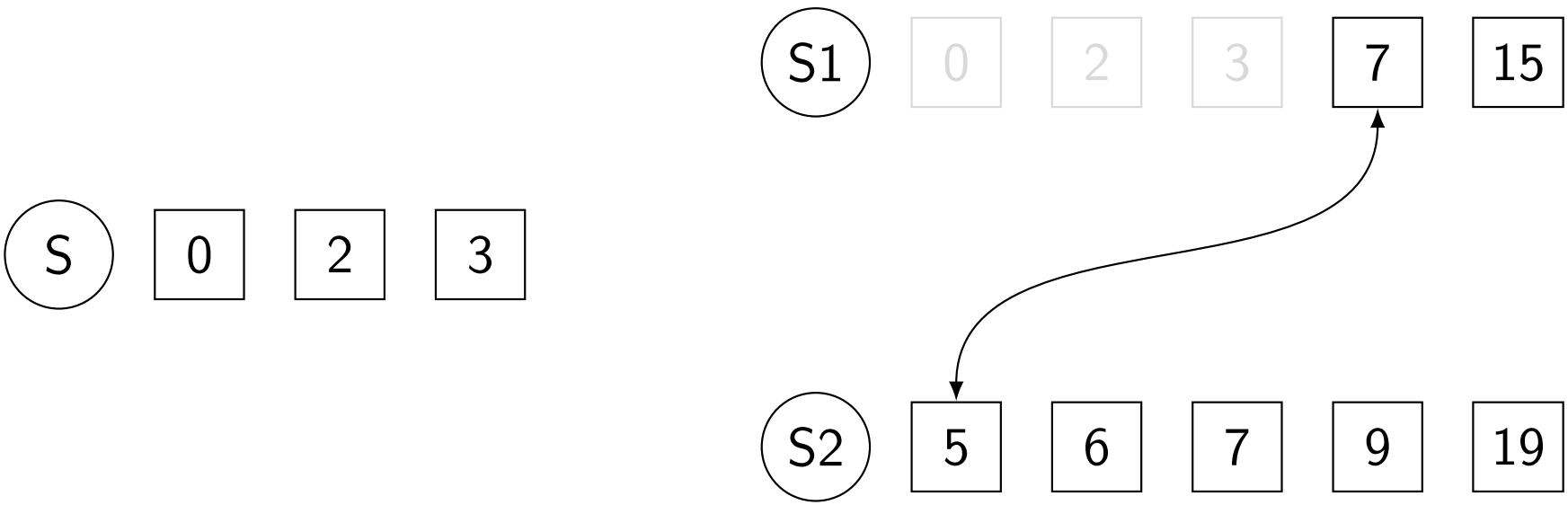
# Exempel Merge

# Exempel Merge

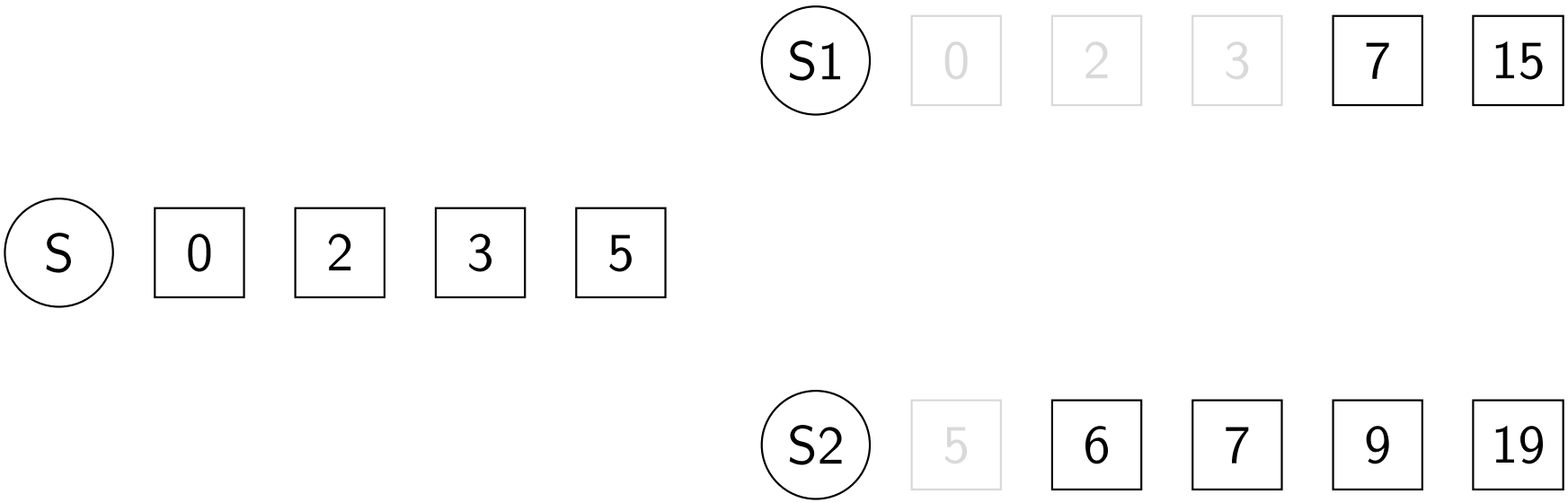


# Exempel Merge

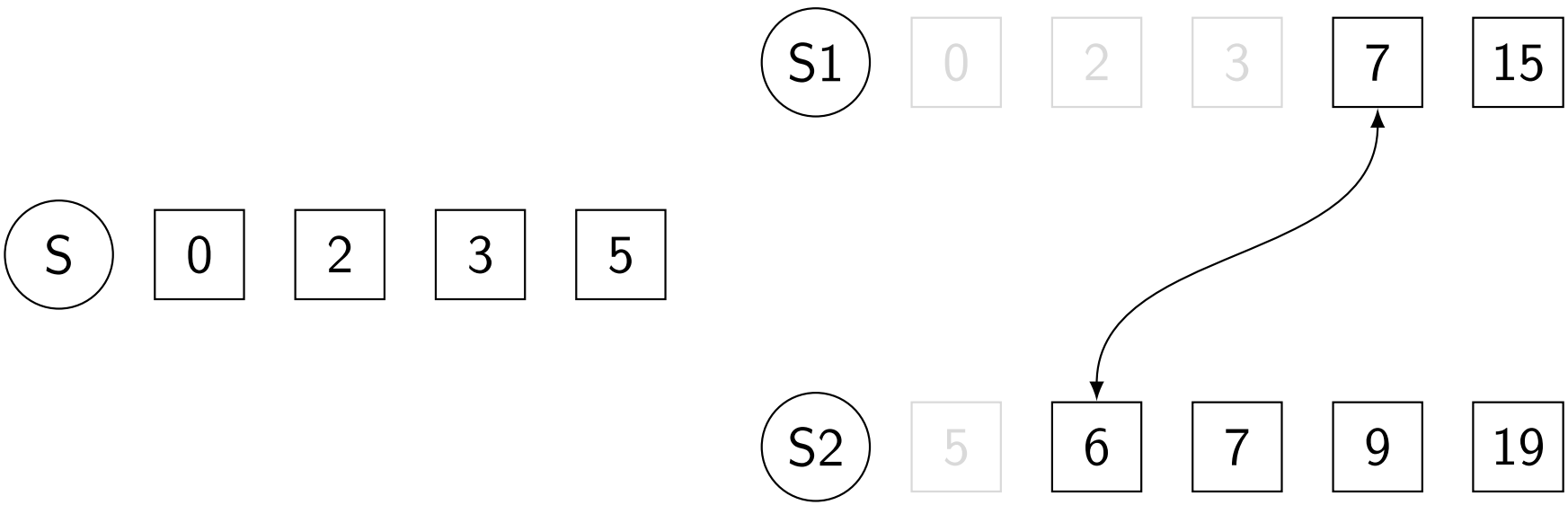
# Exempel Merge



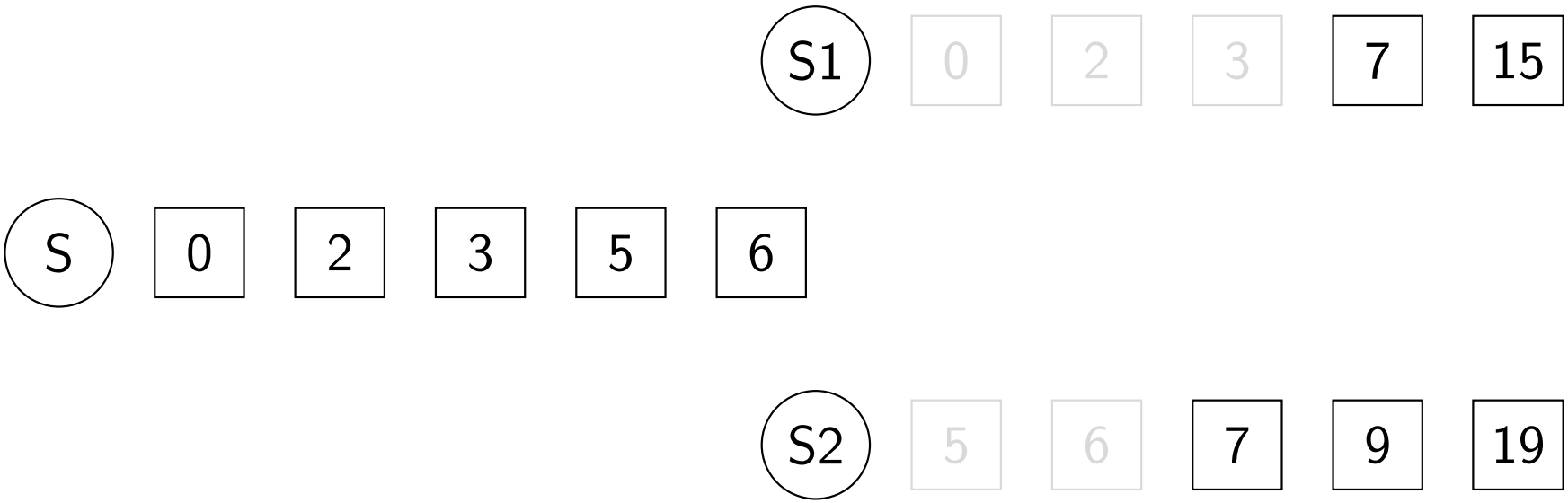
# Exempel Merge



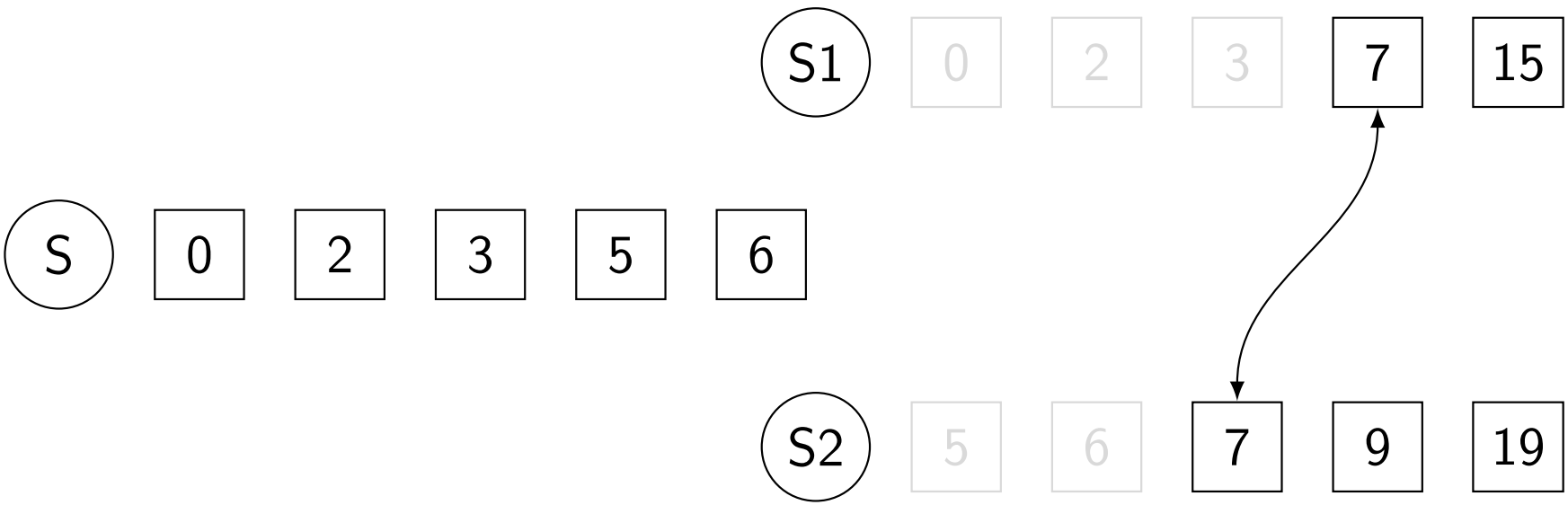
# Exempel Merge



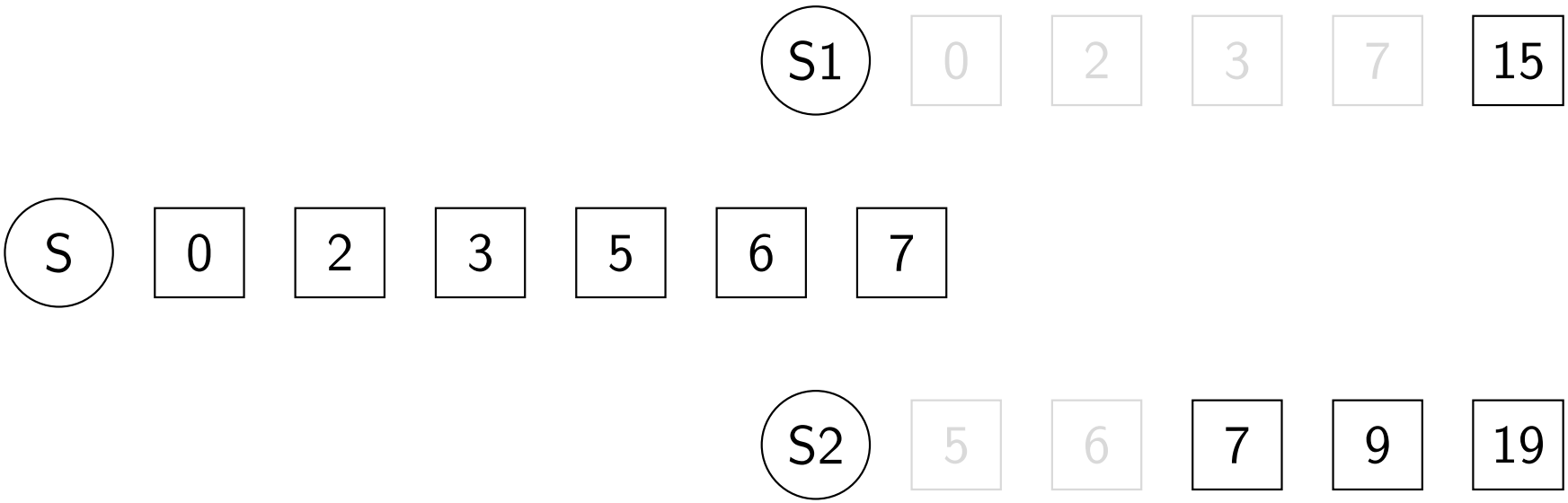
# Exempel Merge



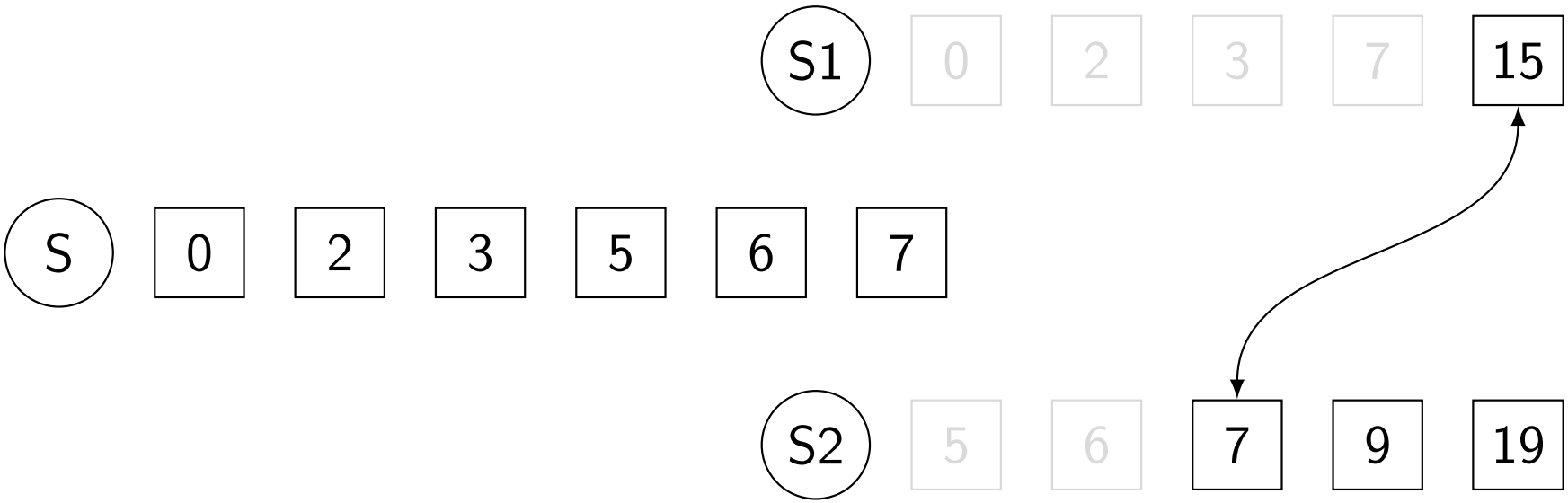
# Exempel Merge



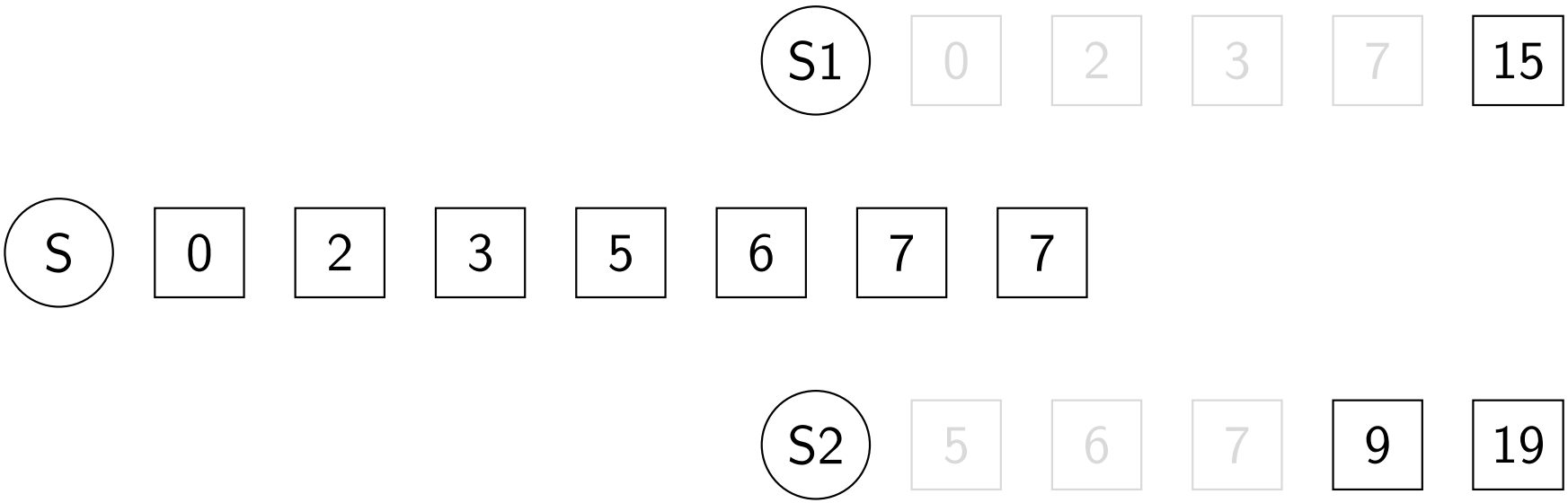
# Exempel Merge



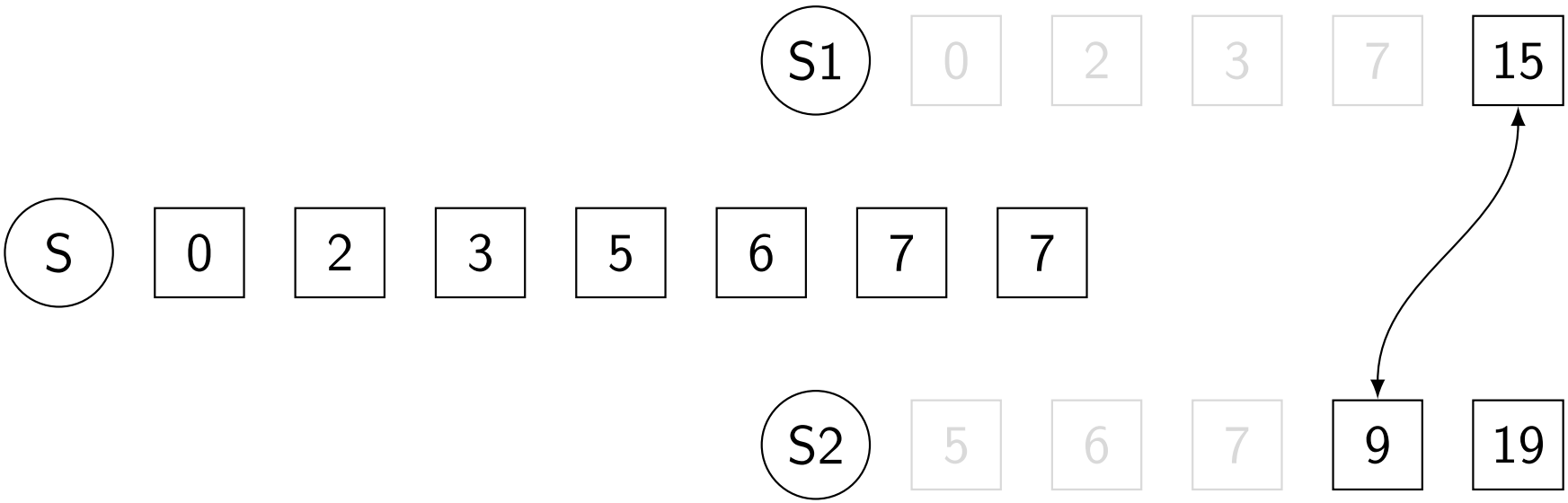
# Exempel Merge



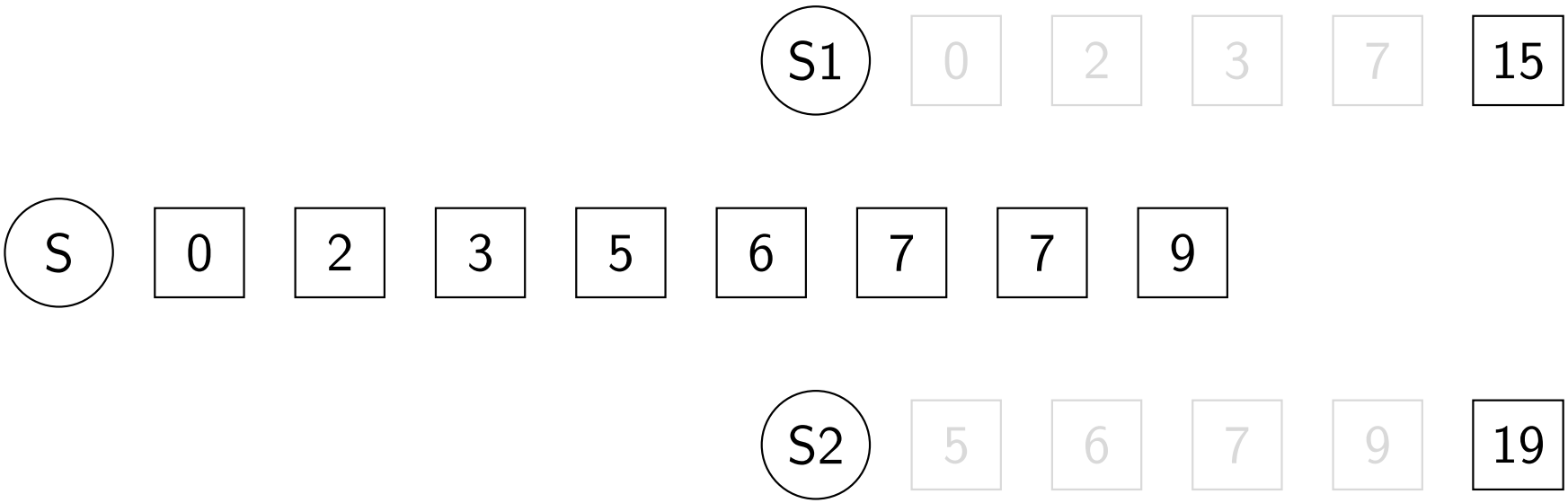
# Exempel Merge



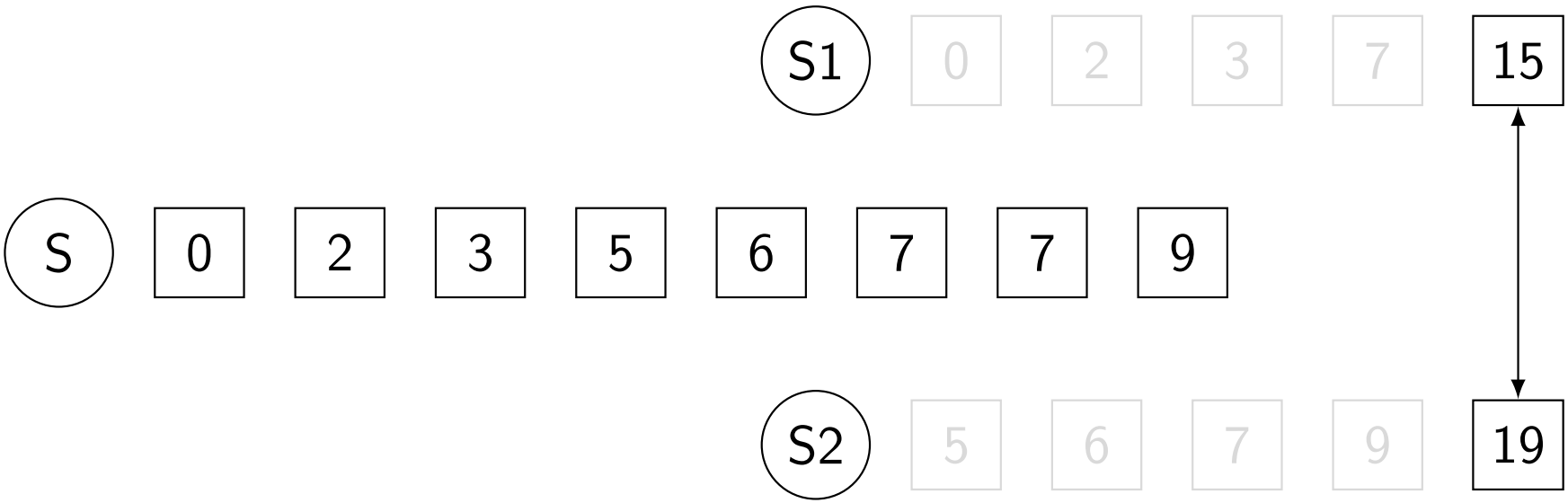
# Exempel Merge



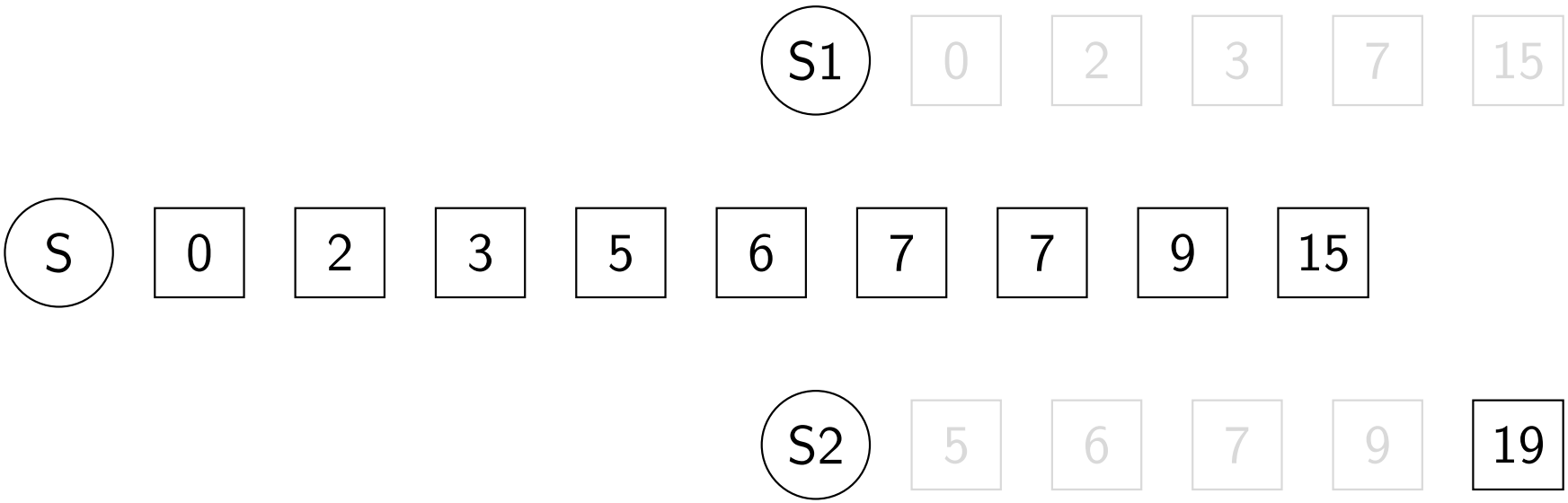
# Exempel Merge



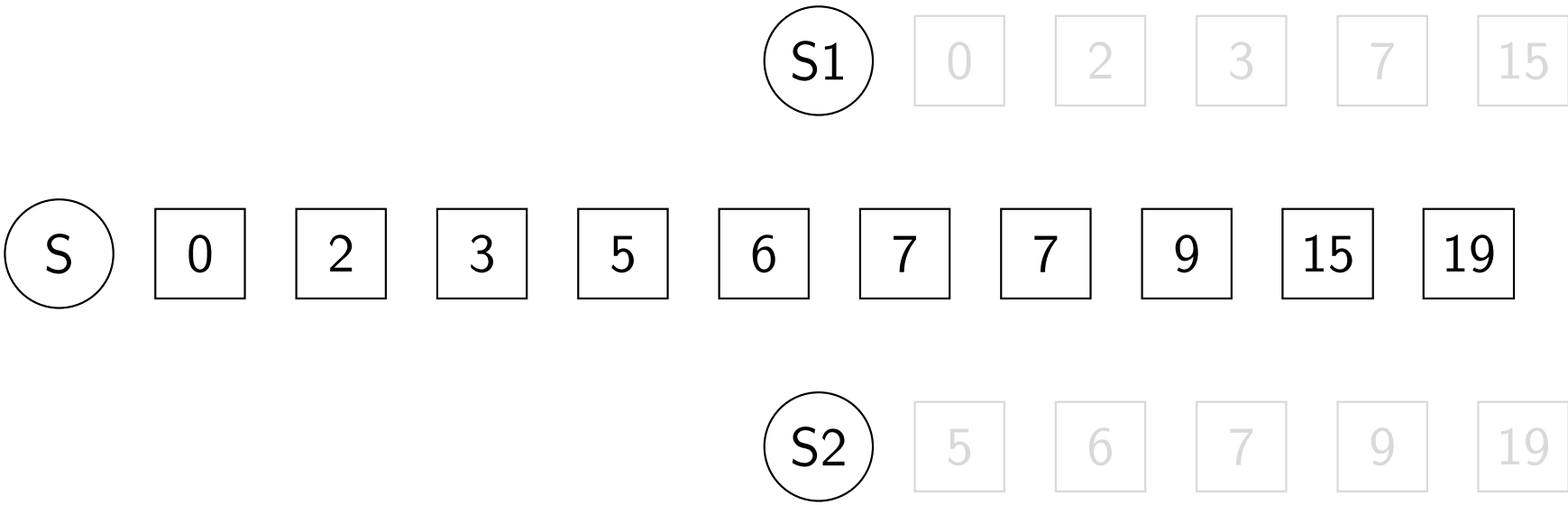
# Exempel Merge



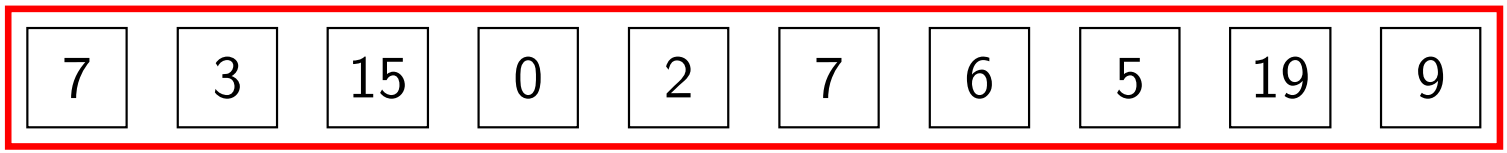
# Exempel Merge



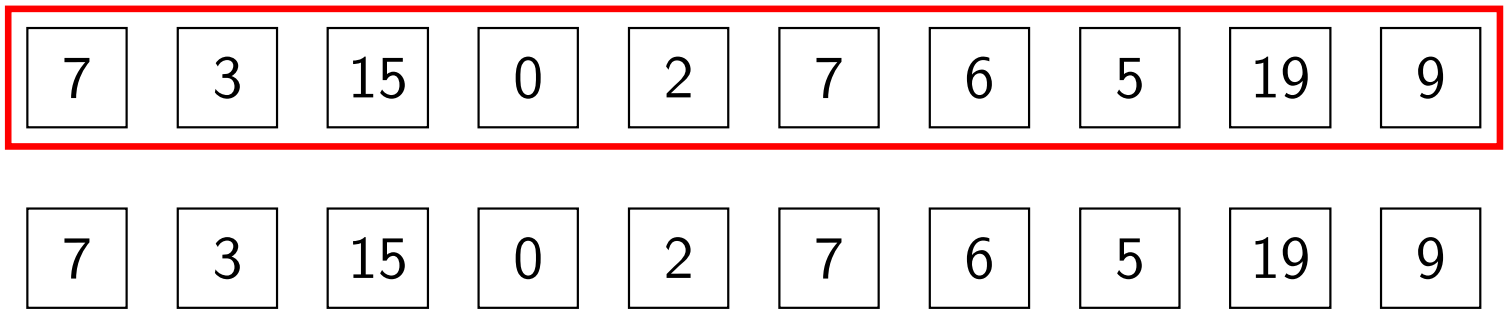
# Exempel Merge



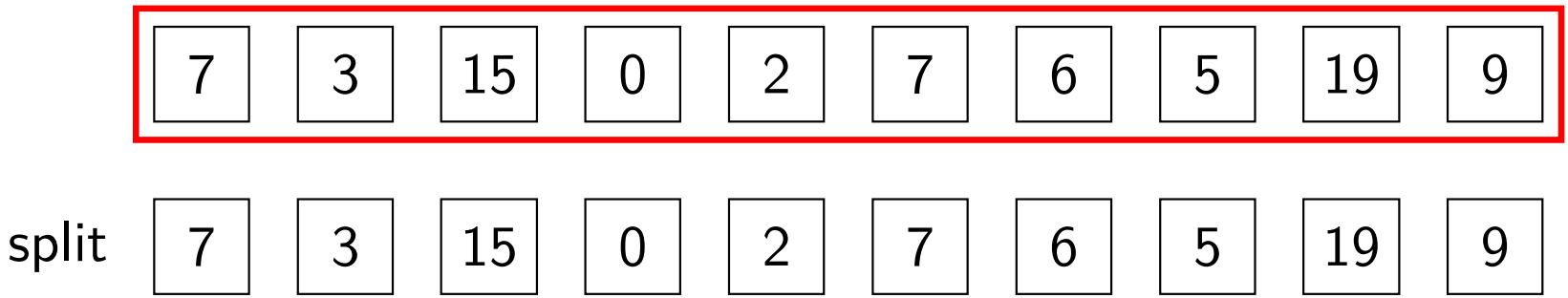
# Exempel Mergesort



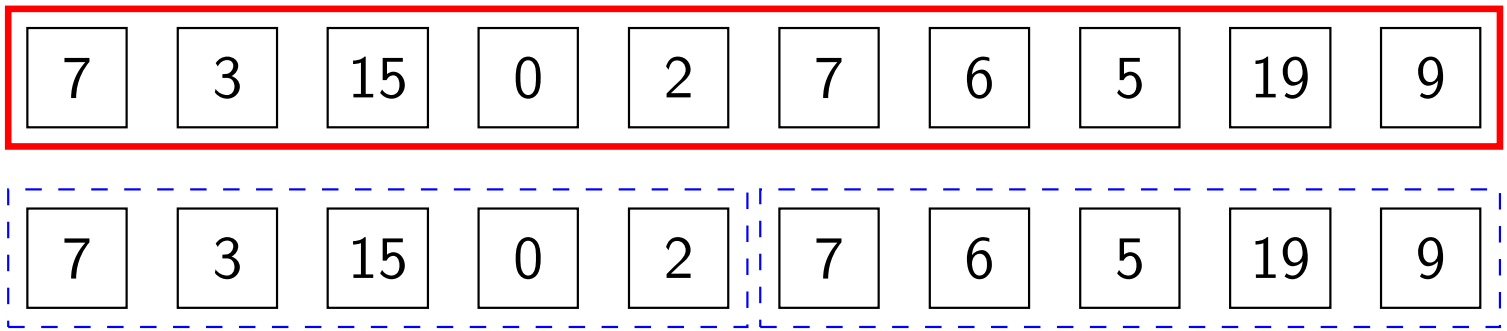
# Exempel Mergesort



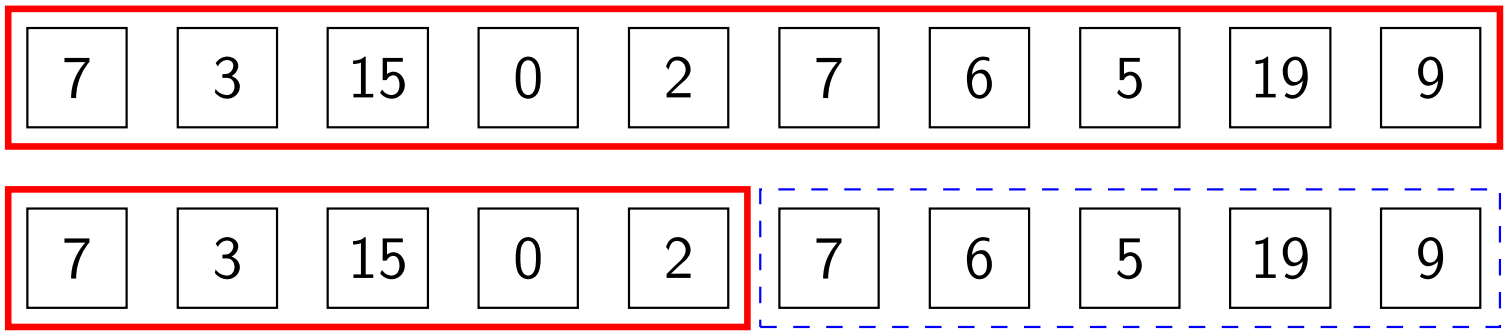
# Exempel Mergesort



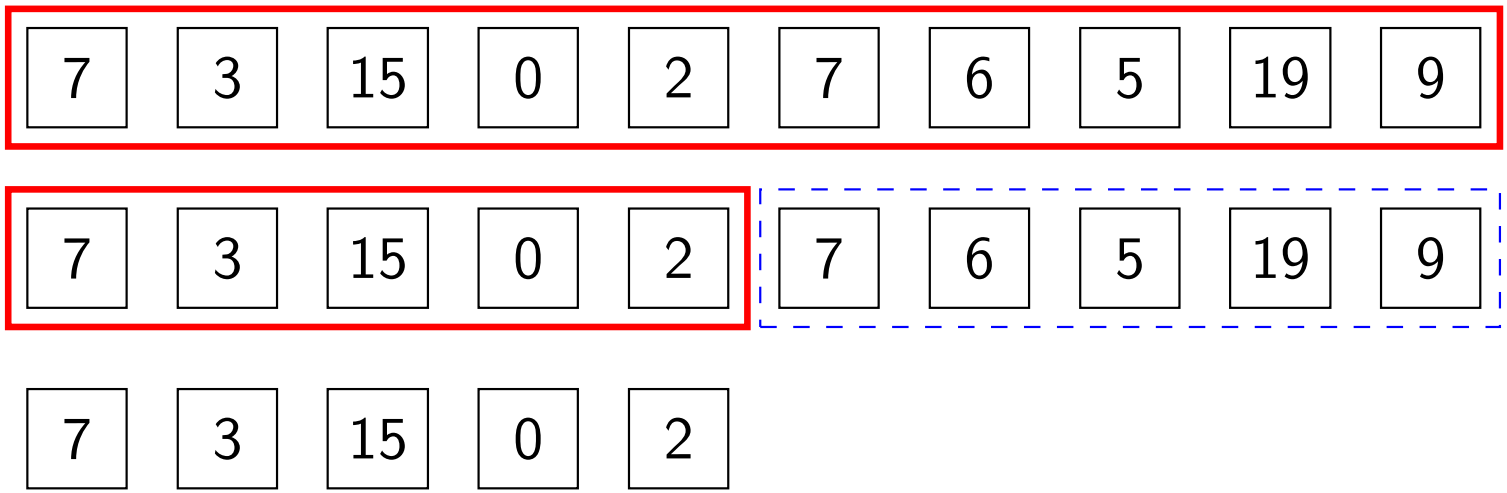
# Exempel Mergesort



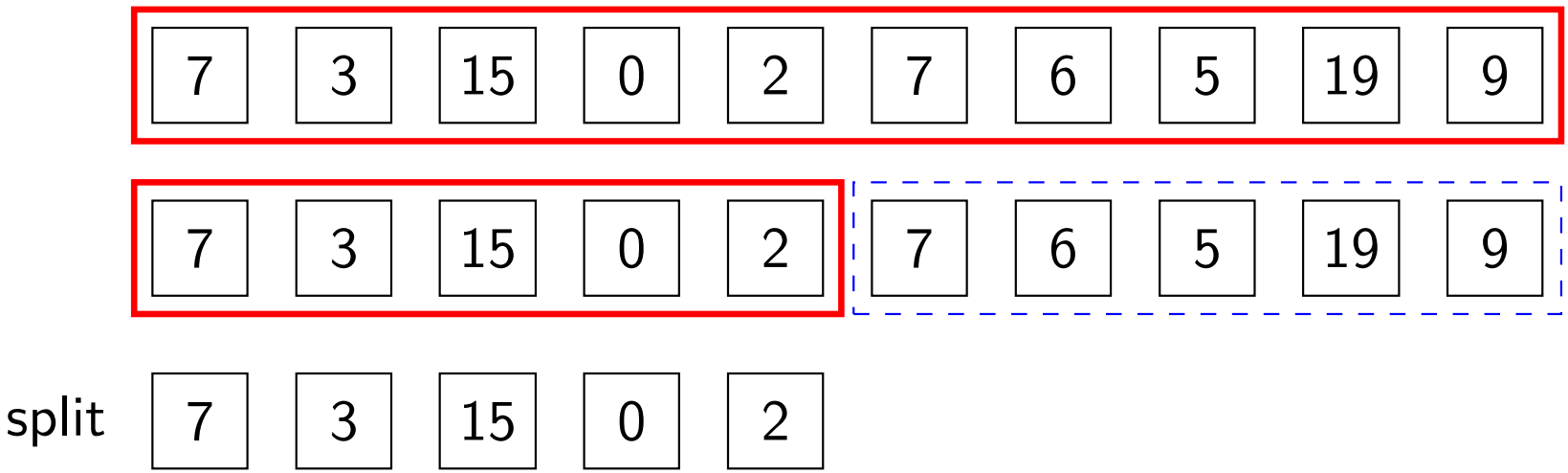
# Exempel Mergesort



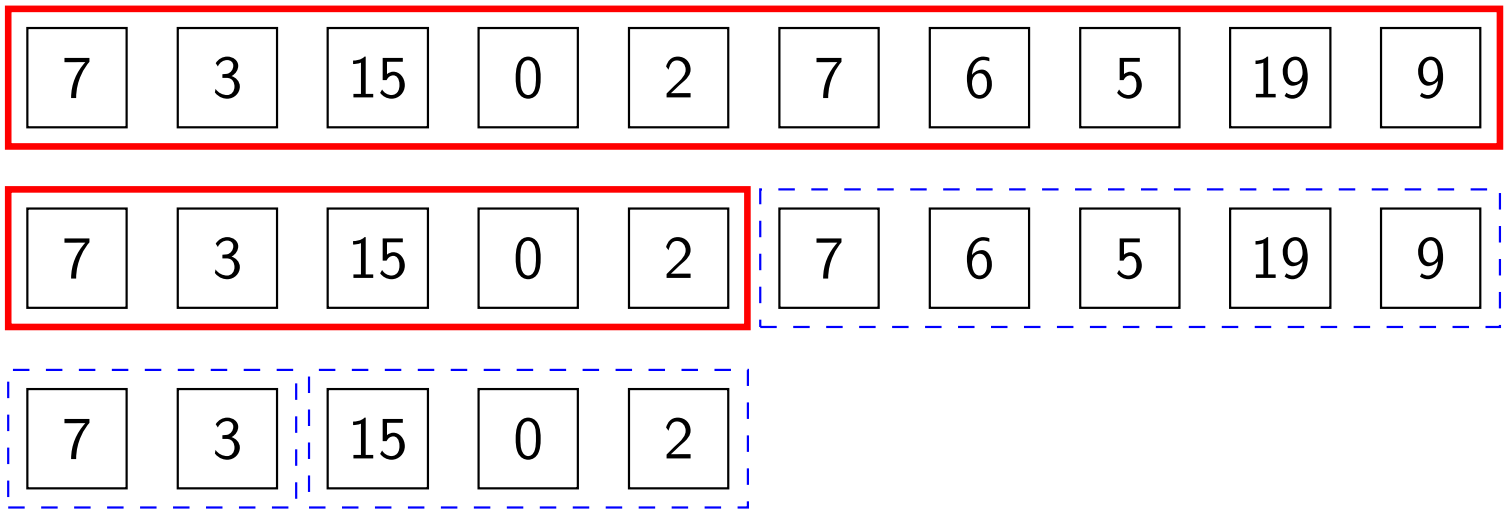
# Exempel Mergesort



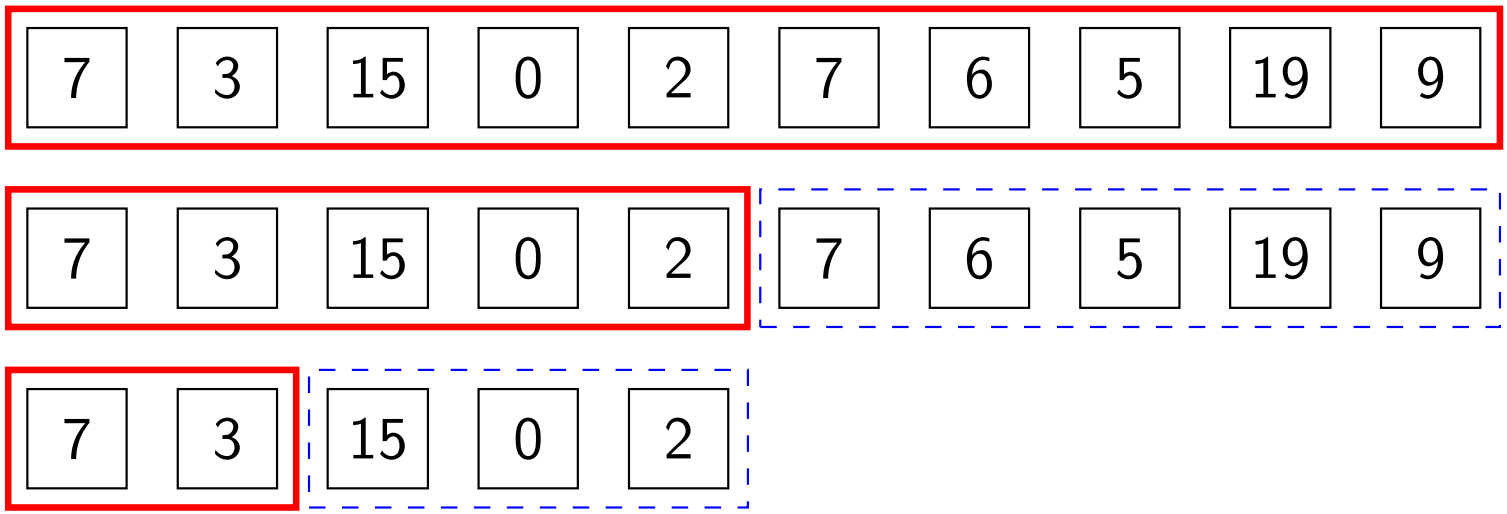
# Exempel Mergesort



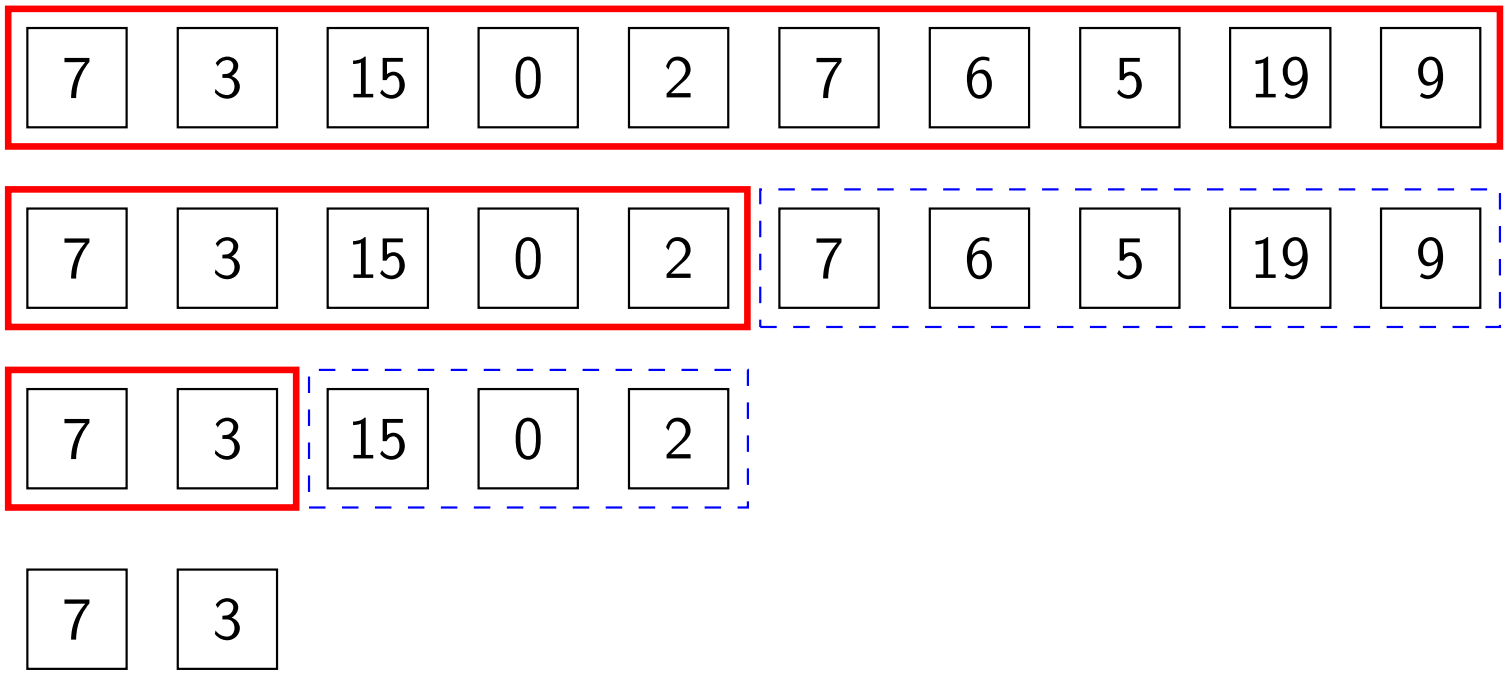
# Exempel Mergesort



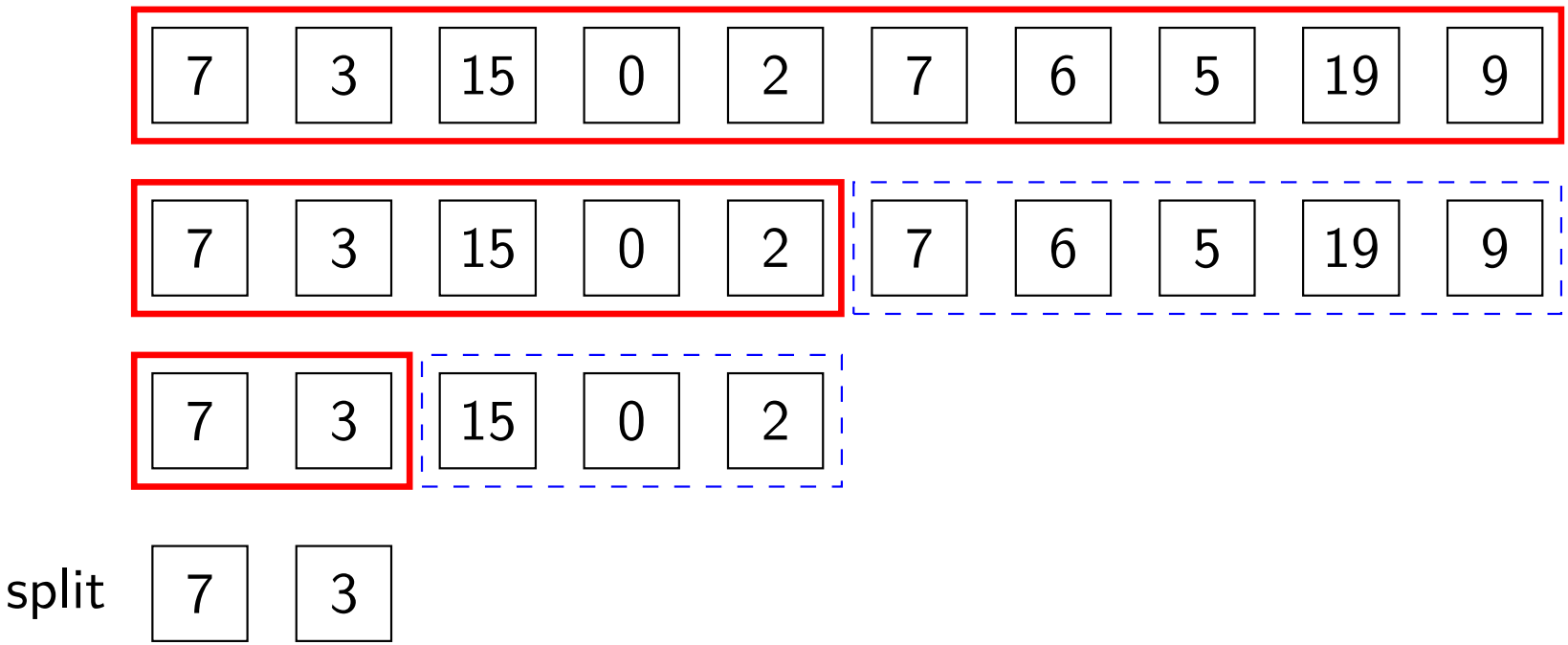
# Exempel Mergesort



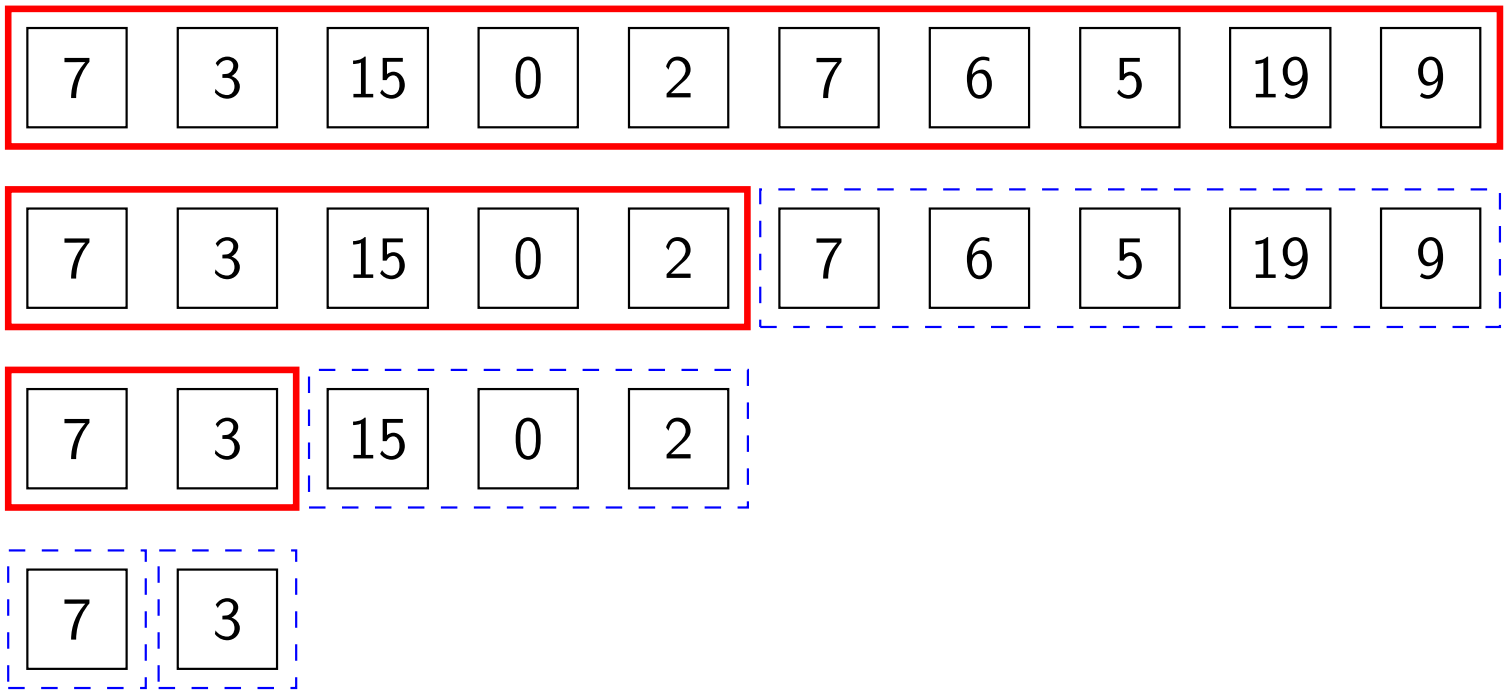
# Exempel Mergesort



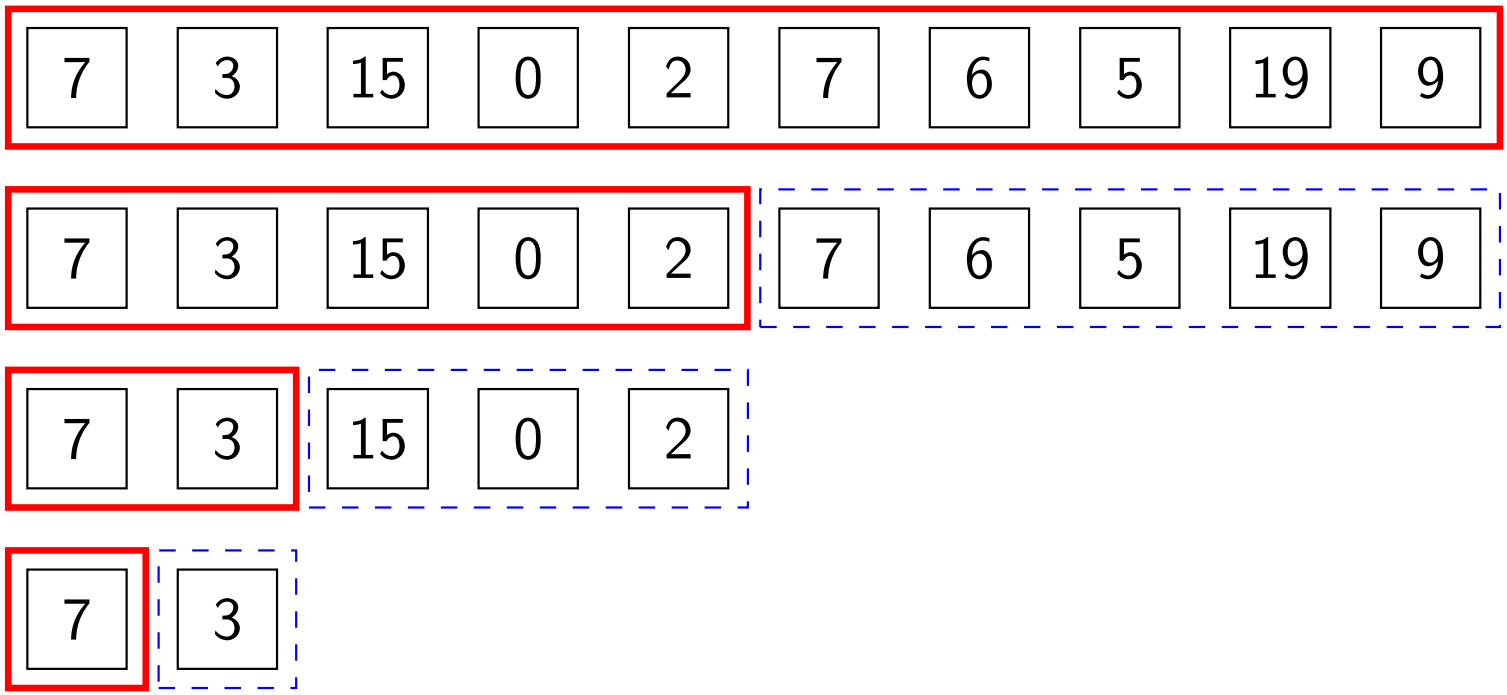
# Exempel Mergesort



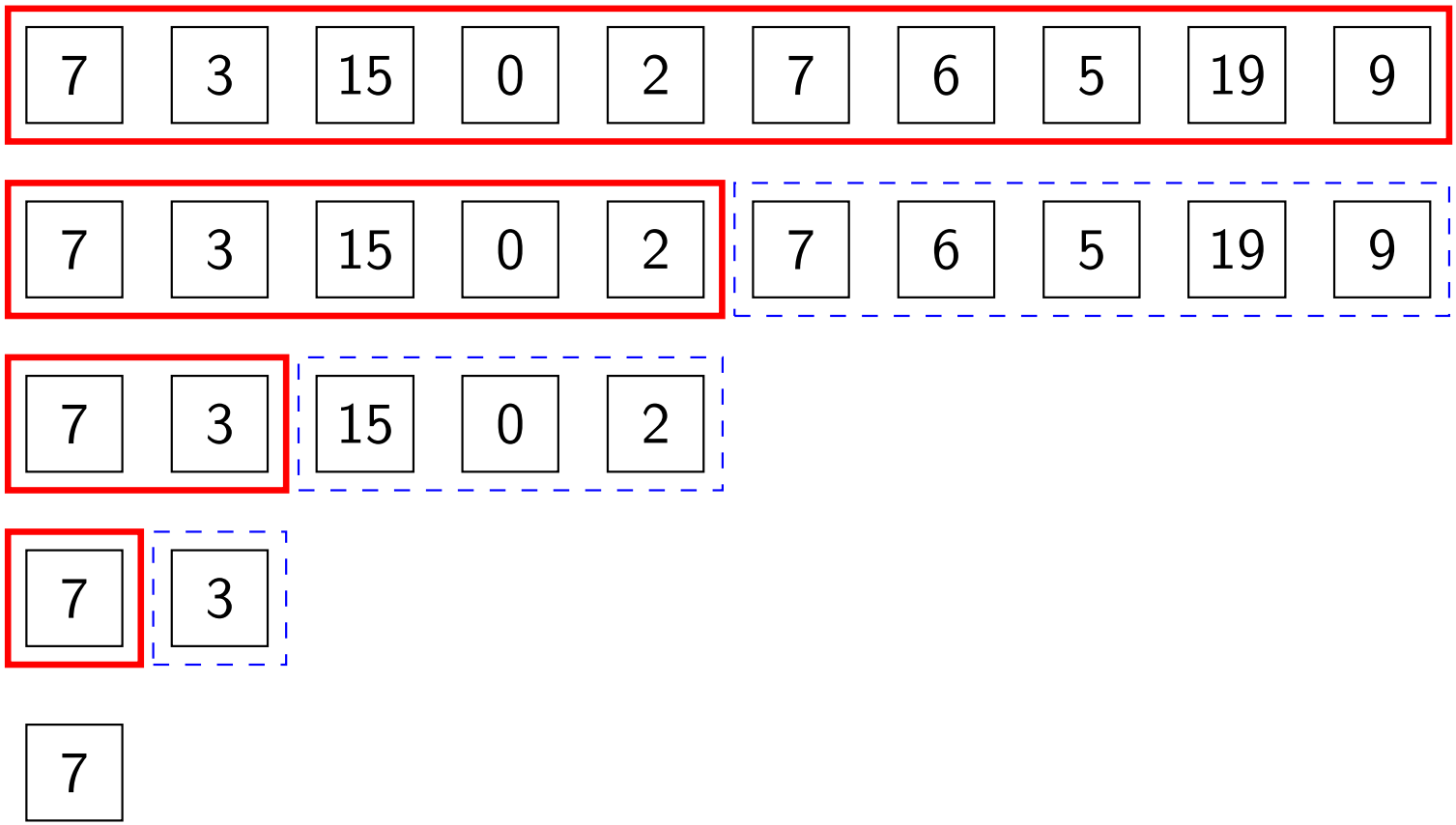
# Exempel Mergesort



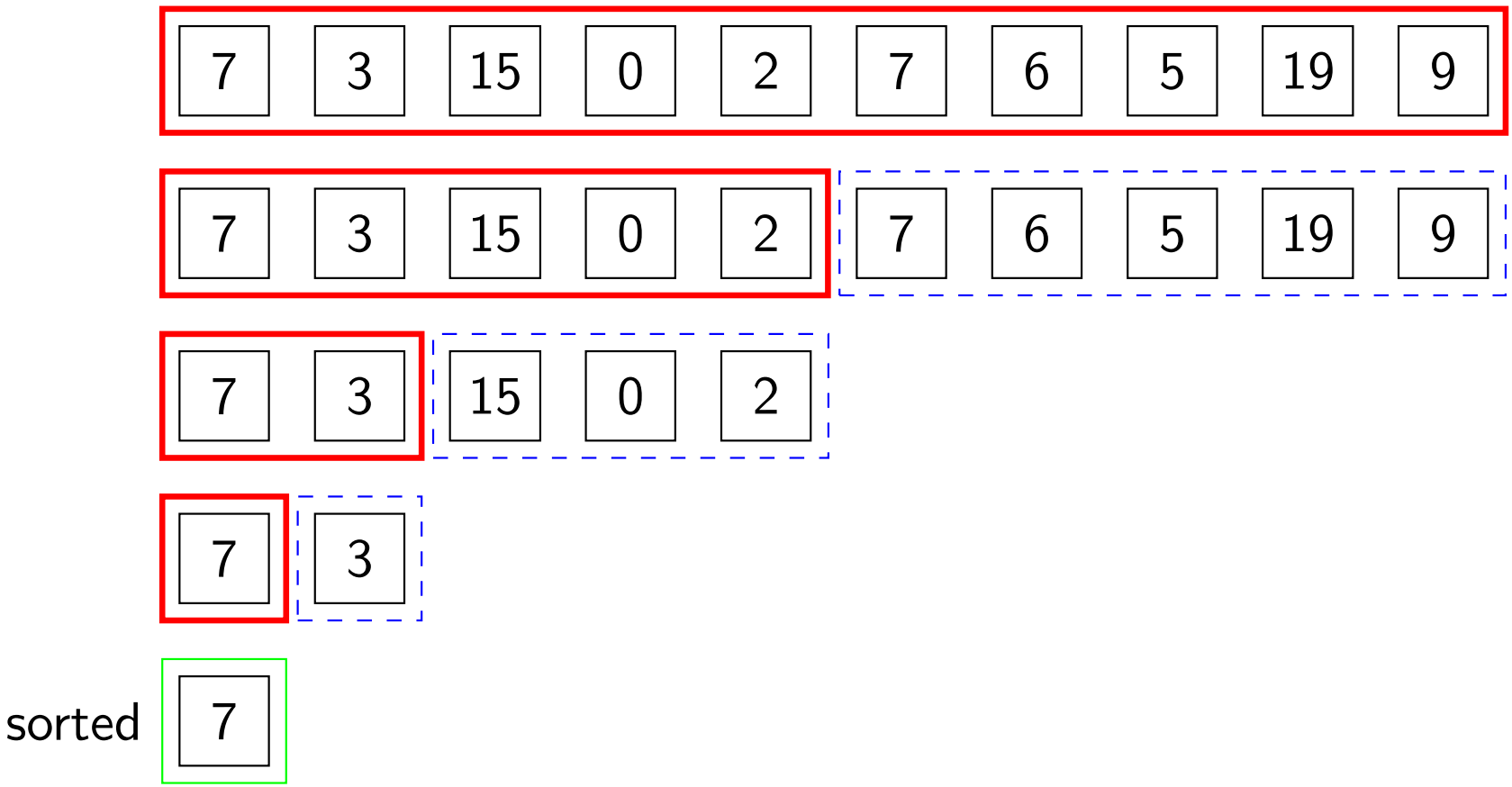
# Exempel Mergesort



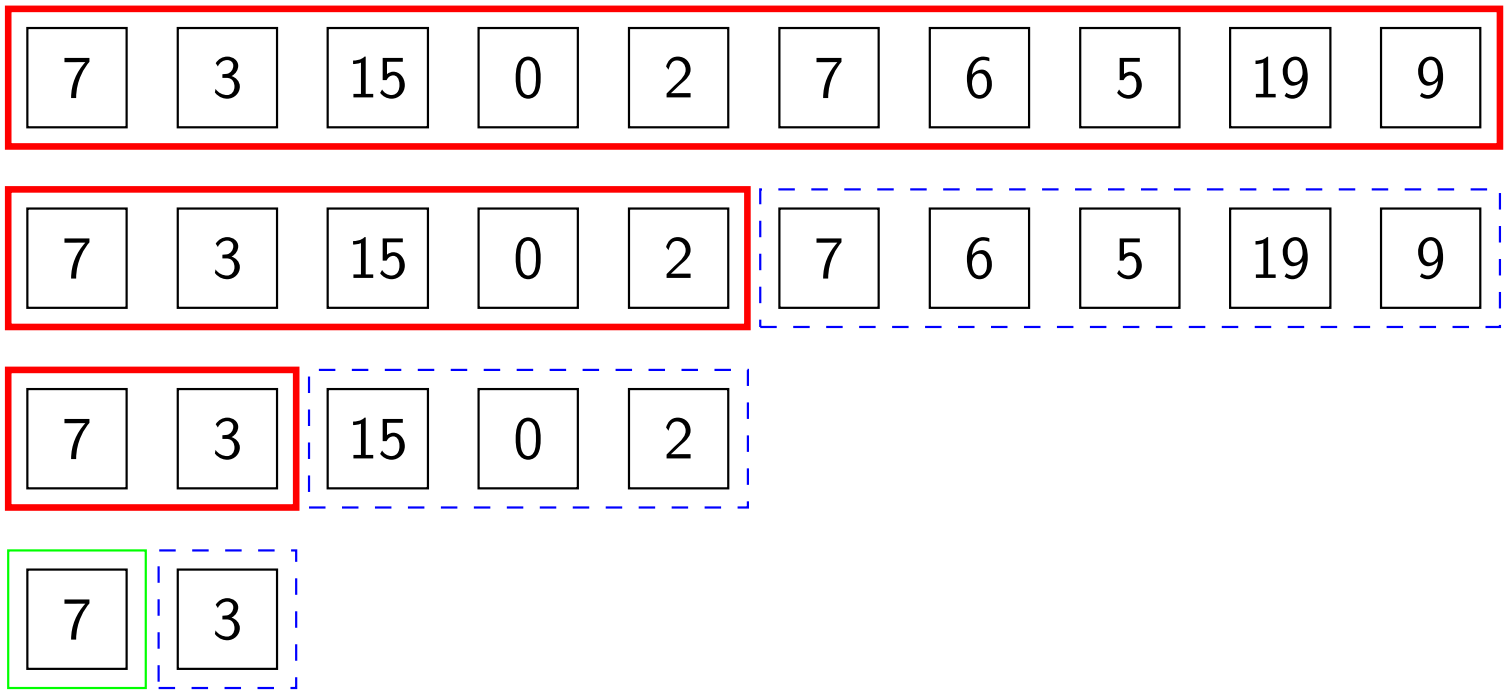
# Exempel Mergesort



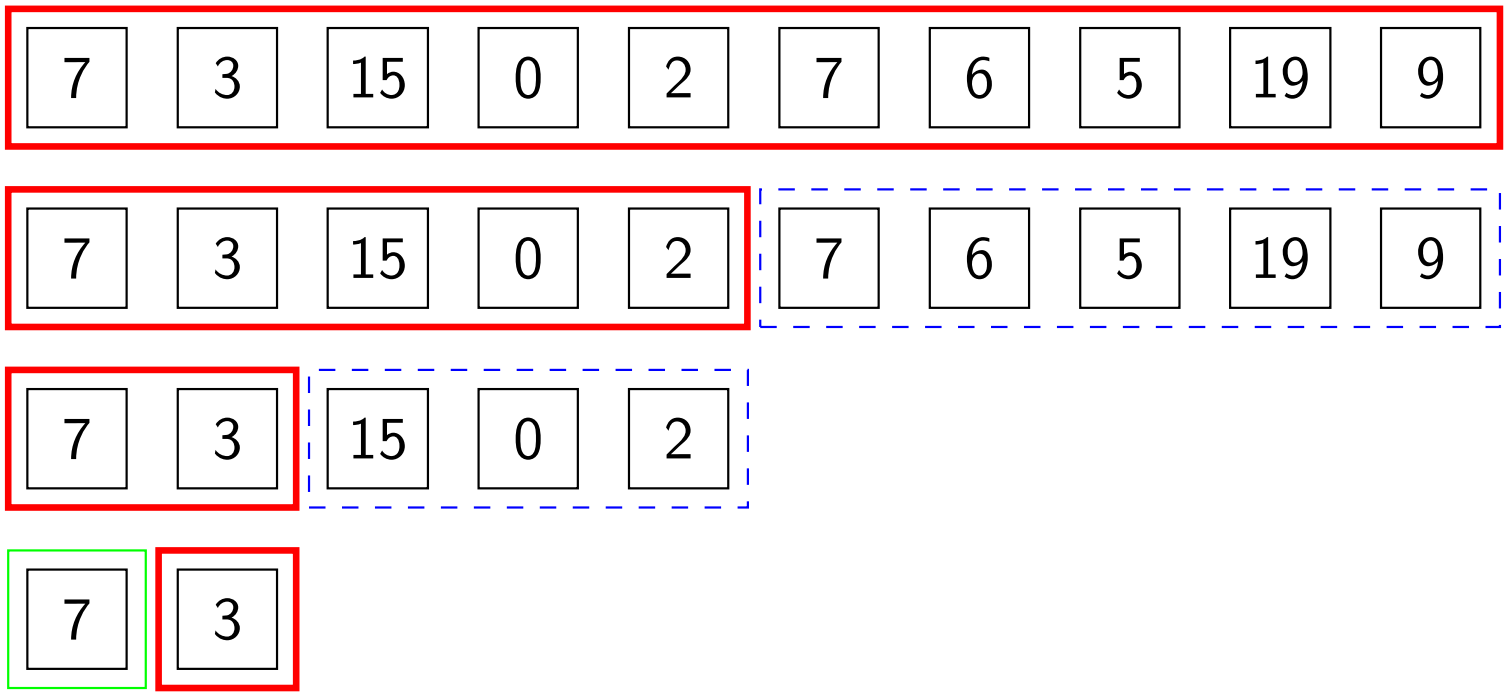
# Exempel Mergesort



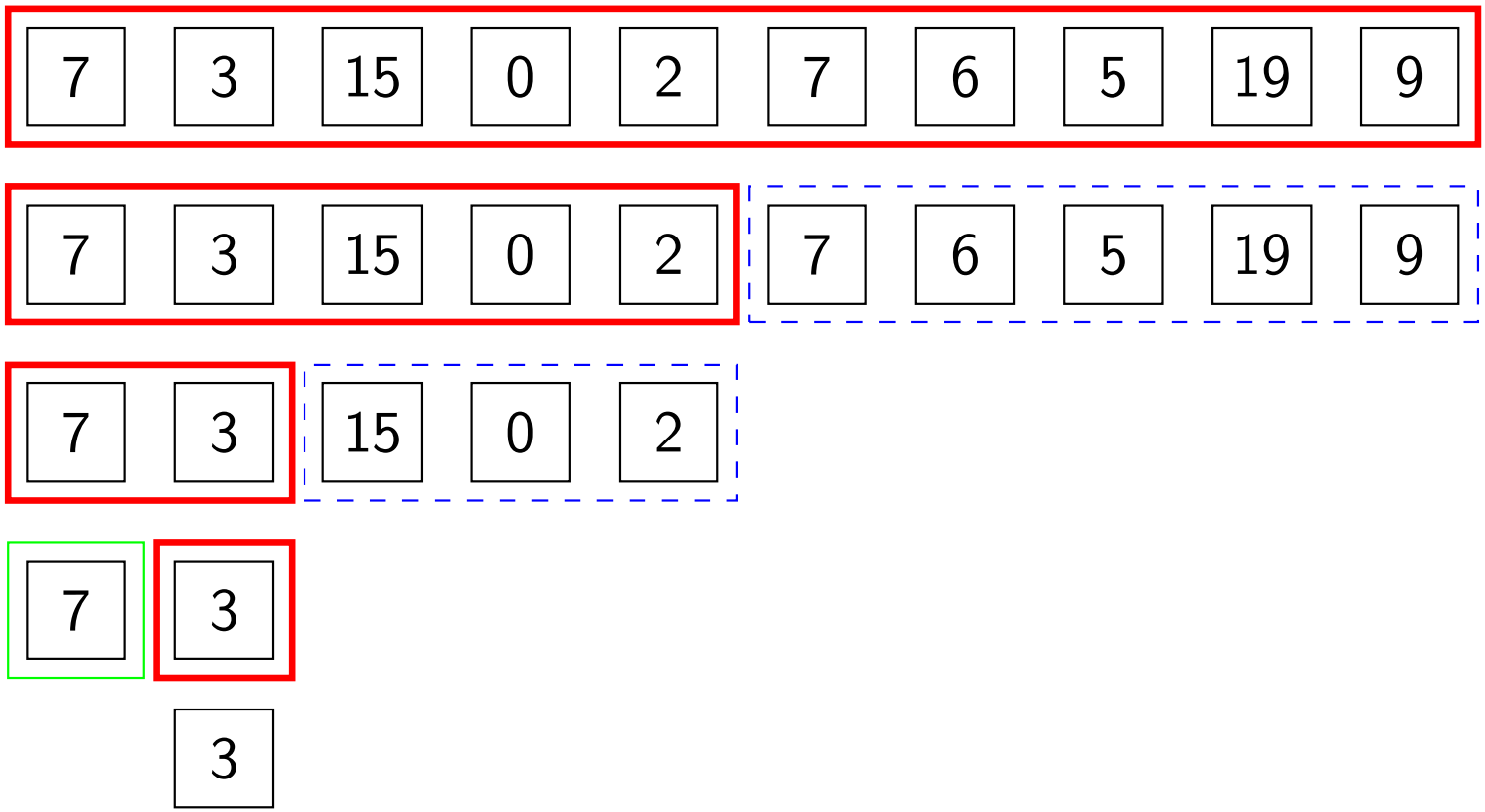
# Exempel Mergesort



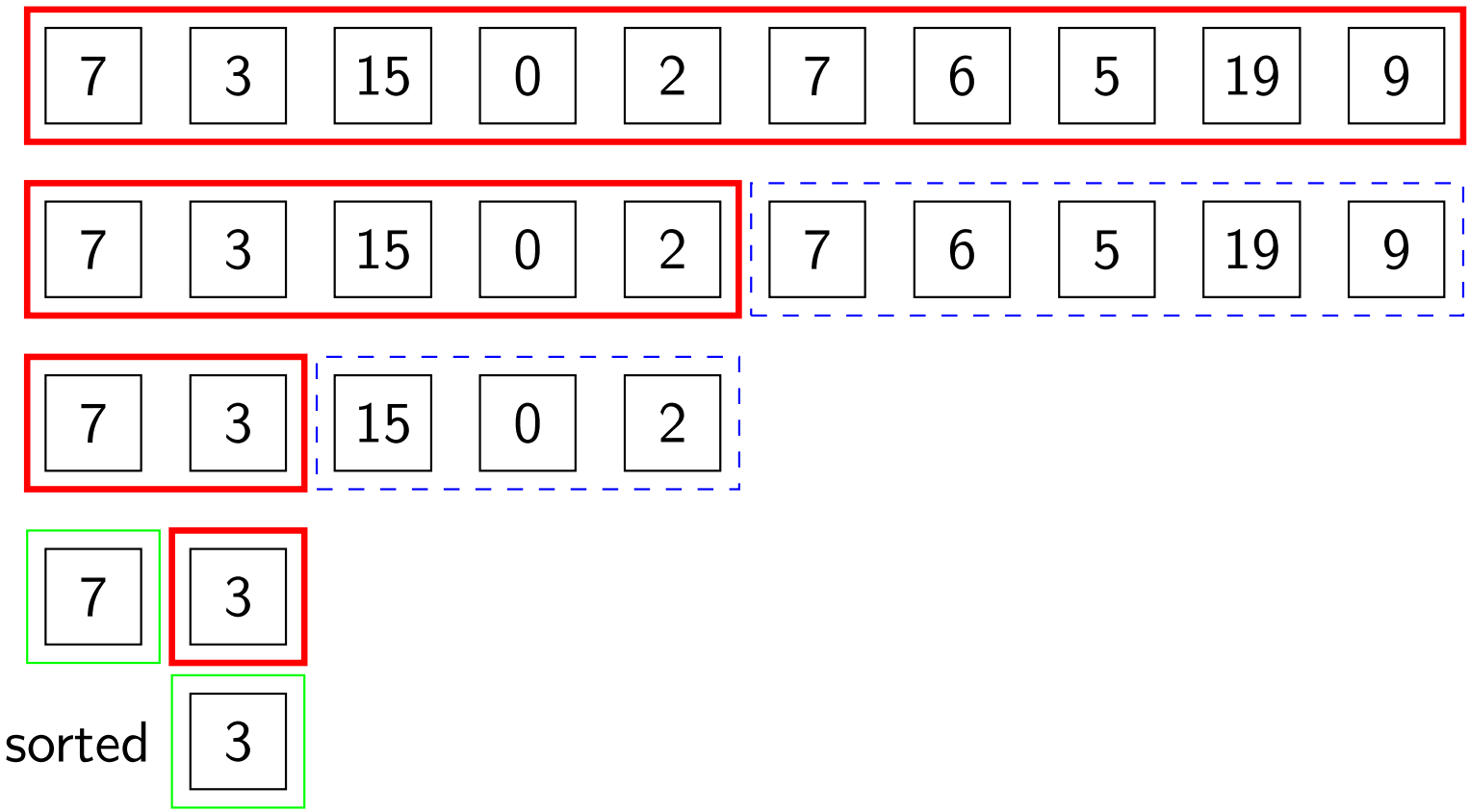
# Exempel Mergesort



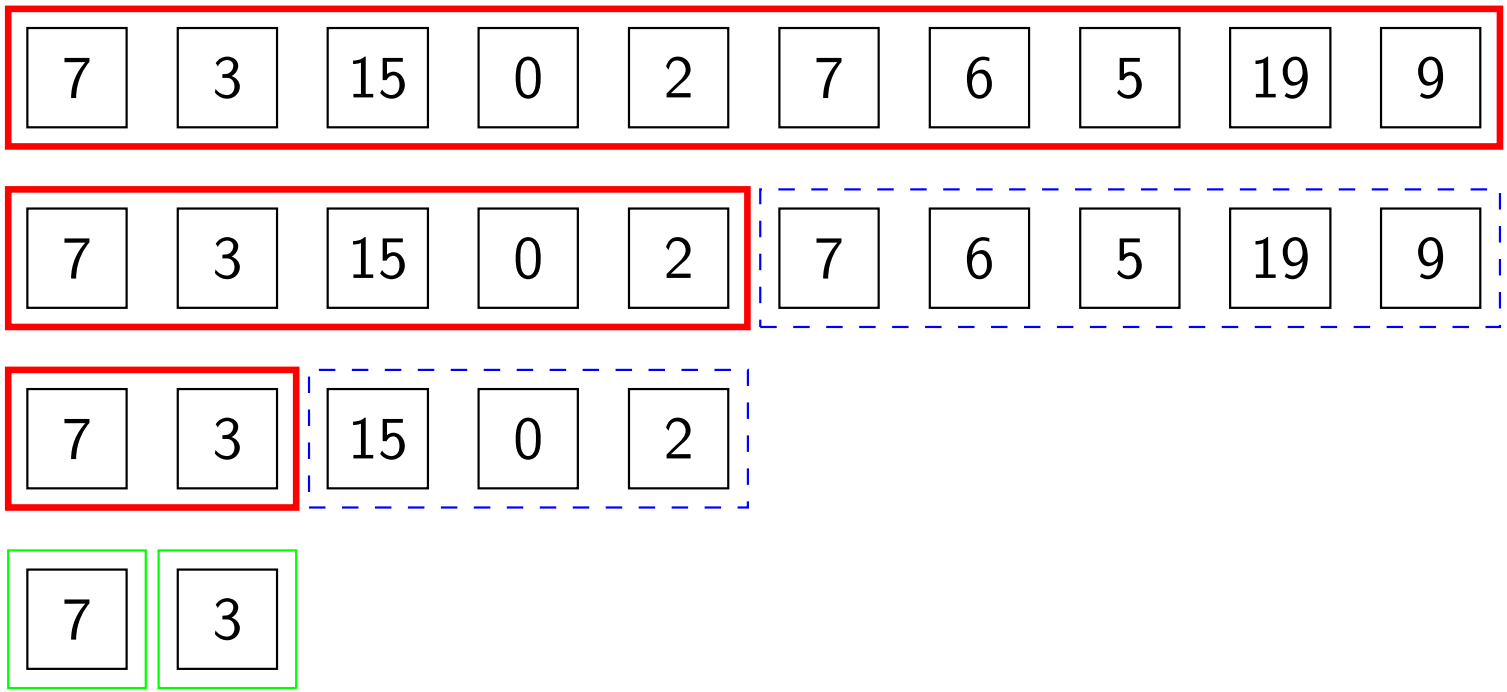
# Exempel Mergesort



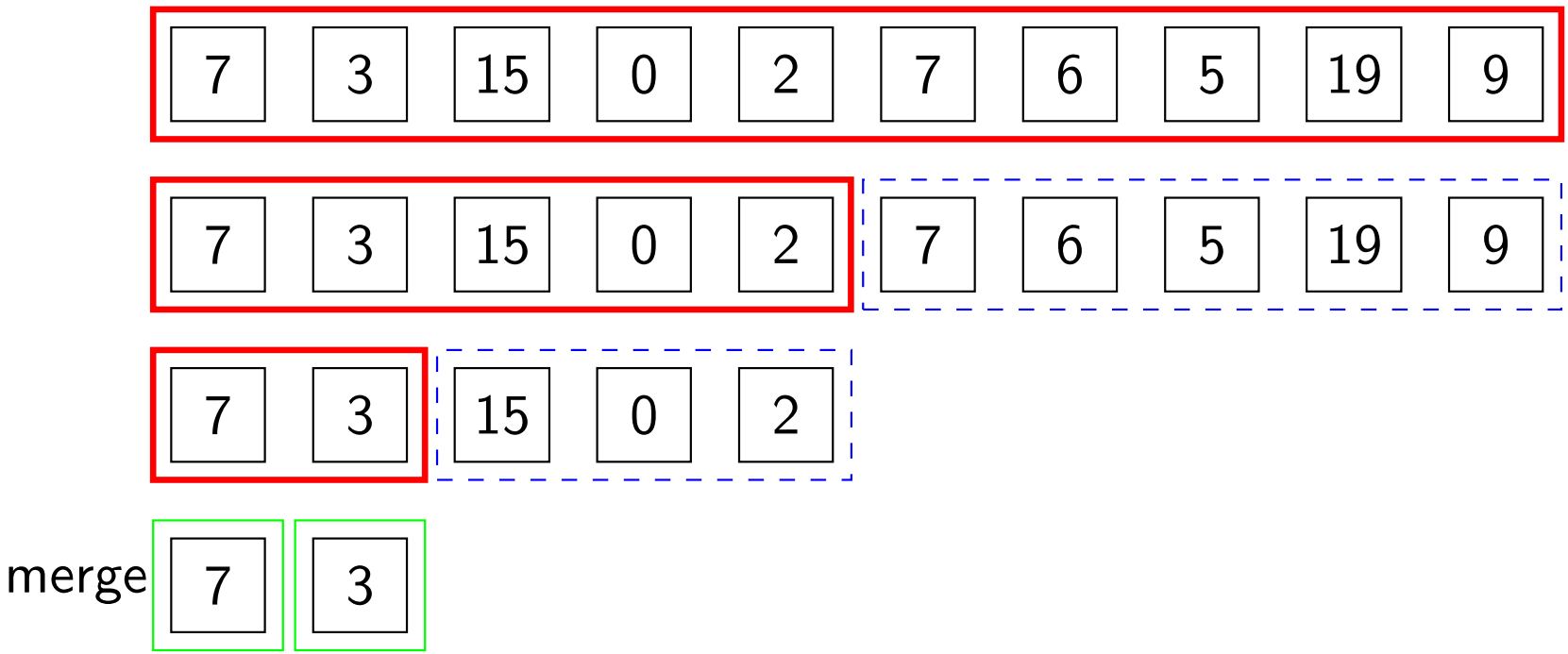
# Exempel Mergesort



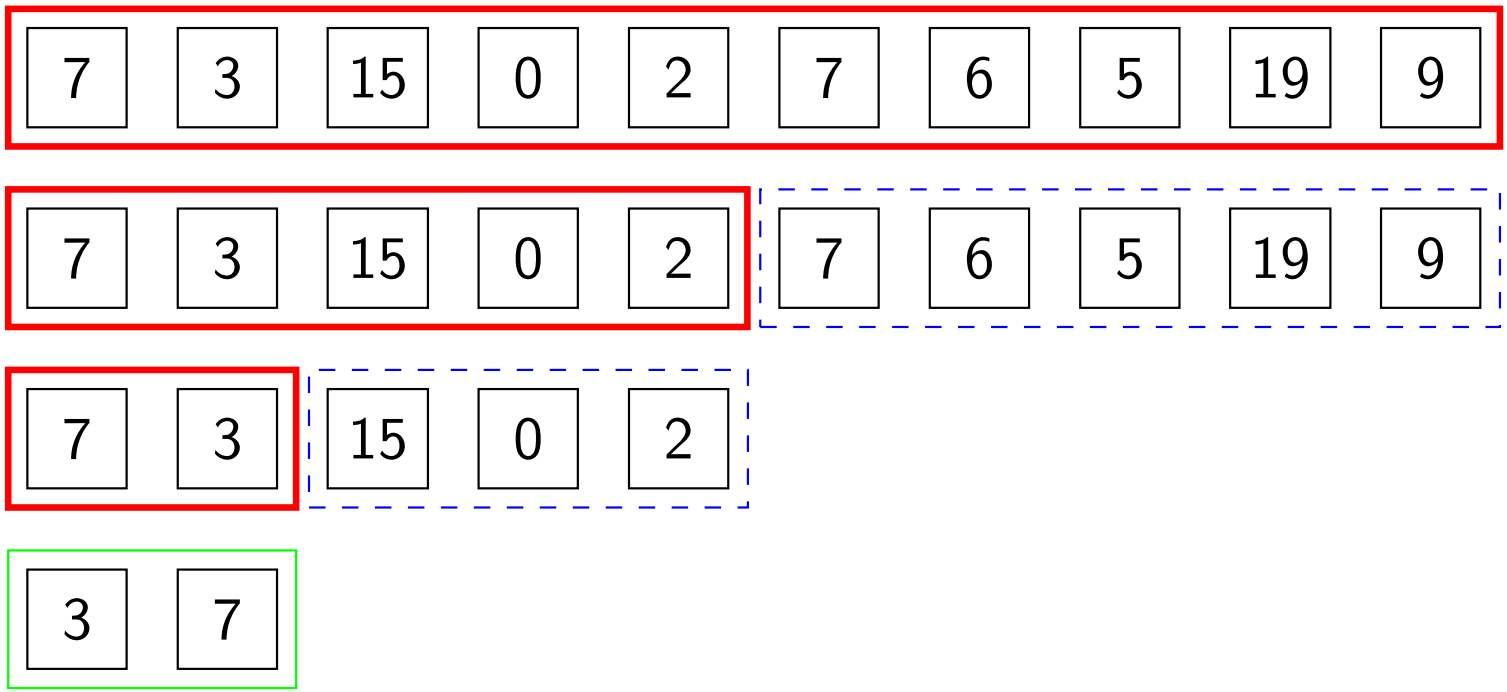
# Exempel Mergesort



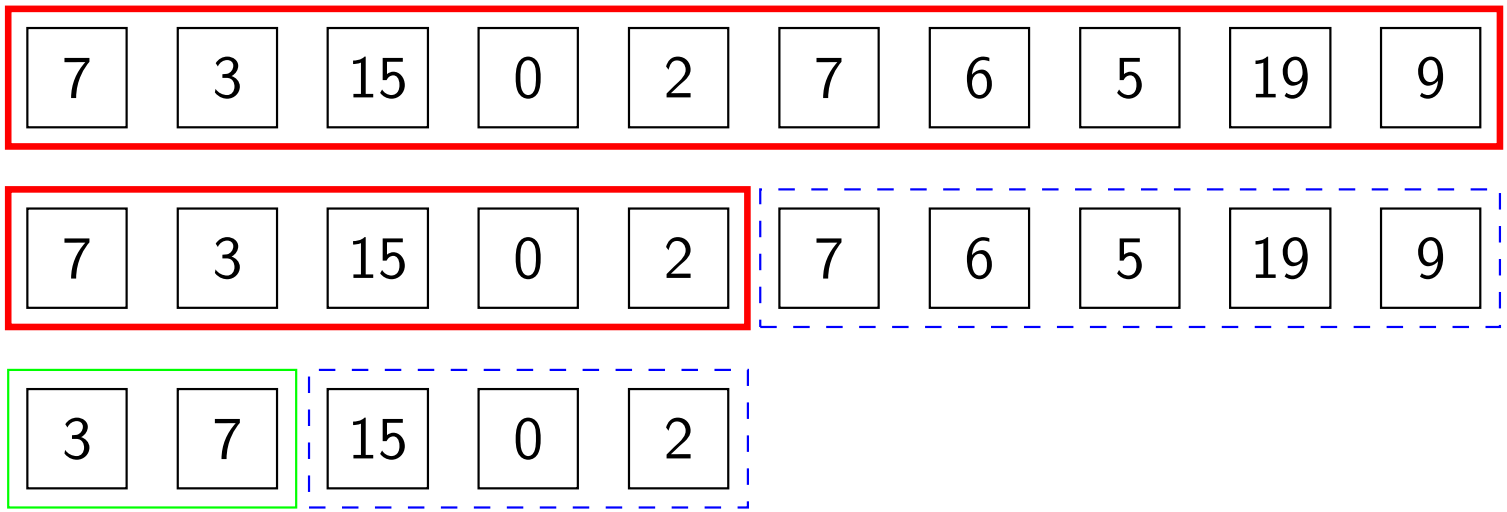
# Exempel Mergesort



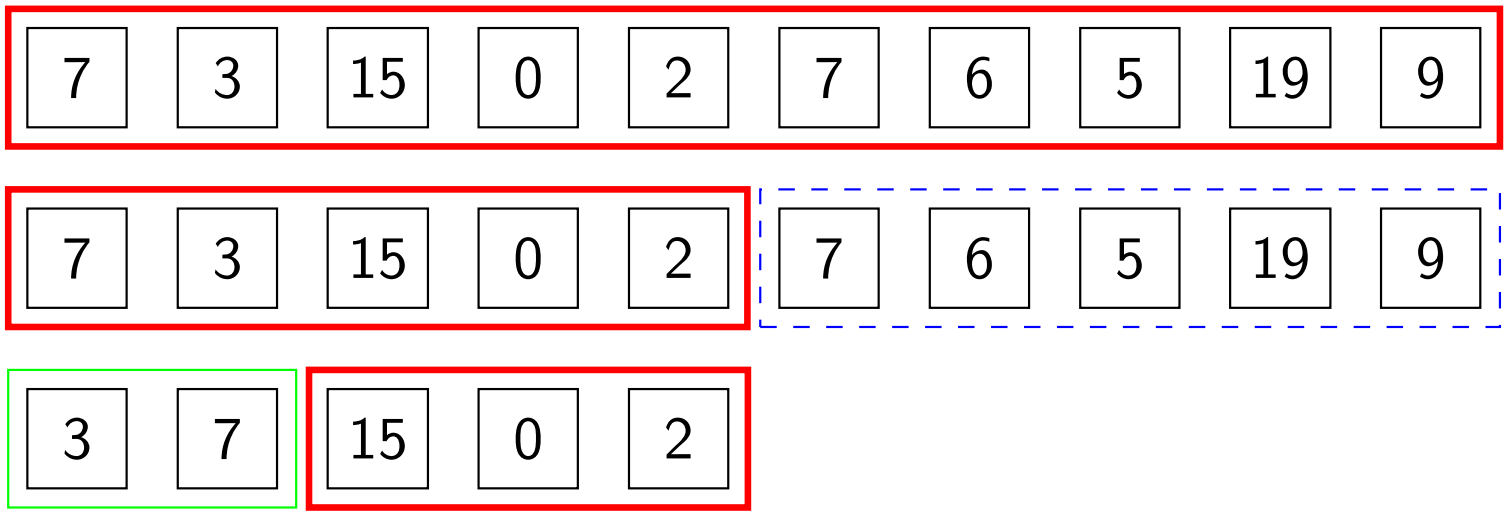
# Exempel Mergesort



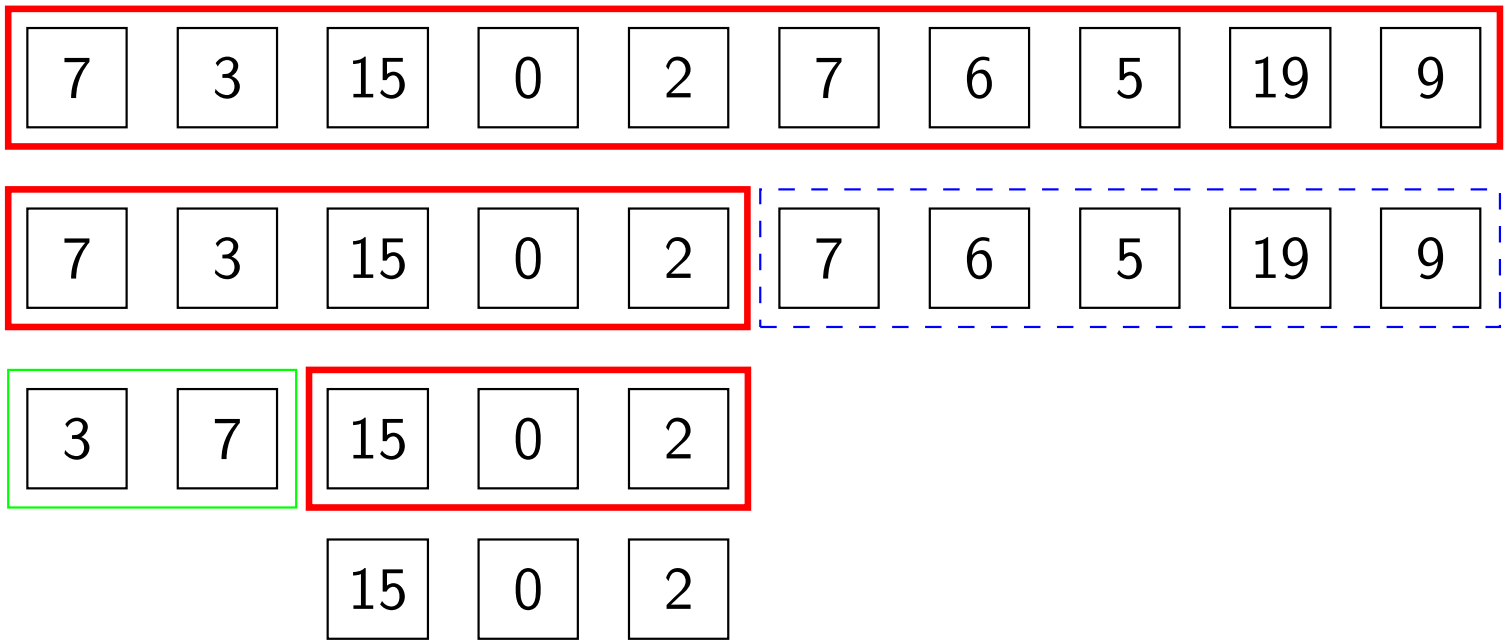
# Exempel Mergesort



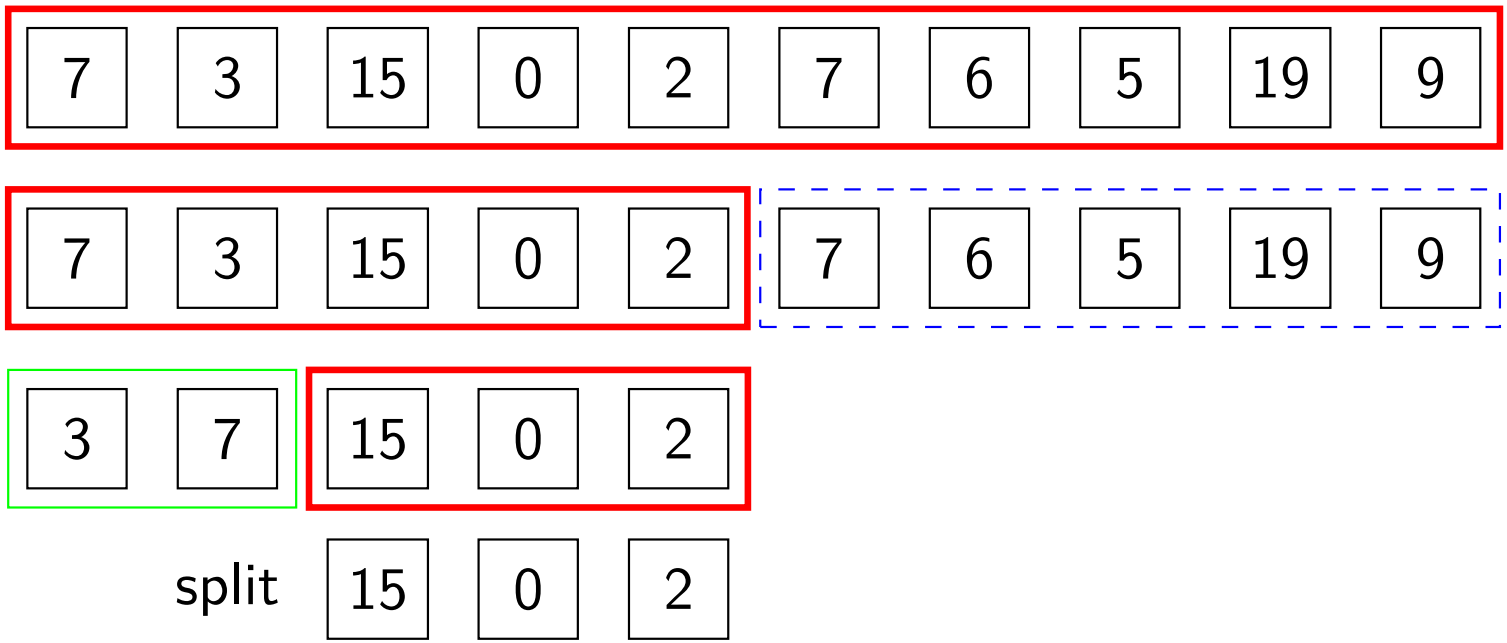
# Exempel Mergesort



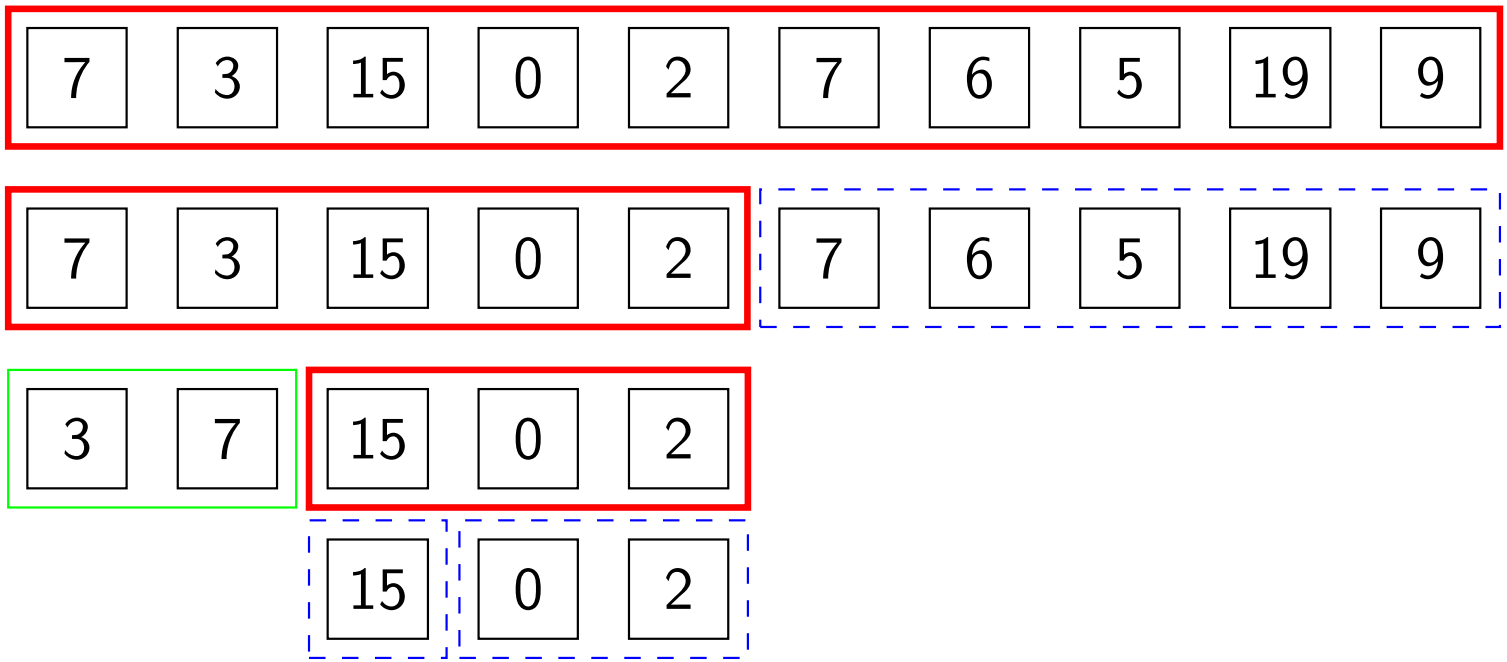
# Exempel Mergesort



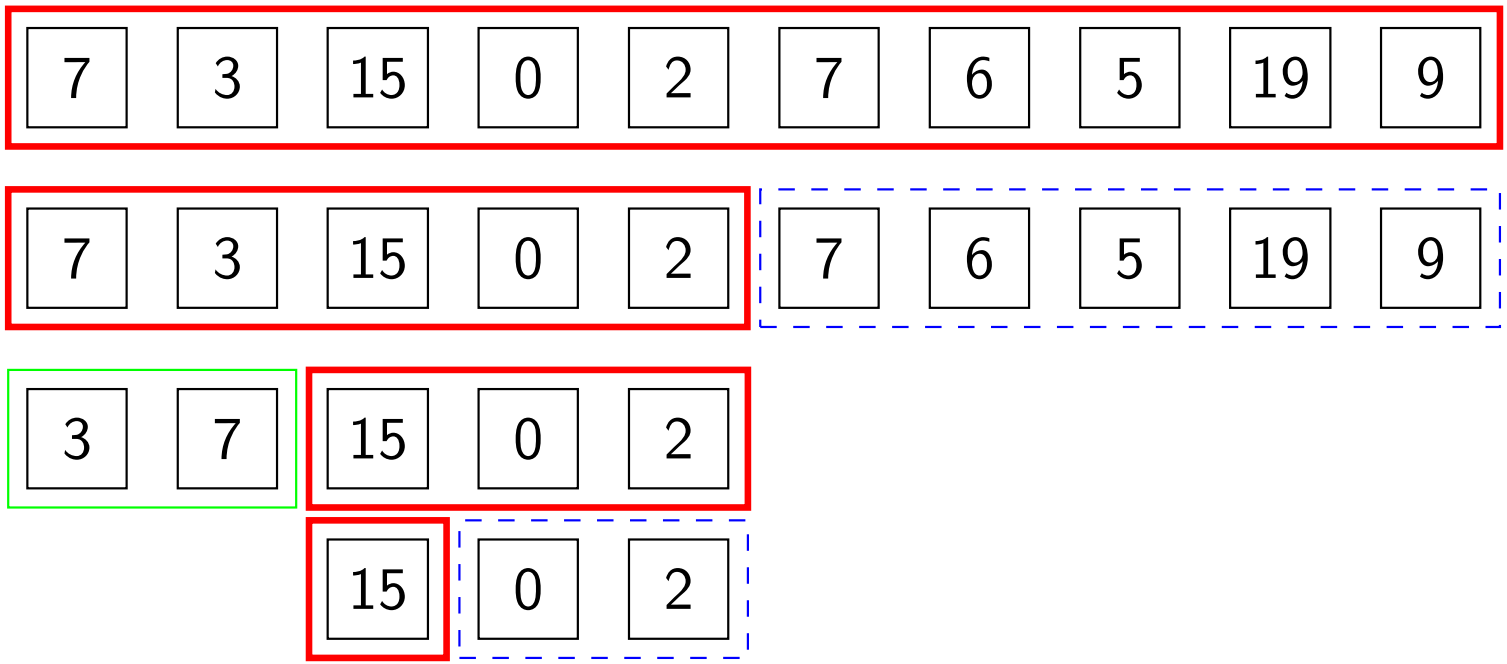
# Exempel Mergesort



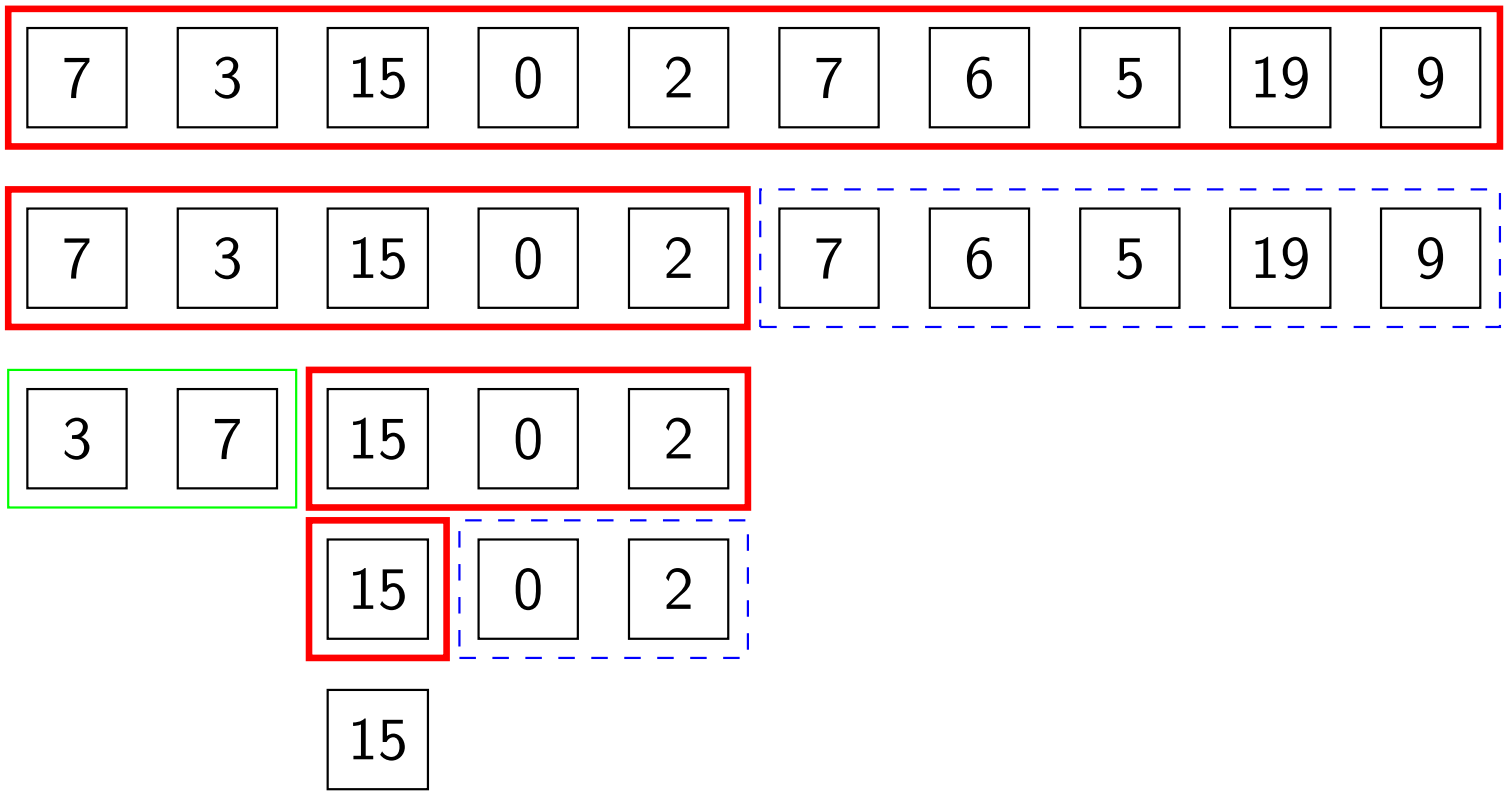
# Exempel Mergesort



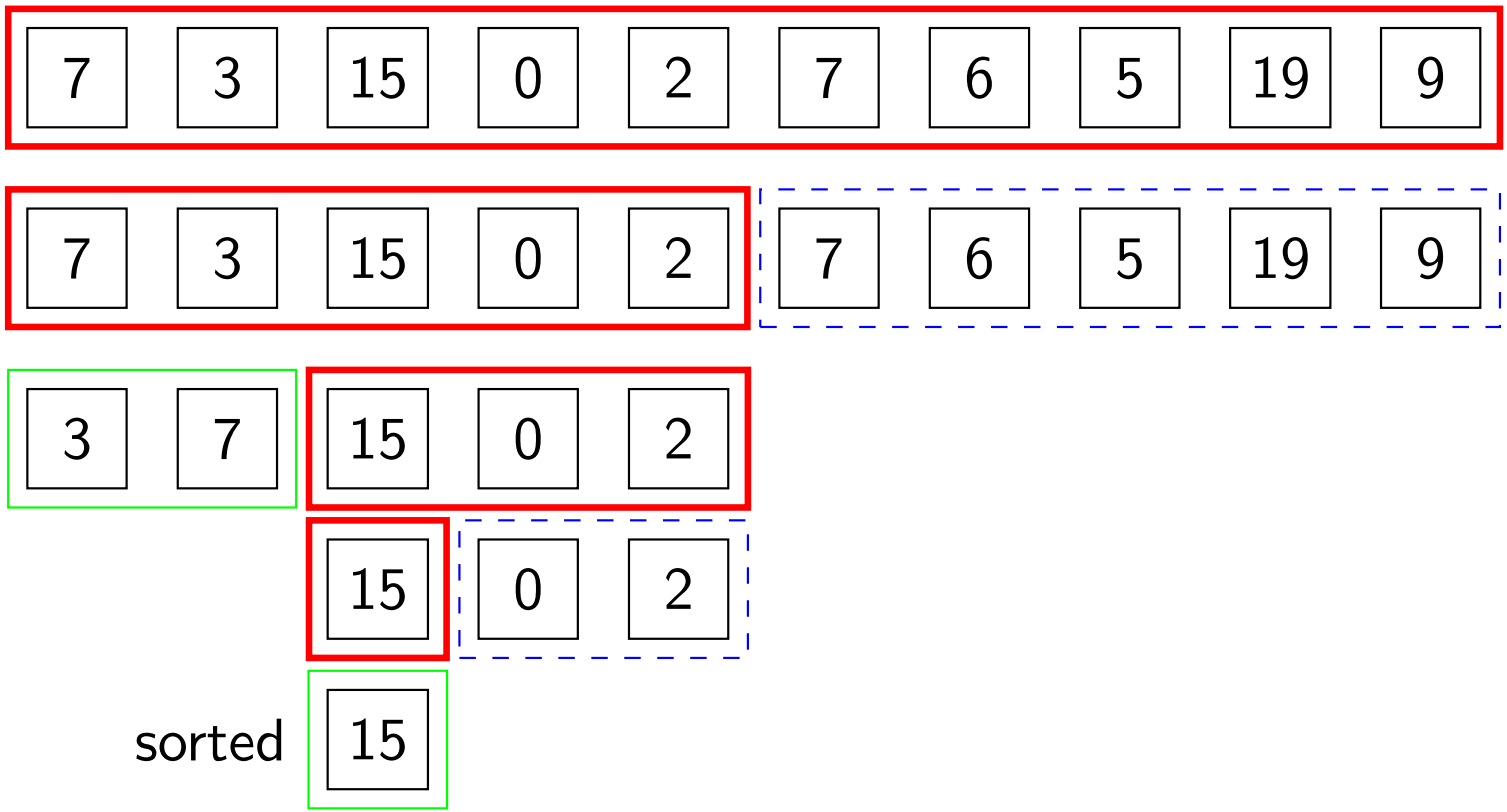
# Exempel Mergesort



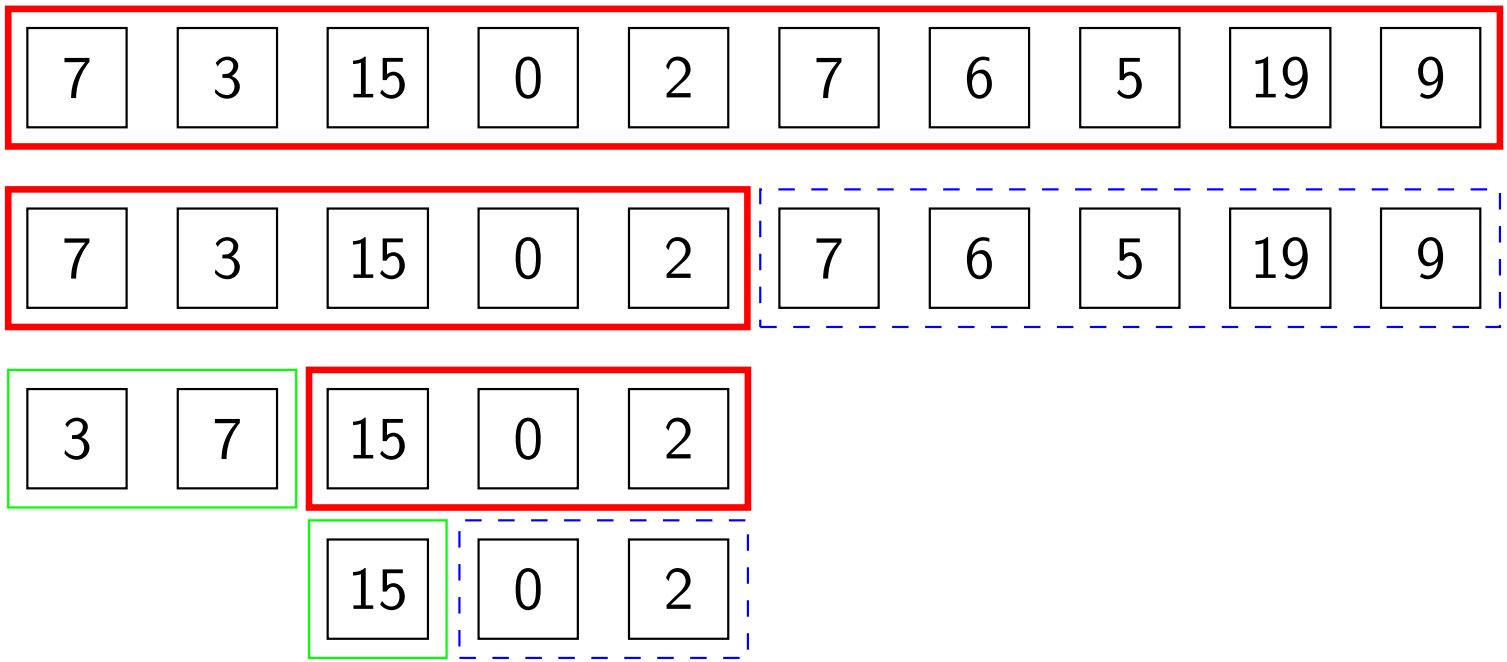
# Exempel Mergesort



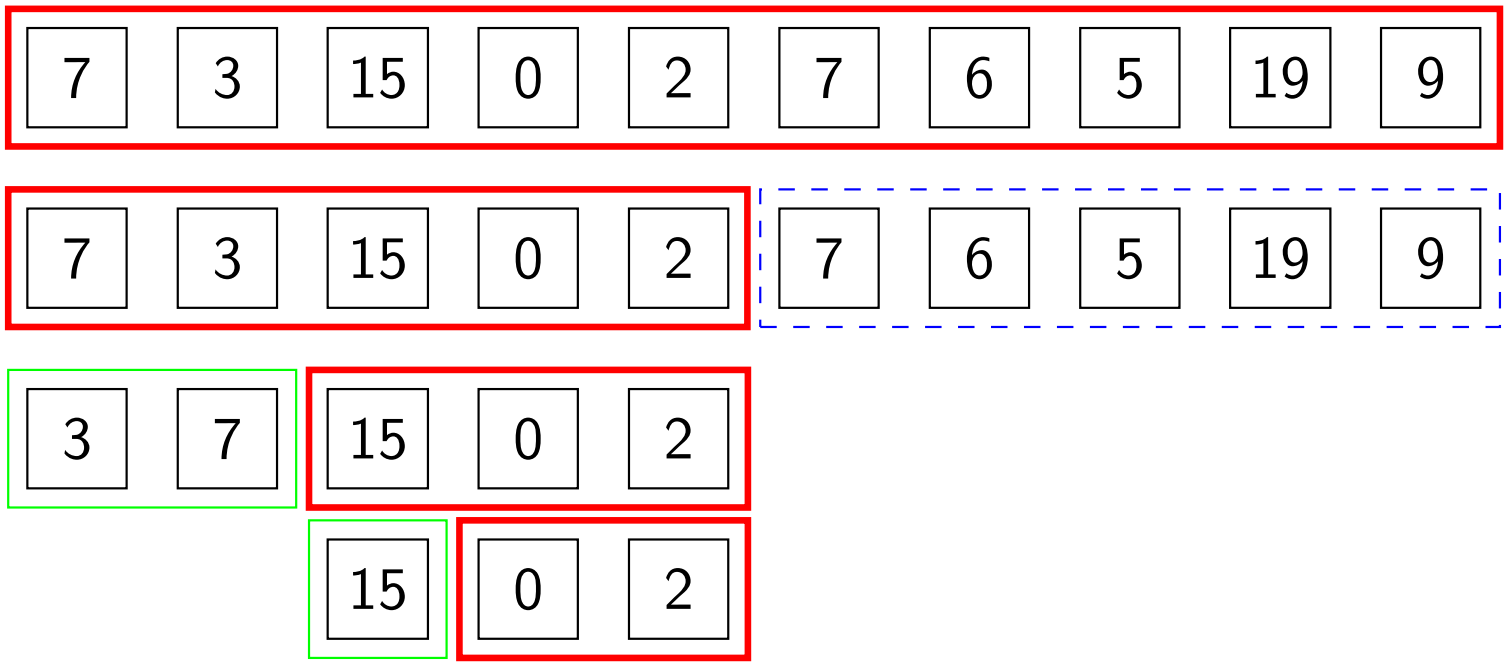
# Exempel Mergesort



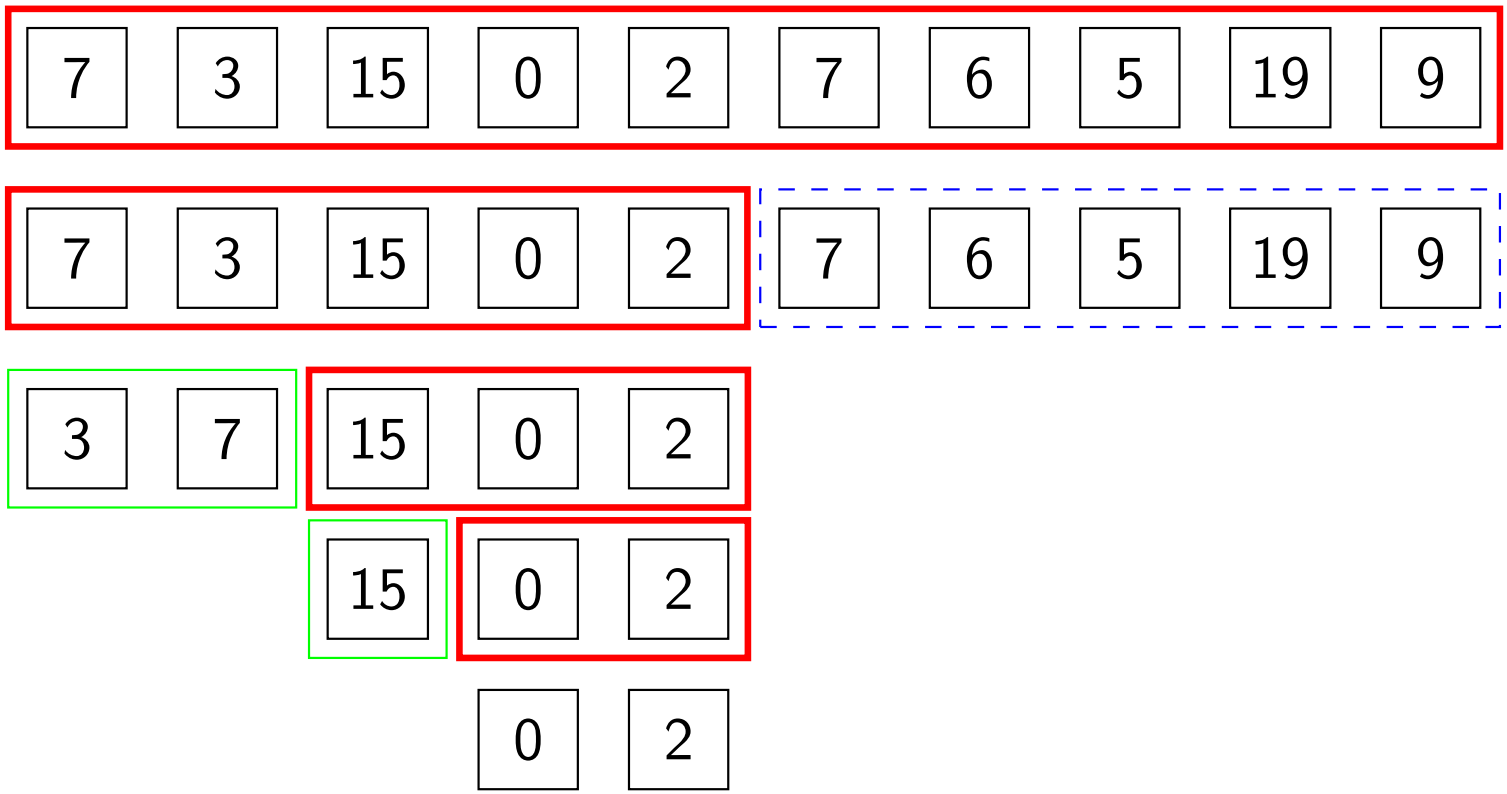
# Exempel Mergesort



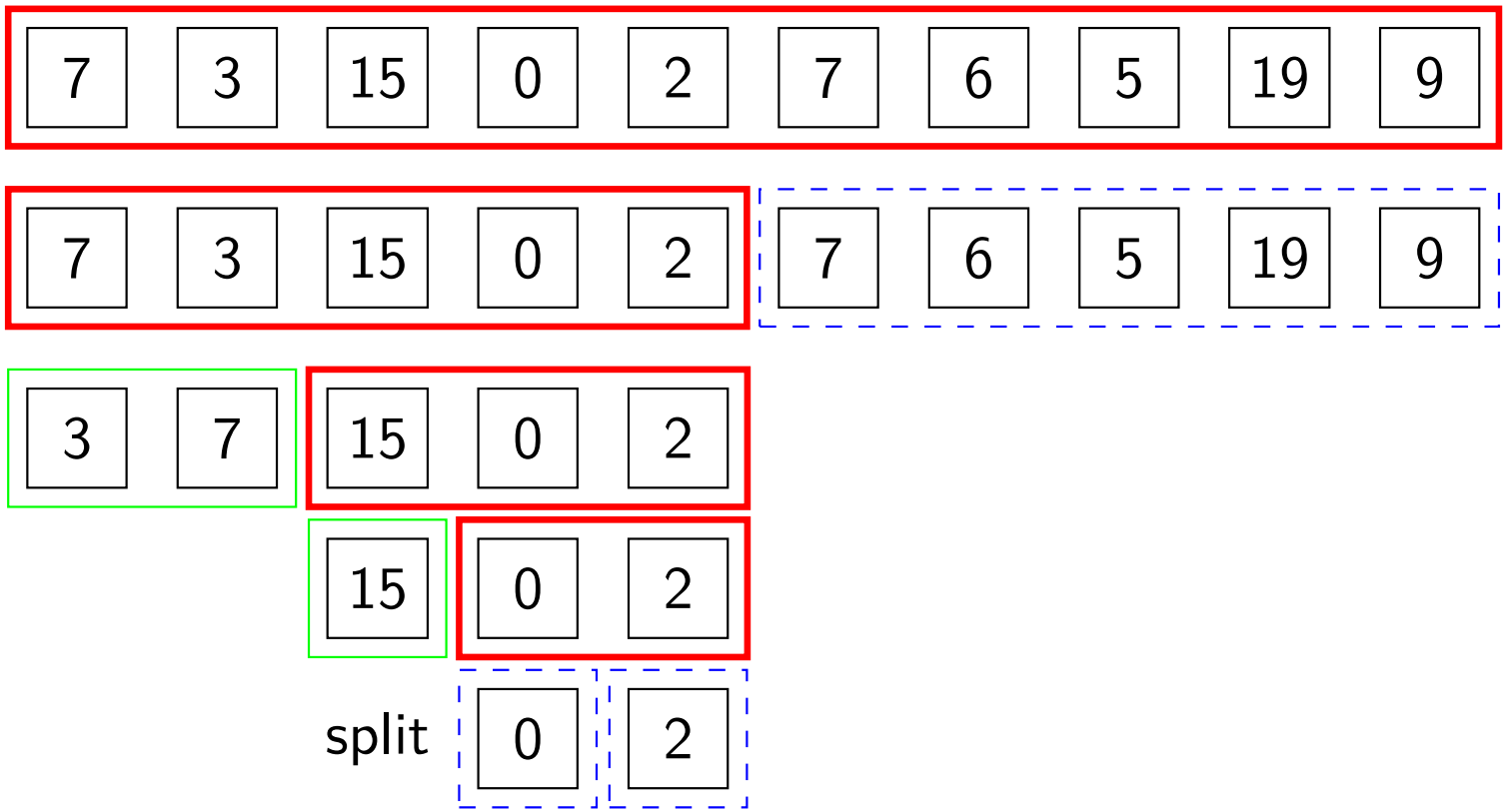
# Exempel Mergesort



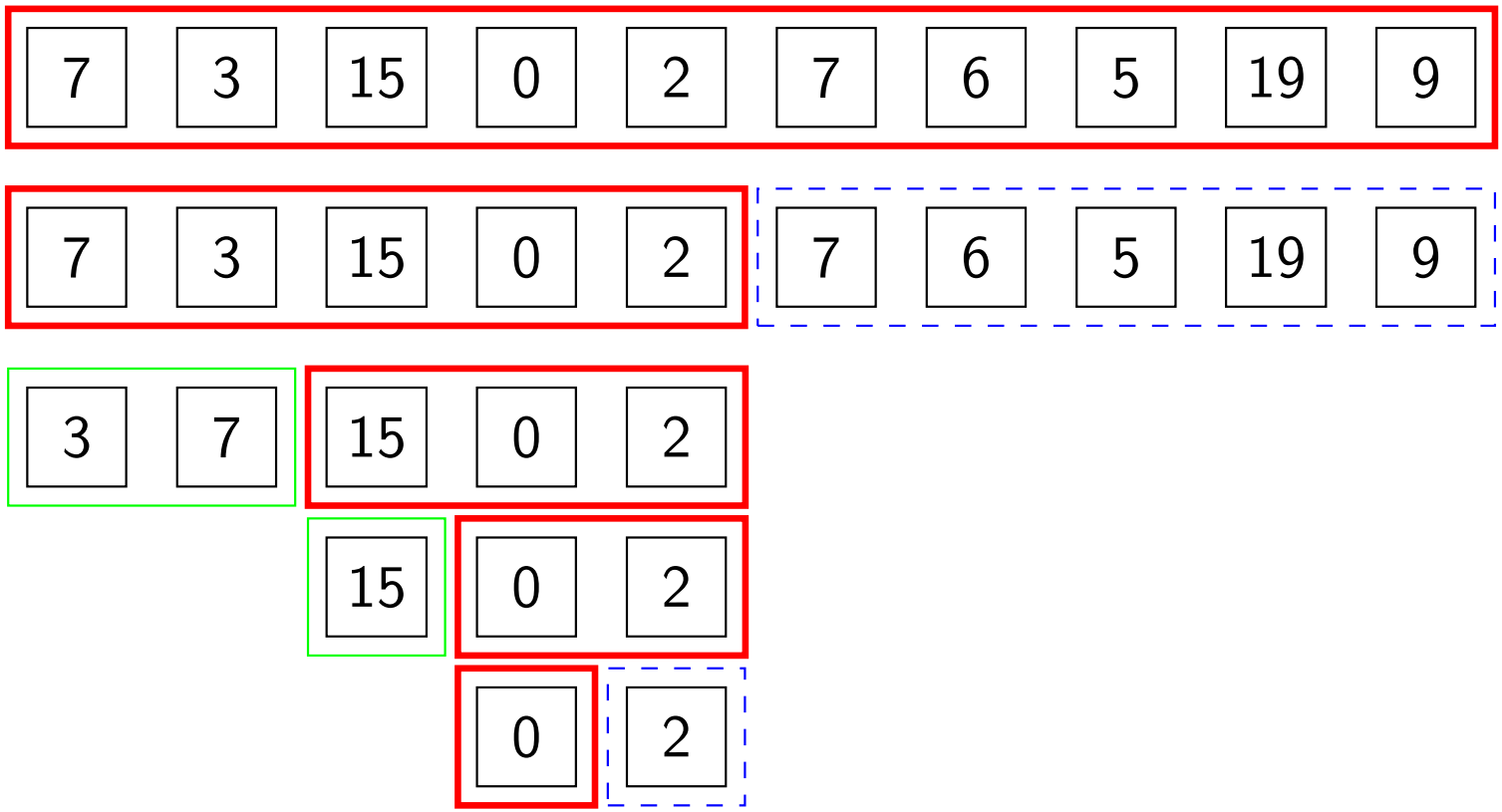
# Exempel Mergesort



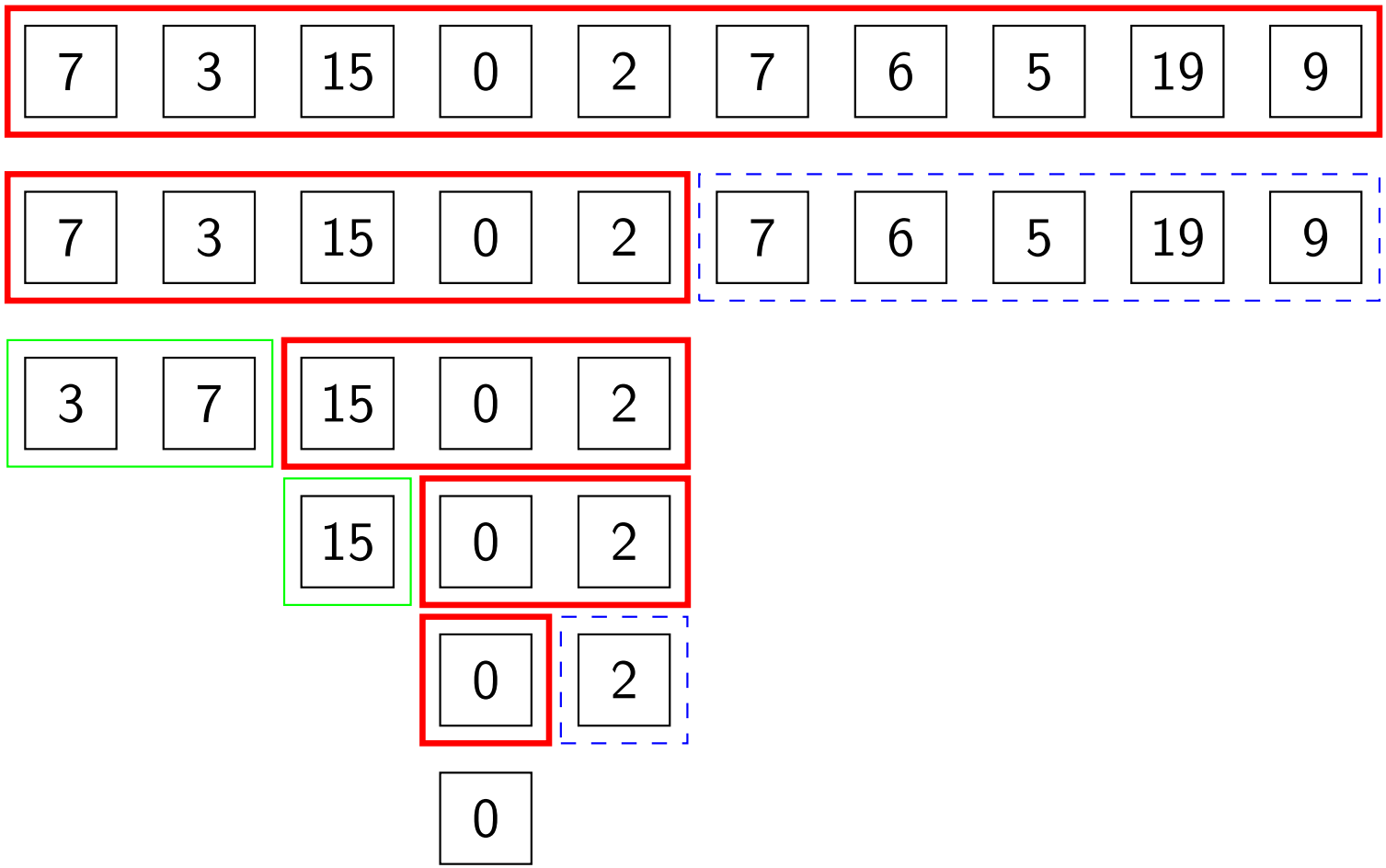
# Exempel Mergesort



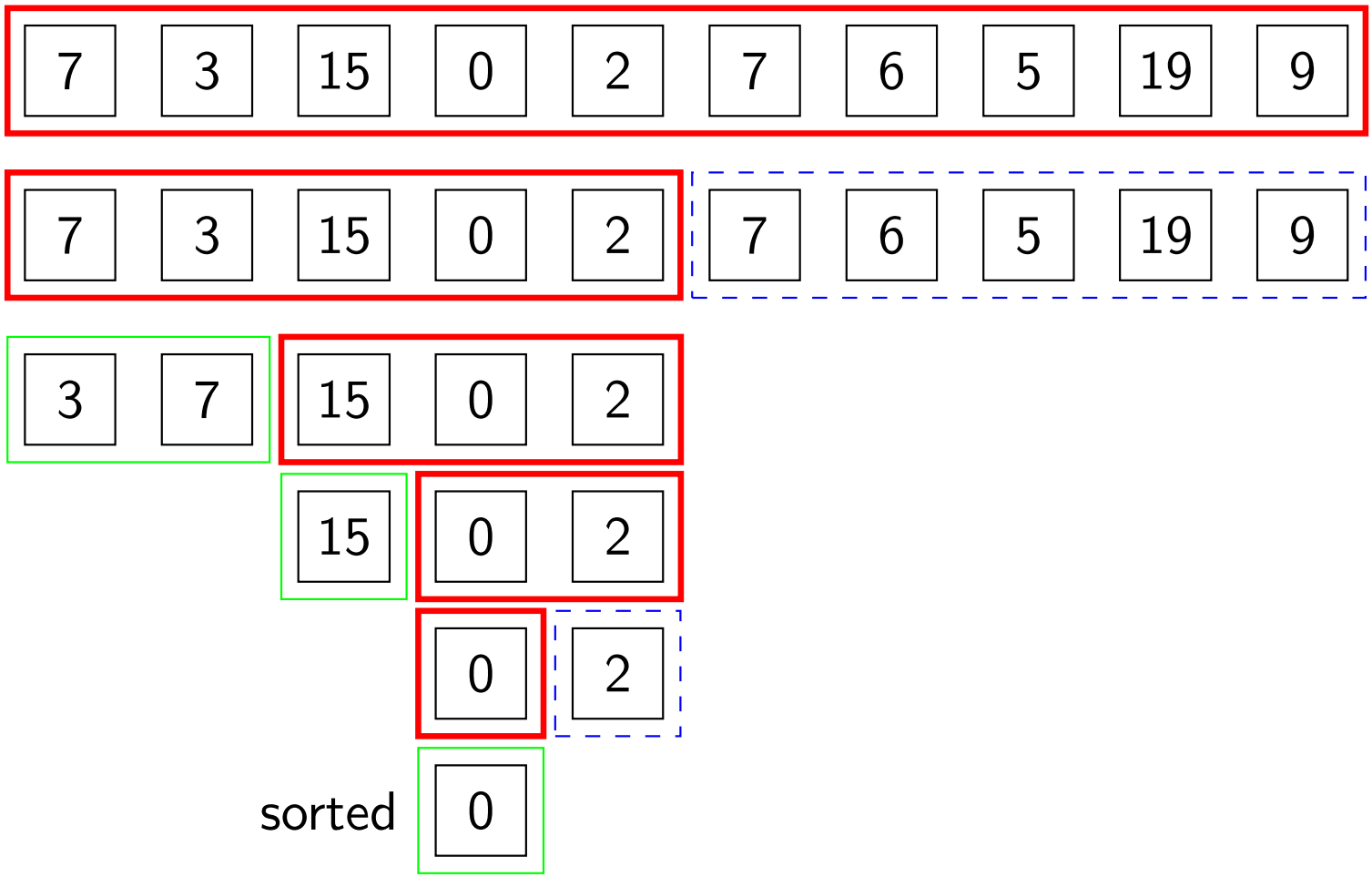
# Exempel Mergesort



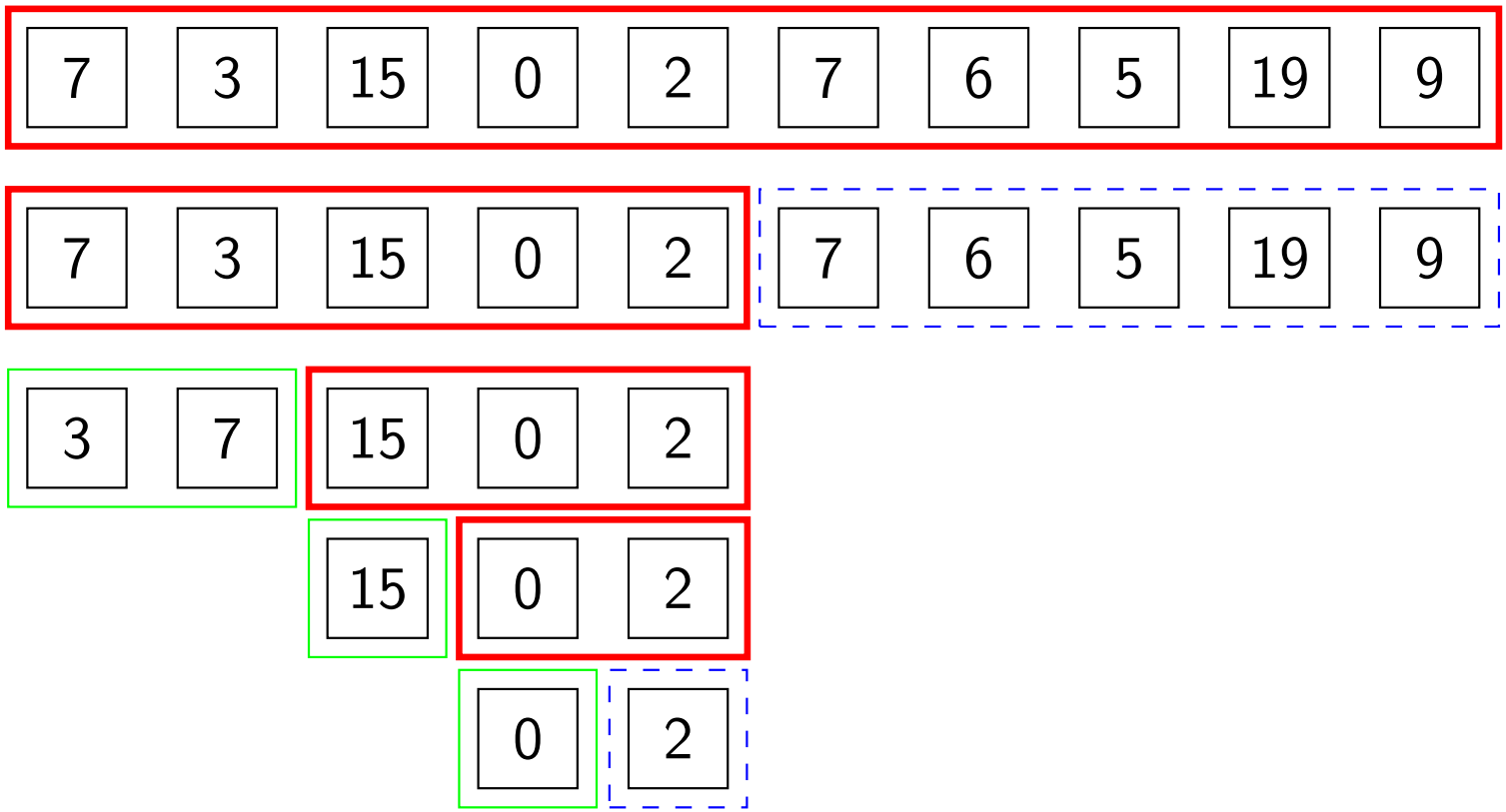
# Exempel Mergesort



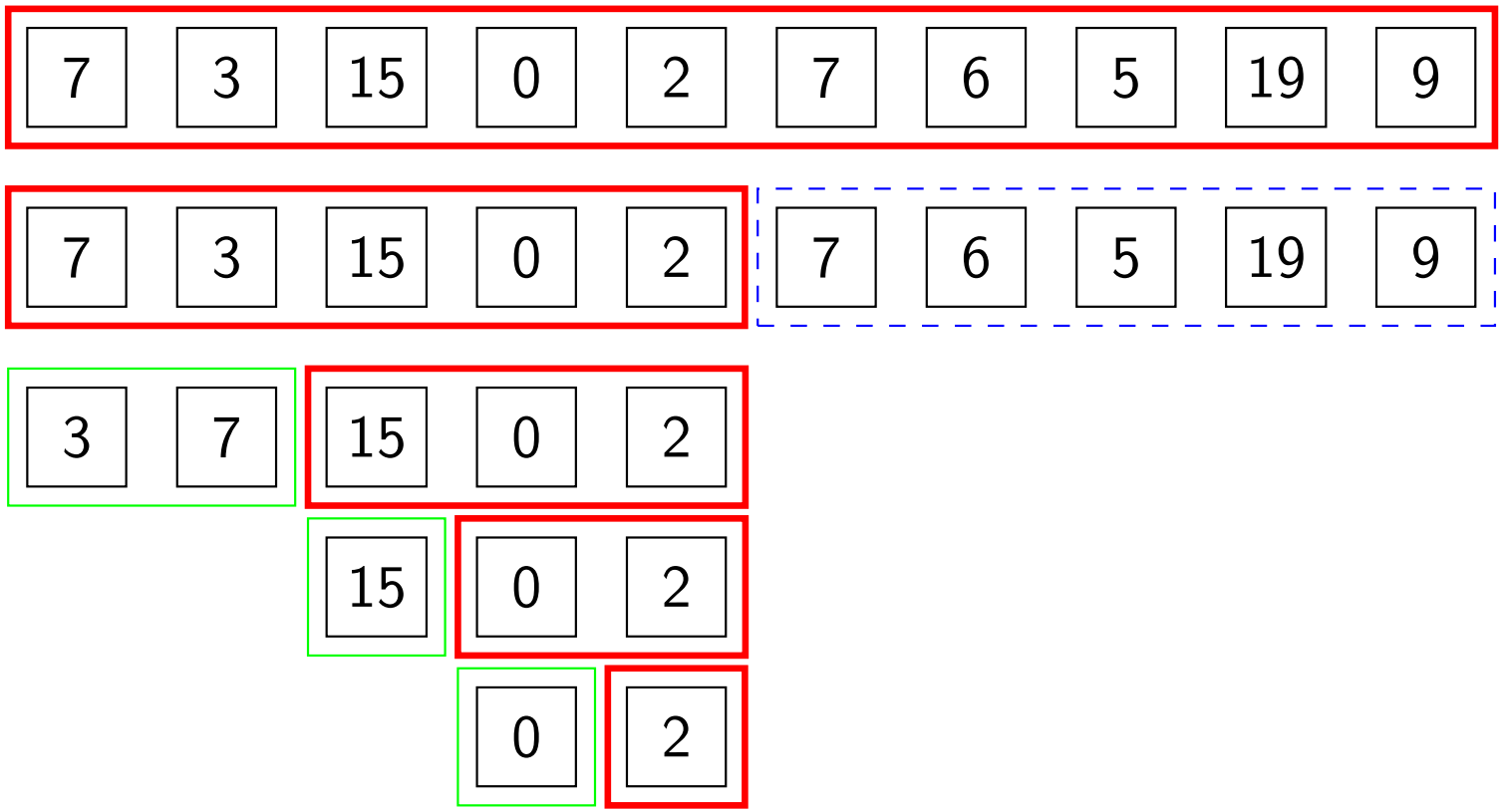
# Exempel Mergesort



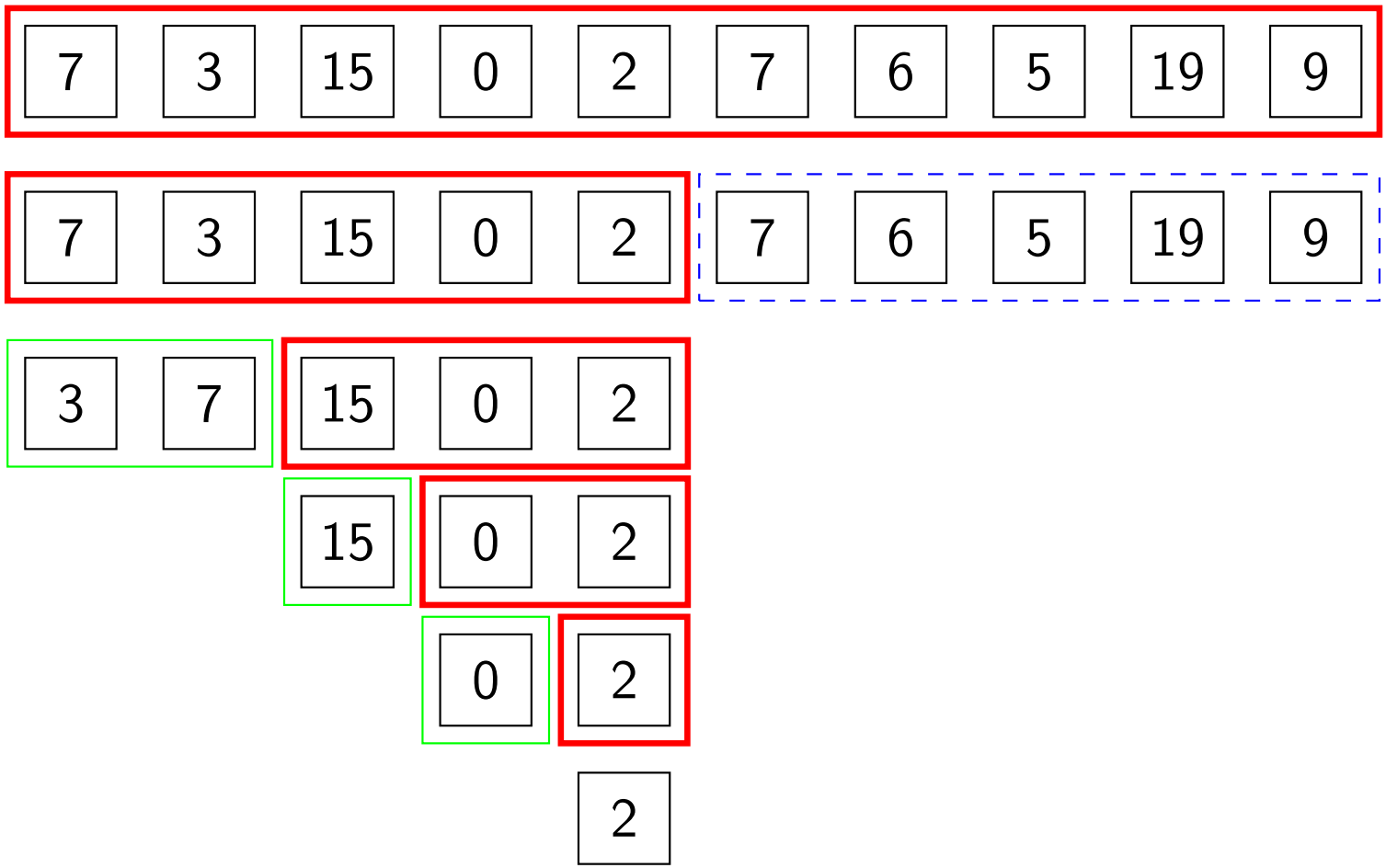
# Exempel Mergesort



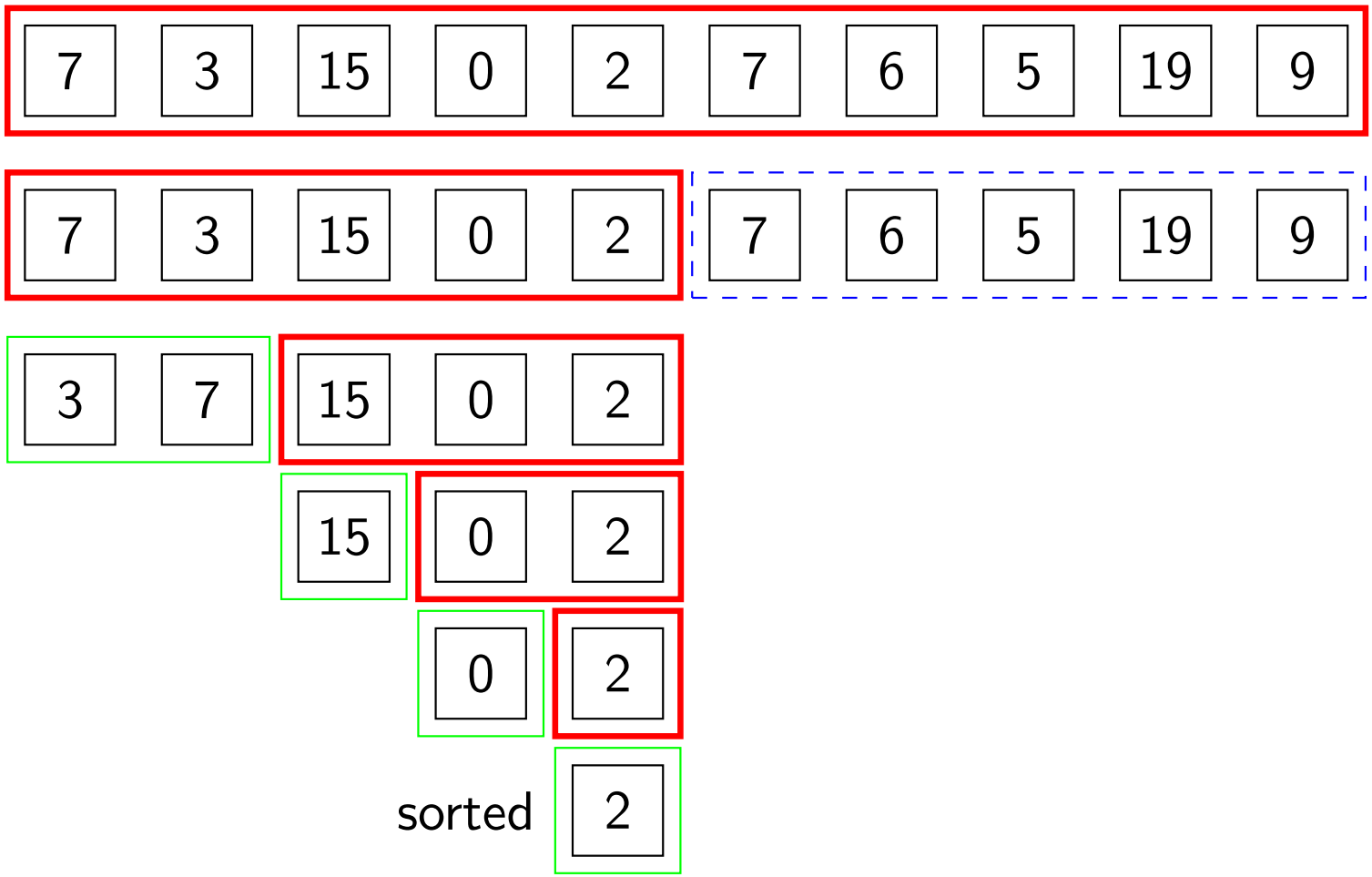
# Exempel Mergesort



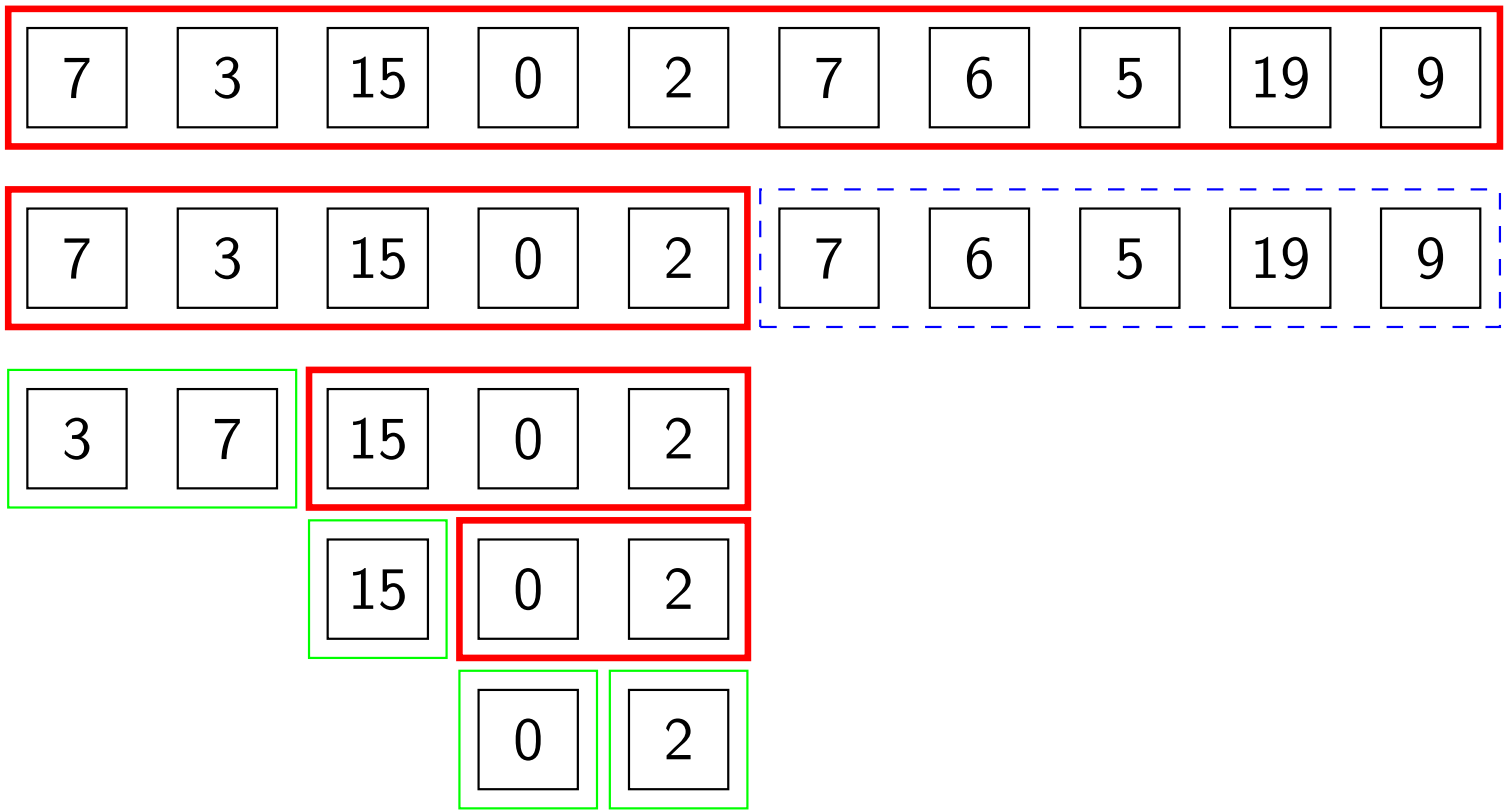
# Exempel Mergesort



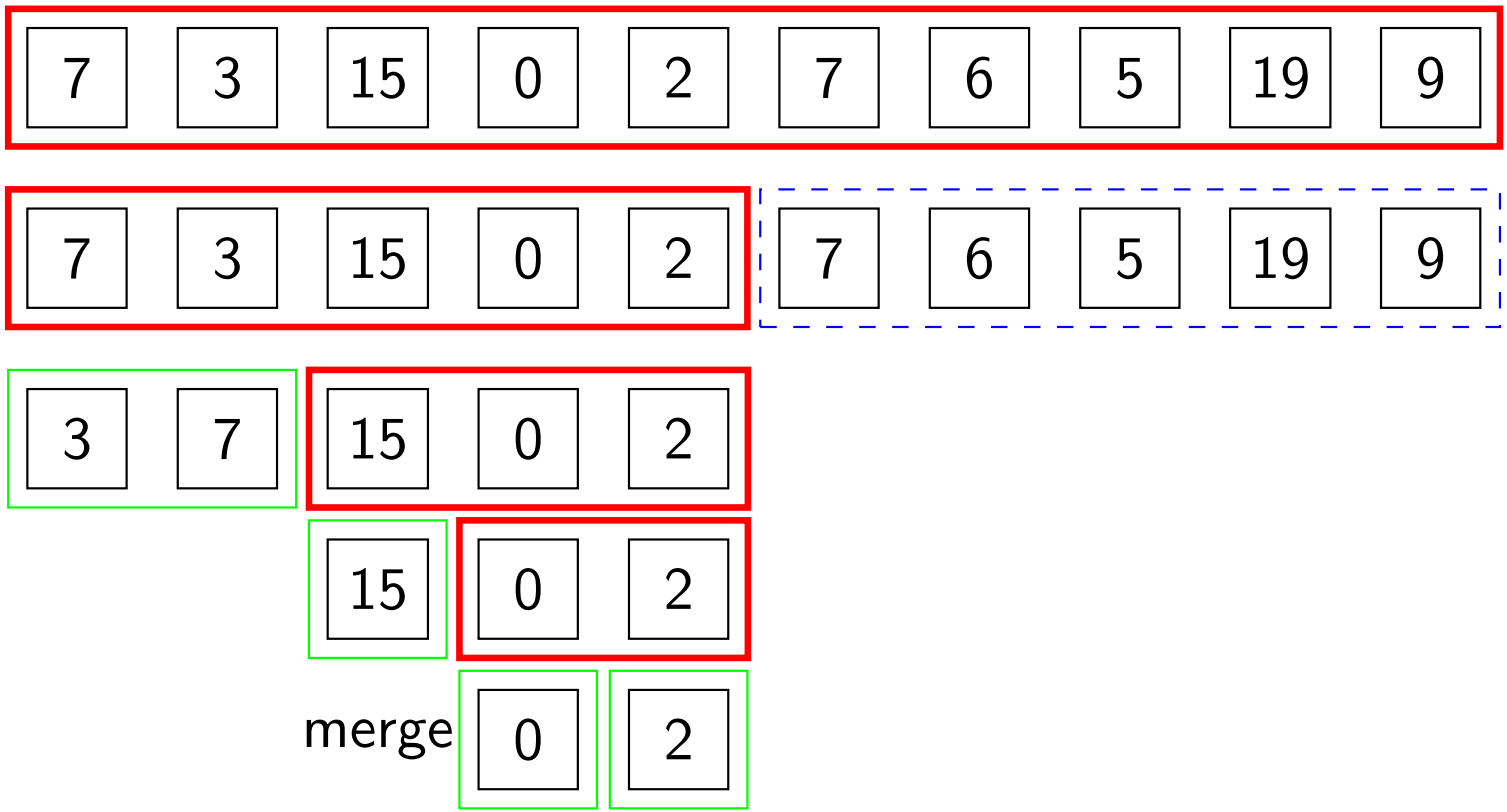
# Exempel Mergesort



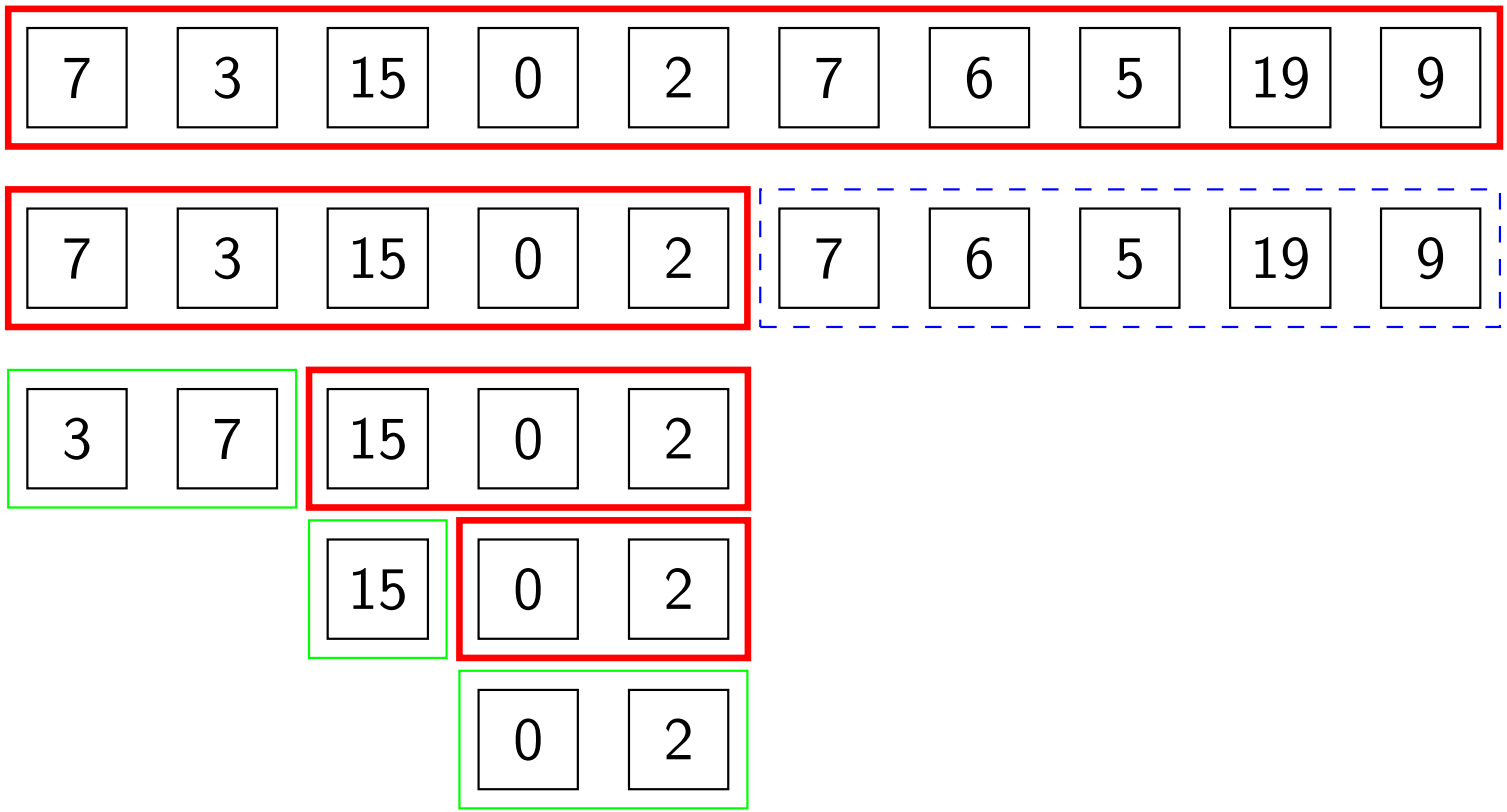
# Exempel Mergesort



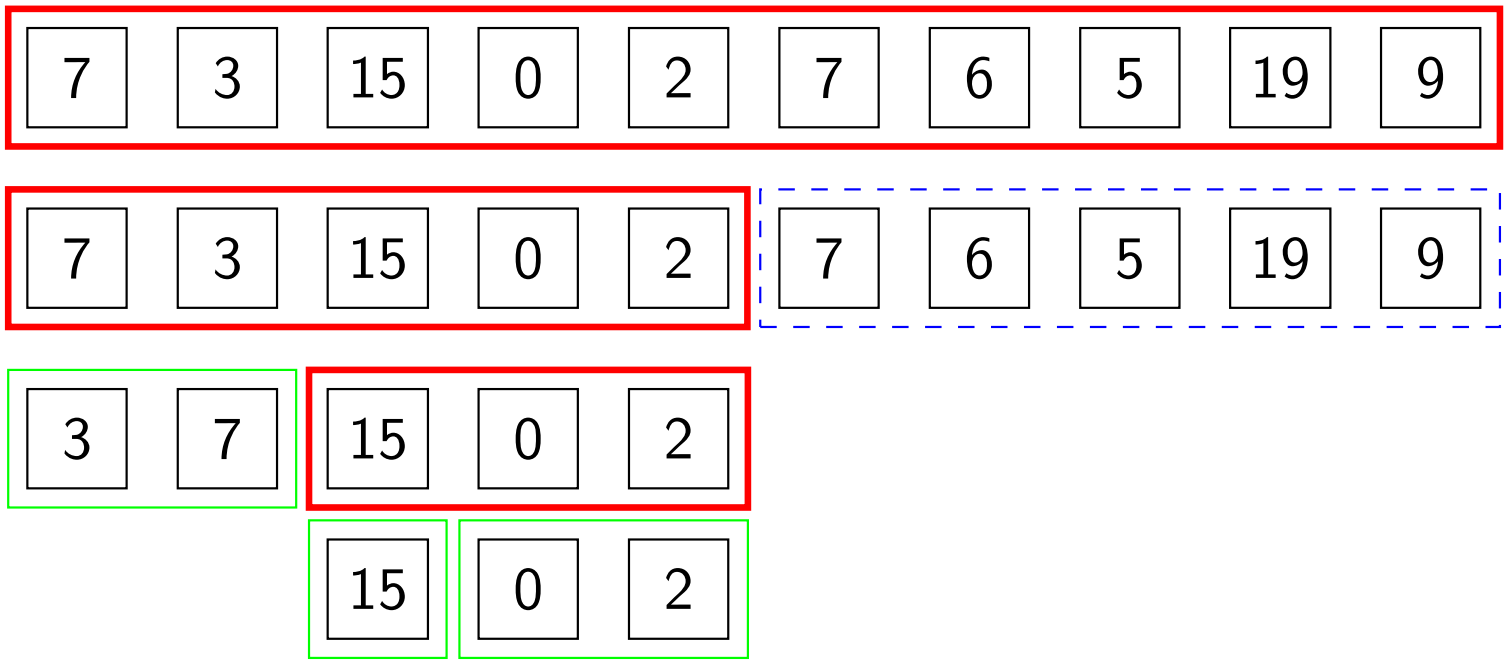
# Exempel Mergesort



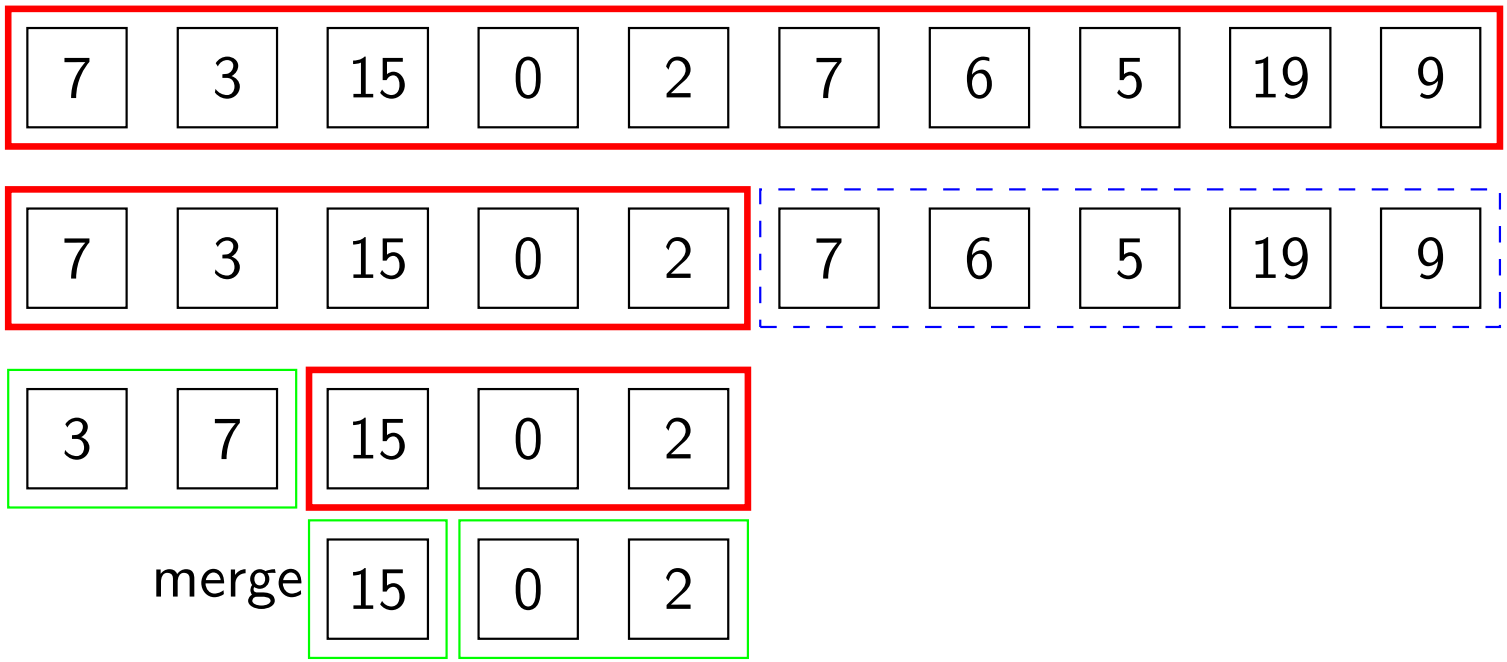
# Exempel Mergesort



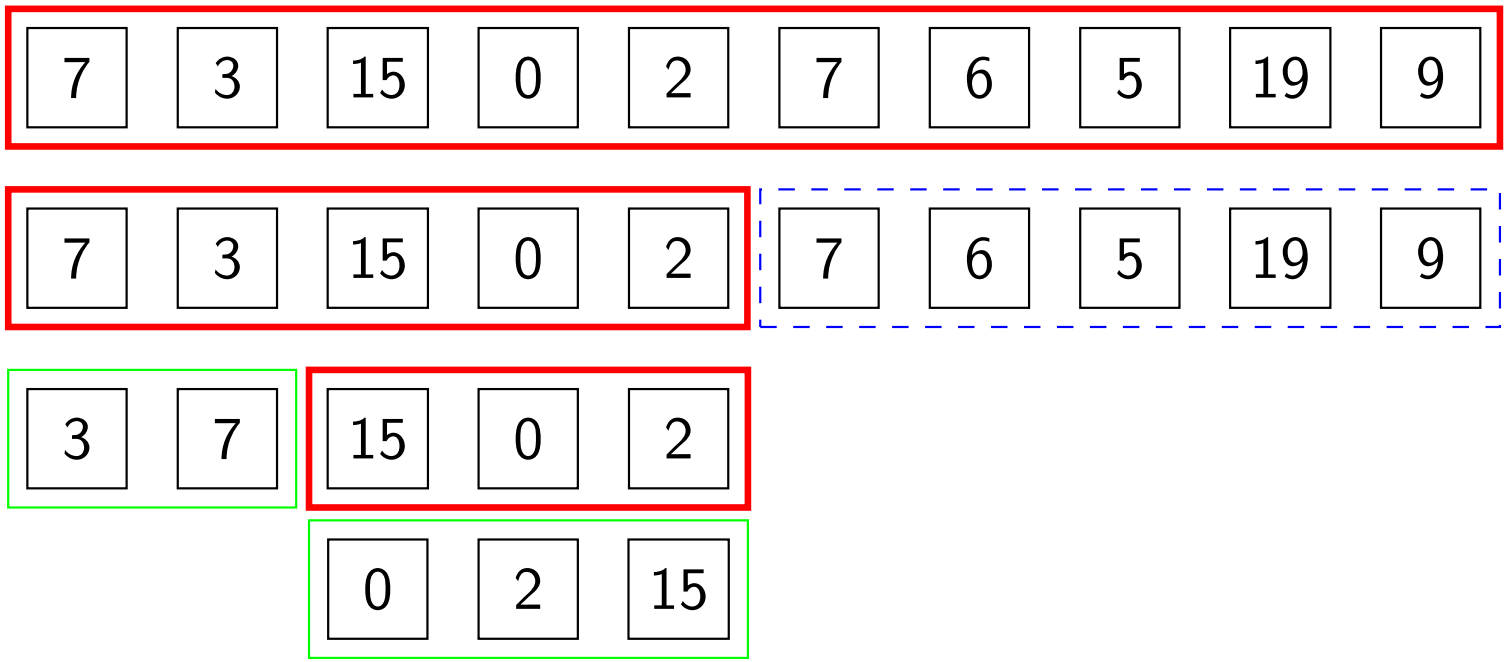
# Exempel Mergesort



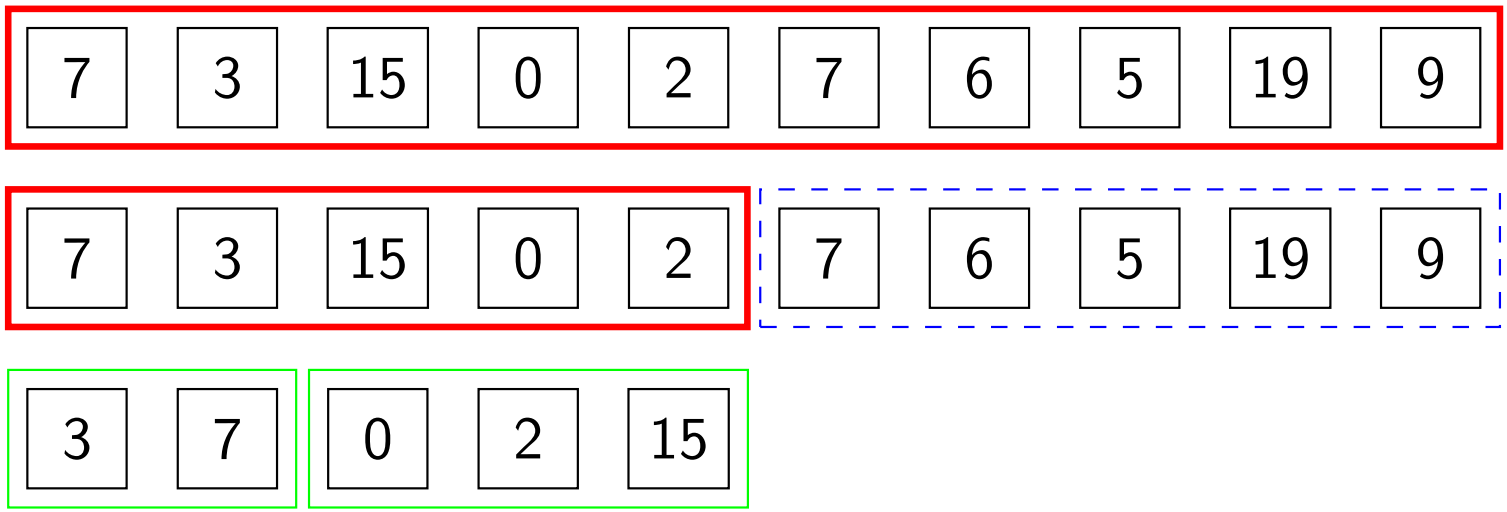
# Exempel Mergesort



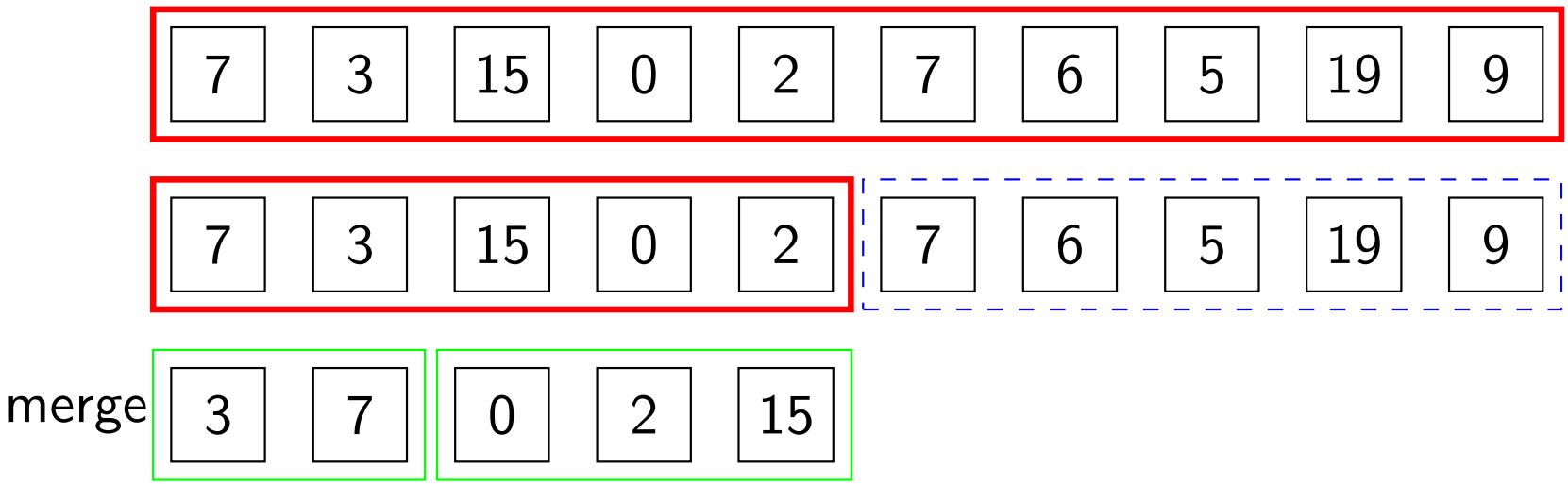
# Exempel Mergesort



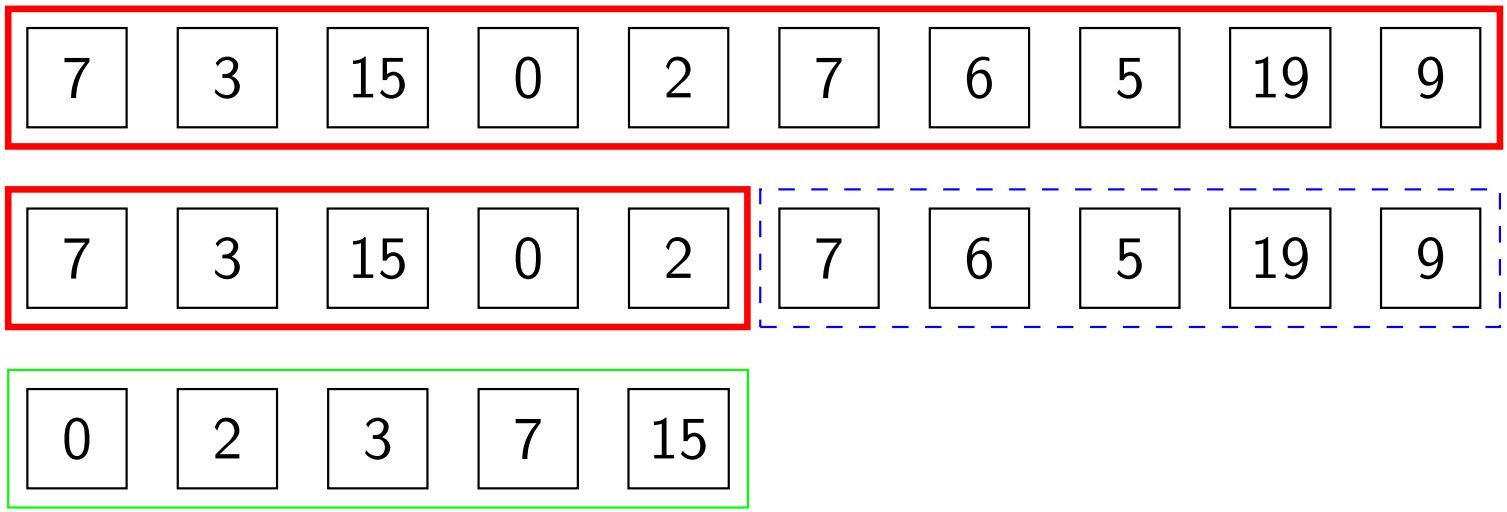
# Exempel Mergesort



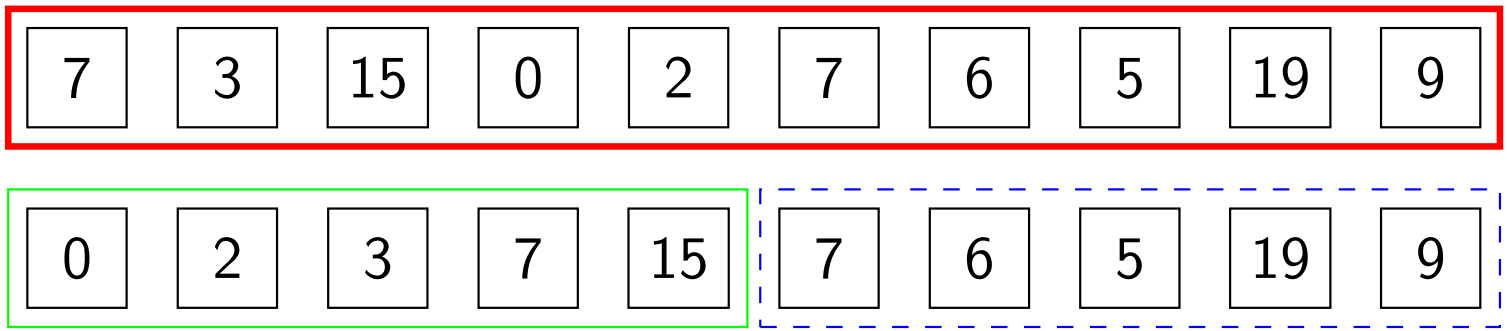
# Exempel Mergesort



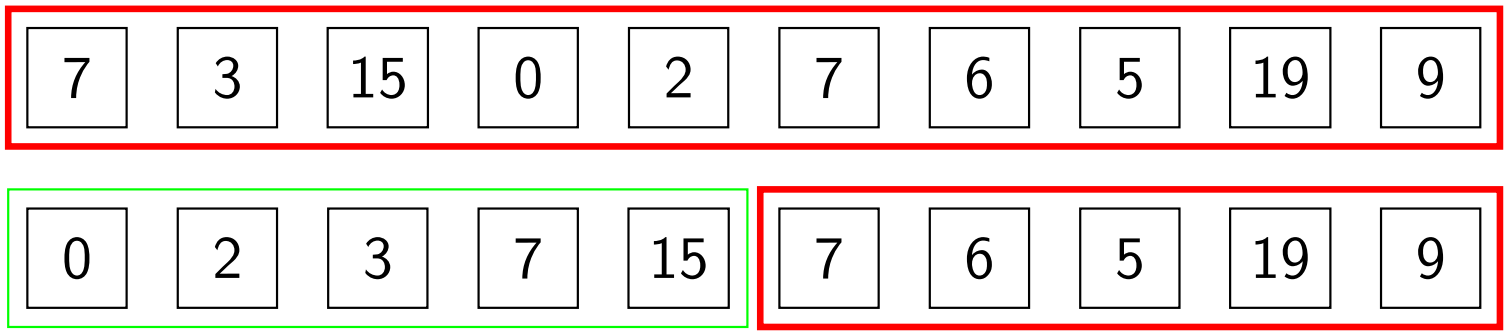
# Exempel Mergesort



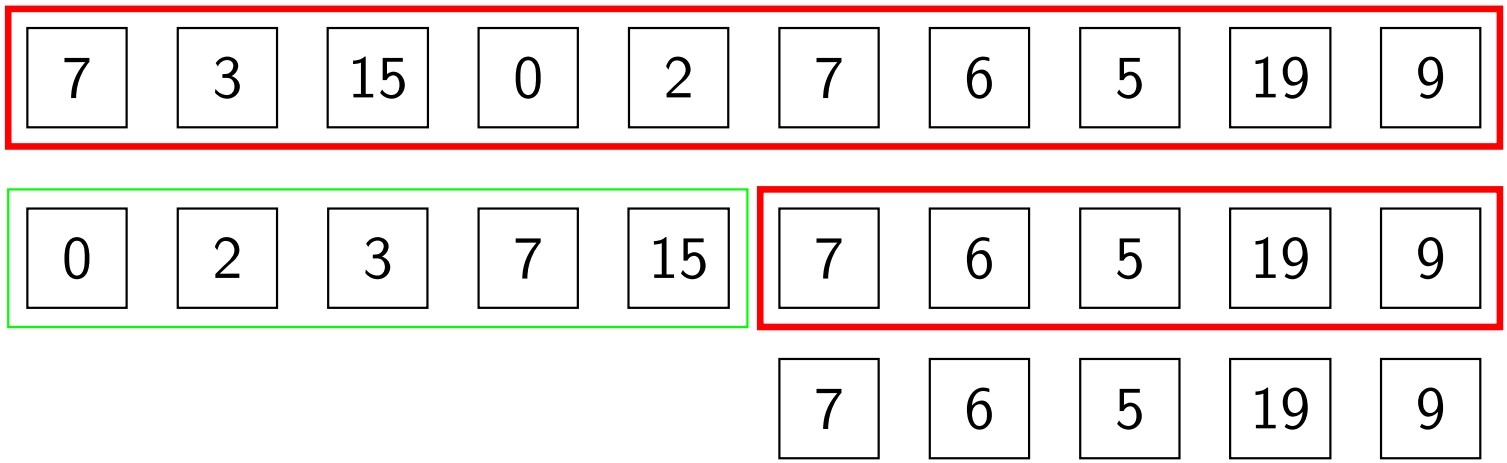
# Exempel Mergesort



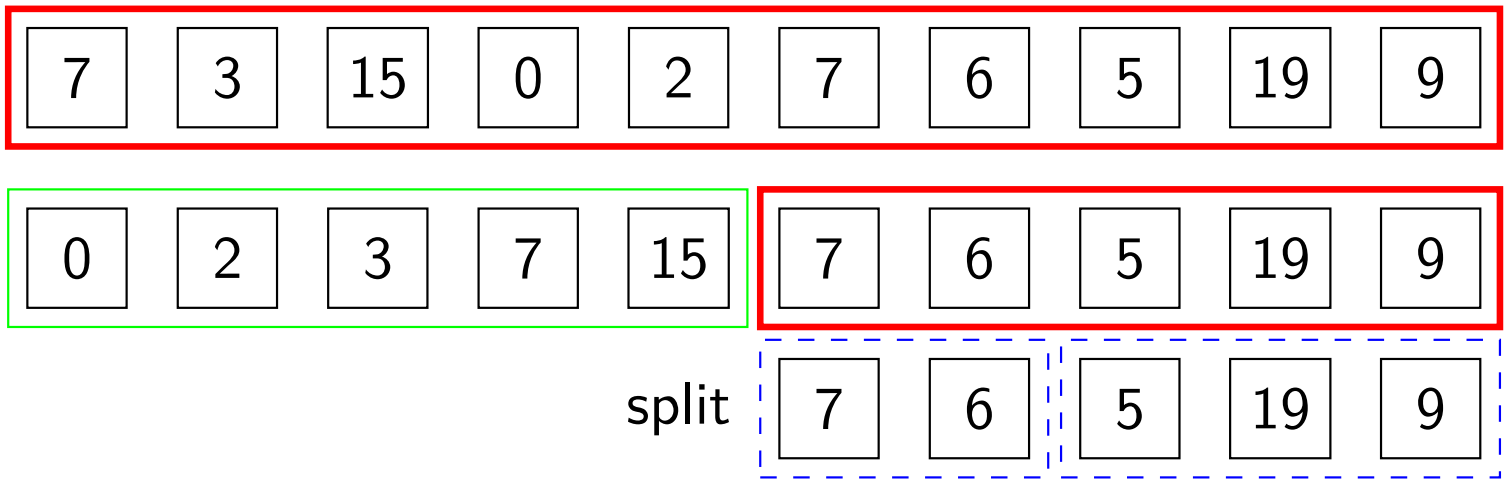
# Exempel Mergesort



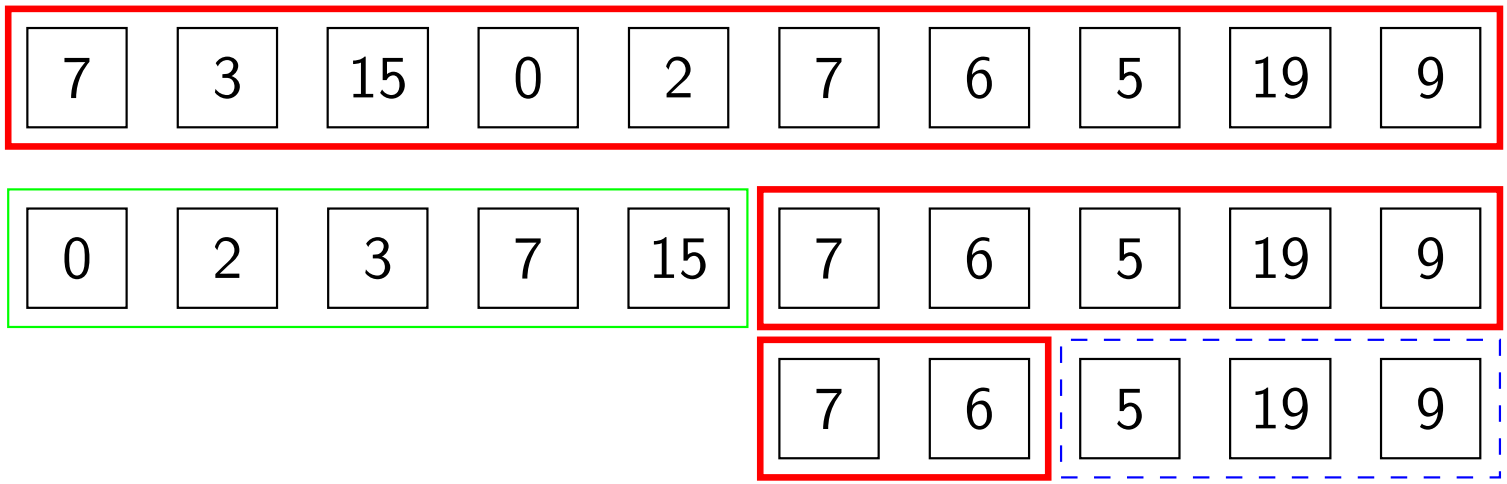
# Exempel Mergesort



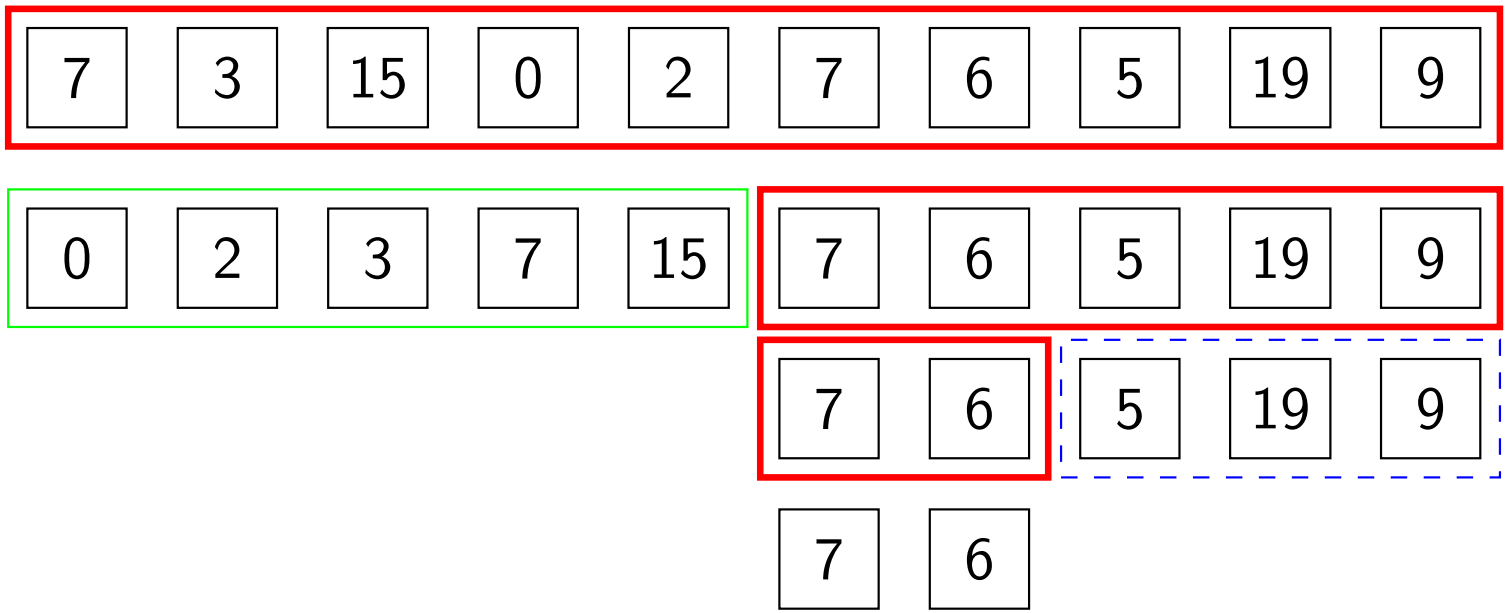
# Exempel Mergesort



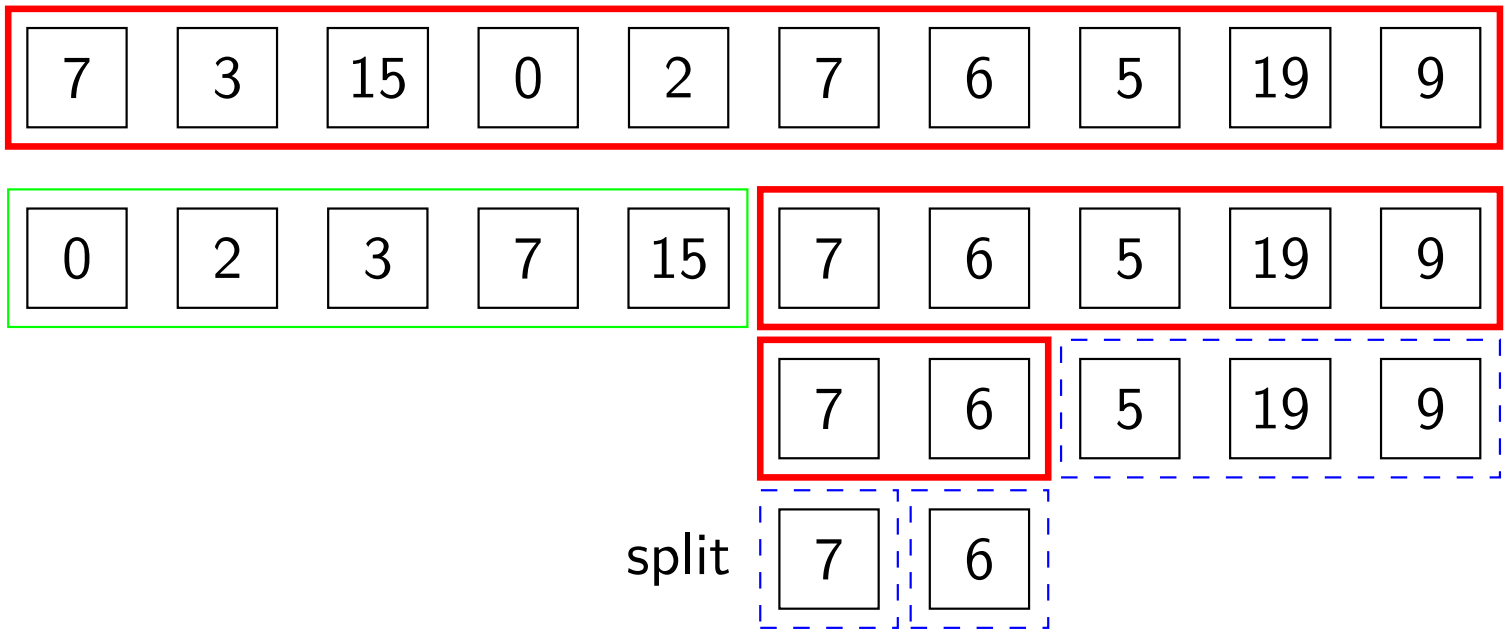
# Exempel Mergesort



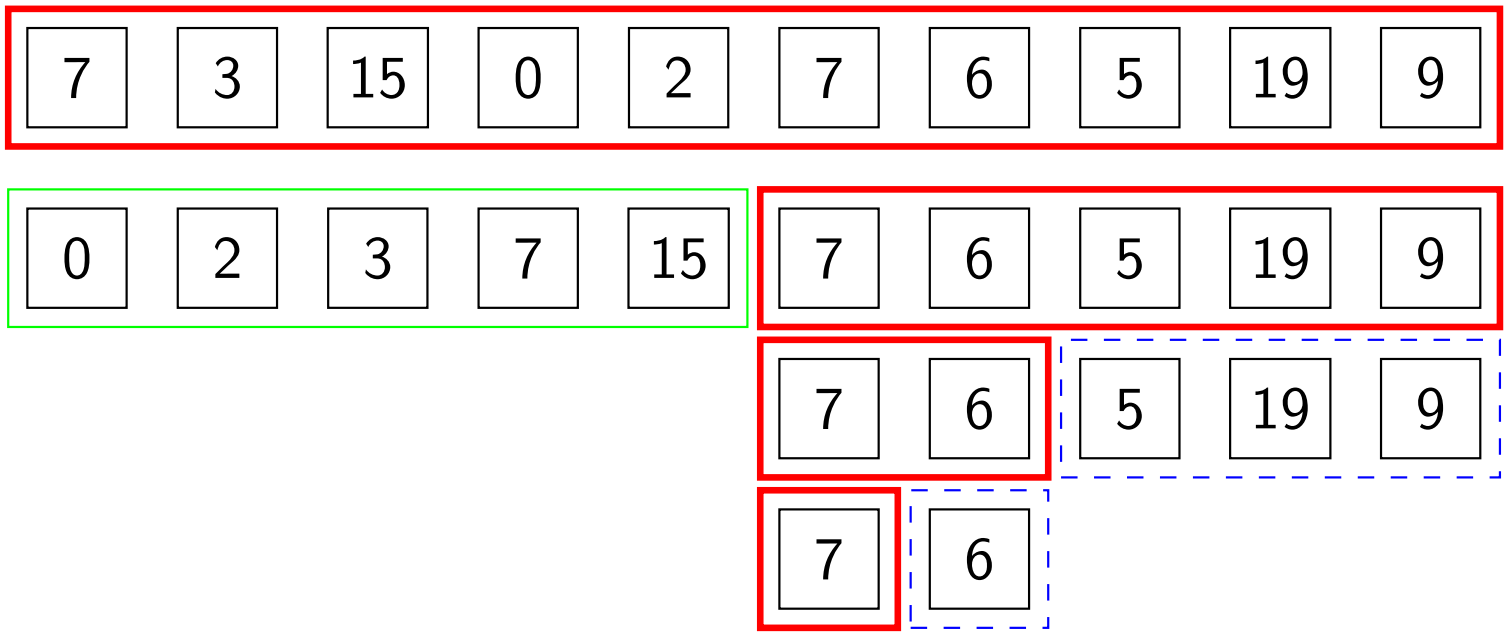
# Exempel Mergesort



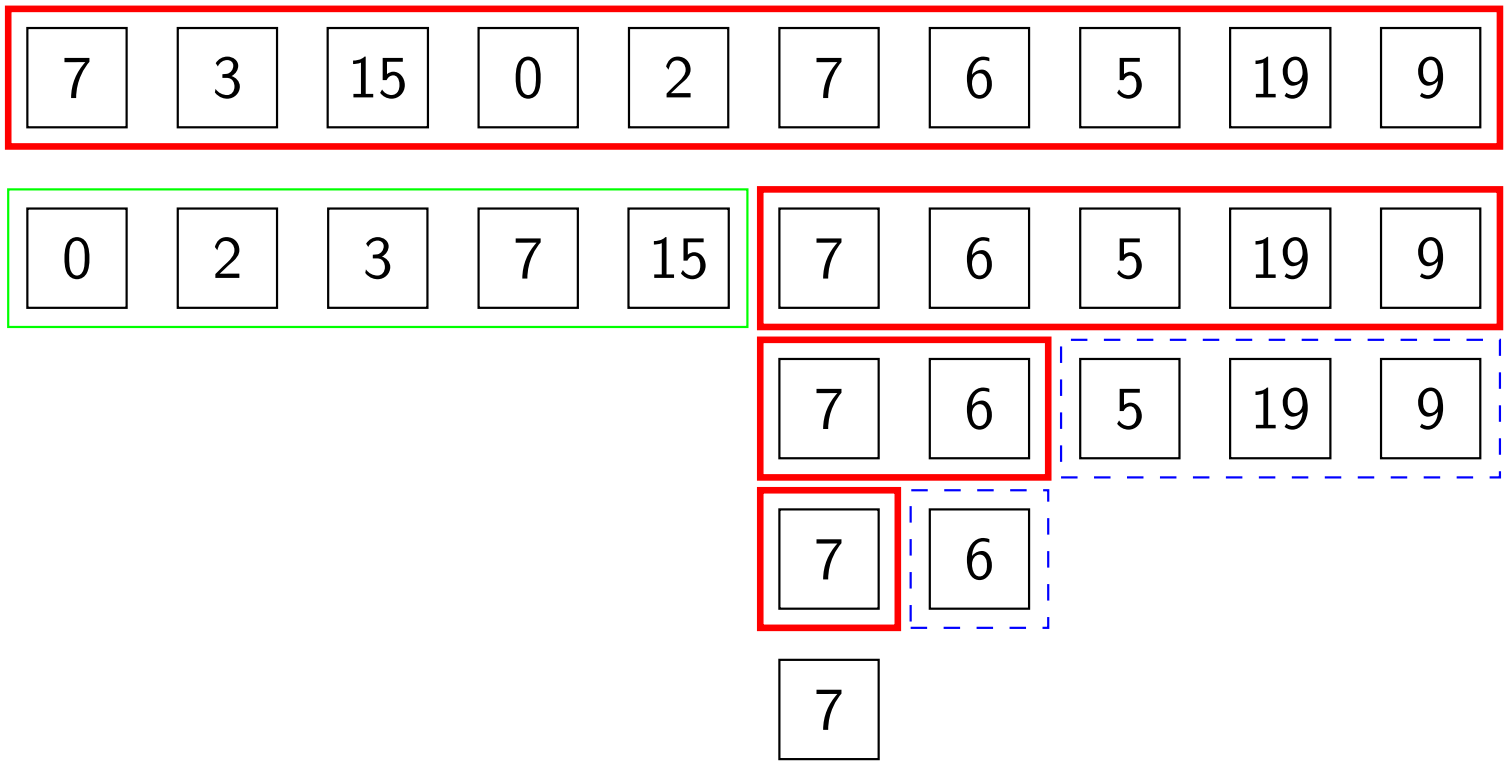
# Exempel Mergesort



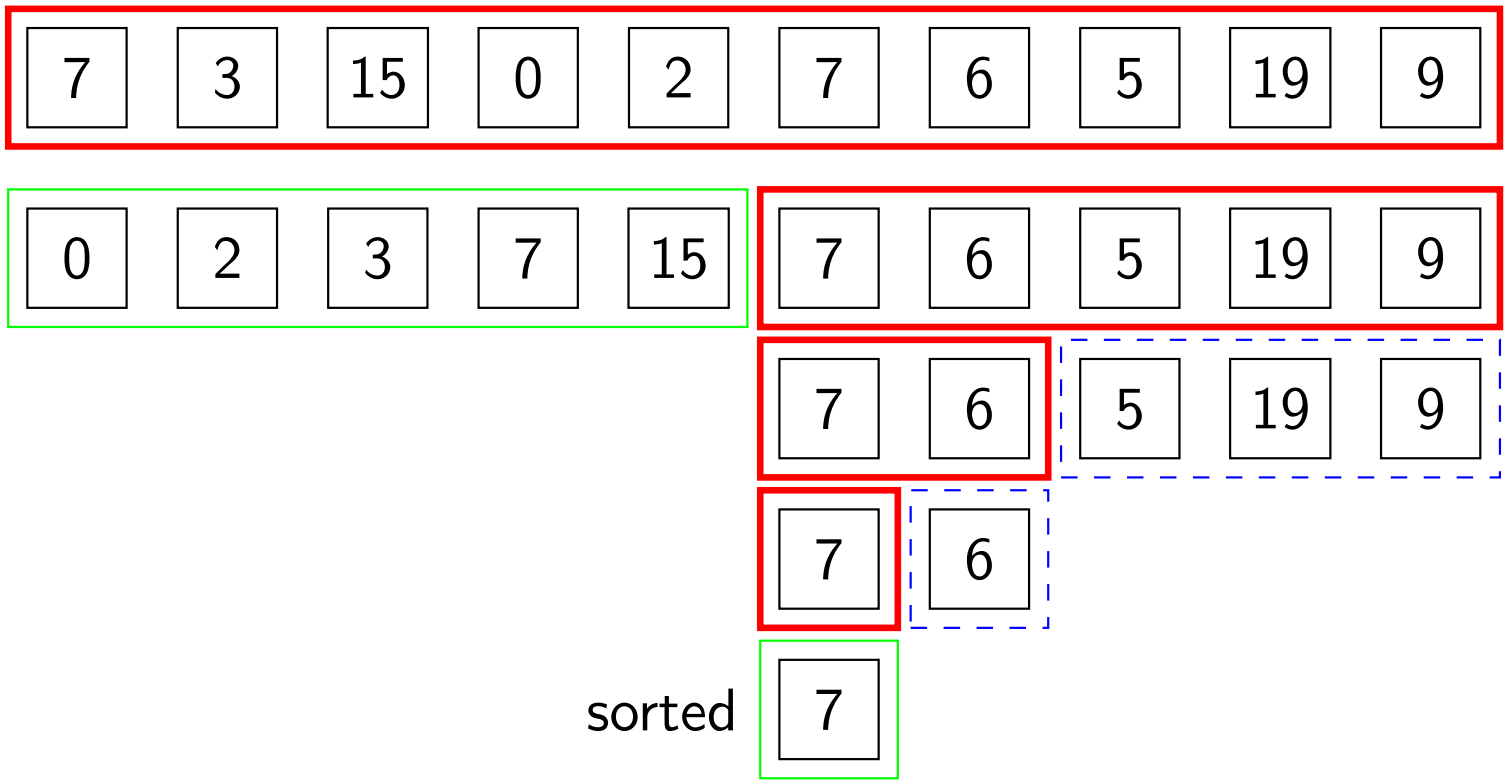
# Exempel Mergesort



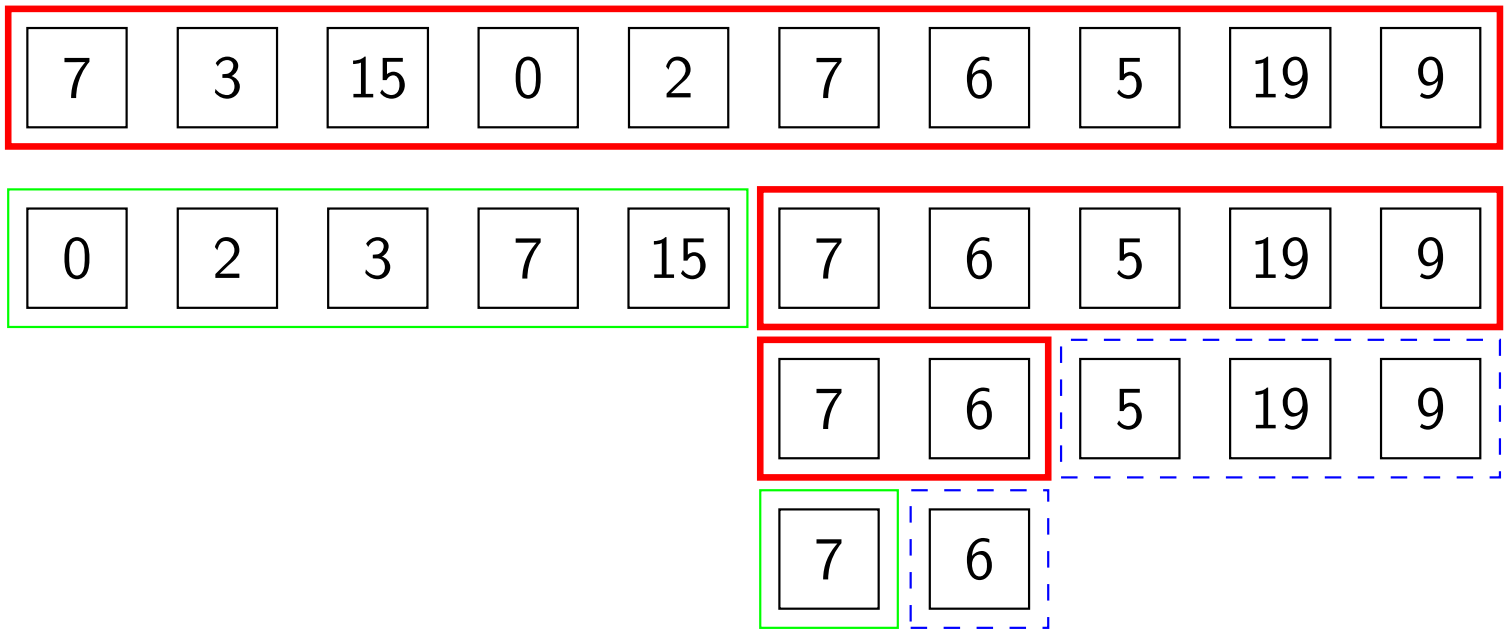
# Exempel Mergesort



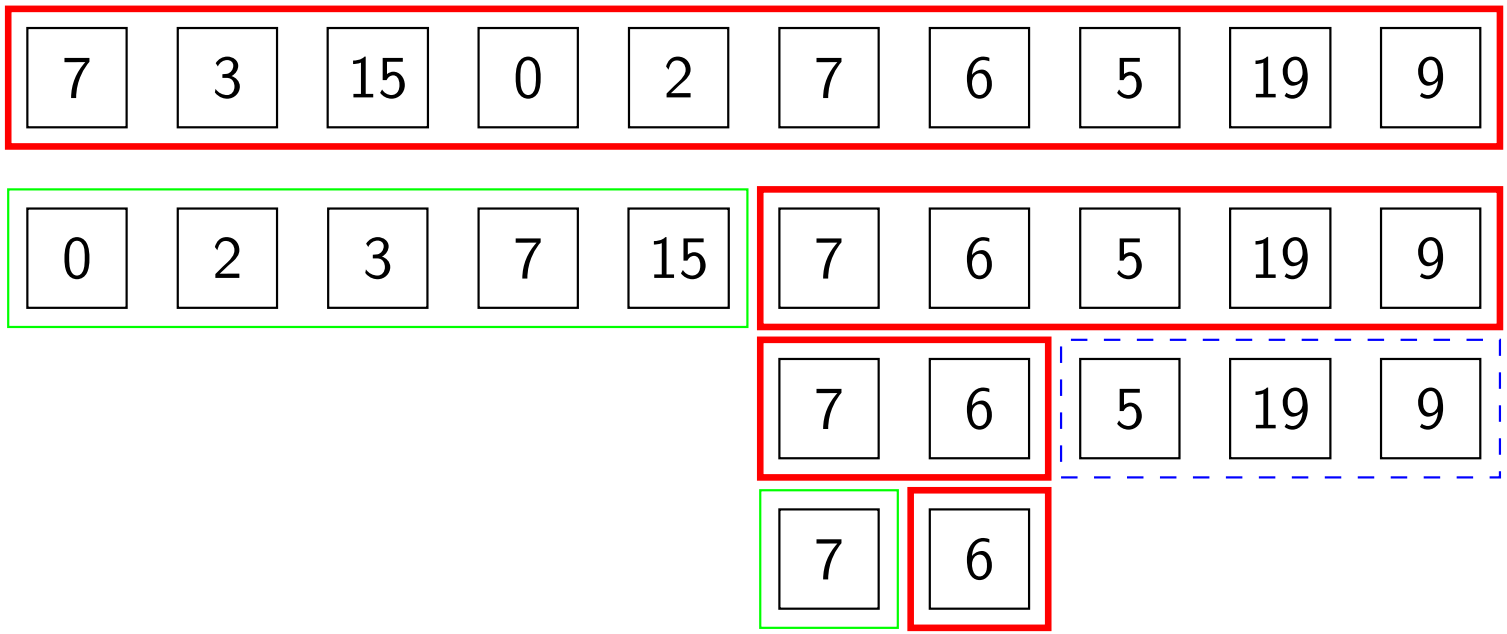
# Exempel Mergesort



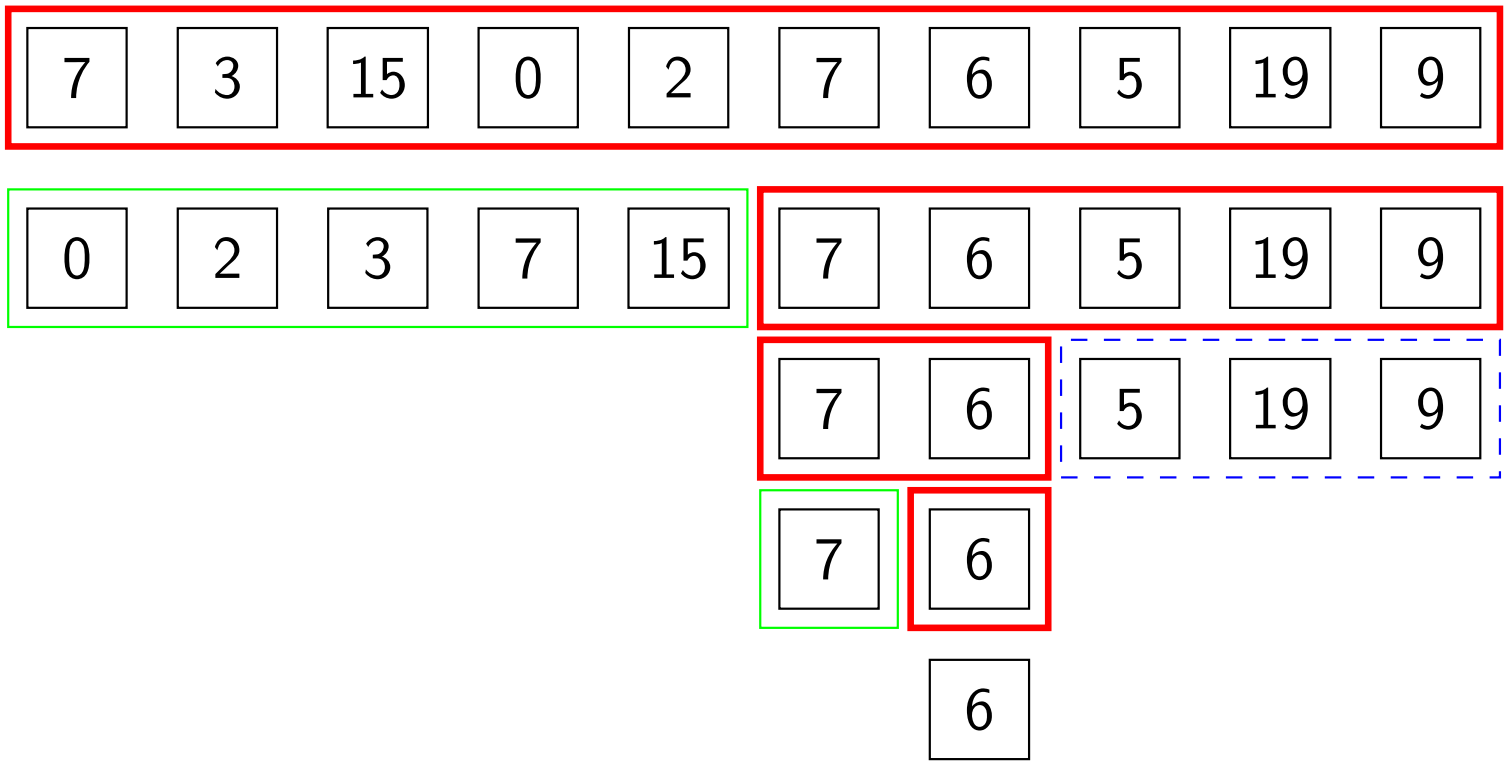
# Exempel Mergesort



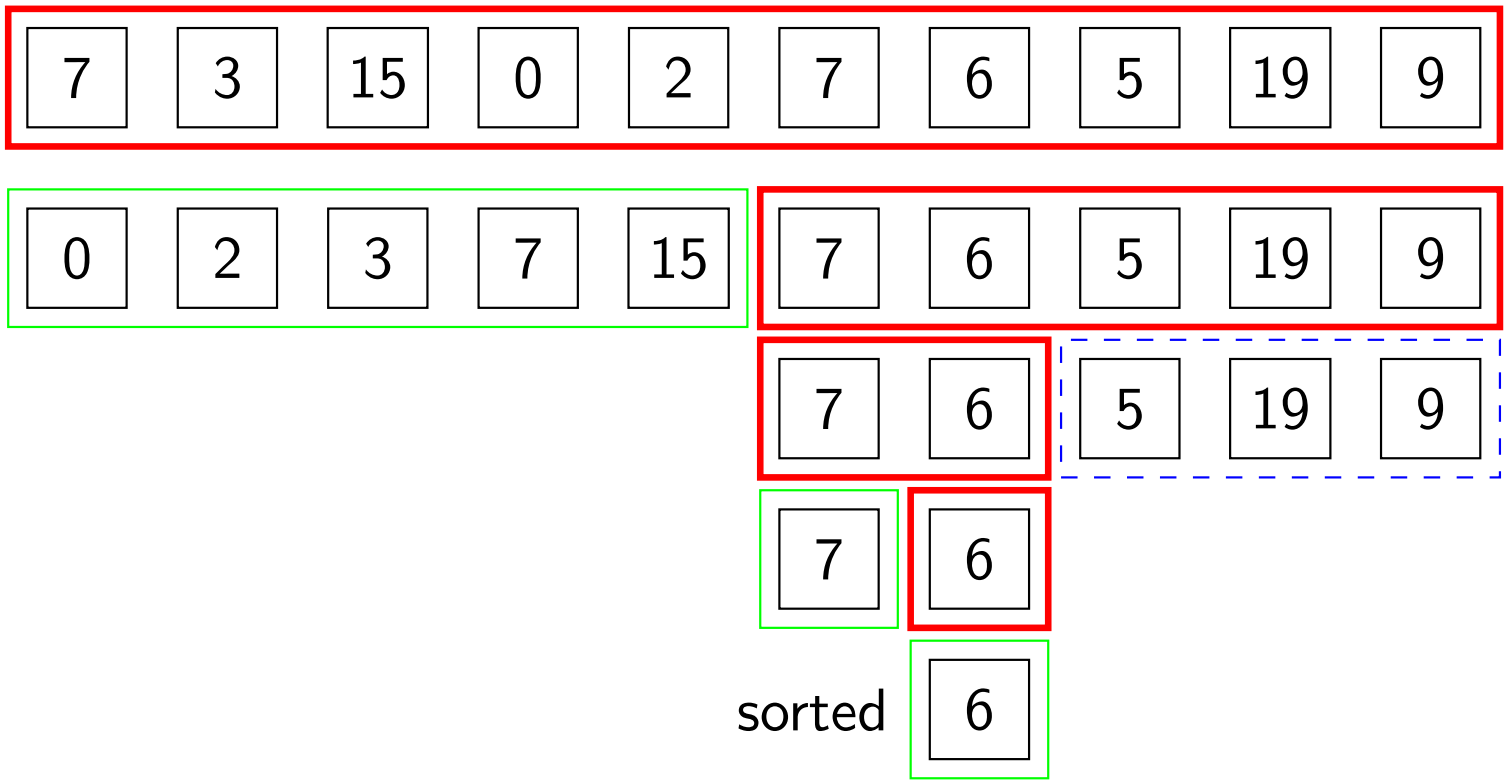
# Exempel Mergesort



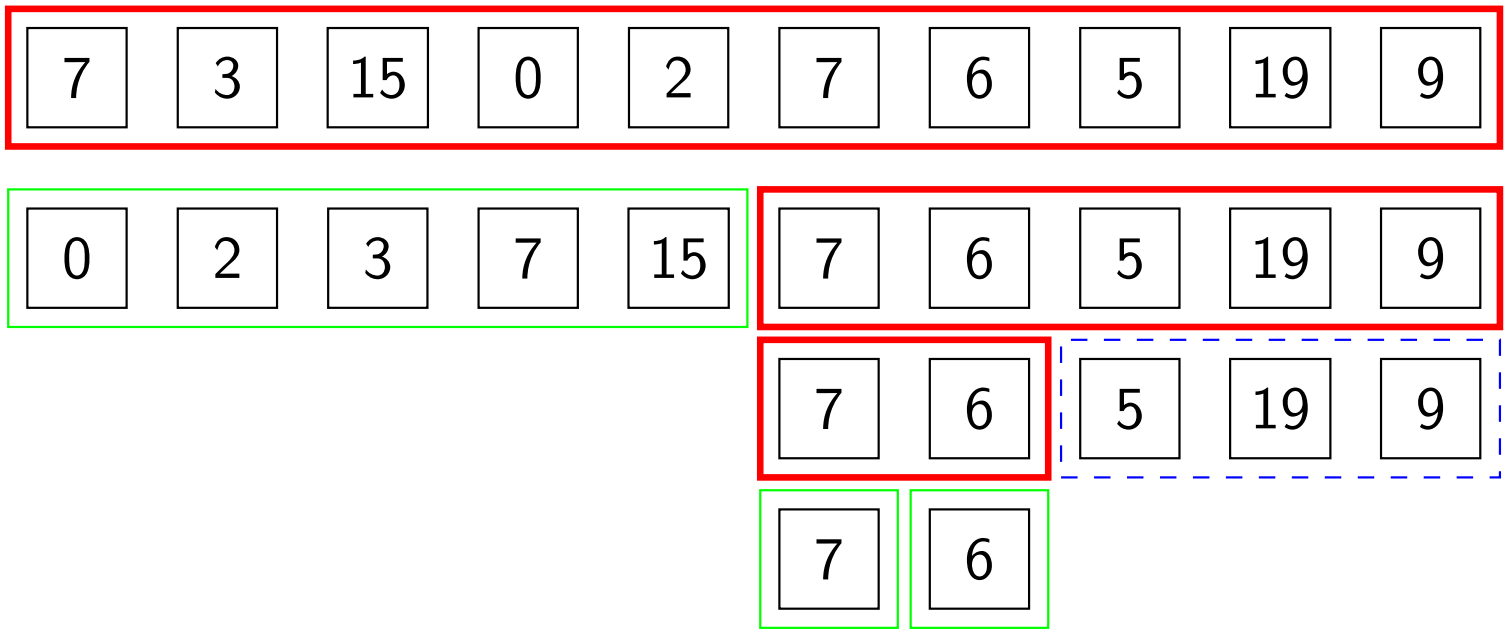
# Exempel Mergesort



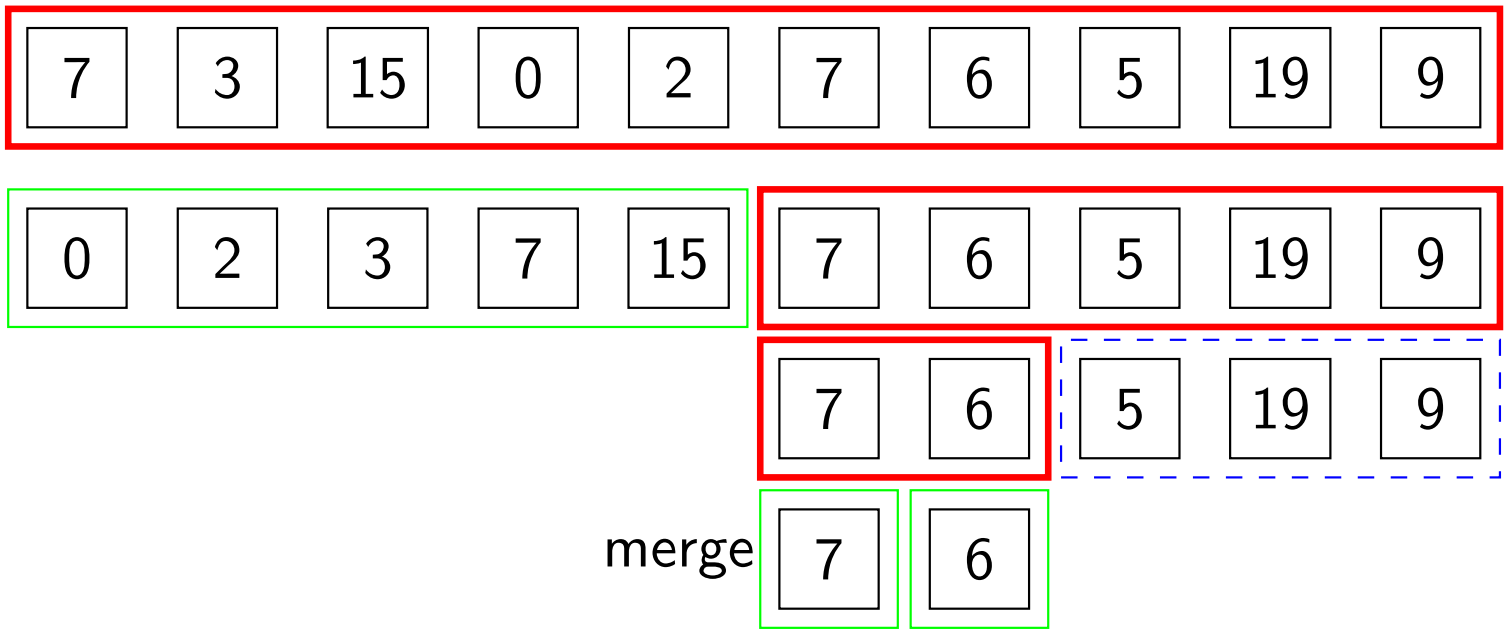
# Exempel Mergesort



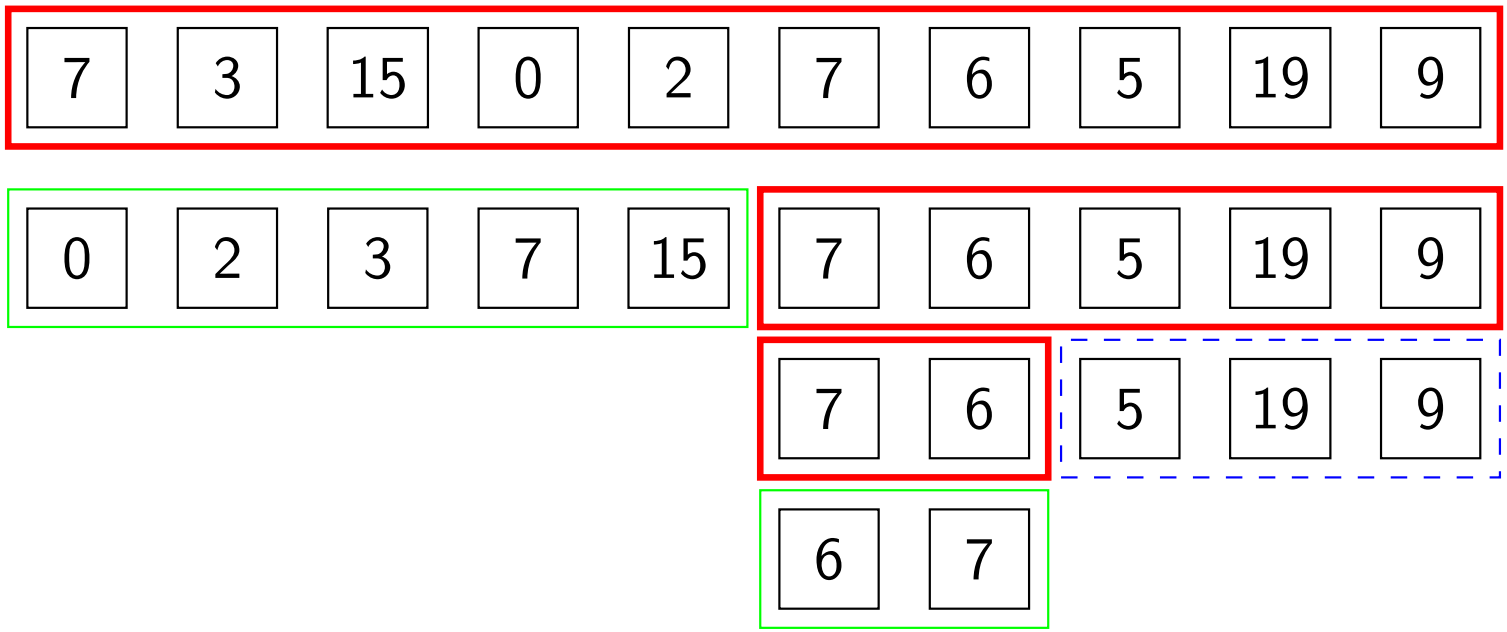
# Exempel Mergesort



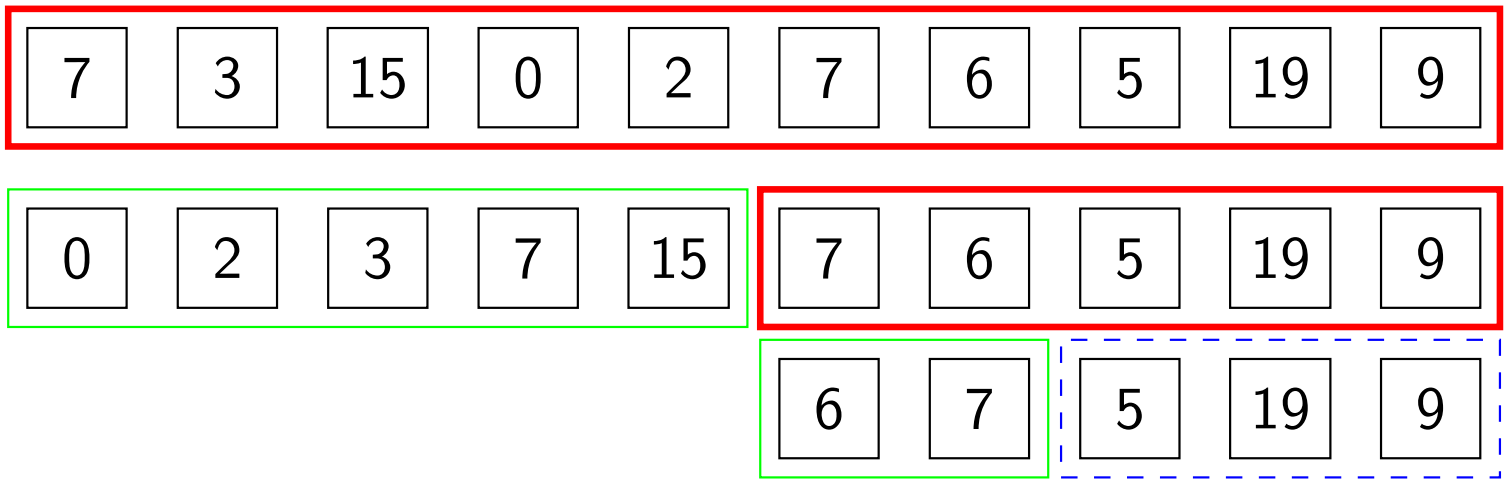
# Exempel Mergesort



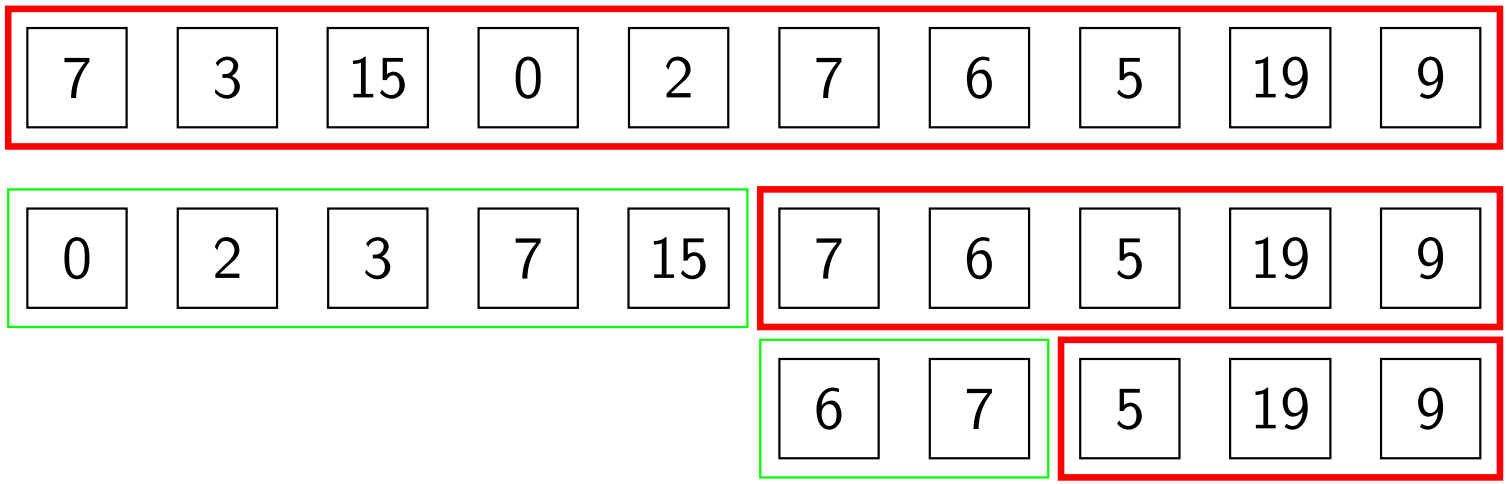
# Exempel Mergesort



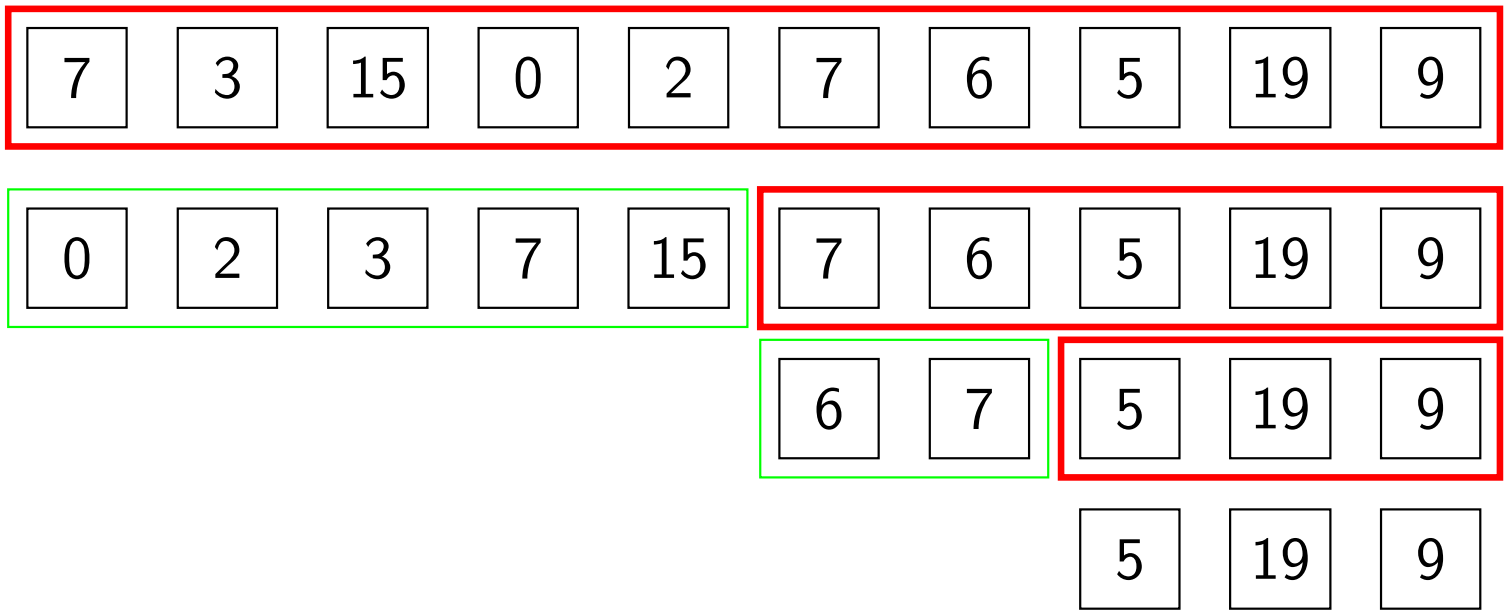
# Exempel Mergesort



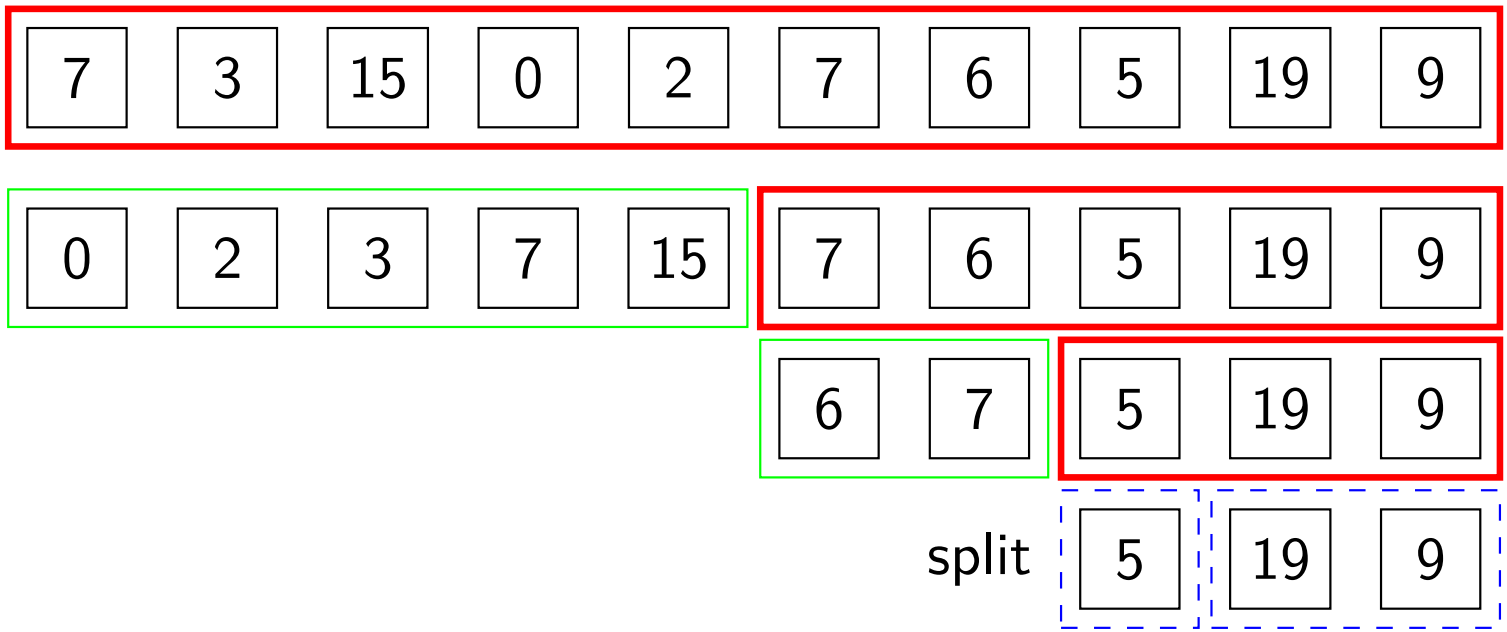
# Exempel Mergesort



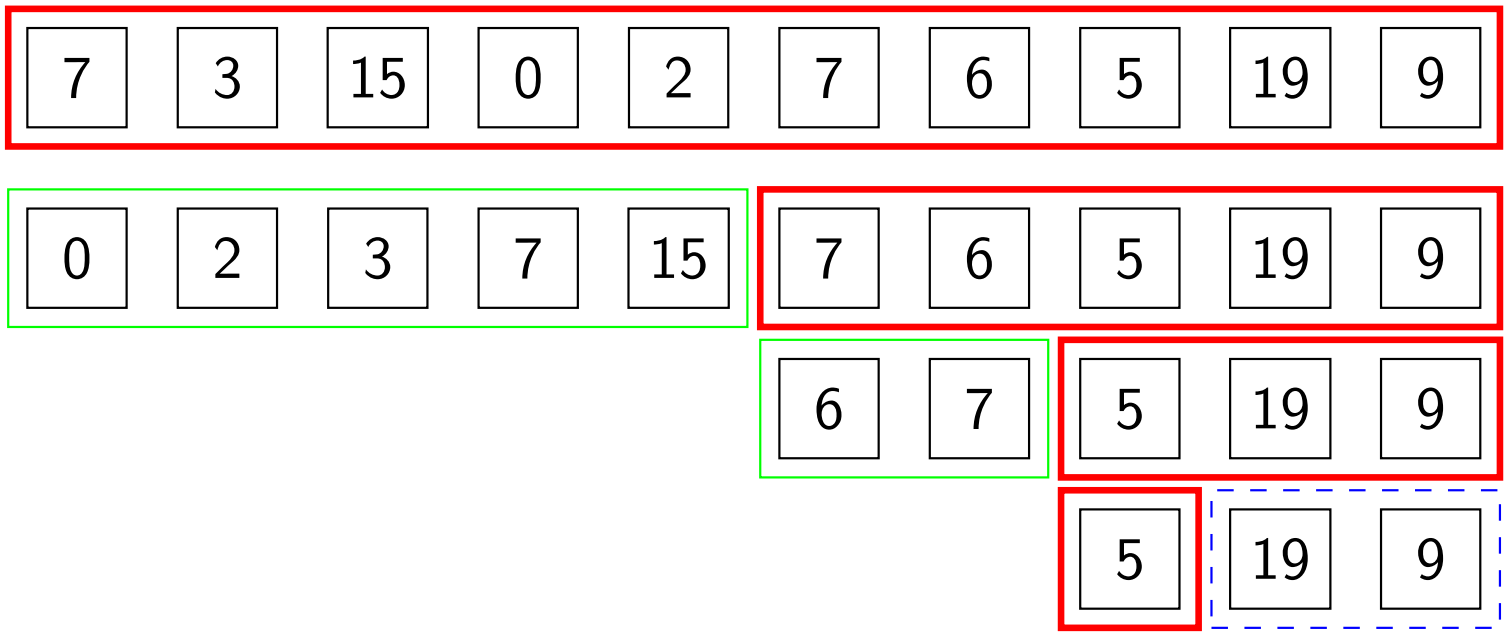
# Exempel Mergesort



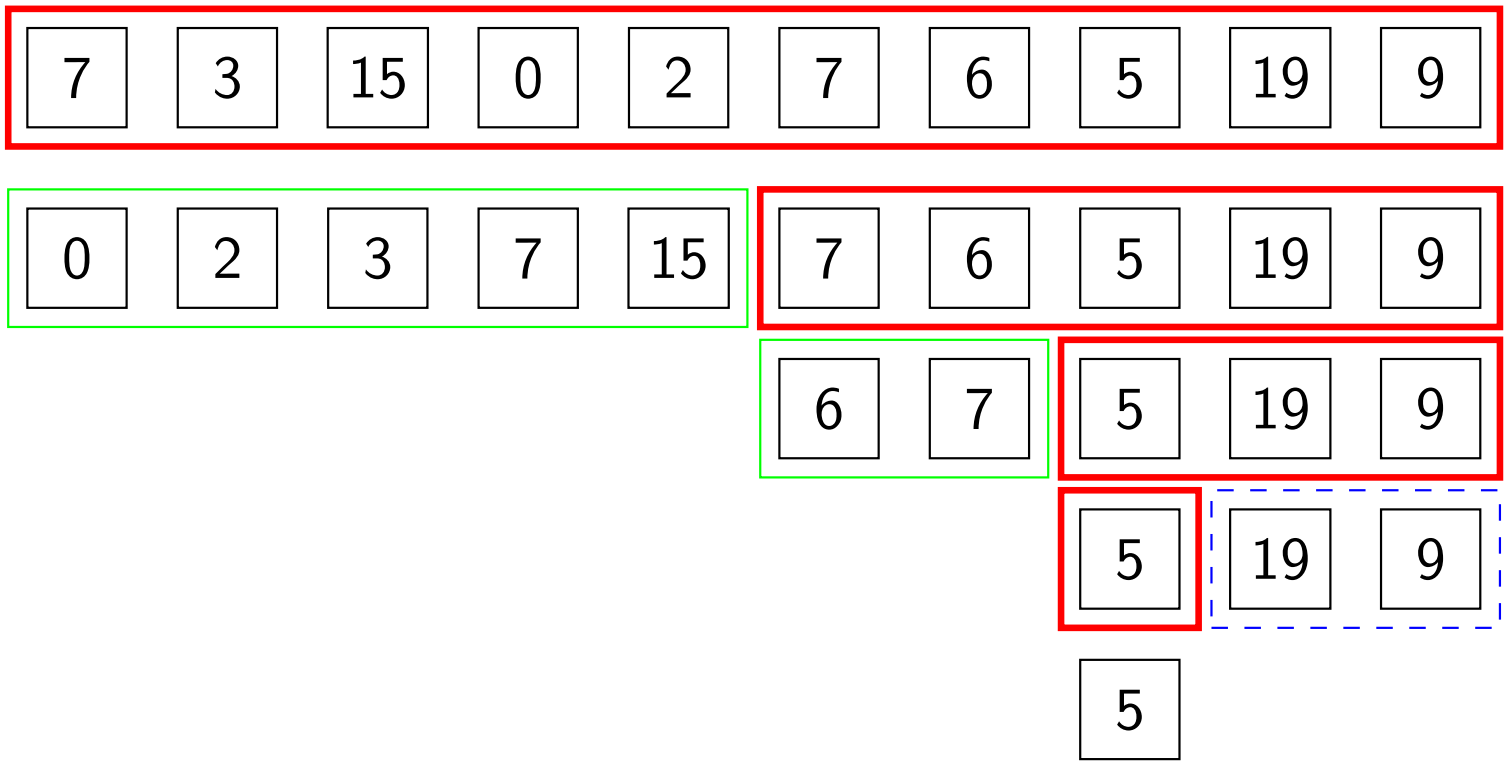
# Exempel Mergesort



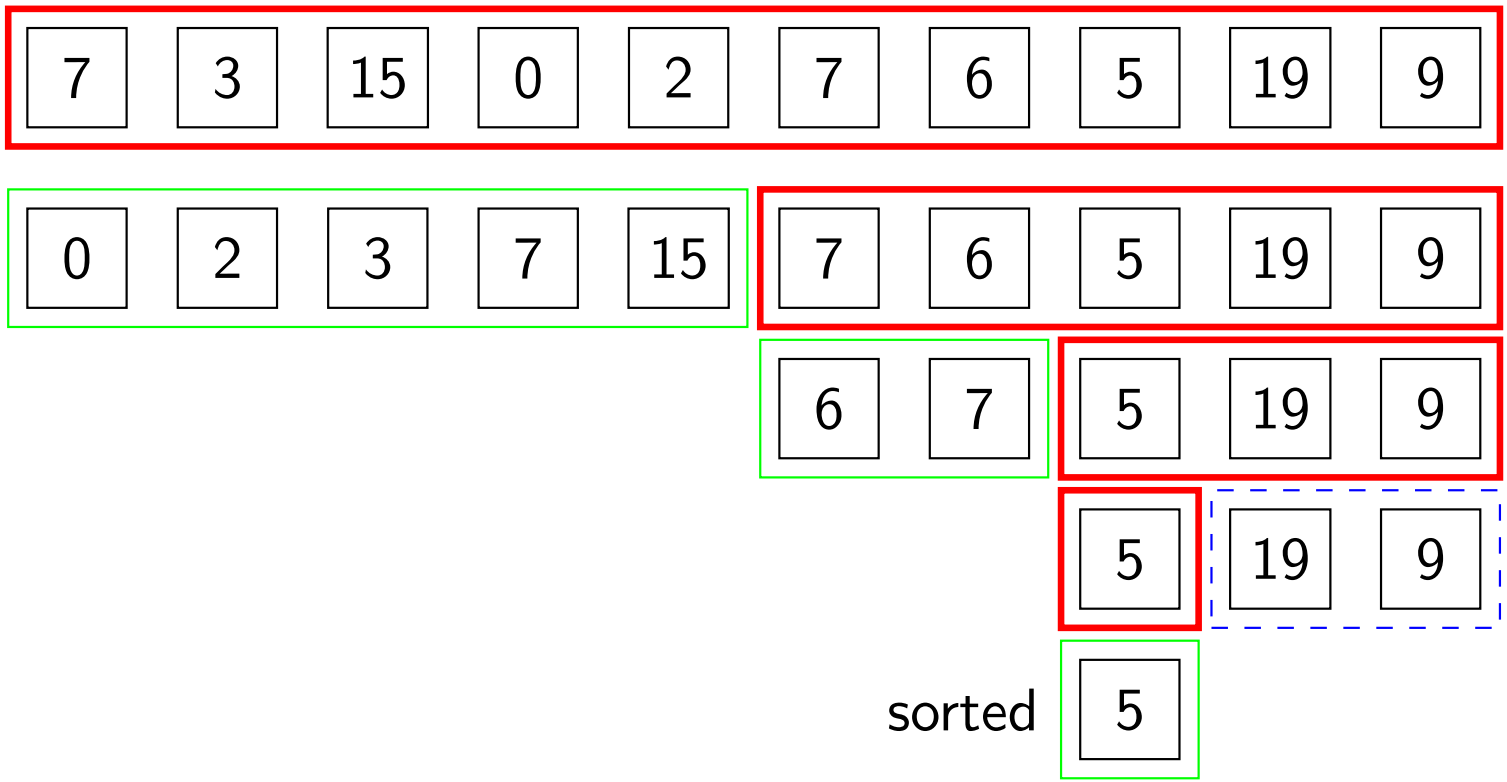
# Exempel Mergesort



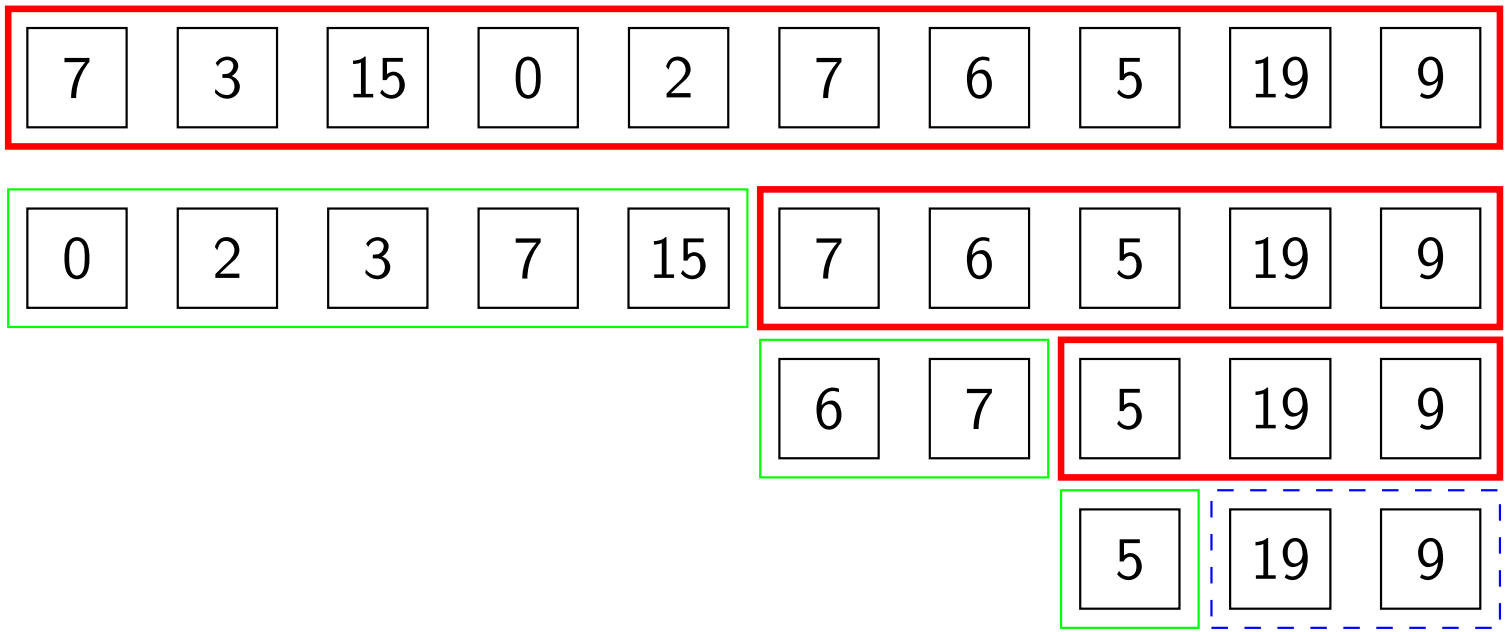
# Exempel Mergesort



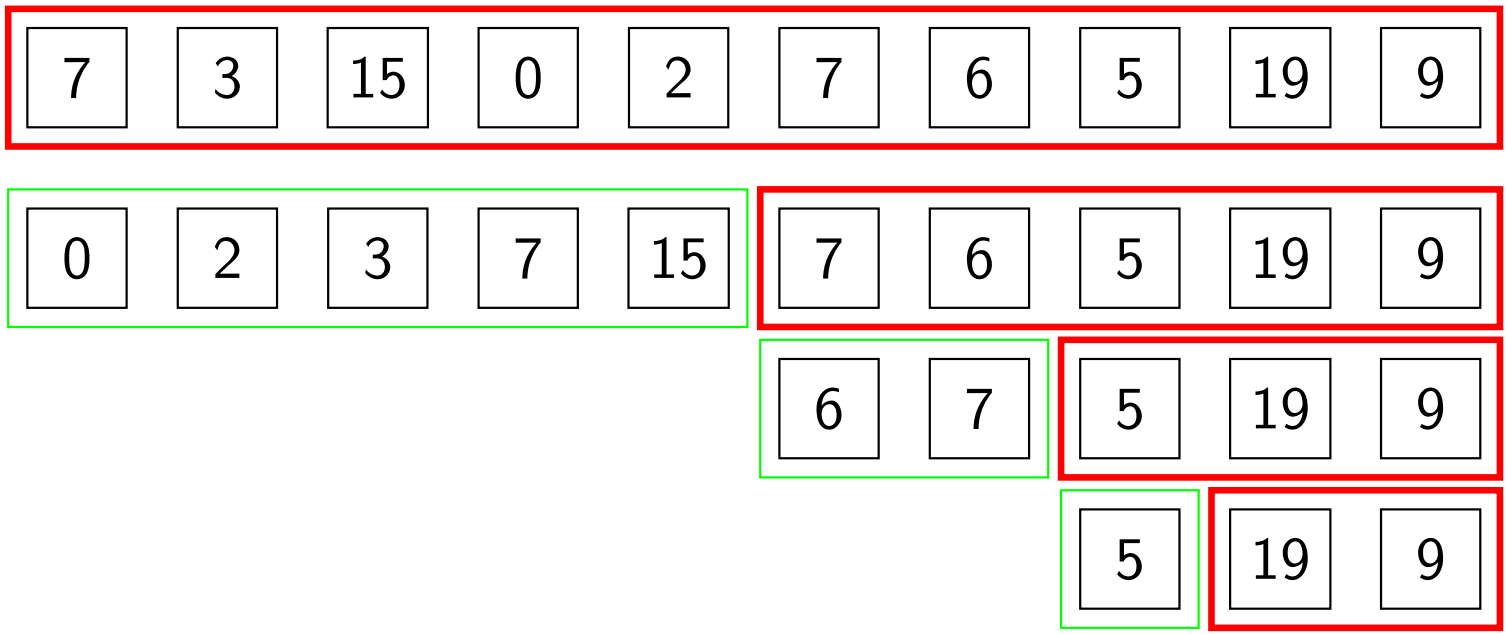
# Exempel Mergesort



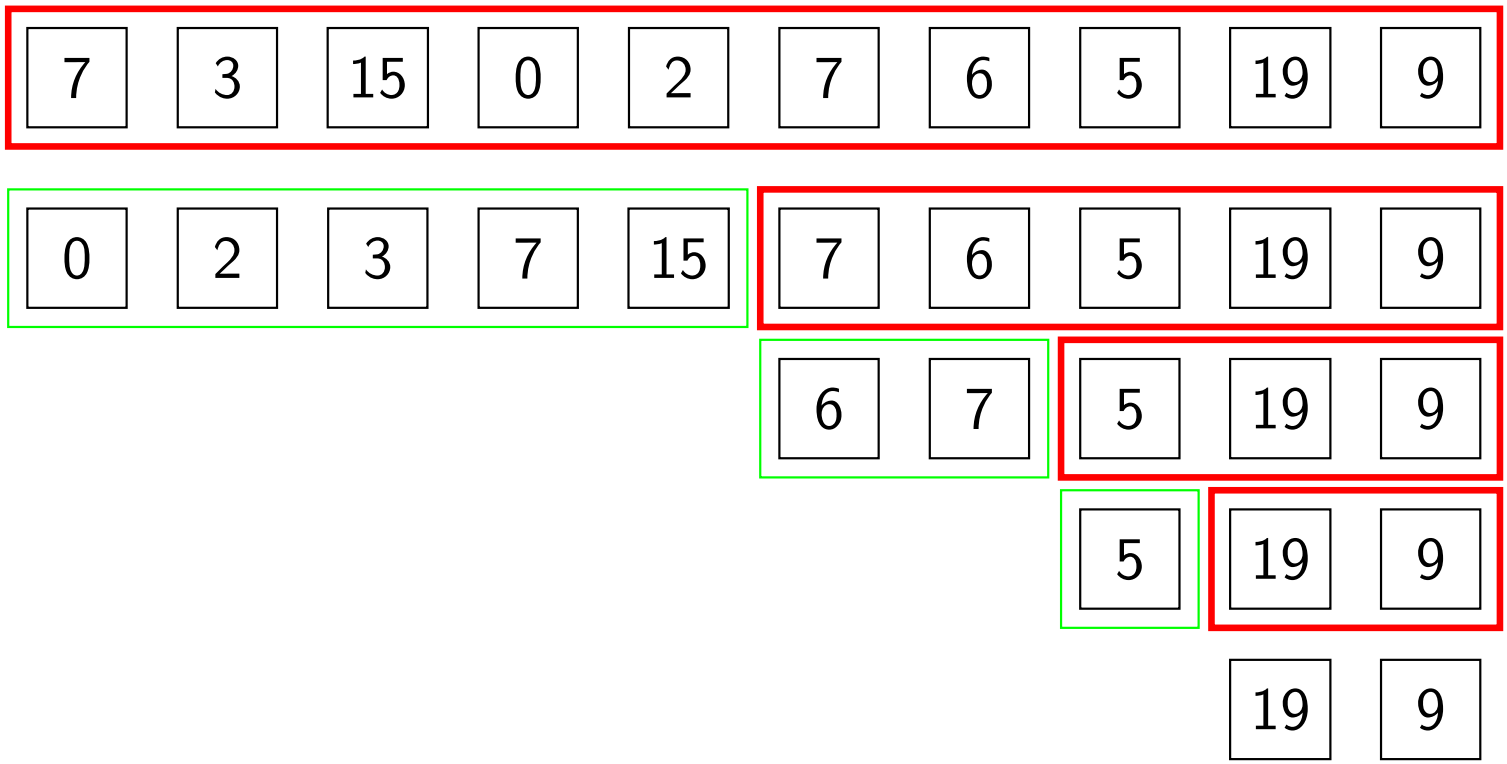
# Exempel Mergesort



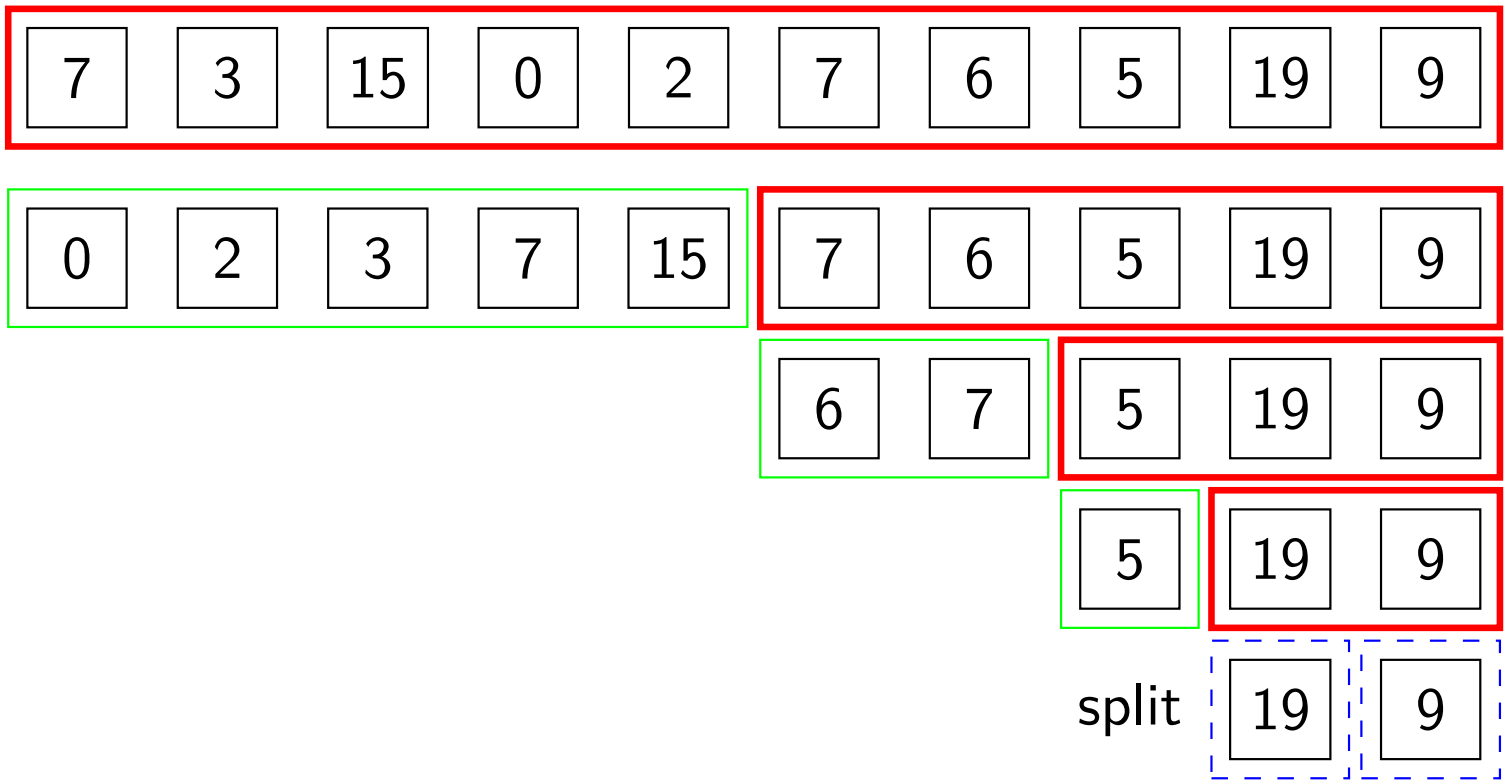
# Exempel Mergesort



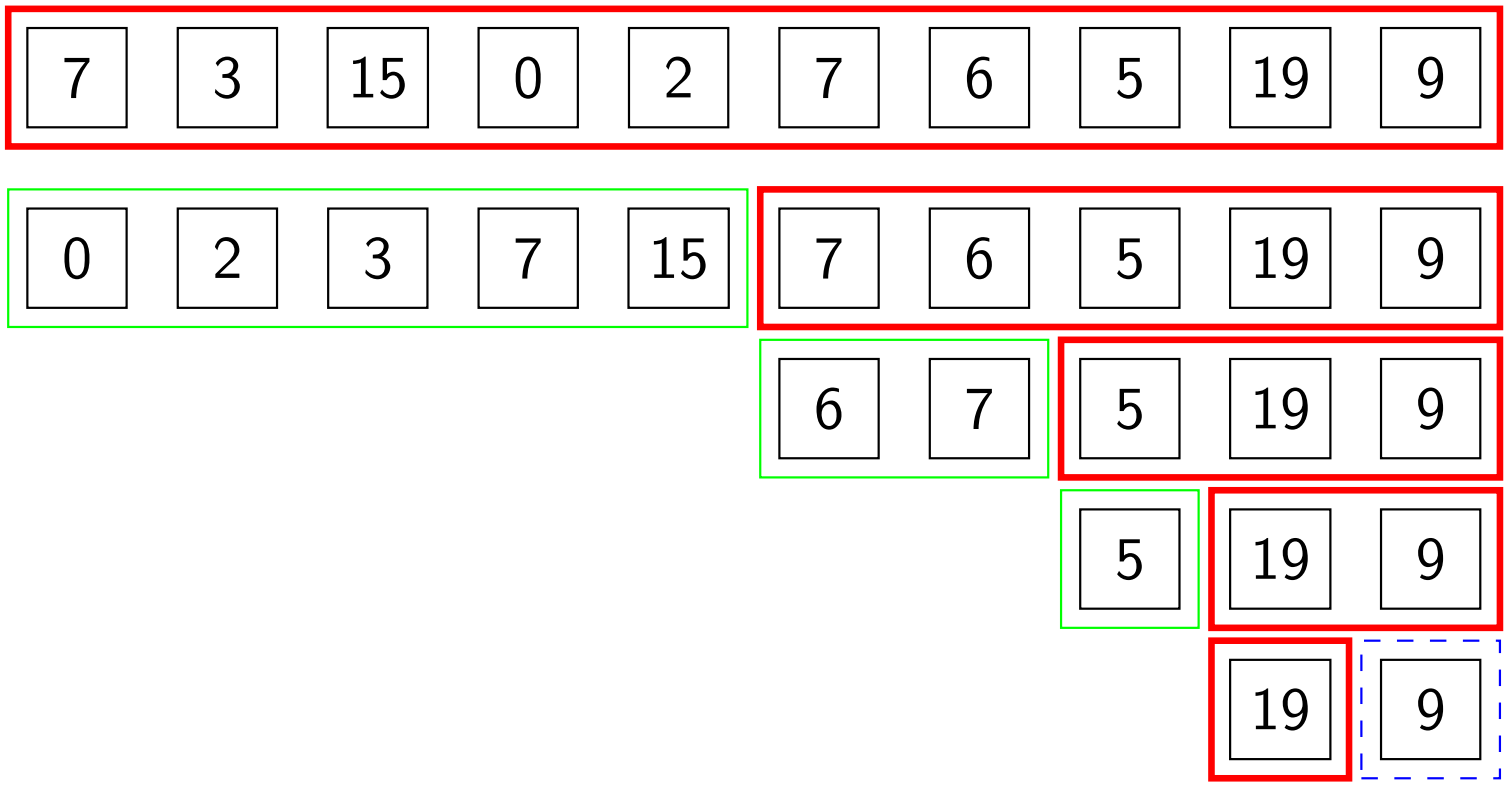
# Exempel Mergesort



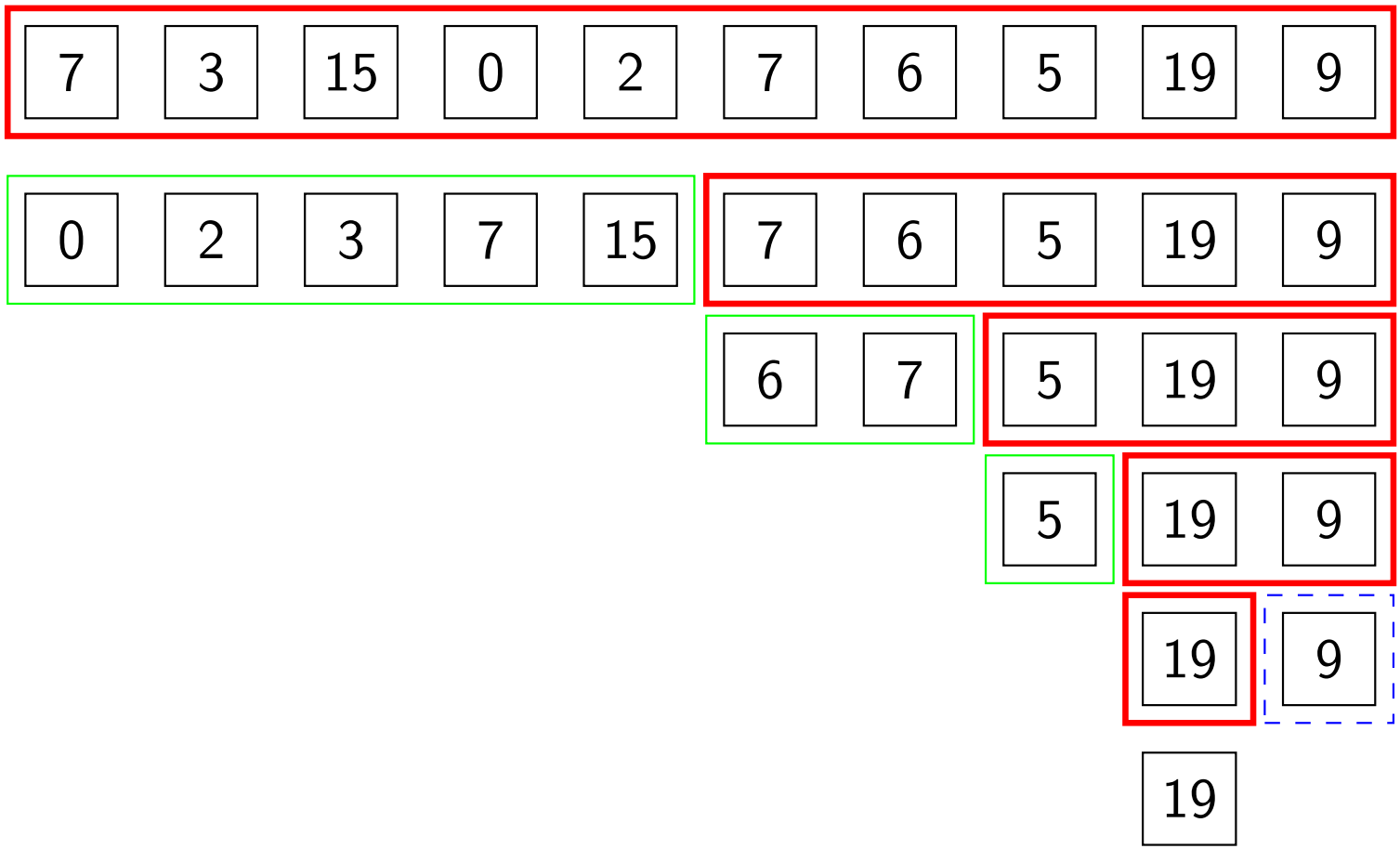
# Exempel Mergesort



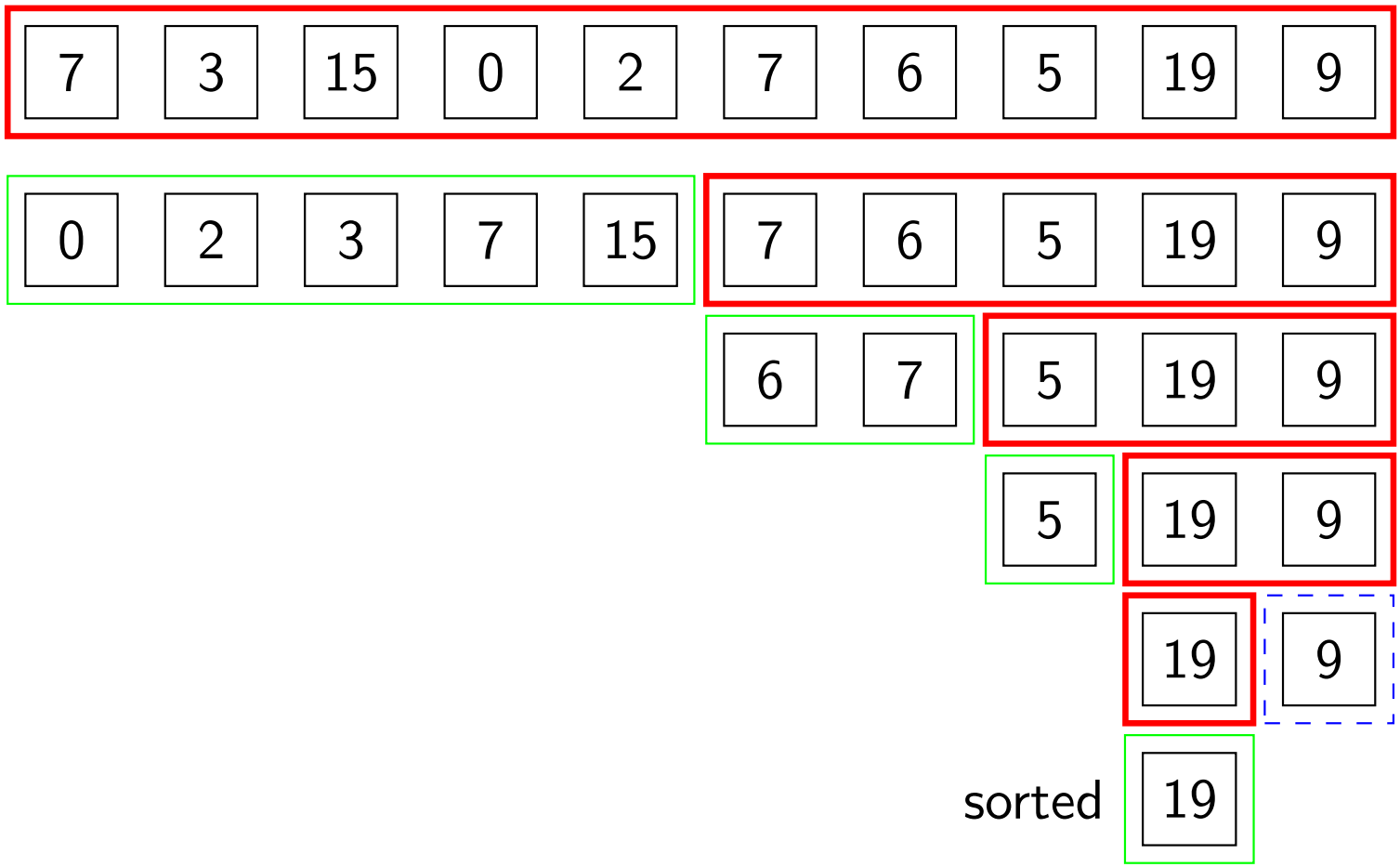
# Exempel Mergesort



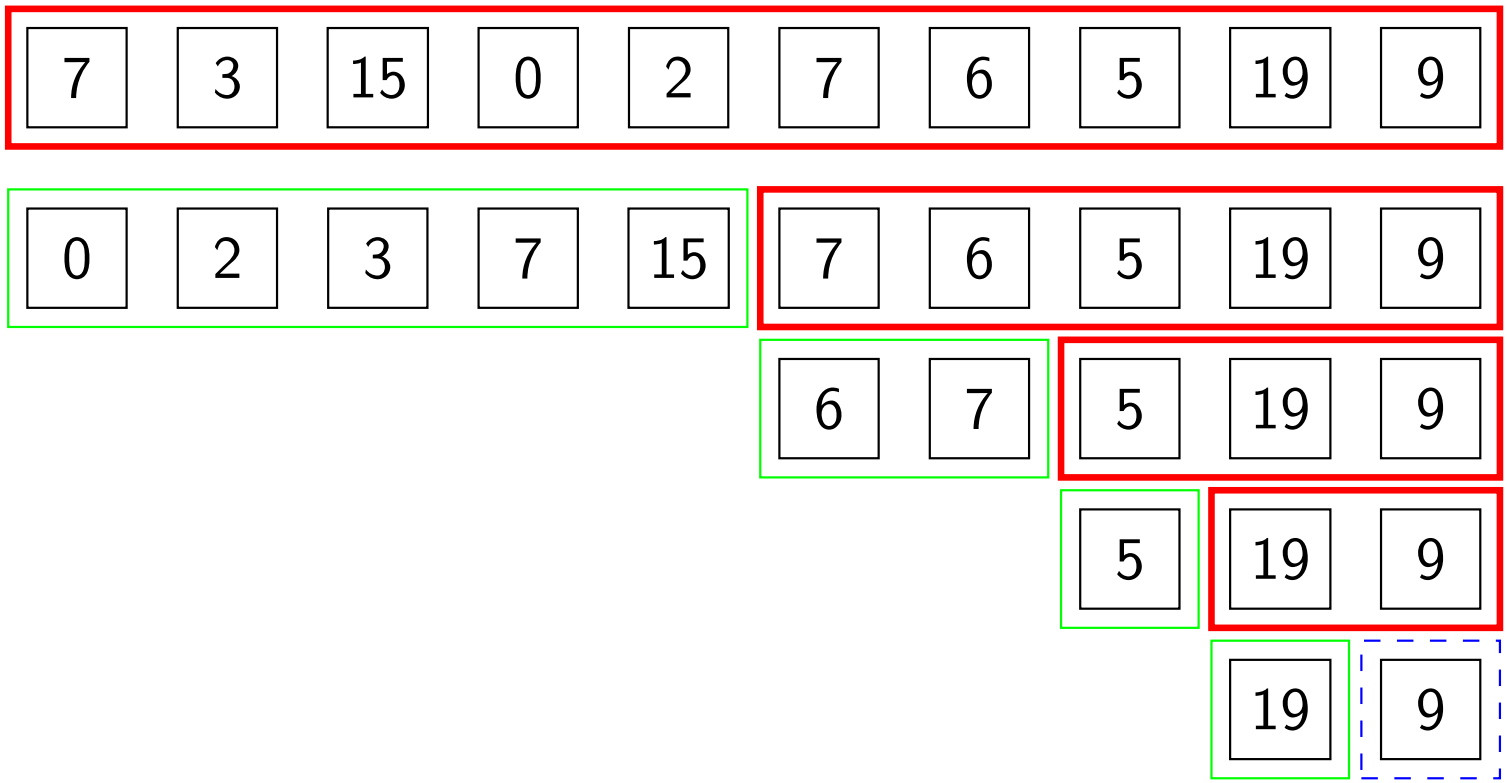
# Exempel Mergesort



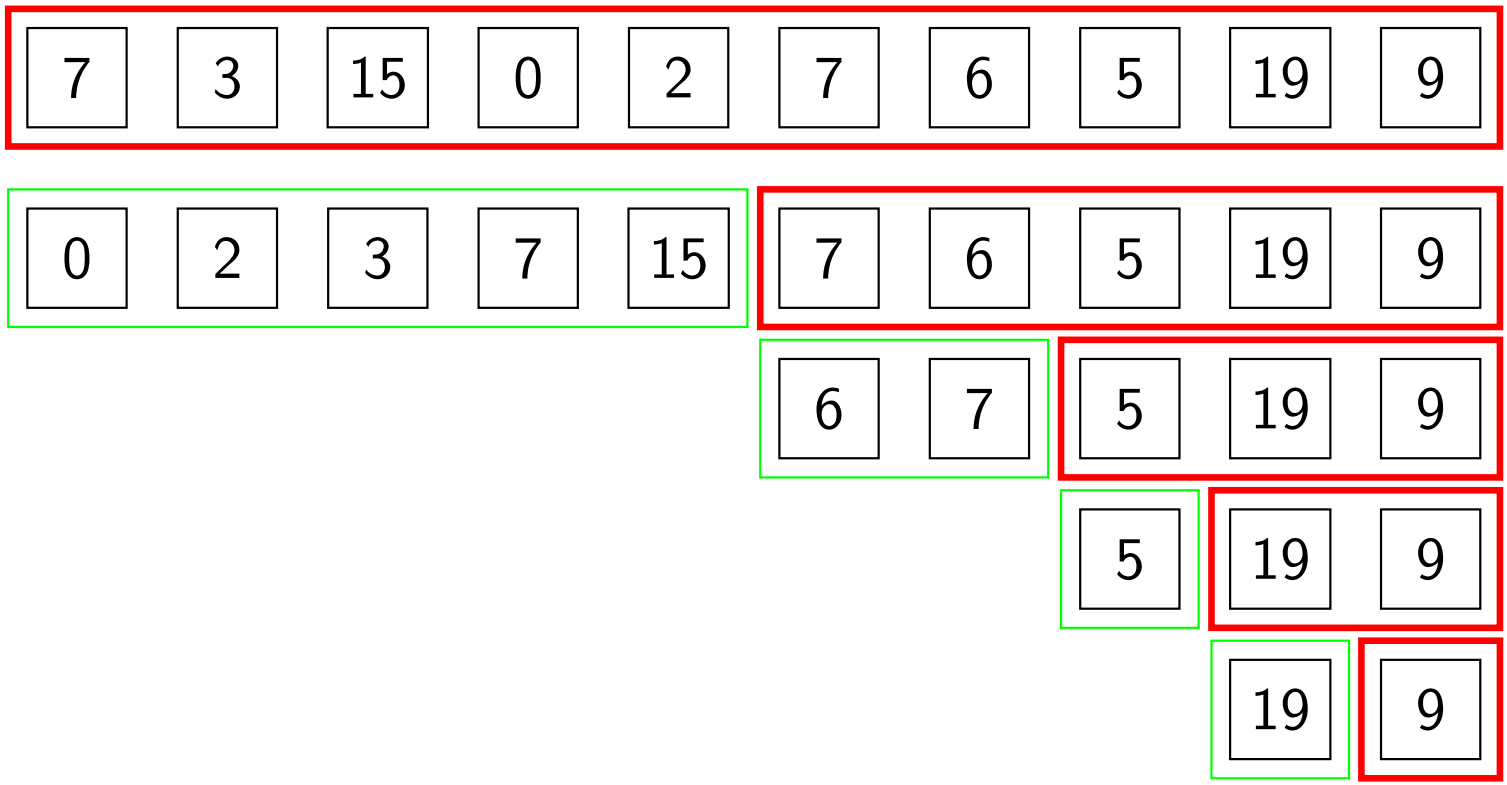
# Exempel Mergesort



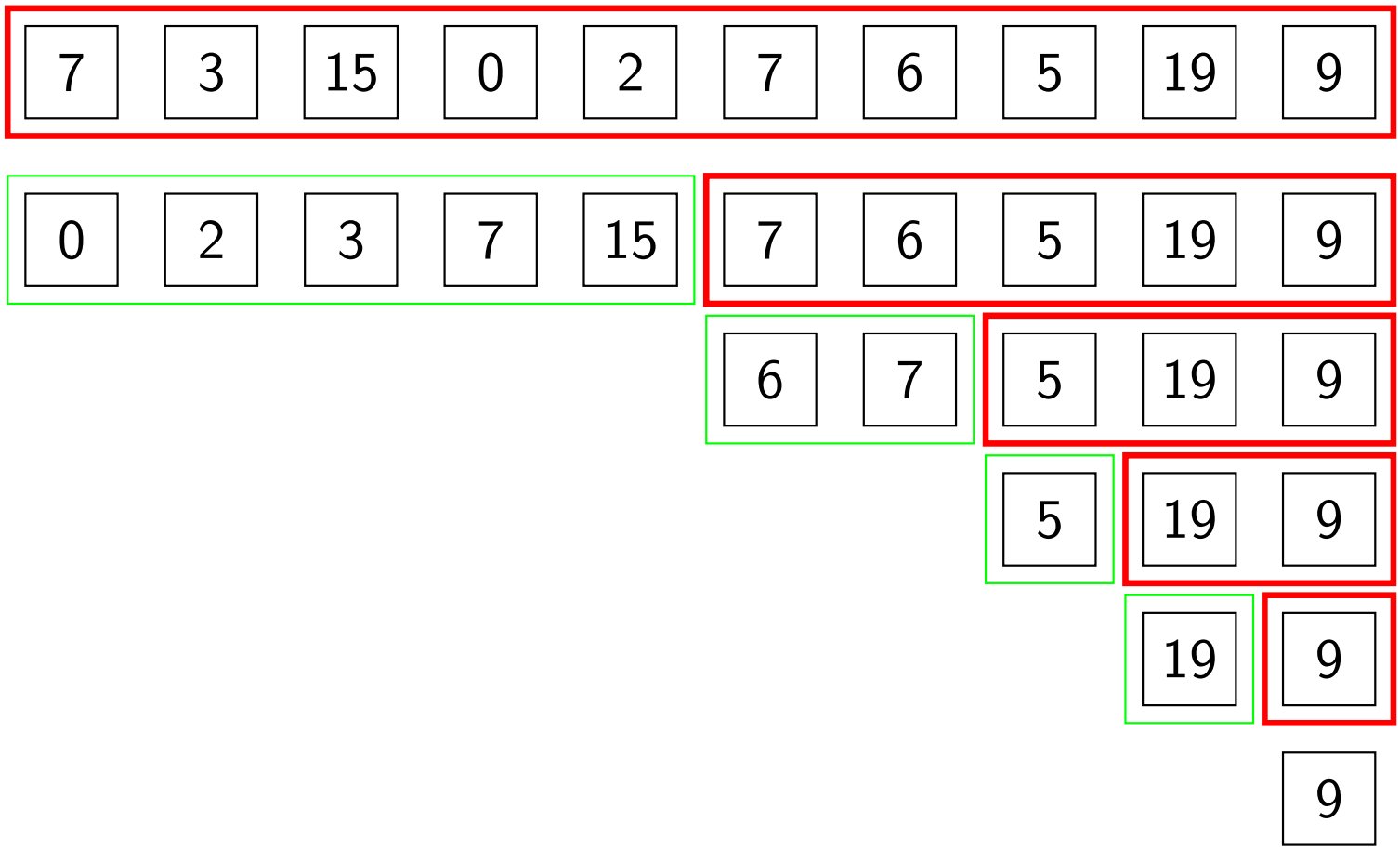
# Exempel Mergesort



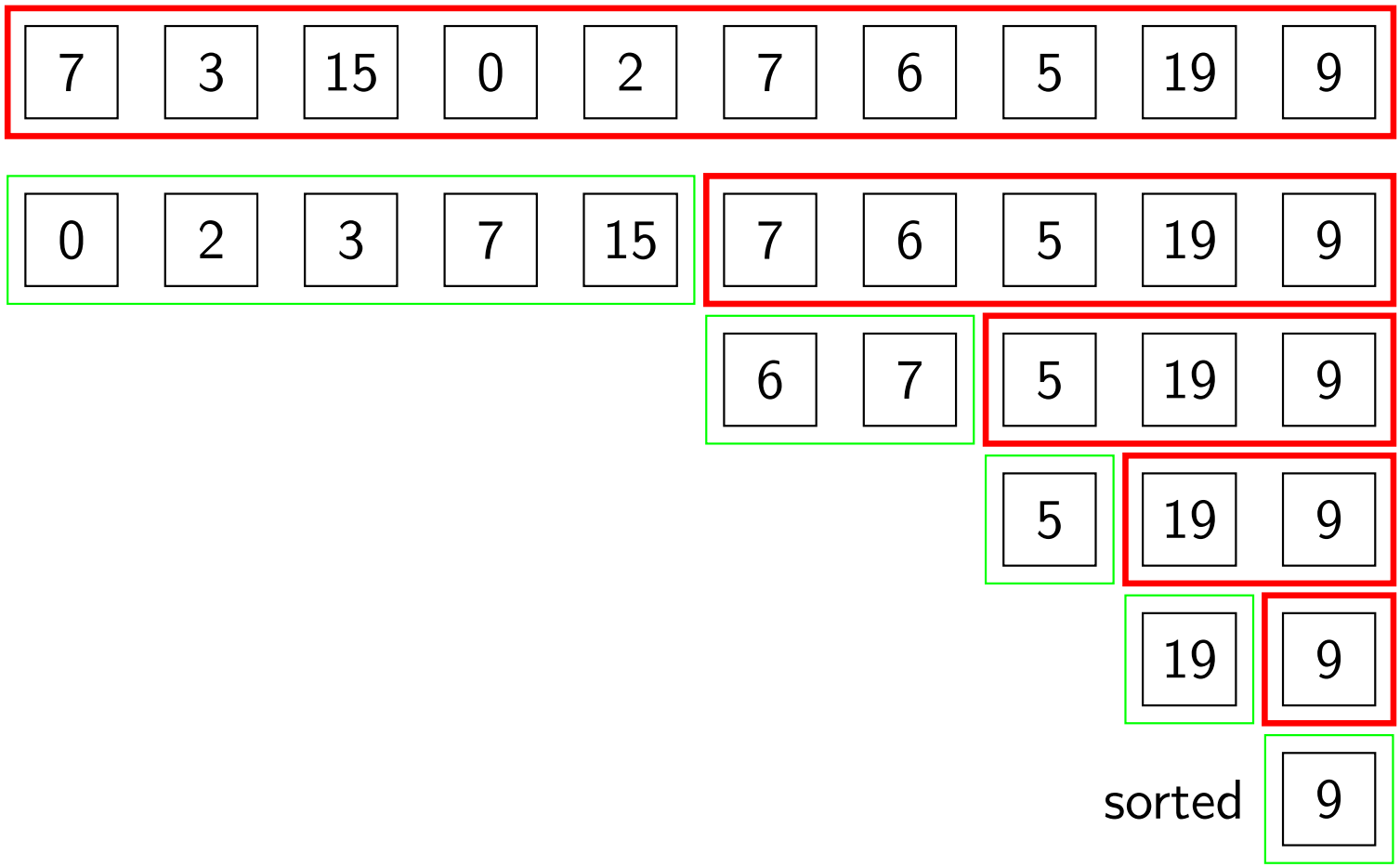
# Exempel Mergesort



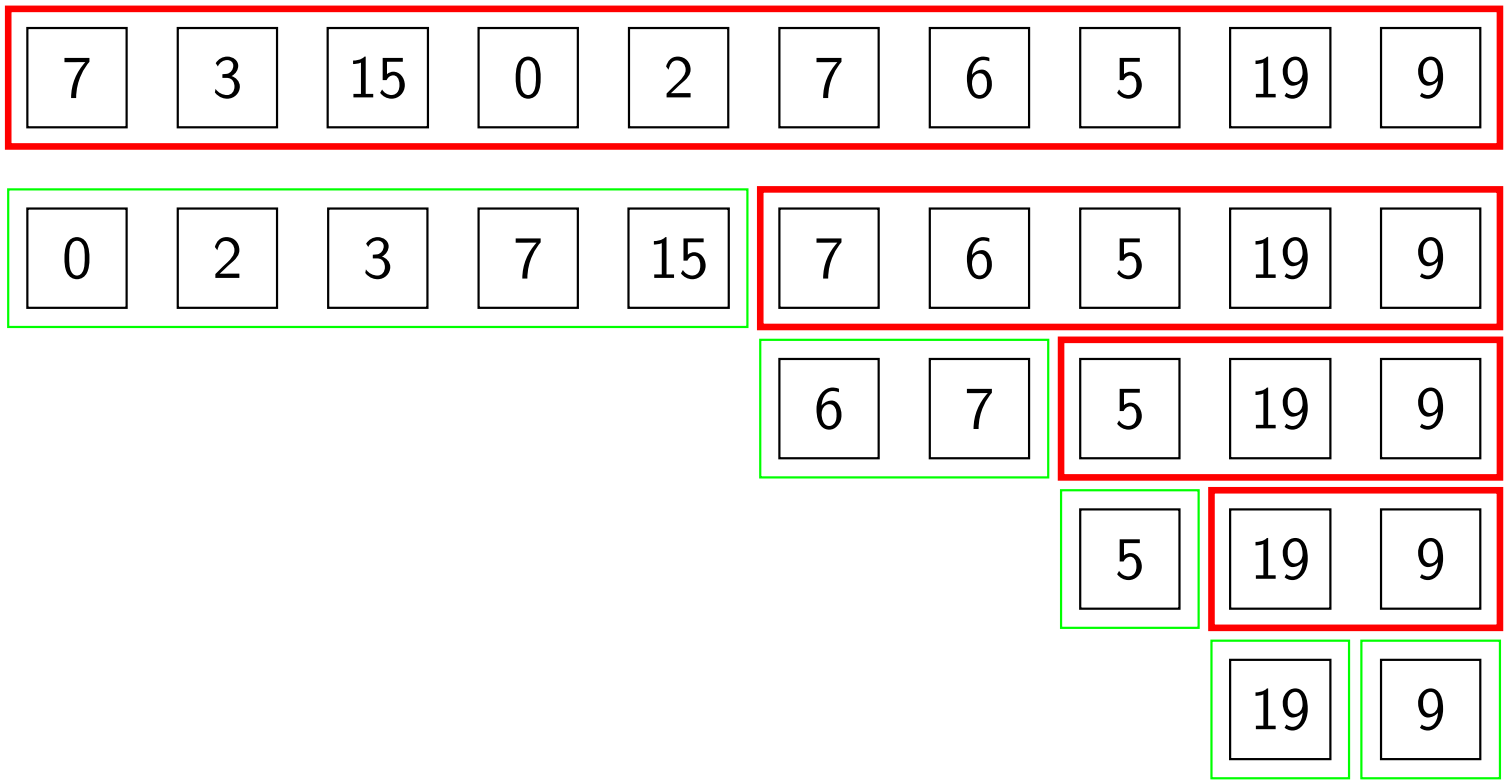
# Exempel Mergesort



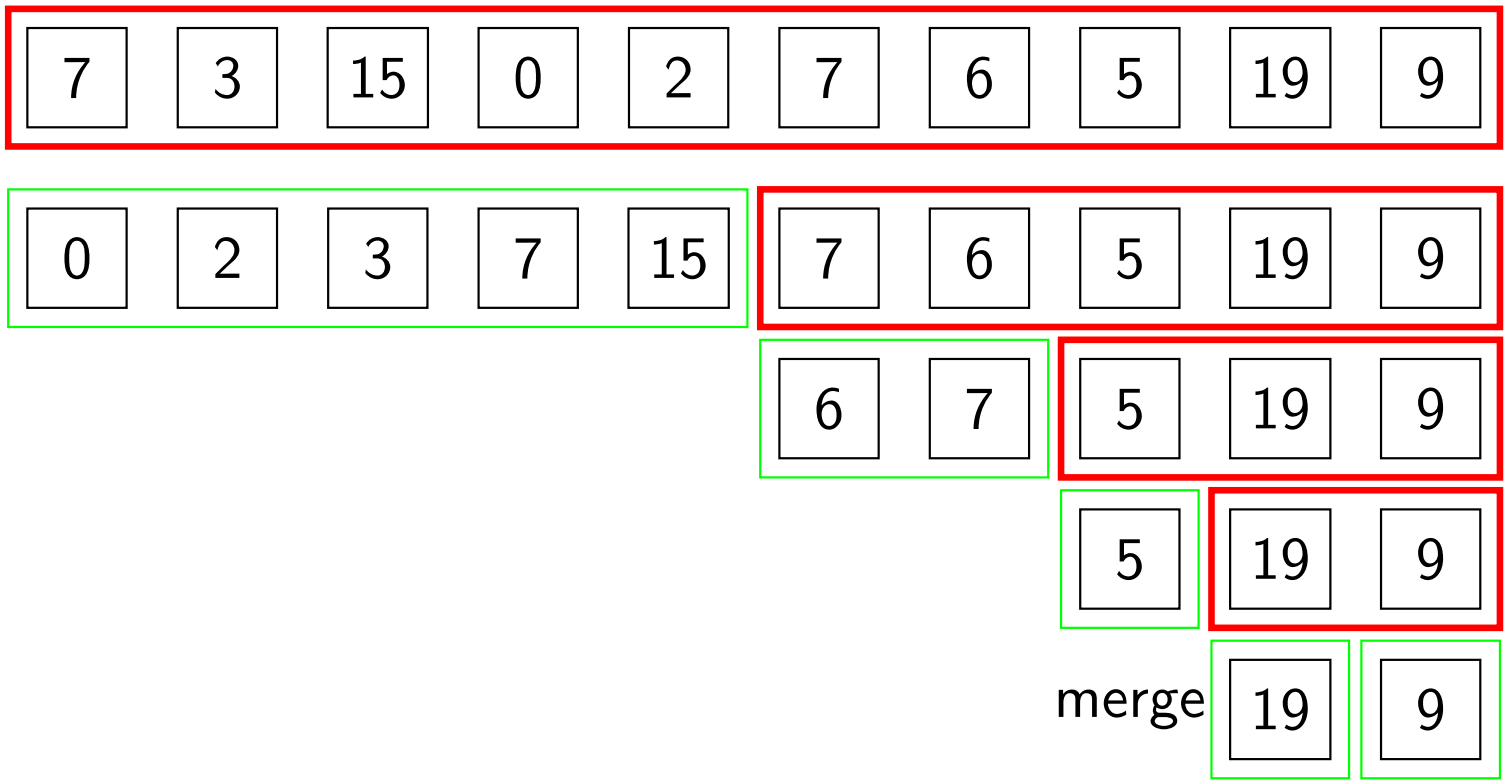
# Exempel Mergesort



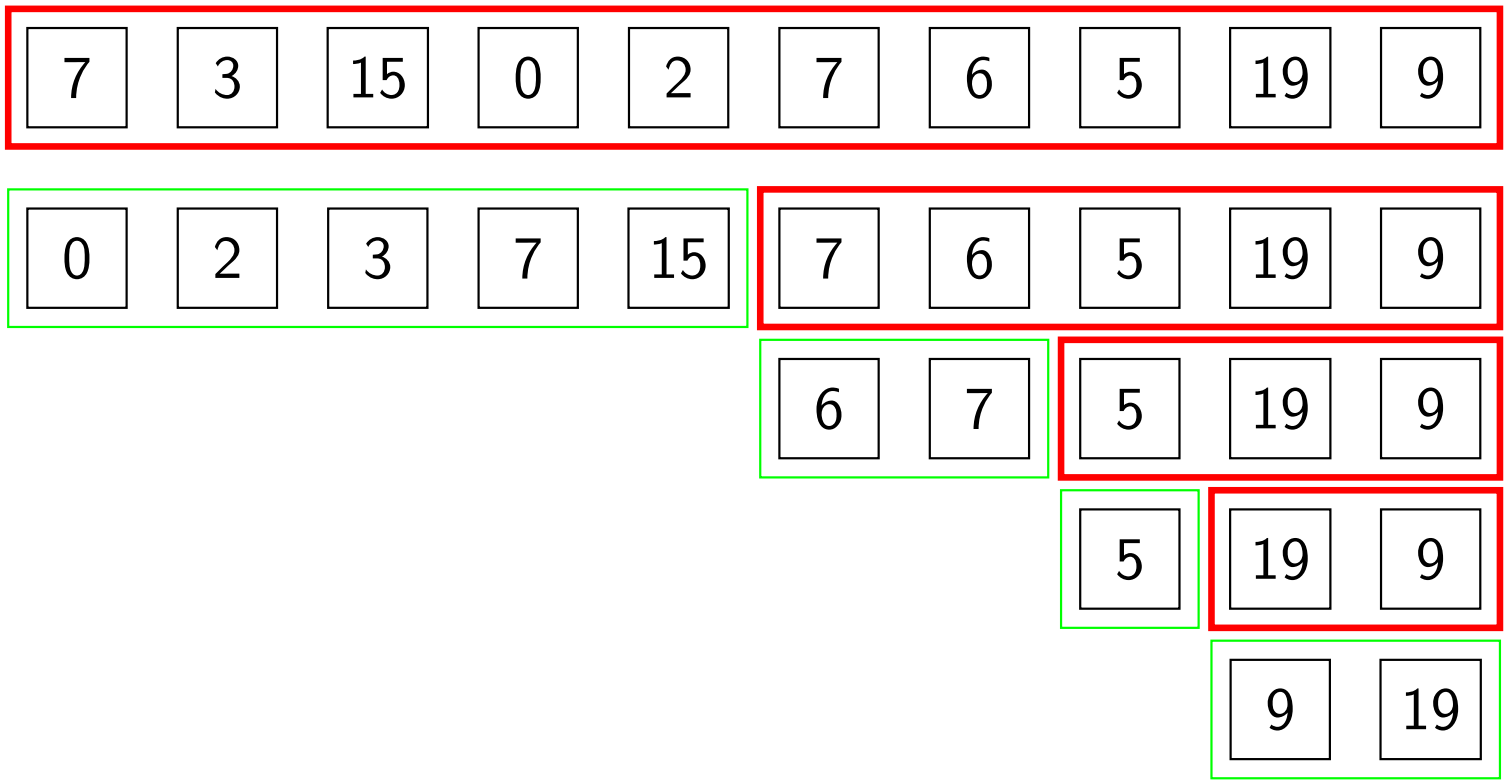
# Exempel Mergesort



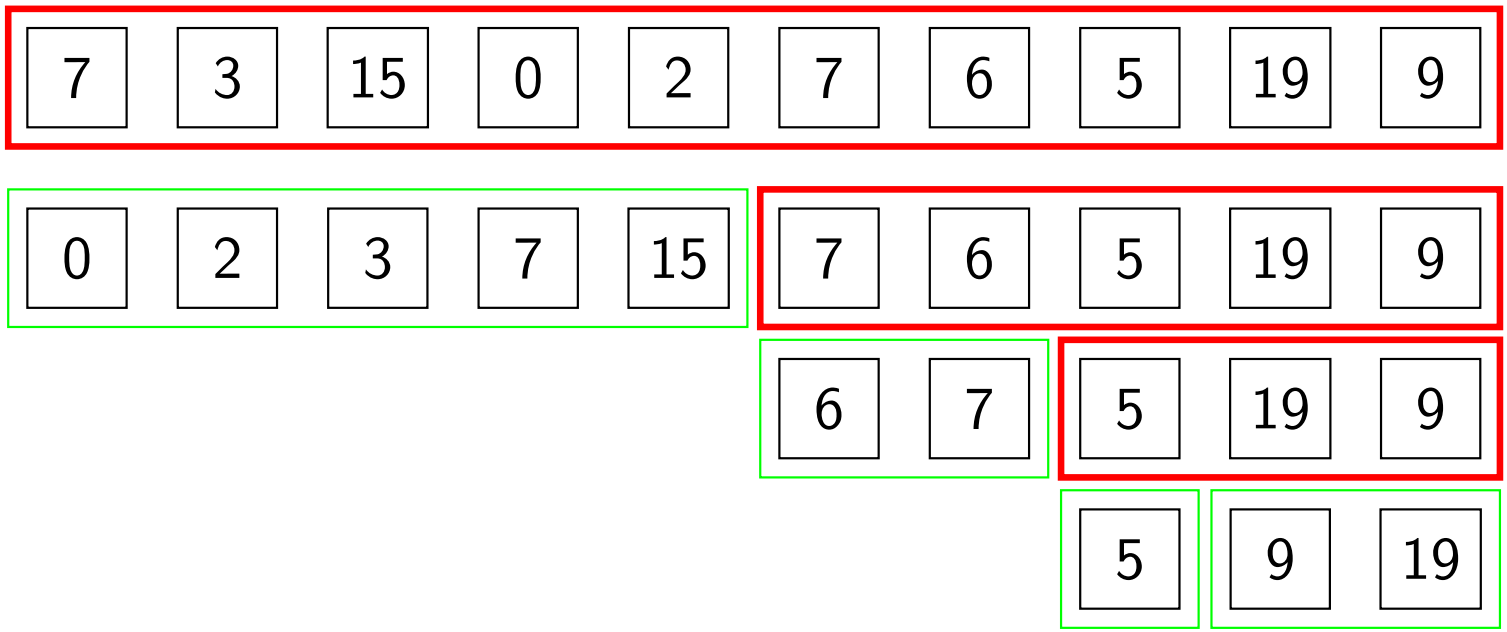
# Exempel Mergesort



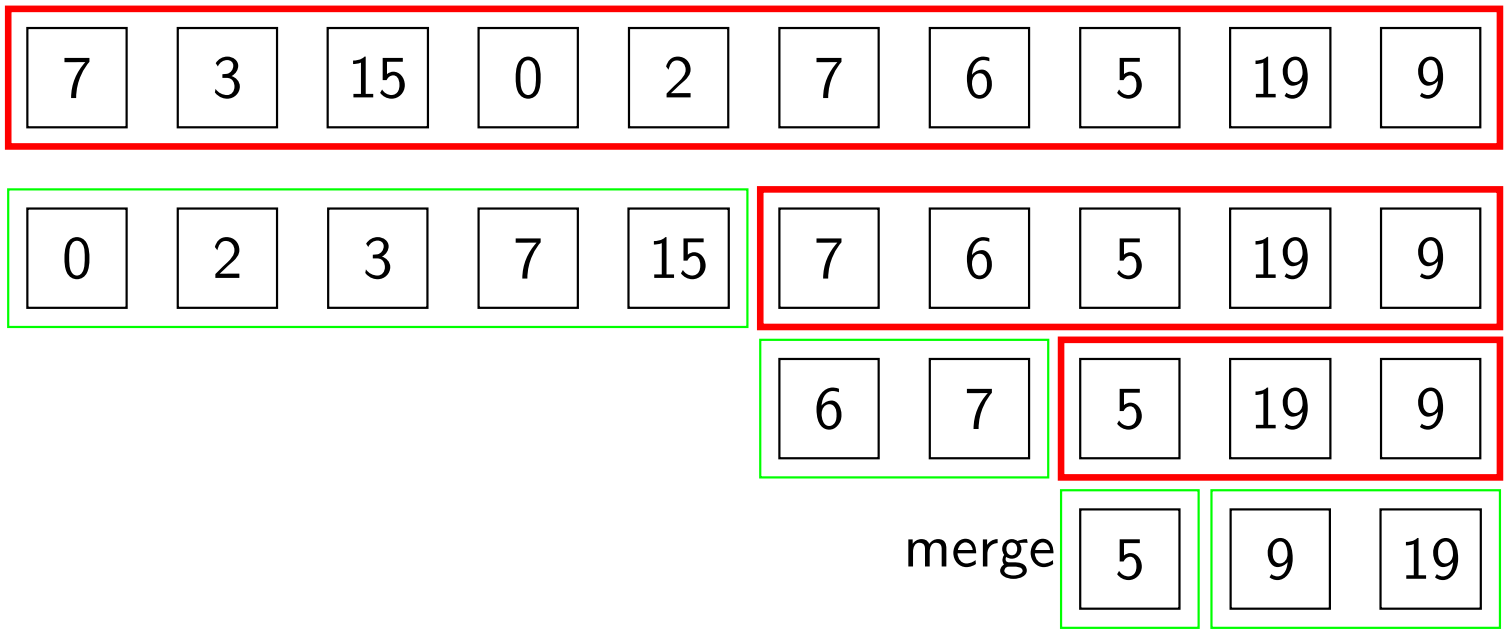
# Exempel Mergesort



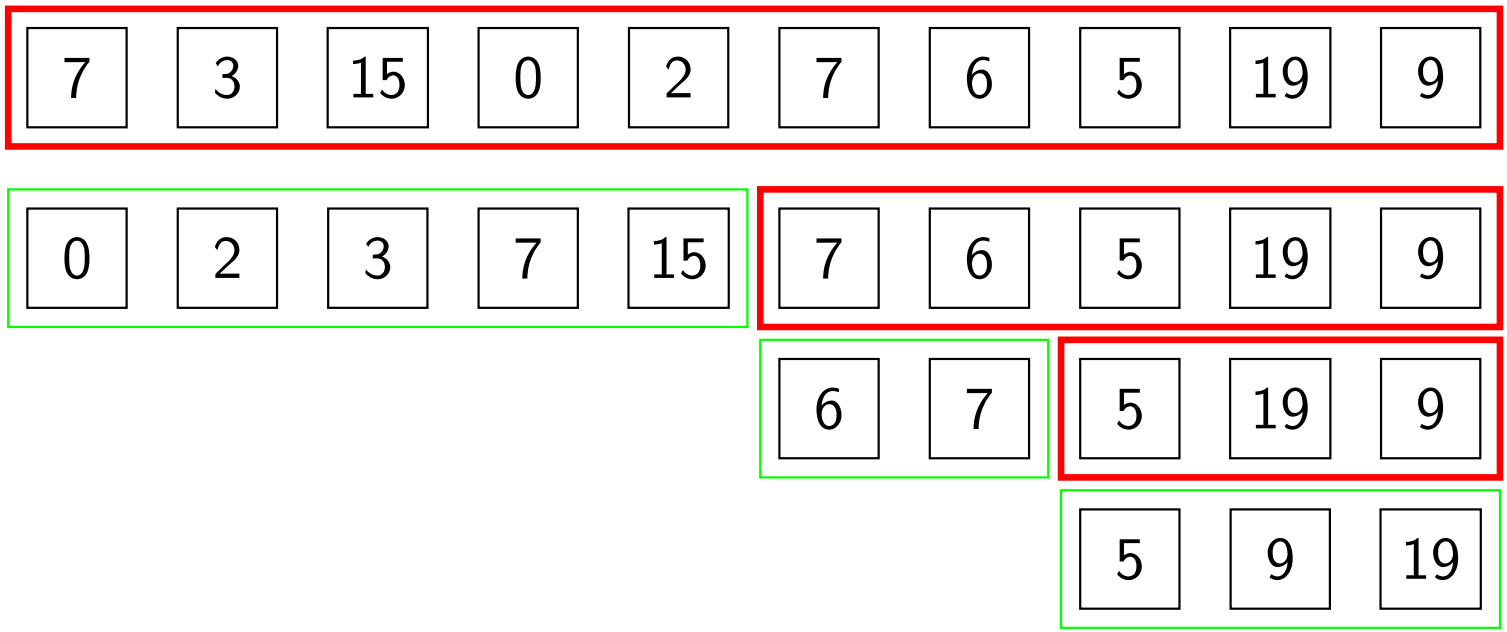
# Exempel Mergesort



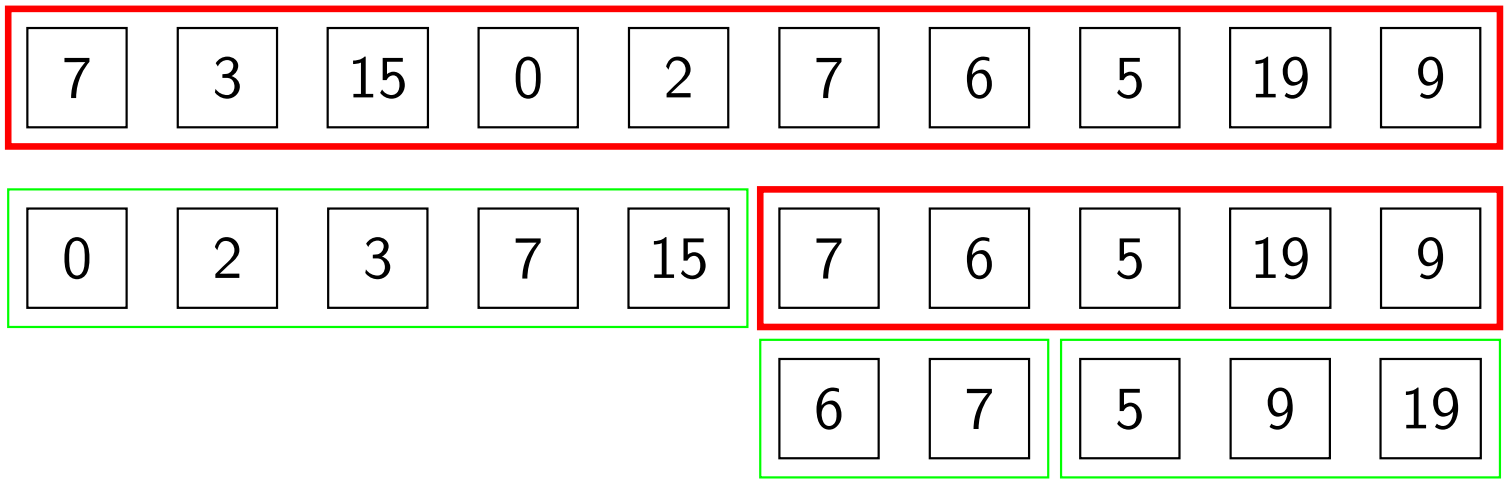
# Exempel Mergesort



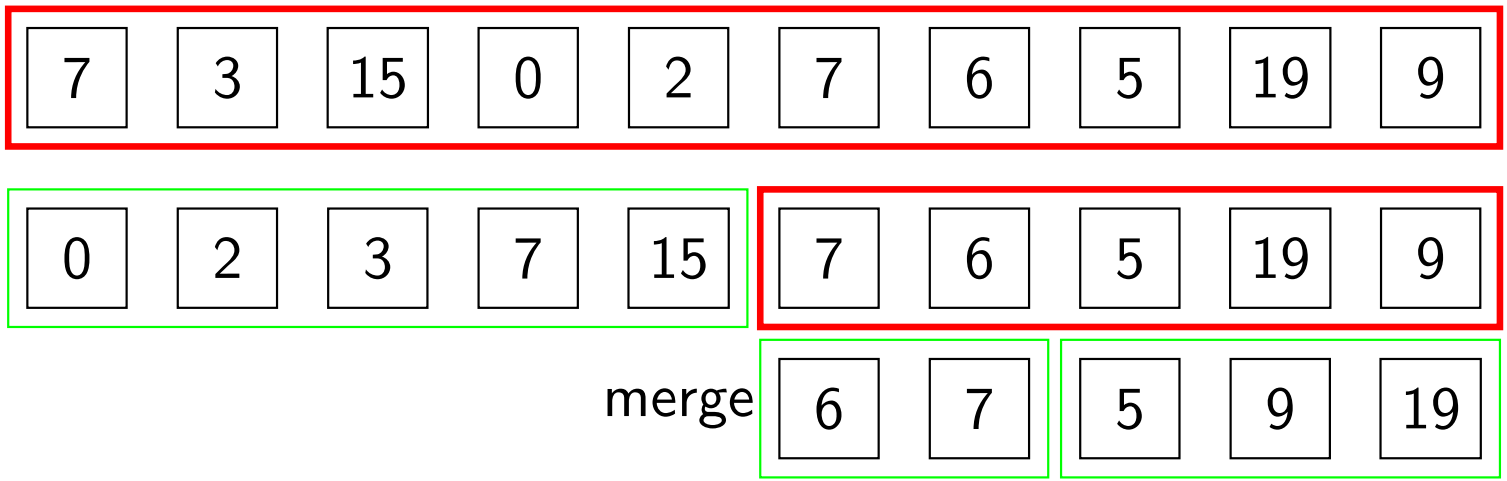
# Exempel Mergesort



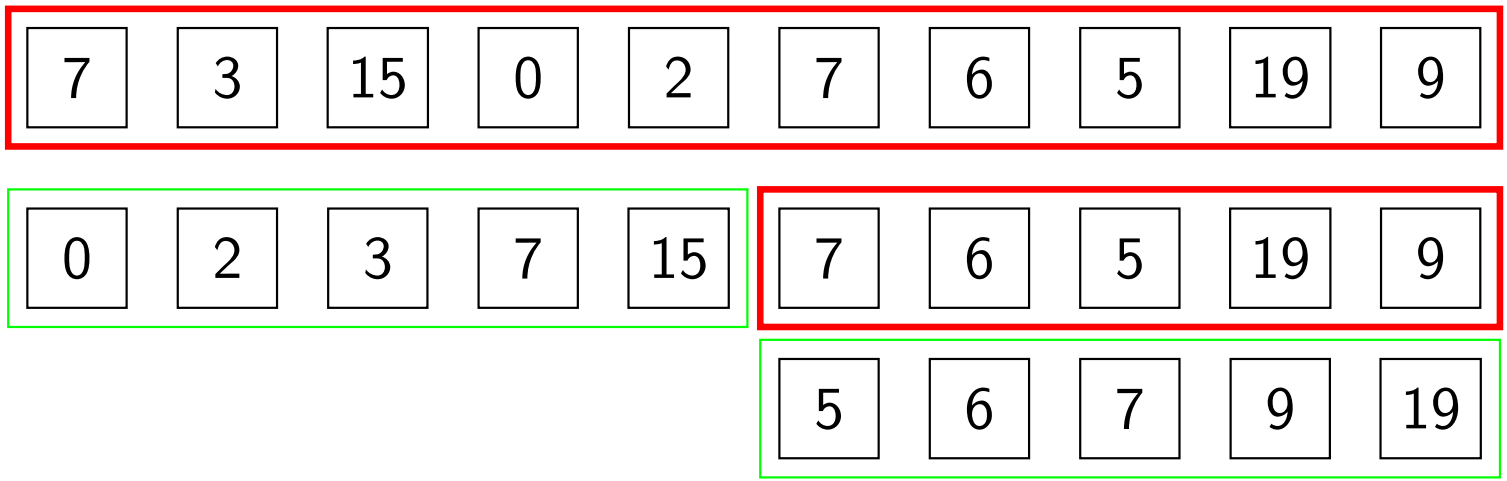
# Exempel Mergesort



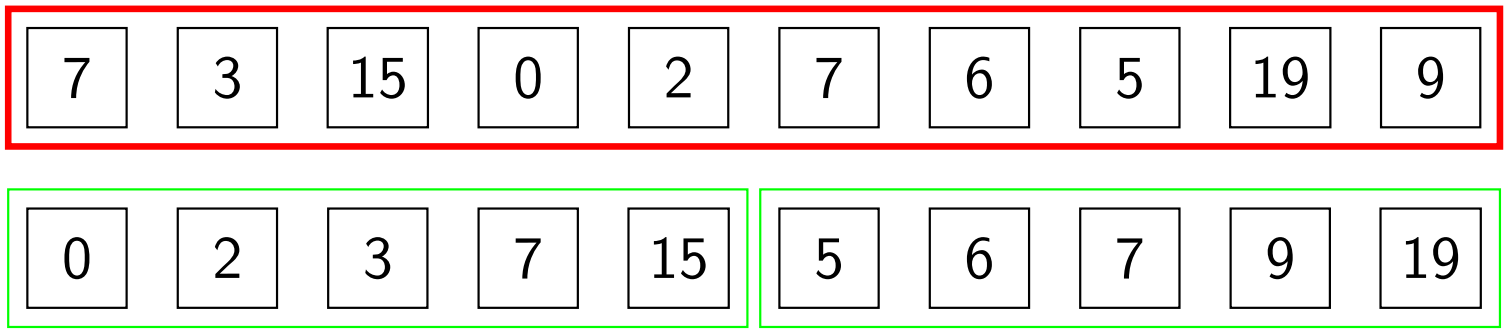
# Exempel Mergesort



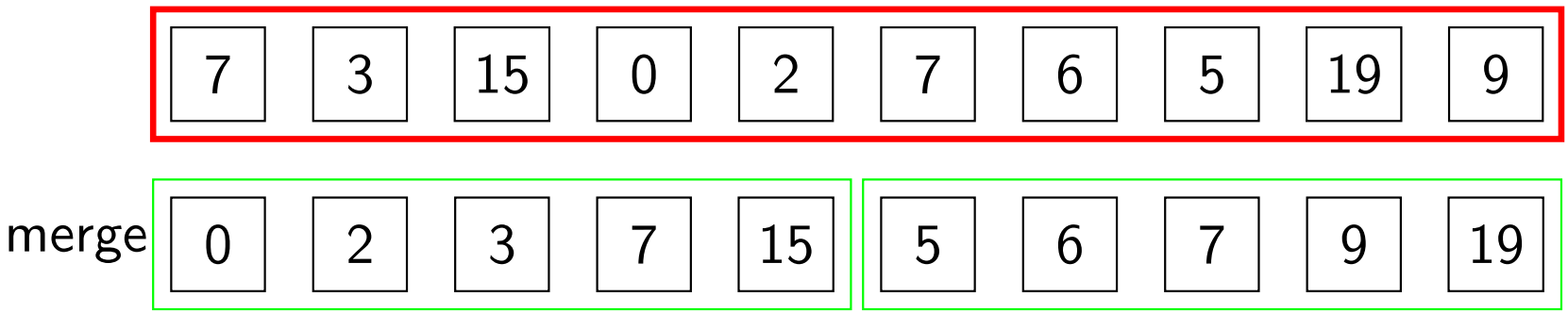
# Exempel Mergesort



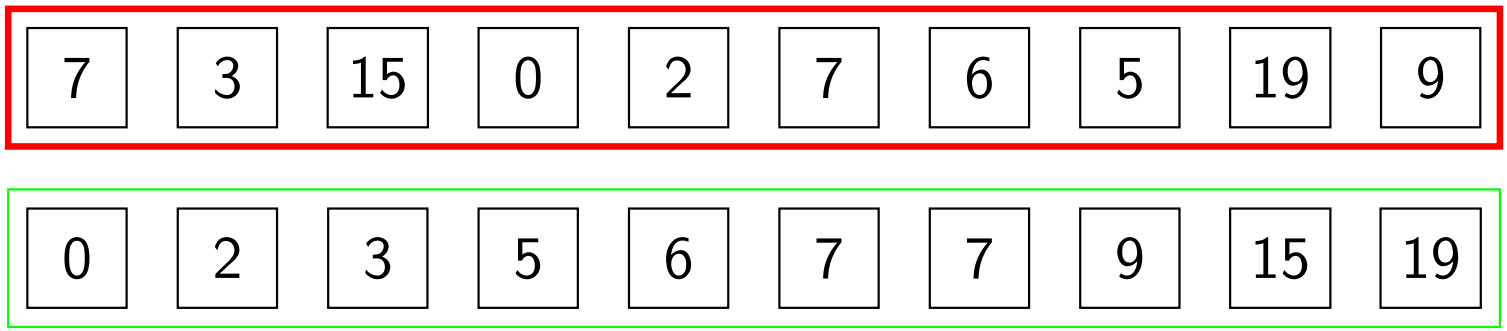
# Exempel Mergesort



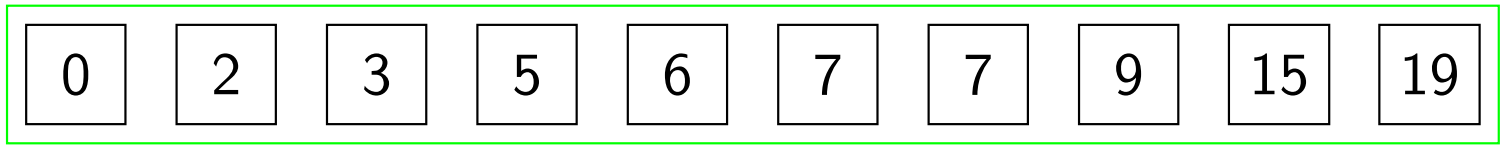
# Exempel Mergesort



# Exempel Mergesort



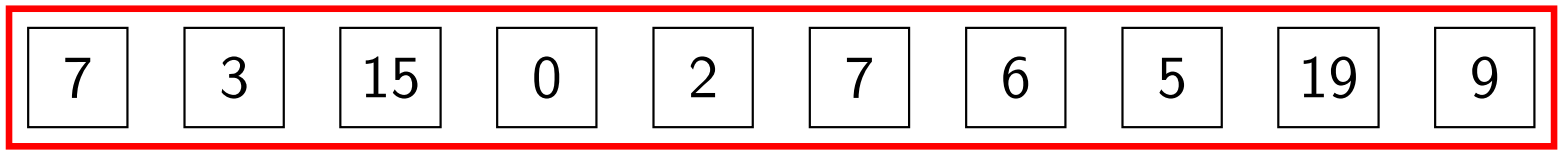
# Exempel Mergesort



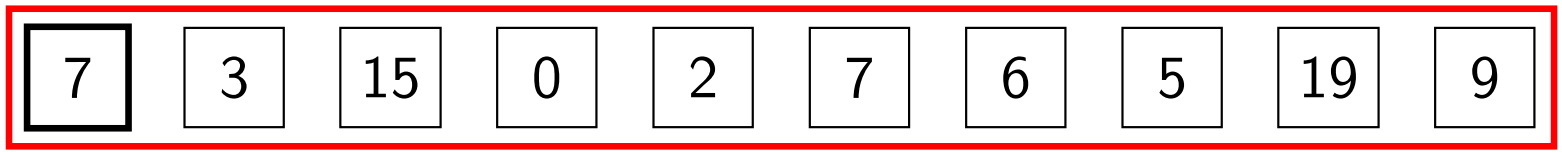
# Quicksort

- Algoritm:
  - Välj ut ett pivåelement.
  - Dela upp listan i tre delar: Less, Equal, Greater.
  - Sortera Less och Greater rekursivt.
  - Slå ihop Less+Equal+Greater.

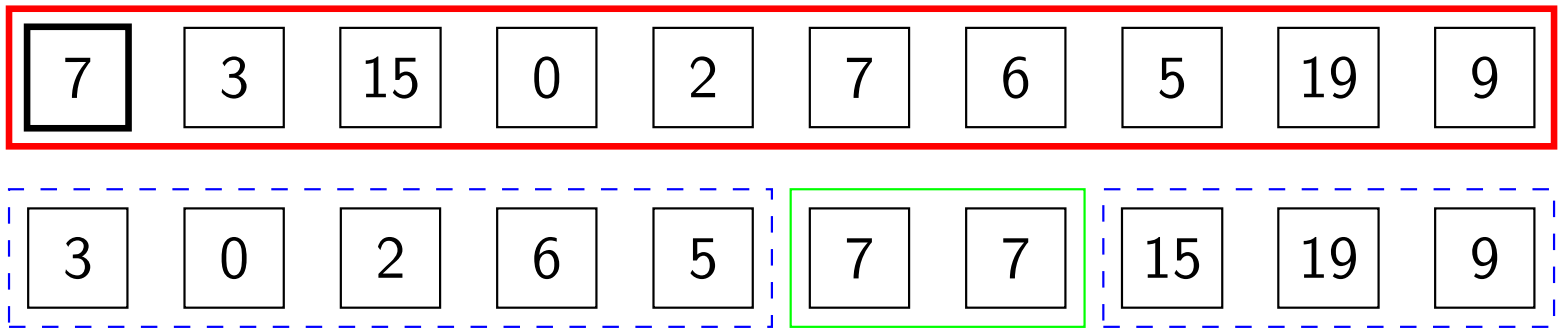
# Quicksort — Exempel



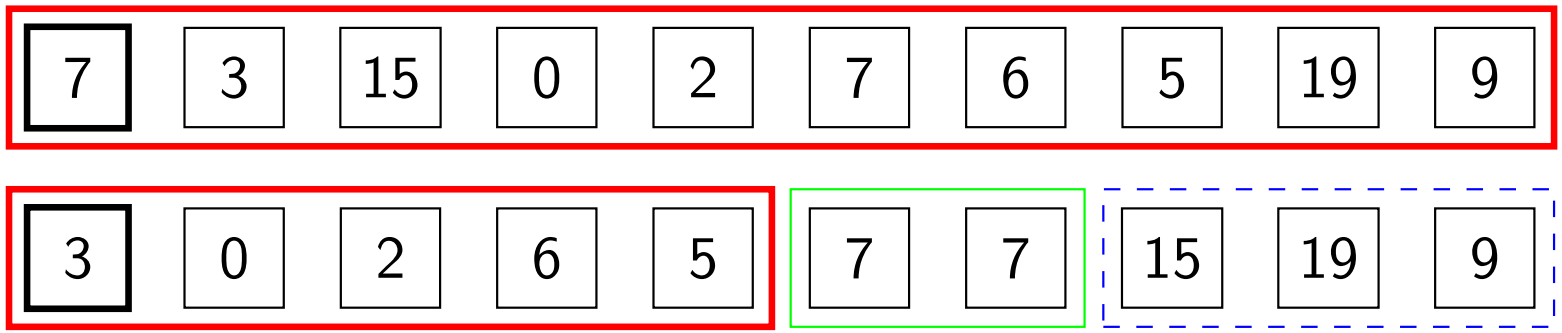
# Quicksort — Exempel



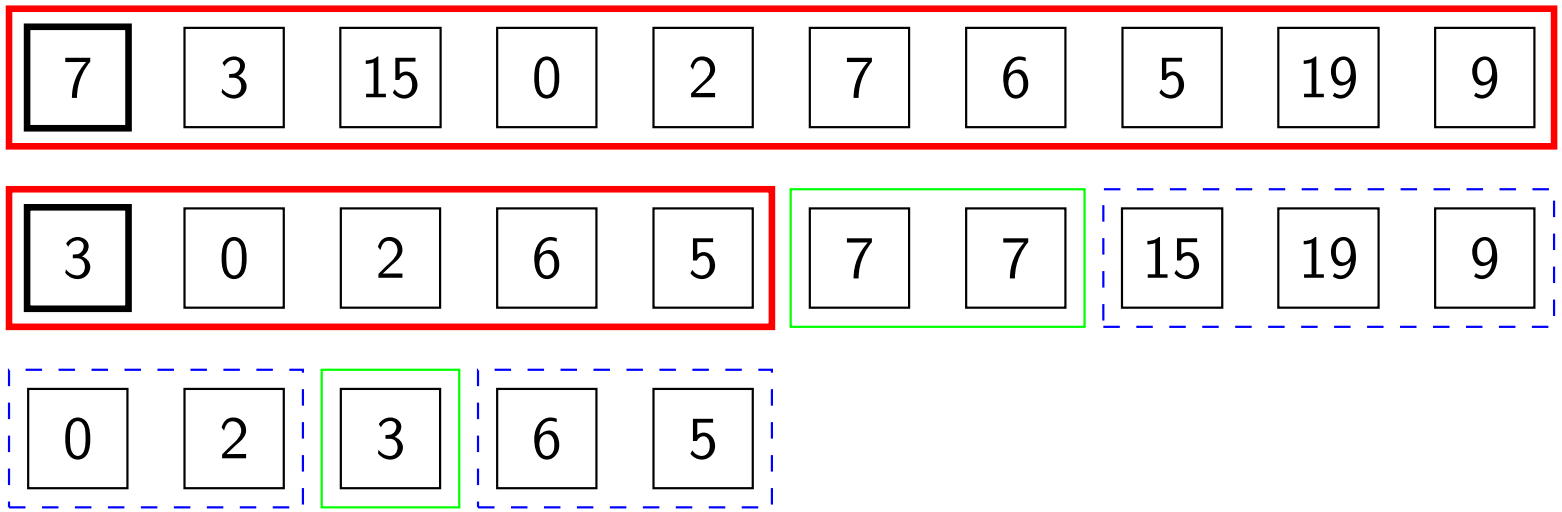
# Quicksort — Exempel



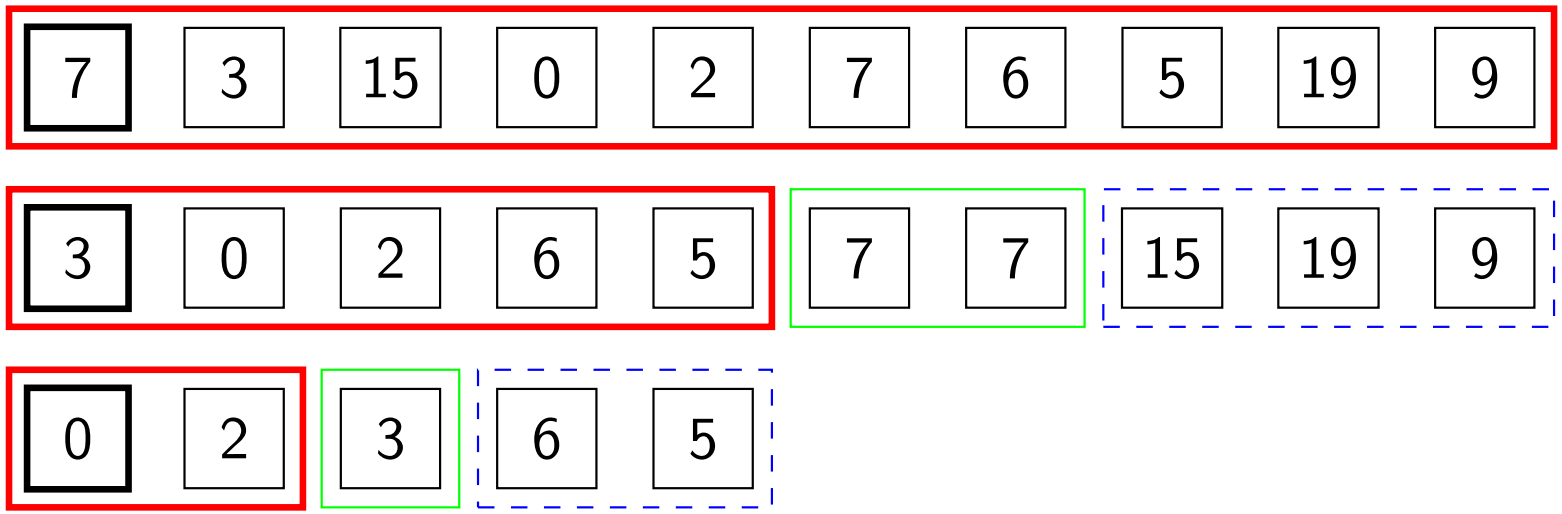
# Quicksort — Exempel



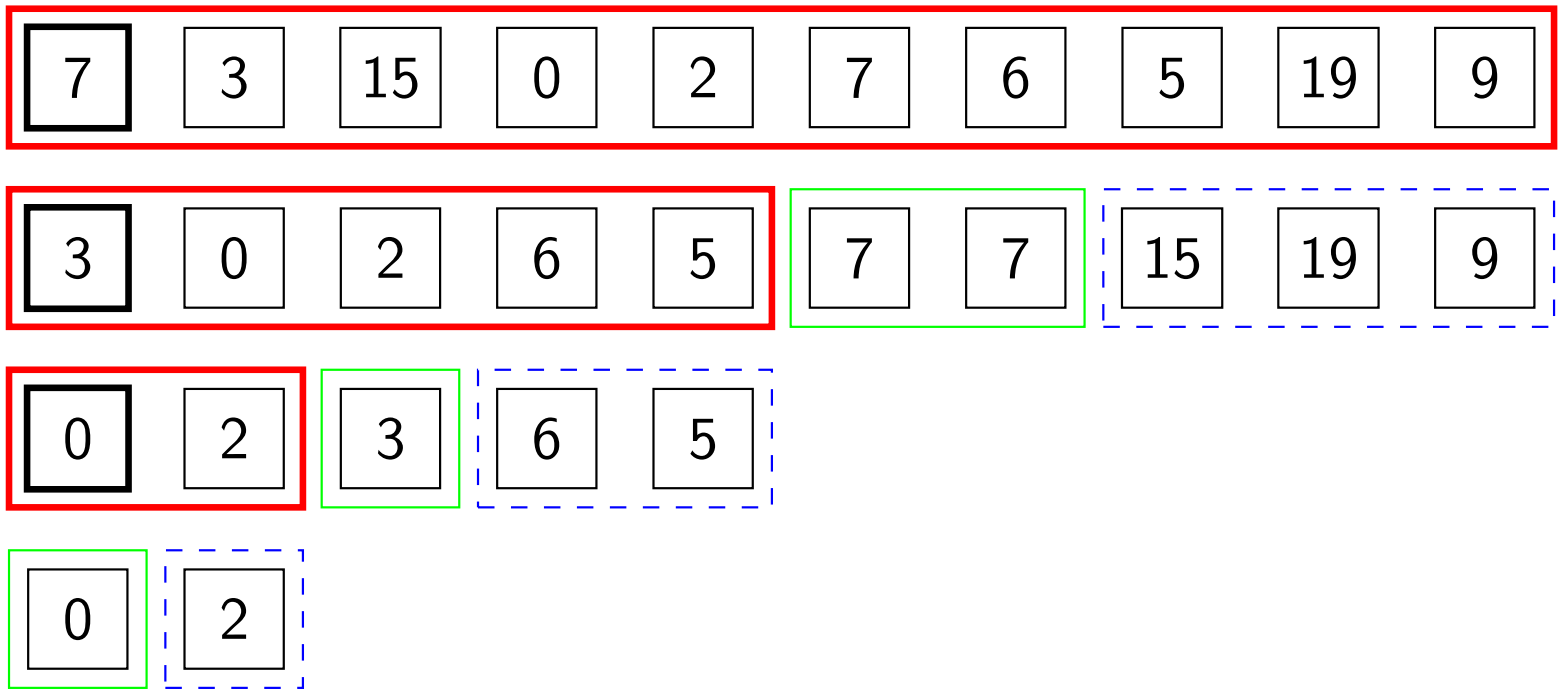
# Quicksort — Exempel



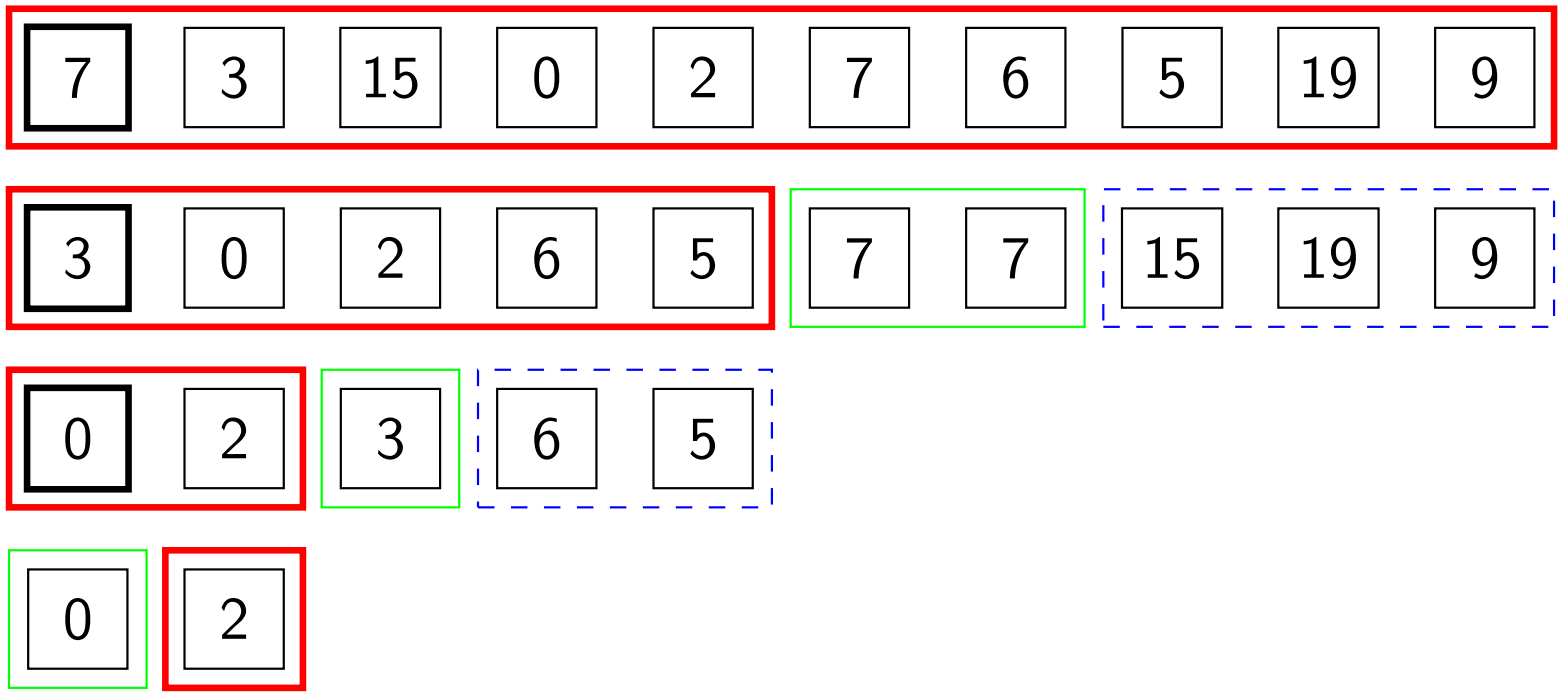
# Quicksort — Exempel



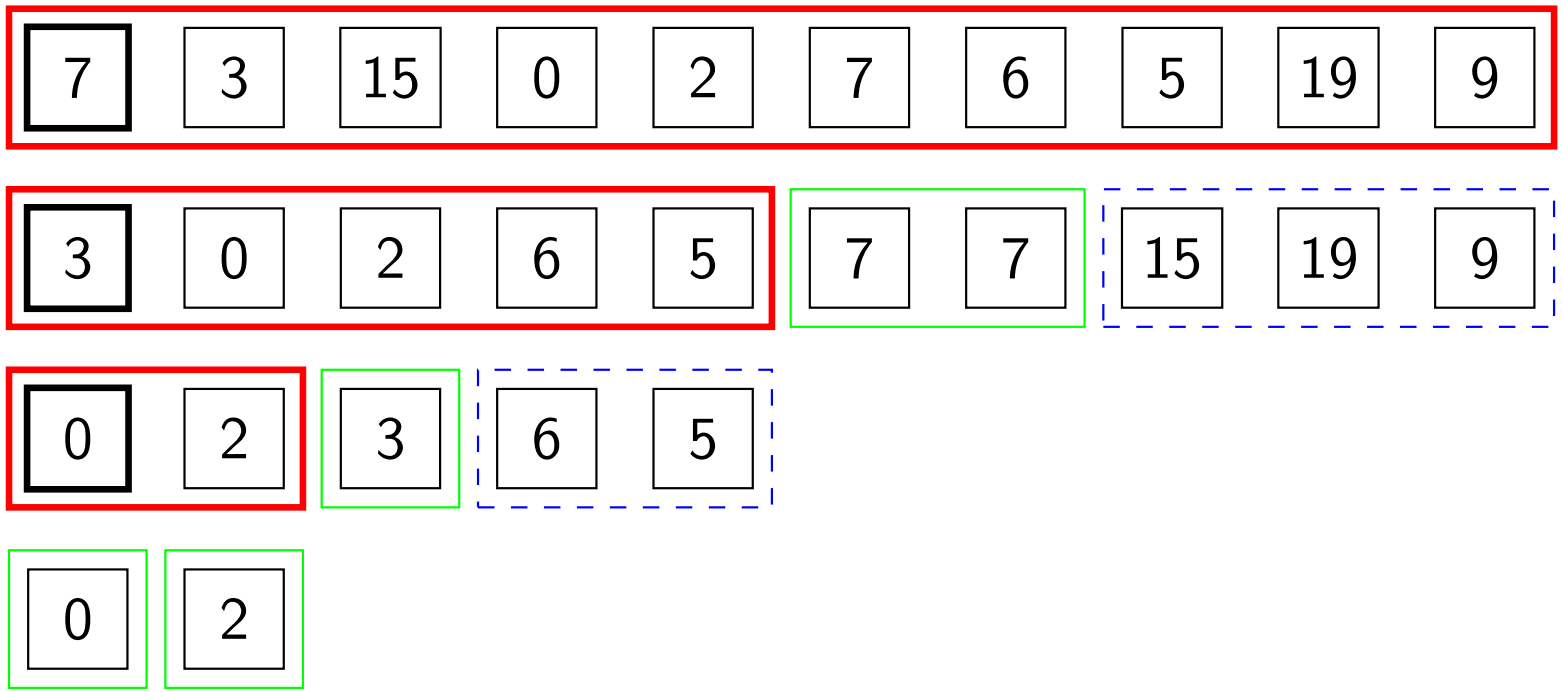
# Quicksort — Exempel



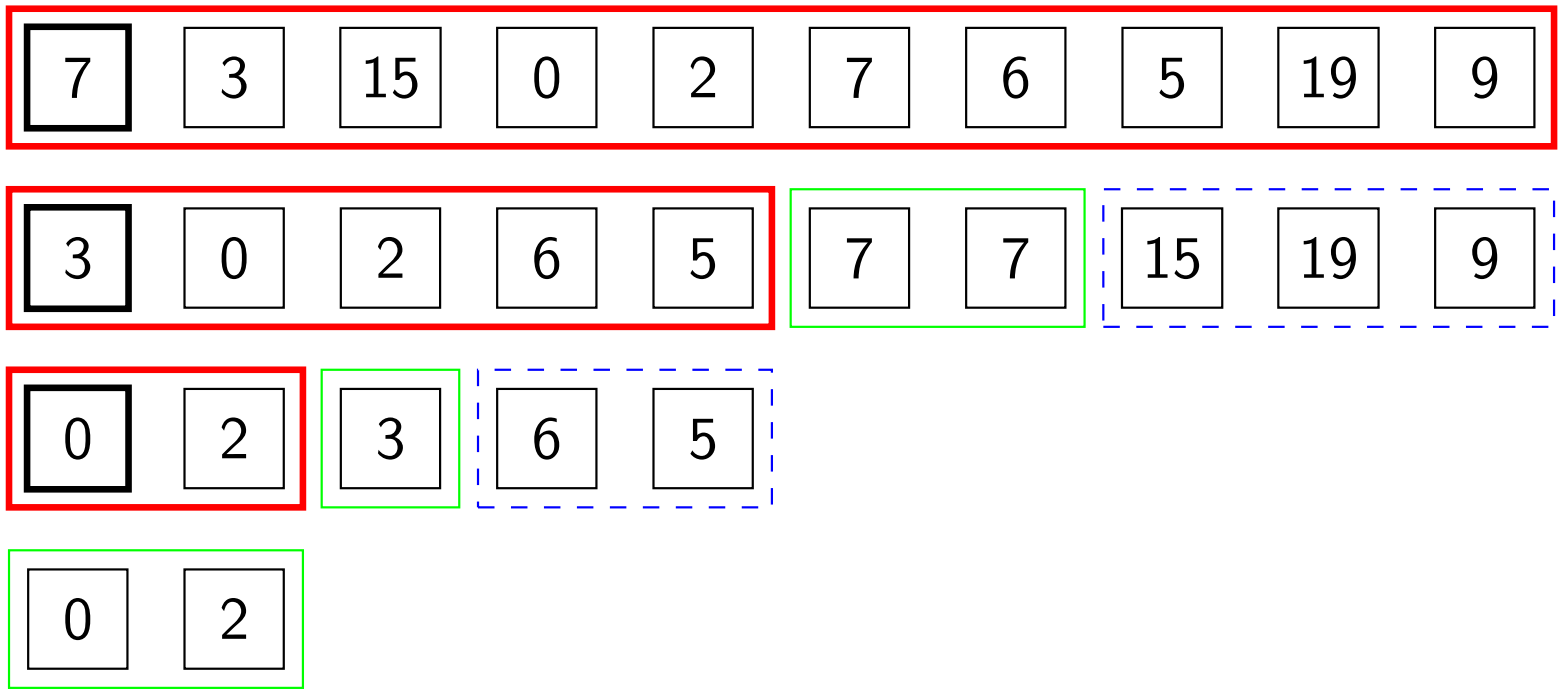
# Quicksort — Exempel



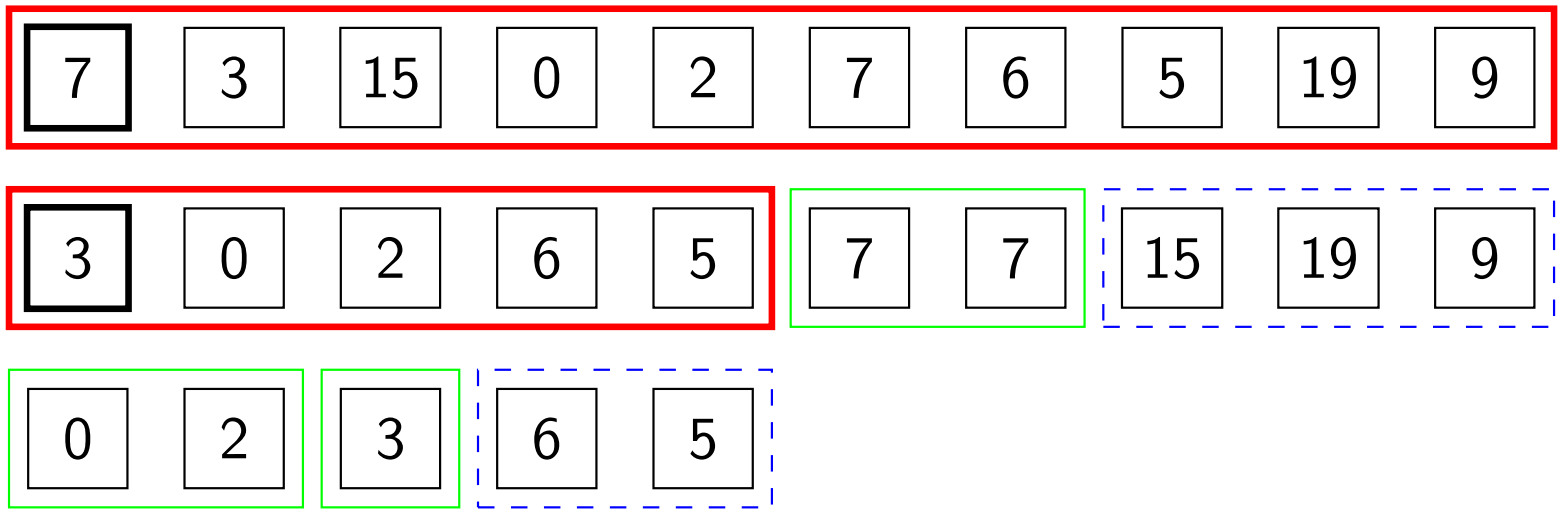
# Quicksort — Exempel



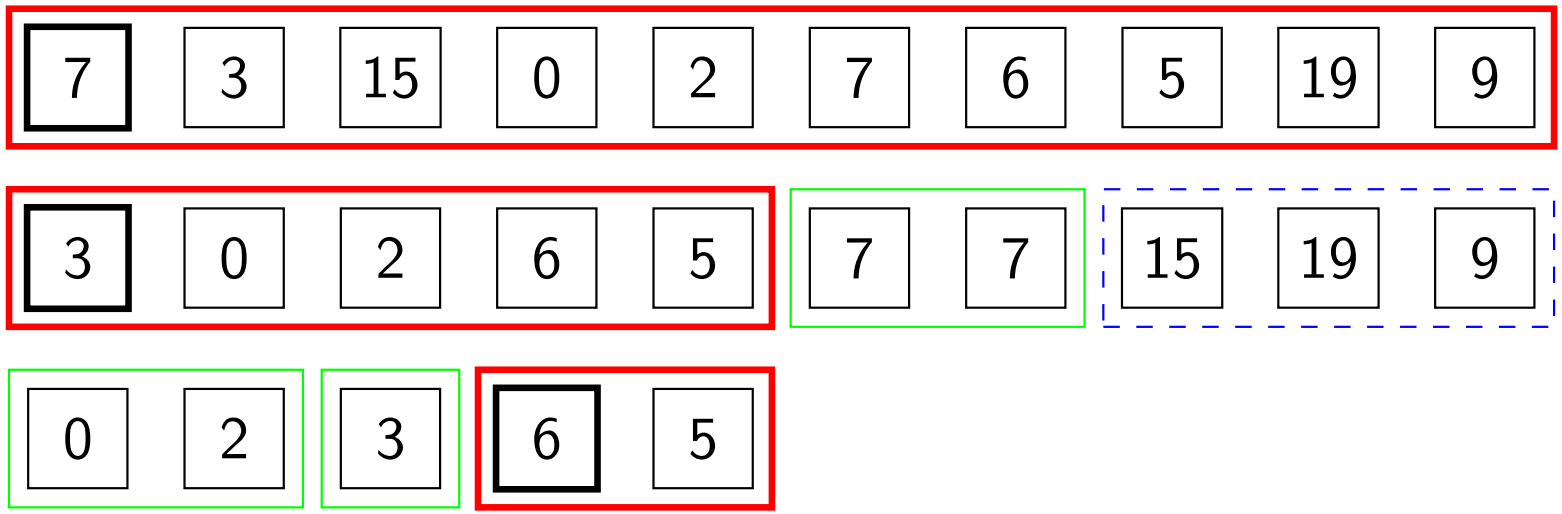
# Quicksort — Exempel



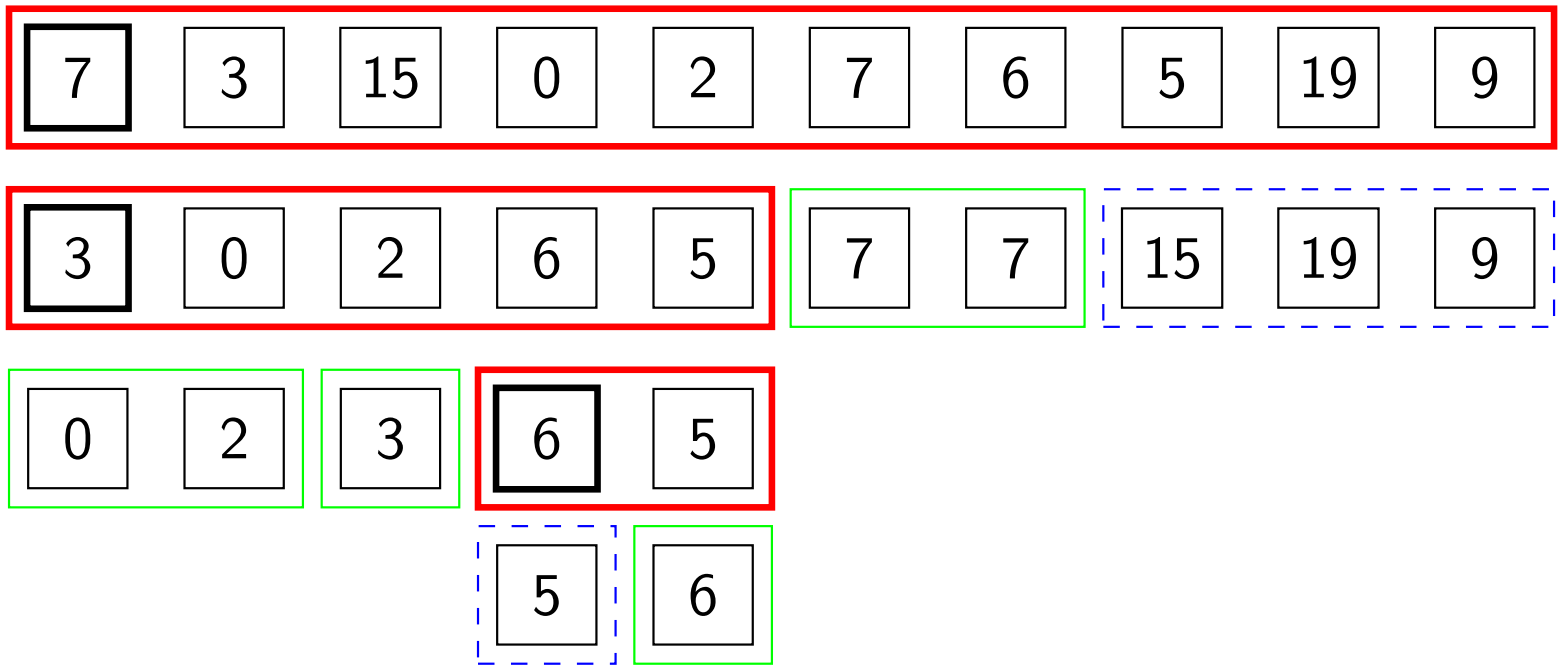
# Quicksort — Exempel



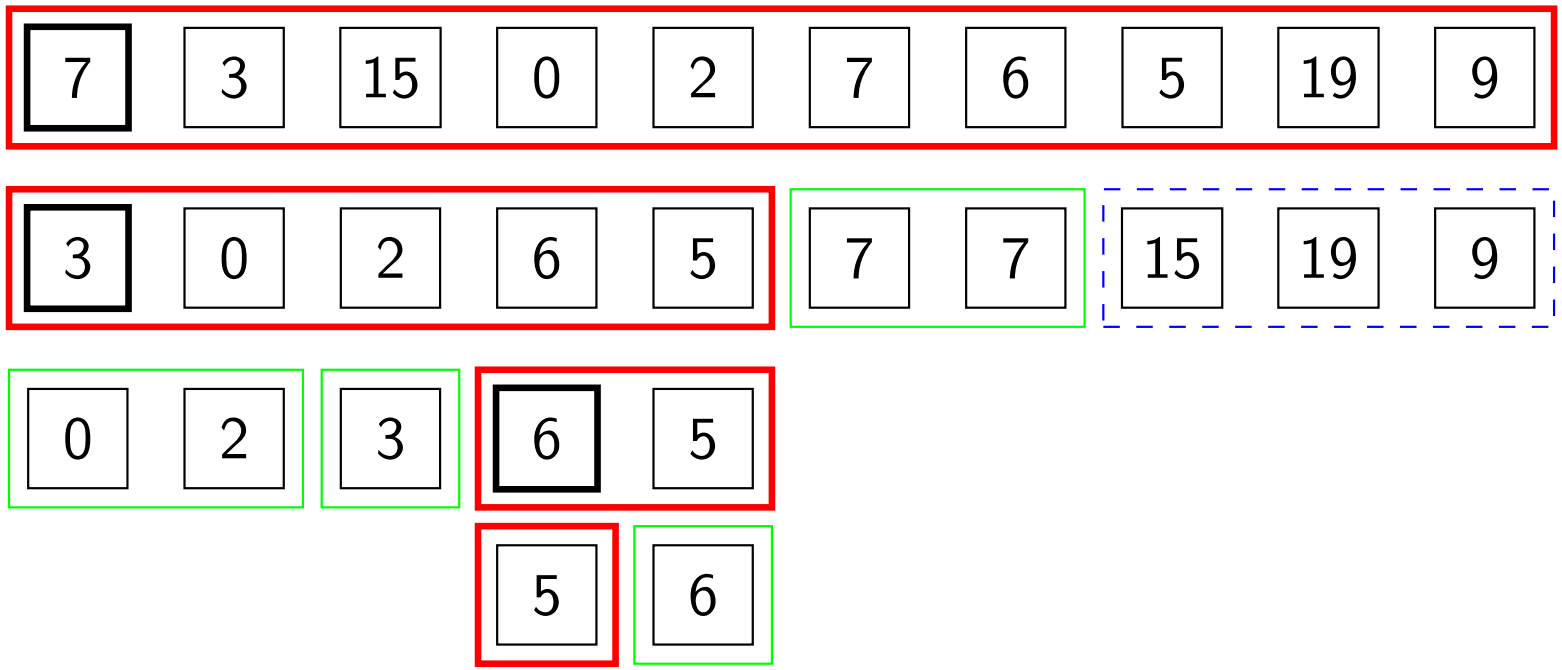
# Quicksort — Exempel



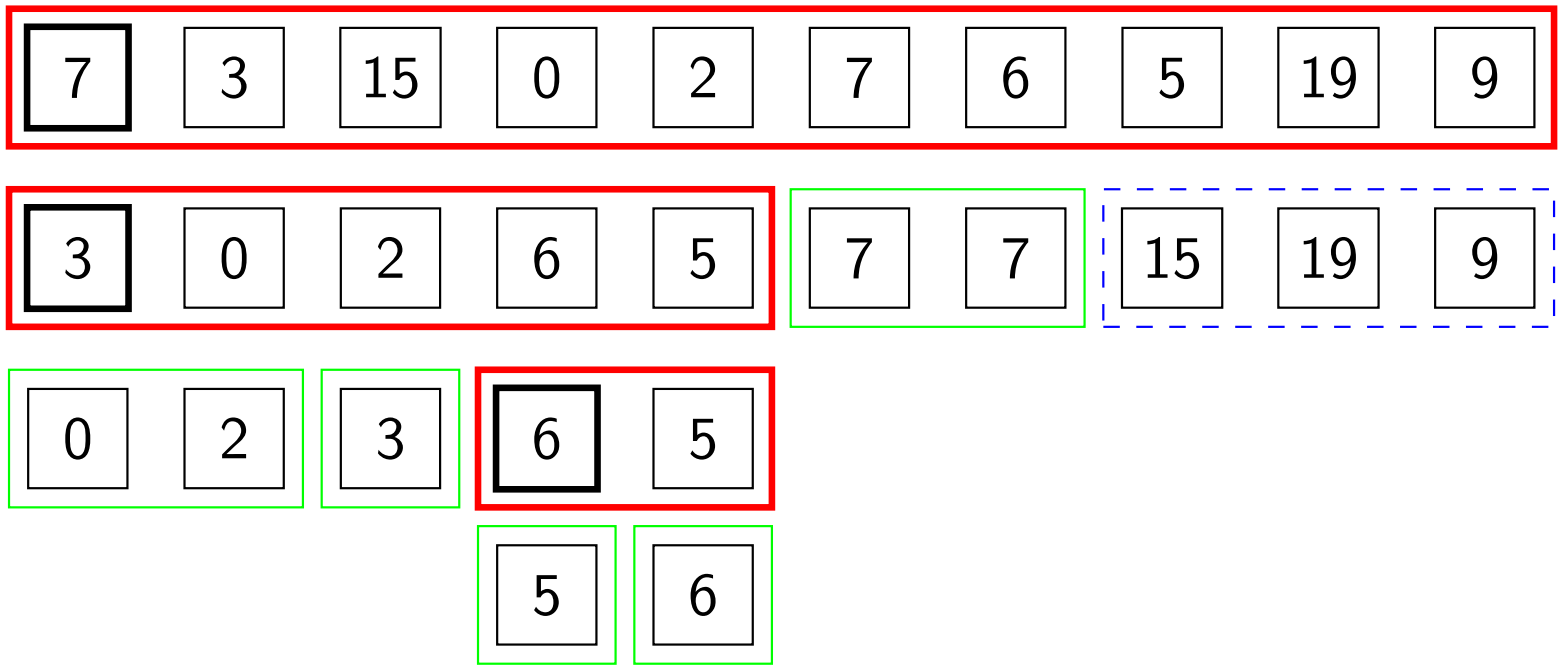
# Quicksort — Exempel



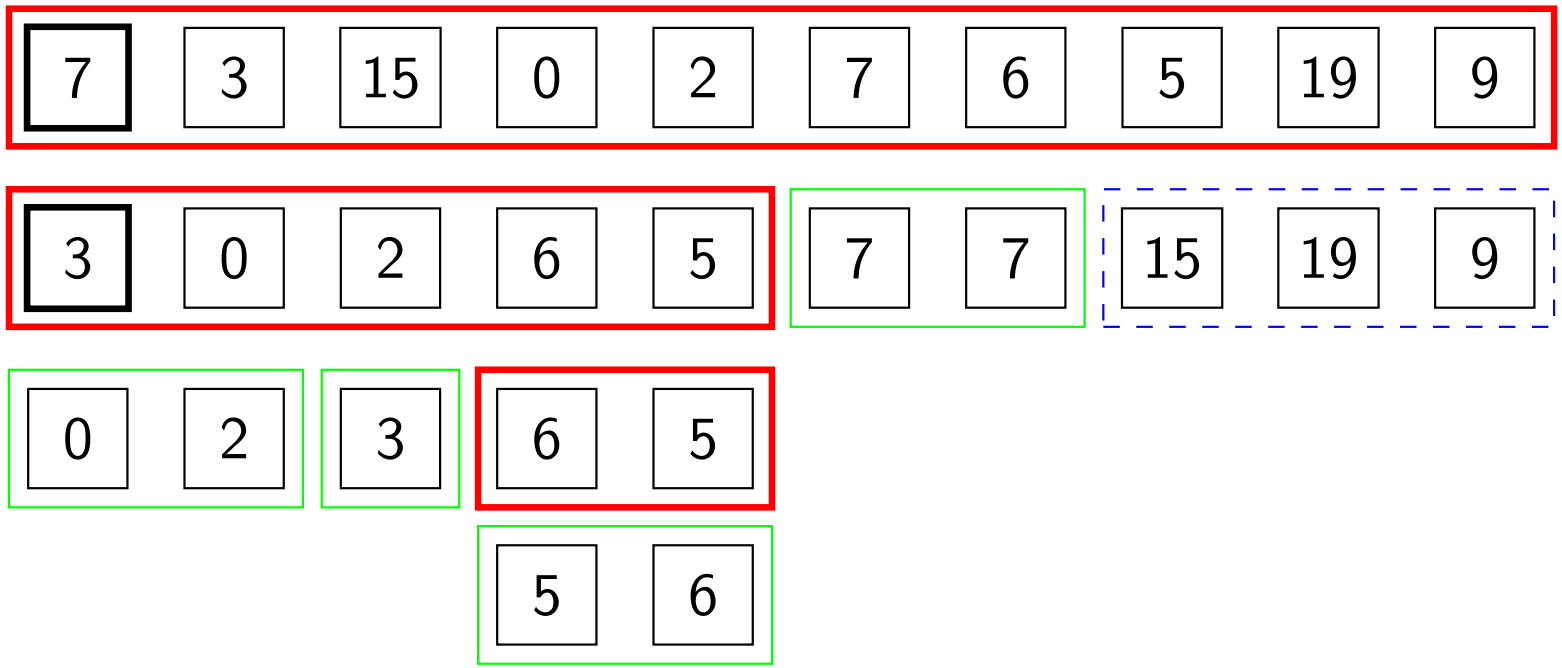
# Quicksort — Exempel



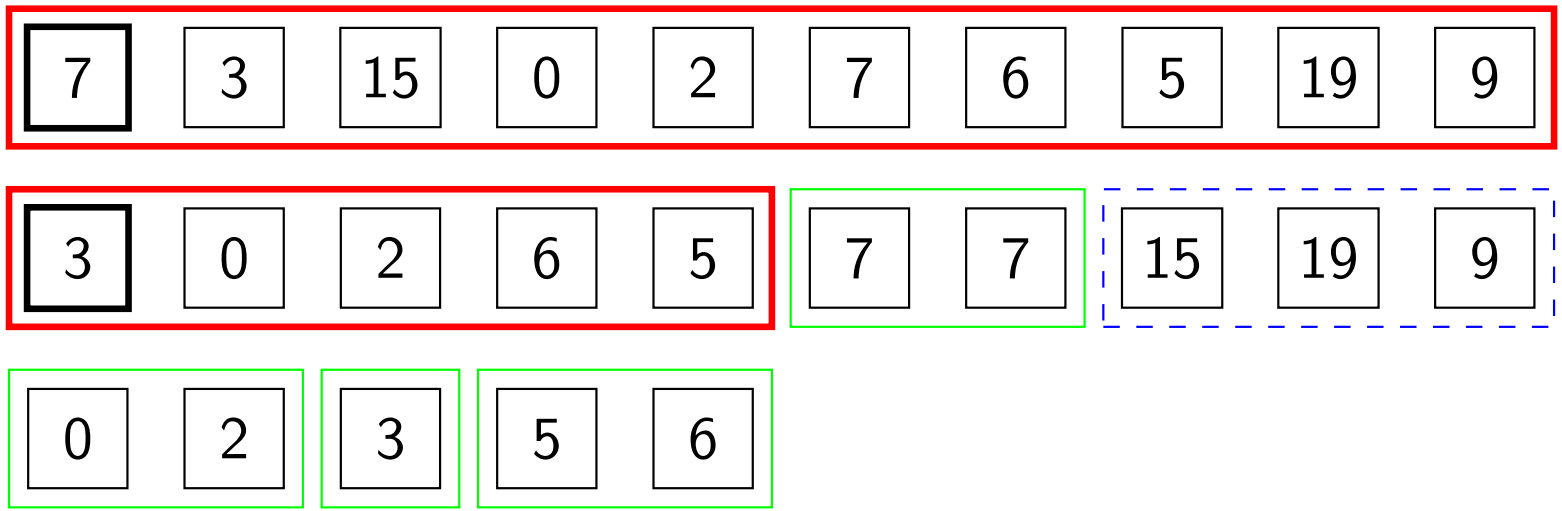
# Quicksort — Exempel



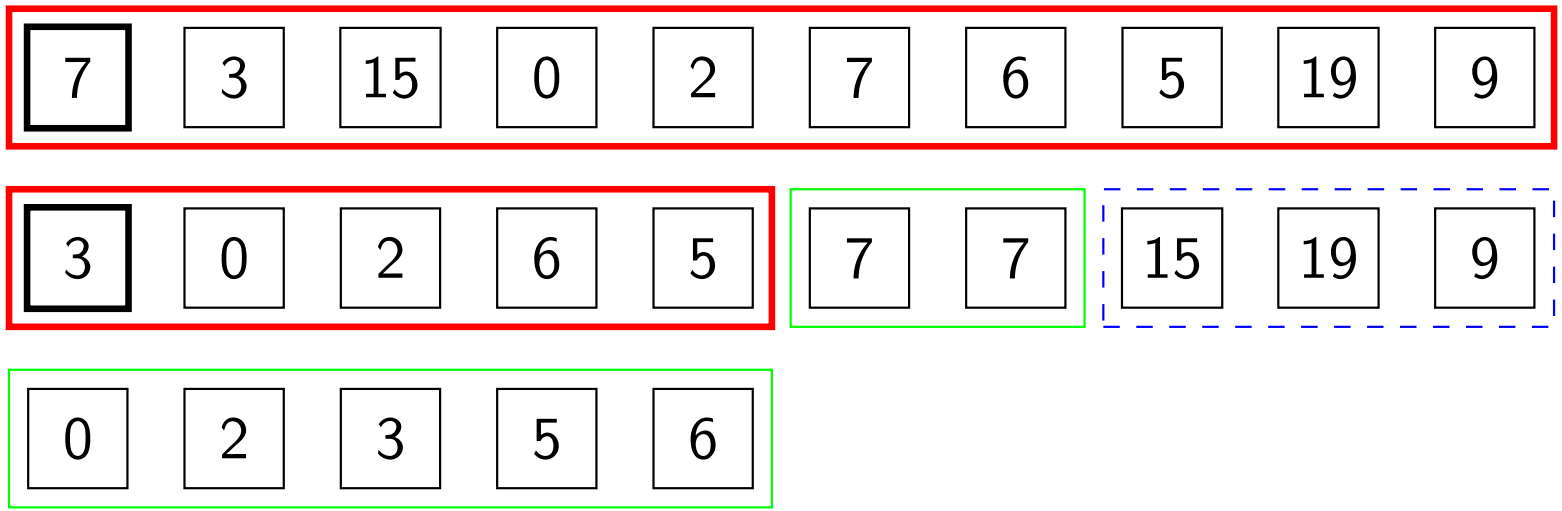
# Quicksort — Exempel



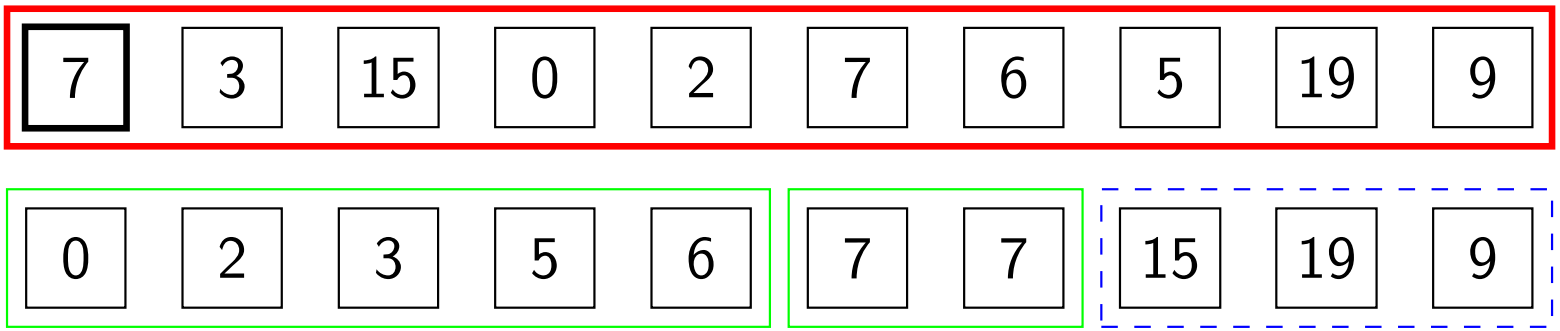
# Quicksort — Exempel



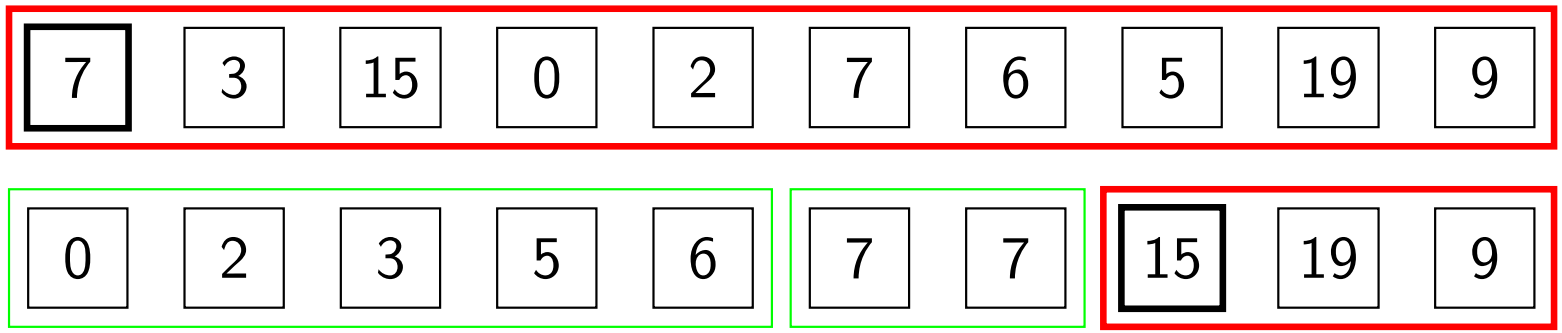
# Quicksort — Exempel



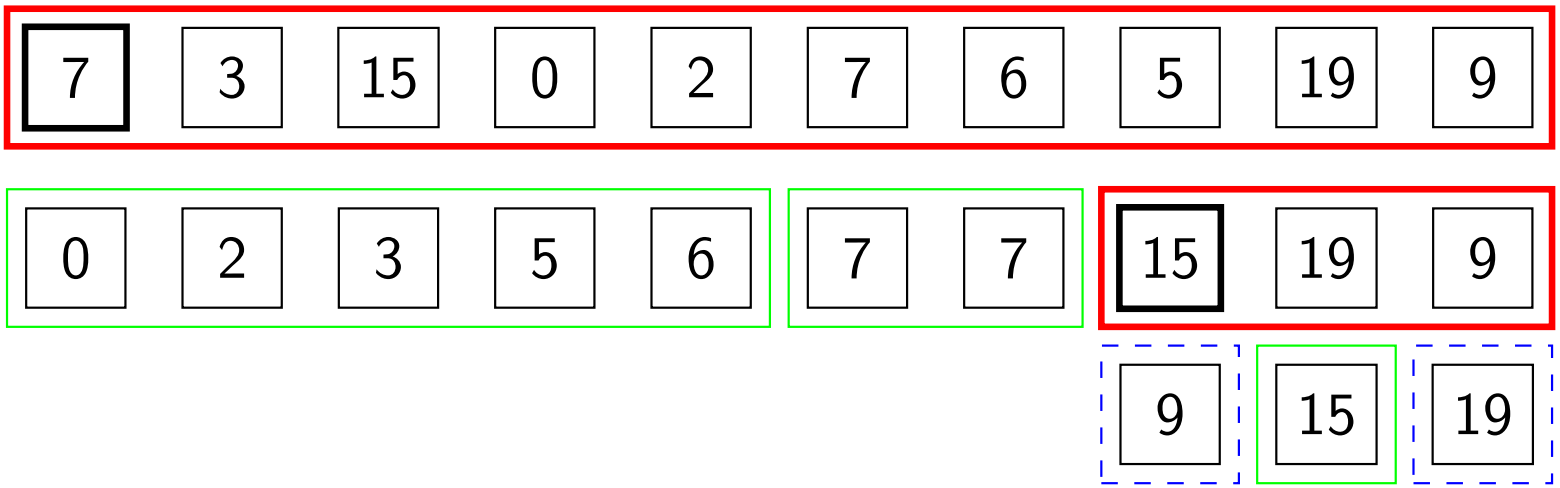
# Quicksort — Exempel



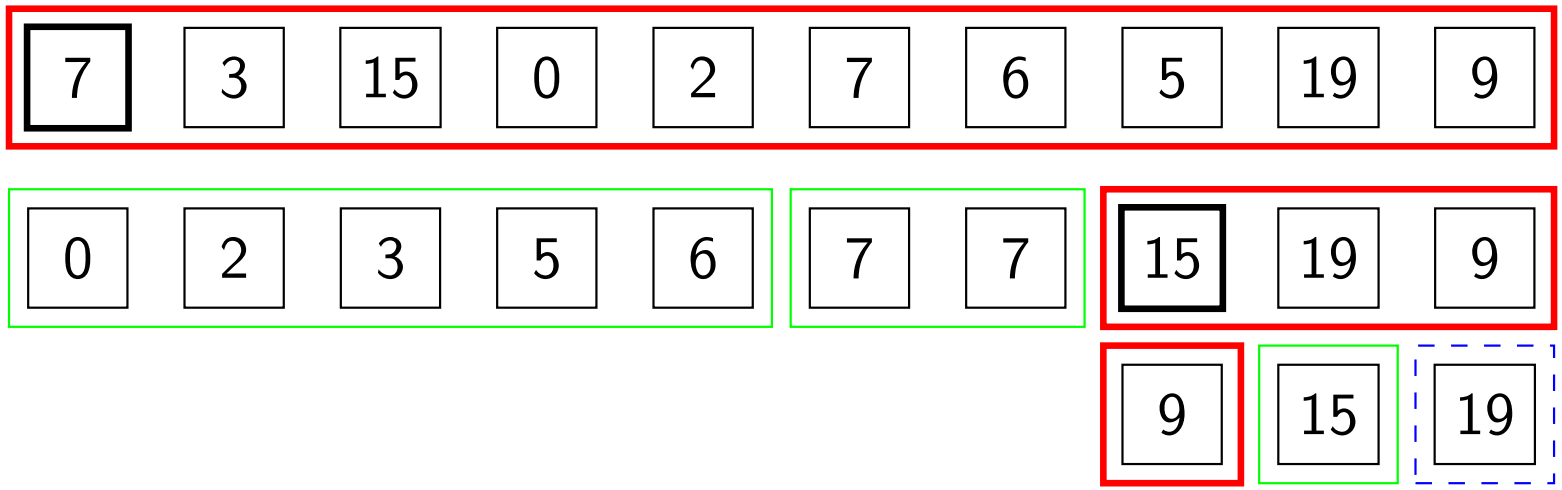
# Quicksort — Exempel



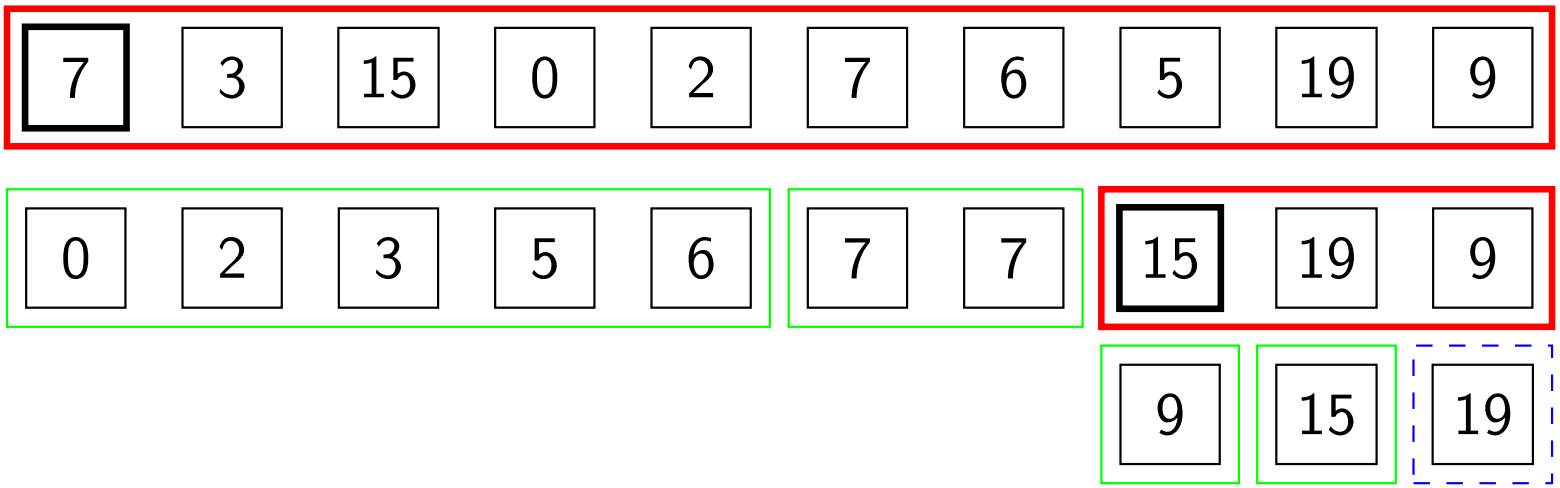
# Quicksort — Exempel



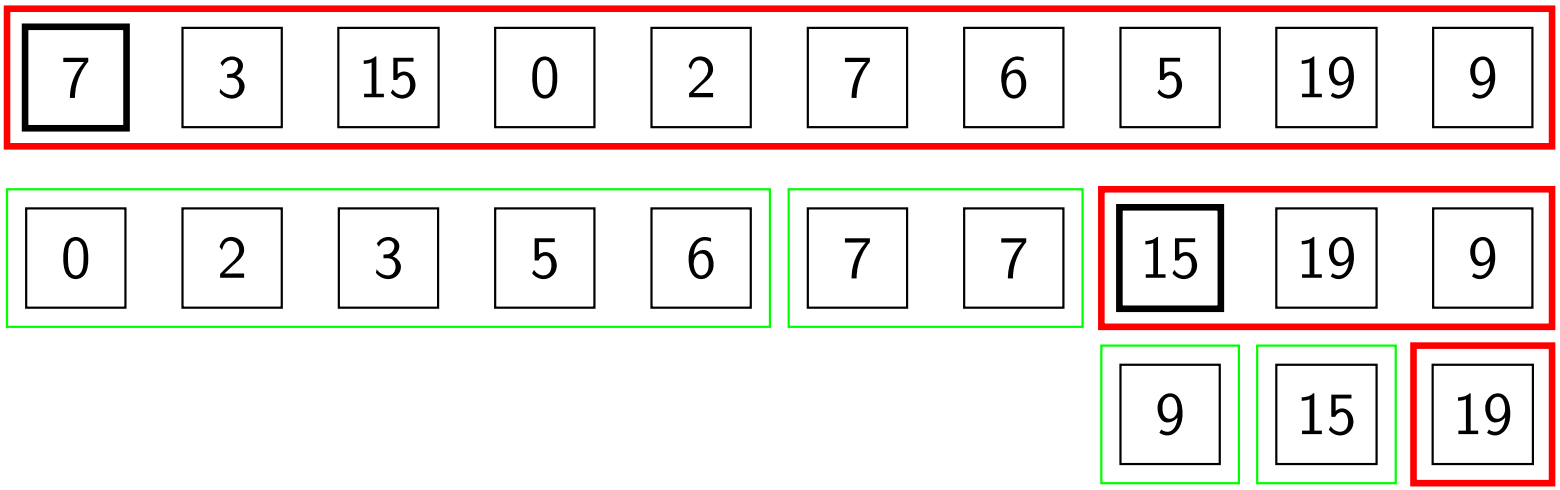
# Quicksort — Exempel



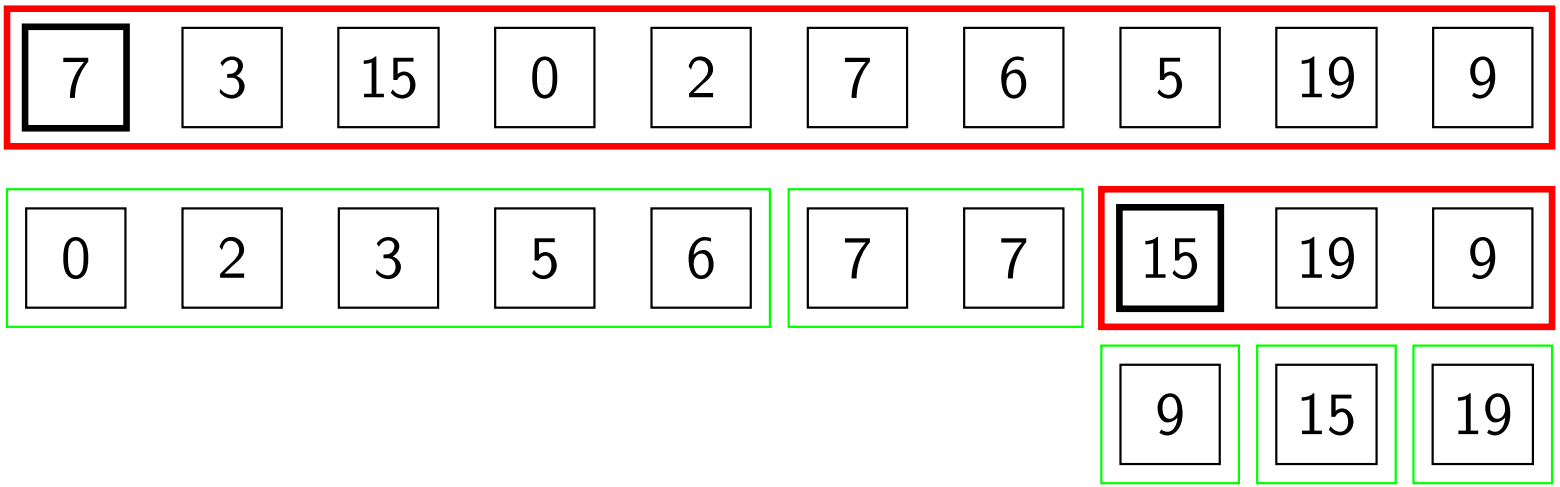
# Quicksort — Exempel



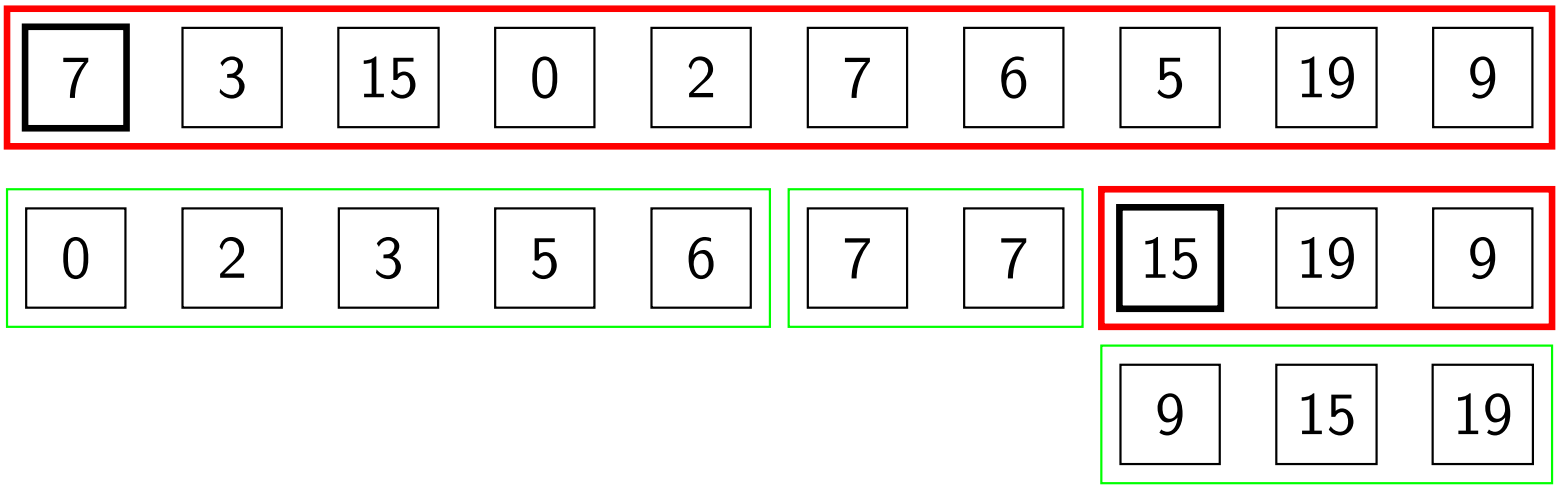
# Quicksort — Exempel



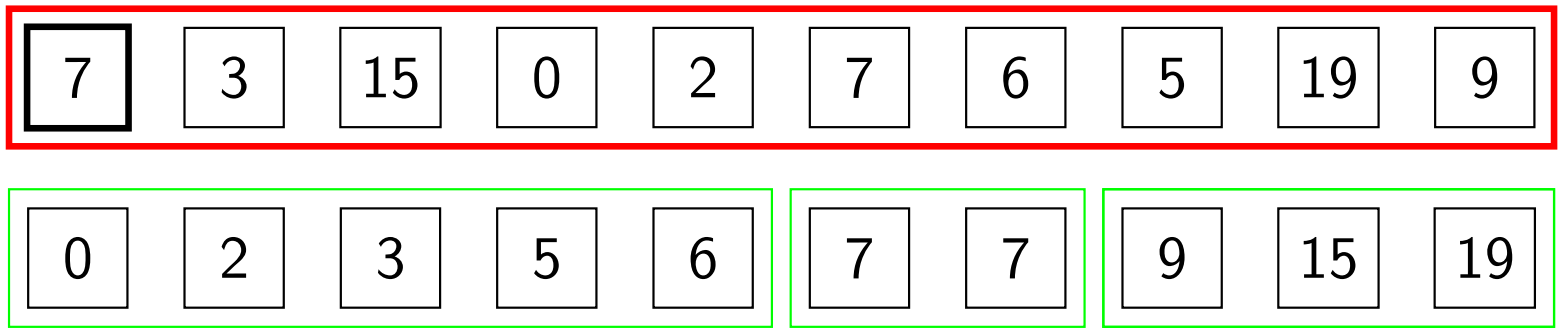
# Quicksort — Exempel



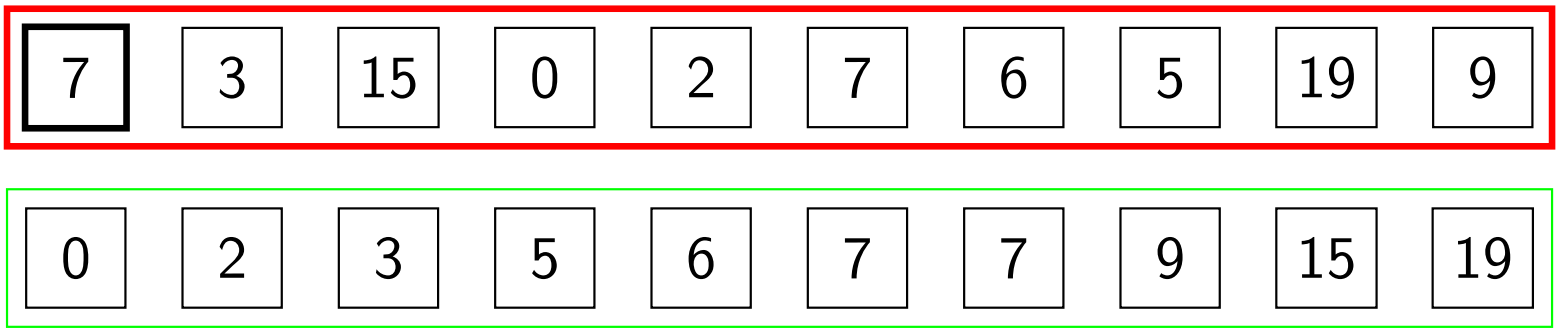
# Quicksort — Exempel



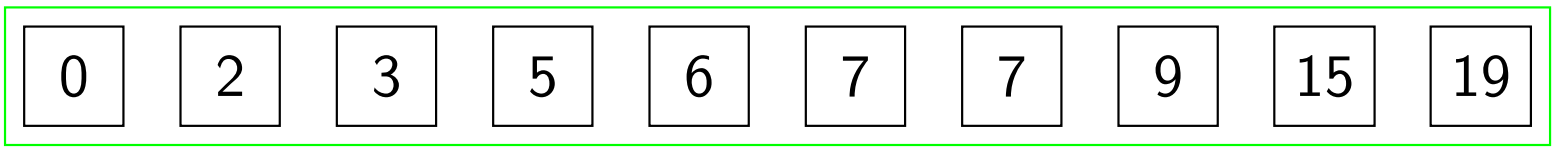
# Quicksort — Exempel



# Quicksort — Exempel



# Quicksort — Exempel



# Quicksort — val av pivåelement

- $O(n \log n)$ i bästa fallet.
- Valet av pivåelement kritiskt:
  - Vill ha ett pivåelement som har ett mitten-värde.
  - Vid sned fördelning får man insticks-/urvalssortering med $O(n^2)$.
- Alternativ (eftersträvar en enkel tilldelning):
  - Välj första/sista, slumpmässigt.
  - Medel/median mellan några stycken.
  - Största av de två första som skiljer sig åt.

# Några algoritmer sammanfattade

| Algoritm | Tidskomplexitet | Stabil? | Minnesbehov |
|---|---|---|---|
| *Insertion Sort* | $O(n^2)$ | Ja | $O(1)$ |
| *Selection Sort* | $O(n^2)$ | Nej[1] | $O(1)$ |
| *Bubble Sort* | $O(n^2)$ | Ja | $O(1)$ |
| *Merge Sort* | $O(n \log n)$ | Ja | $O(n)$ |
| *Quicksort* | $O(n \log n)$–$O(n^2)$ | Nej | $O(\log n)$–$O(n)$ |
| *Heapsort* | $O(n \log n)$ | Nej | $O(1)$ |
| *(Radix Exchange Sort)* | $O(wn)$ | Nej | $O(w)$ |

[1] Ja om $O(n)$ extra minne.