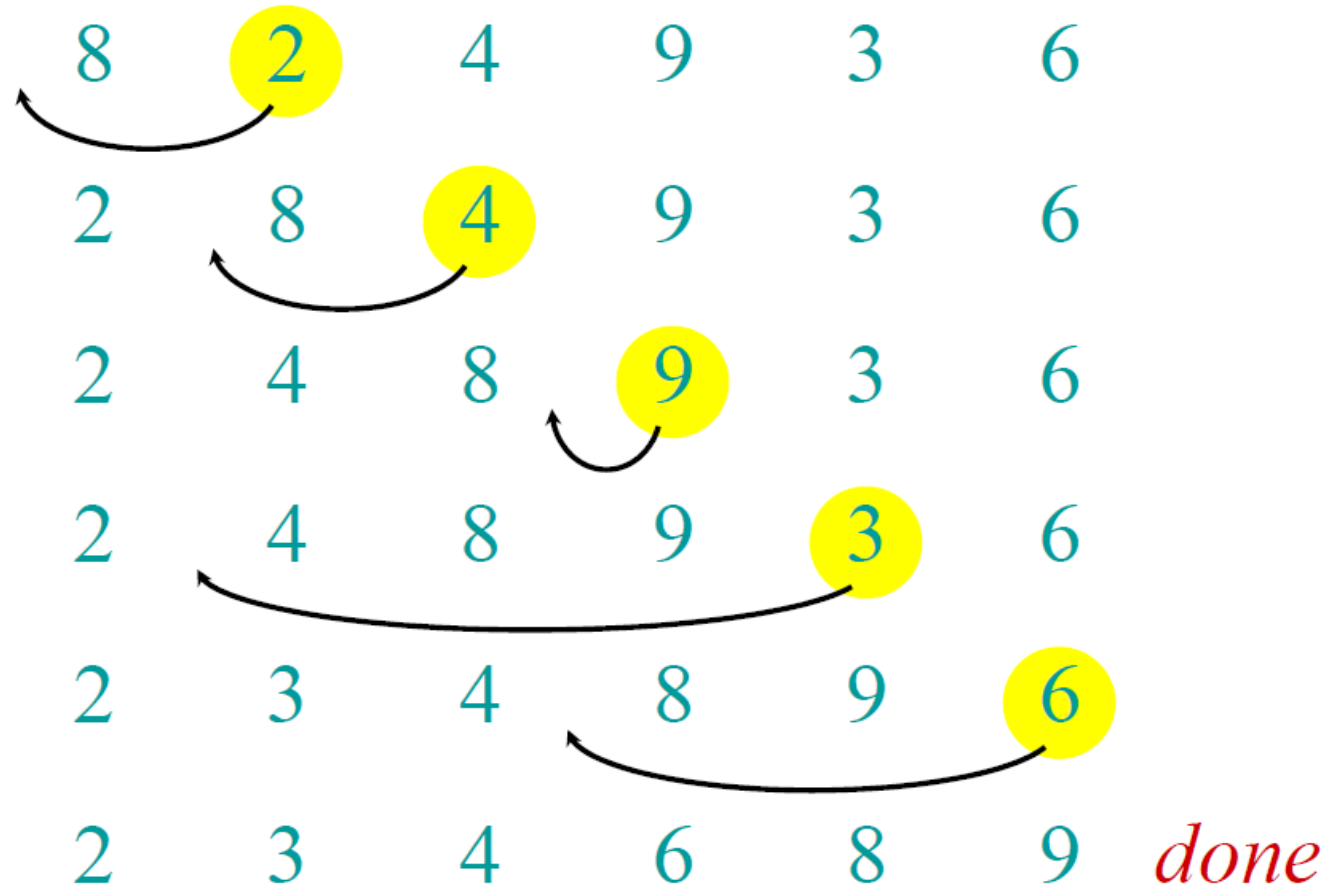


Insertion Sort

Merge Sort

SWE2016-44

Insertion Sort



Insertion Sort

INSERTION-SORT (A, n)	T(n)
<i>for j ← 2 to n</i>	$n-1$
<i>do key ← A[j]</i>	$n-1$
<i>i ← j - 1</i>	$n-1$
<i>while i > 0 and A[i] > key</i>	Best: $\sum_{j=2}^n 1$, Worst: $\sum_{j=2}^n j$, Avg: $\frac{1}{2} \sum_{j=2}^n (j + 1)$
<i>do A[i+1] ← A[i]</i>	Best: $\sum_{j=2}^n (1 - 1)$, Worst: $\sum_{j=2}^n (j - 1)$, Avg: $\frac{1}{2} \sum_{j=2}^n (j - 1)$
<i>i ← i - 1</i>	Best: $\sum_{j=2}^n (1 - 1)$, Worst: $\sum_{j=2}^n (j - 1)$, Avg: $\frac{1}{2} \sum_{j=2}^n (j - 1)$
<i>A[i+1] = key</i>	$n-1$

- Best Case: $T(n) = 4(n - 1) + (n - 1) = 5n - 5$
- Worst Case: $T(n) = 4(n - 1) + \left\{ \frac{n(n-1)}{2} + 2 \left(\frac{n(n-1)}{2} - (n - 1) \right) \right\} = \frac{3}{2}n^2 + \frac{1}{2}n - 2$
- Average Case: $T(n) = 4(n - 1) + \frac{1}{2}(n - 1) + \frac{1}{2} \left\{ \frac{n(n-1)}{2} + 2 \left(\frac{n(n-1)}{2} - (n - 1) \right) \right\} = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2}$

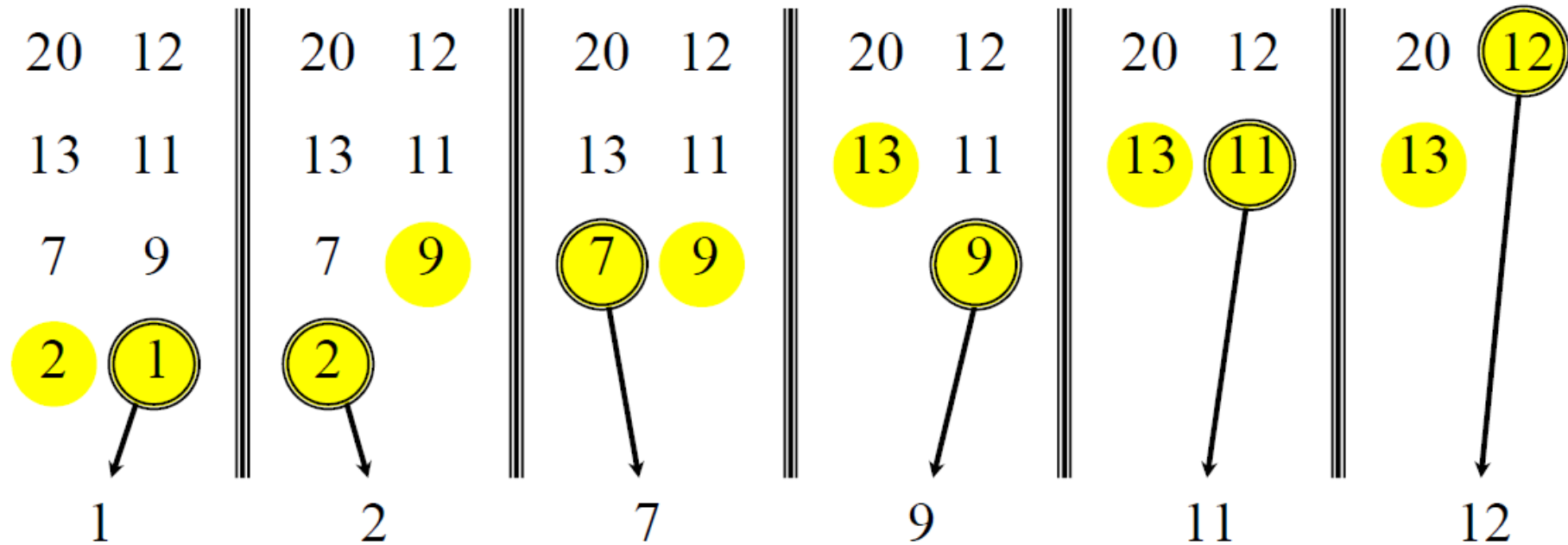
Insertion Sort

- Best Case: $T(n) = 5n - 5$
 - Θ notation: $\Theta(n)$ $\blacktriangleleft 4n \leq T(n) \leq 5n$ for $n \geq 5$
- Worst Case: $T(n) = \frac{3}{2}n^2 + \frac{1}{2}n - 2$
 - Θ notation: $\Theta(n^2)$ $\blacktriangleleft \frac{3}{2}n^2 \leq T(n) \leq 2n^2$ for $n \geq 4$
- Average Case: $T(n) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2}$
 - Θ notation: $\Theta(n^2)$ $\blacktriangleleft \frac{3}{4}n^2 \leq T(n) \leq n^2$ for $n \geq 6$

Insertion Sort

- $T(n) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2}$
 - $T(n) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2} \leq \frac{3}{4}n^2 + \frac{11}{4}n \leq \frac{3}{4}n^2 + \frac{11}{4}n^2 = \frac{7}{2}n^2$
 - $T(n) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2} \geq \frac{3}{4}n^2 - \frac{7}{2} = \frac{3}{8}n^2 + \frac{3}{8}n^2 - \frac{7}{2} \geq \frac{3}{8}n^2$ for all $n \geq 4$
- ➔ $\frac{3}{8}n^2 \leq T(n) \leq \frac{7}{2}n^2$ for $n \geq 4$
- ➔ $T(n) = \Theta(n^2)$

Merge two sorted arrays



Merge two sorted arrays

* Best Case Assumption: $A[i] < B[j]$ for all i, j

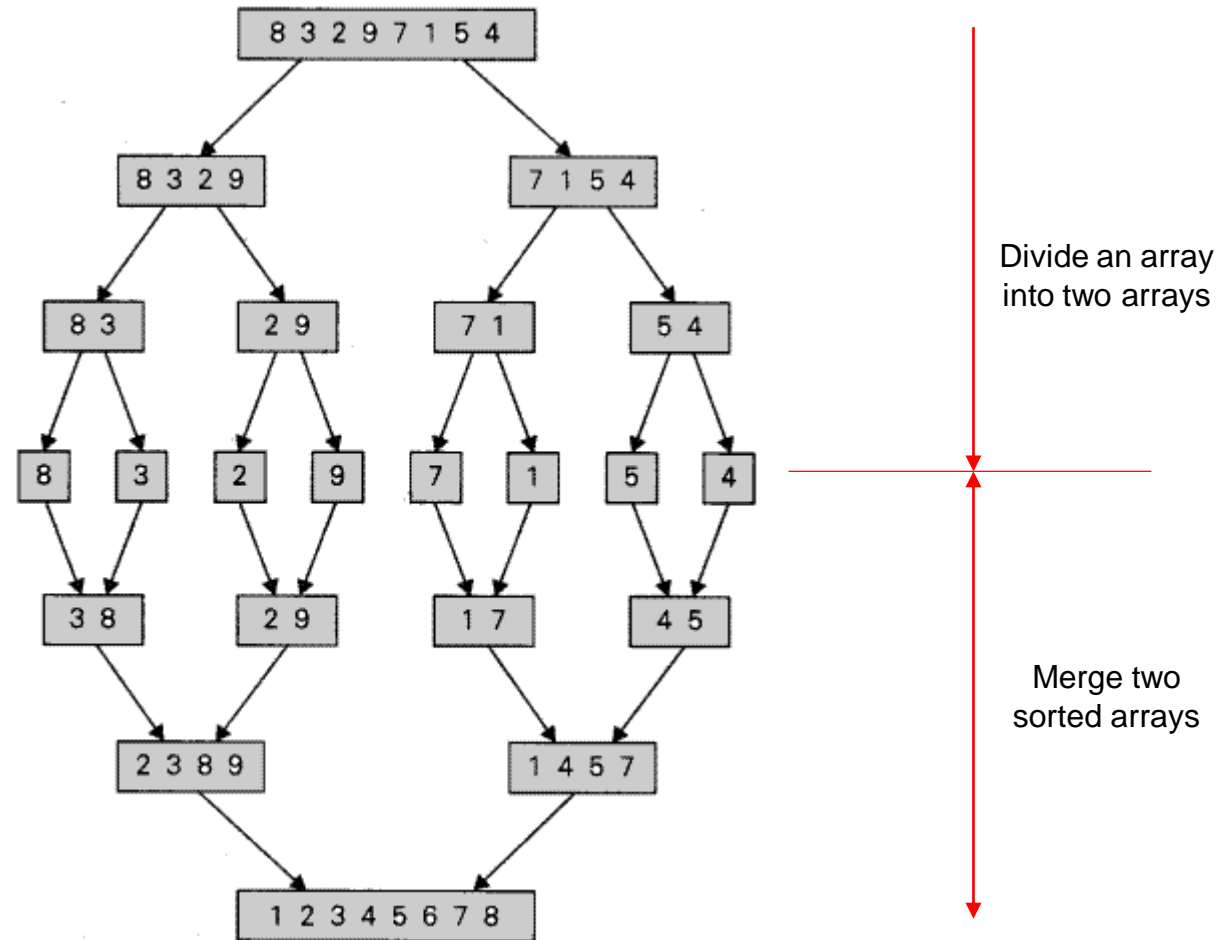
MERGE (A, B, m, n)	$T(m, n)$
$i = 1, j = 1$	1
<i>while</i> $i < m + 1$ <i>and</i> $j < n + 1$	Best: $m + 1$, Worst: $m + n$, Avg: $\frac{1}{2}(m + 1) + \frac{1}{2}(m + n)$
<i>if</i> $A[i] < B[j]$	Best: $3m$, Worst: $3(m + n - 1)$, Avg: $\frac{3}{2}m + \frac{3}{2}(m + n - 1)$
<i>do</i> $Z[i + j - 1] = A[i]$	
$i = i + 1$	
<i>else</i>	
<i>do</i> $Z[i + j - 1] = B[j]$	
$j = j + 1$	Best: n , Worst: 1 , Avg: $\frac{1}{2}n + \frac{1}{2}$
<i>for</i> $k = i$ <i>to</i> m	
$Z[k + n] = A[k]$	
<i>for</i> $k = j$ <i>to</i> n	
$Z[m + k] = B[k]$	

- Best Case: $T(n) = 4m + n + 2$
- Worst Case: $T(n) = 4m + 4n - 1$
- Average Case: $T(n) = 4m + \frac{5}{2}n + \frac{1}{2}$

Merge two sorted arrays

- Best Case: $T(n) = 4m + n + 2$
 - Θ notation: $\Theta(m + n)$ $\blacktriangleleft m + n \leq T(n) \leq 4(m + n)$ for $m \geq 1, n \geq 1$
- Worst Case: $T(n) = 4m + 4n - 1$
 - Θ notation: $\Theta(m + n)$ $\blacktriangleleft m + n \leq T(n) \leq 4(m + n)$ for $m \geq 1, n \geq 1$
- Average Case: $T(n) = 4m + \frac{5}{2}n + \frac{1}{2}$
 - Θ notation: $\Theta(m + n)$ $\blacktriangleleft 2(m + n) \leq T(n) \leq 4(m + n)$ for $m \geq 1, n \geq 1$

Merge Sort



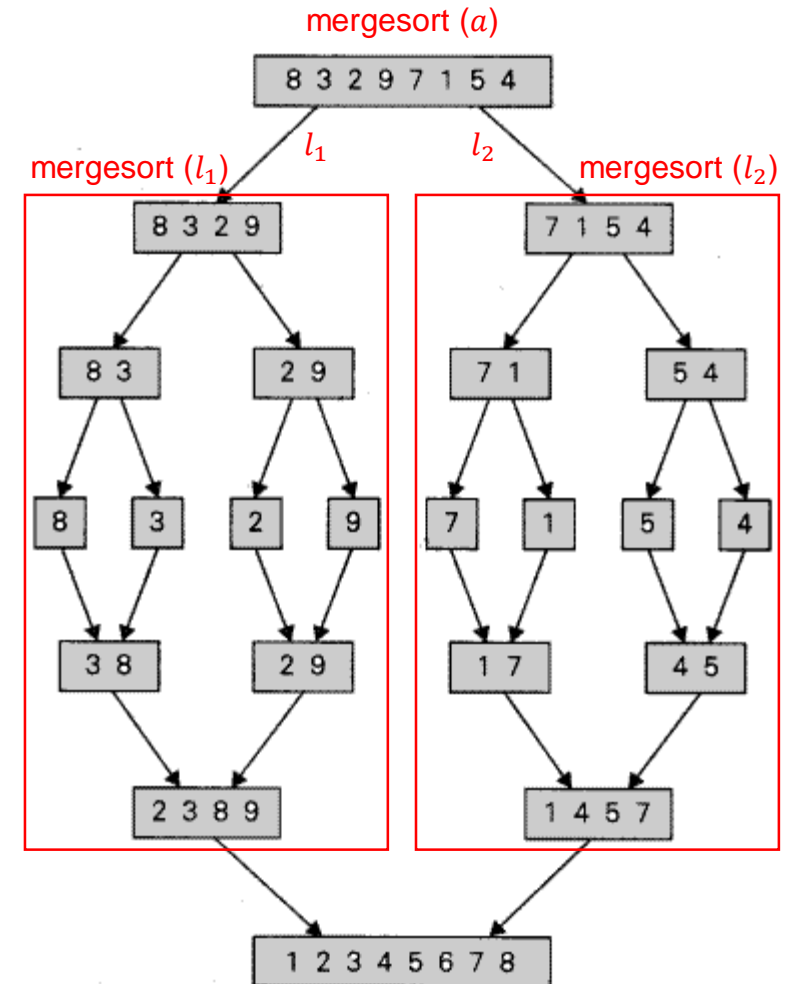
Merge Sort

```
func mergesort( var a as array )  
    if ( n == 1 ) return a  
  
    var l1 as array = a[0] ... a[n/2]  
    var l2 as array = a[n/2+1] ... a[n]  
  
    l1 = mergesort( l1 )  
    l2 = mergesort( l2 )  
  
    return merge( l1, l2 )  
end func
```

```
func merge( var a as array, var b as array )  
    var c as array  
  
    while ( a and b have elements )  
        if ( a[0] > b[0] )  
            add b[0] to the end of c  
            remove b[0] from b  
        else  
            add a[0] to the end of c  
            remove a[0] from a  
        while ( a has elements )  
            add a[0] to the end of c  
            remove a[0] from a  
        while ( b has elements )  
            add b[0] to the end of c  
            remove b[0] from b  
    return c  
end func
```

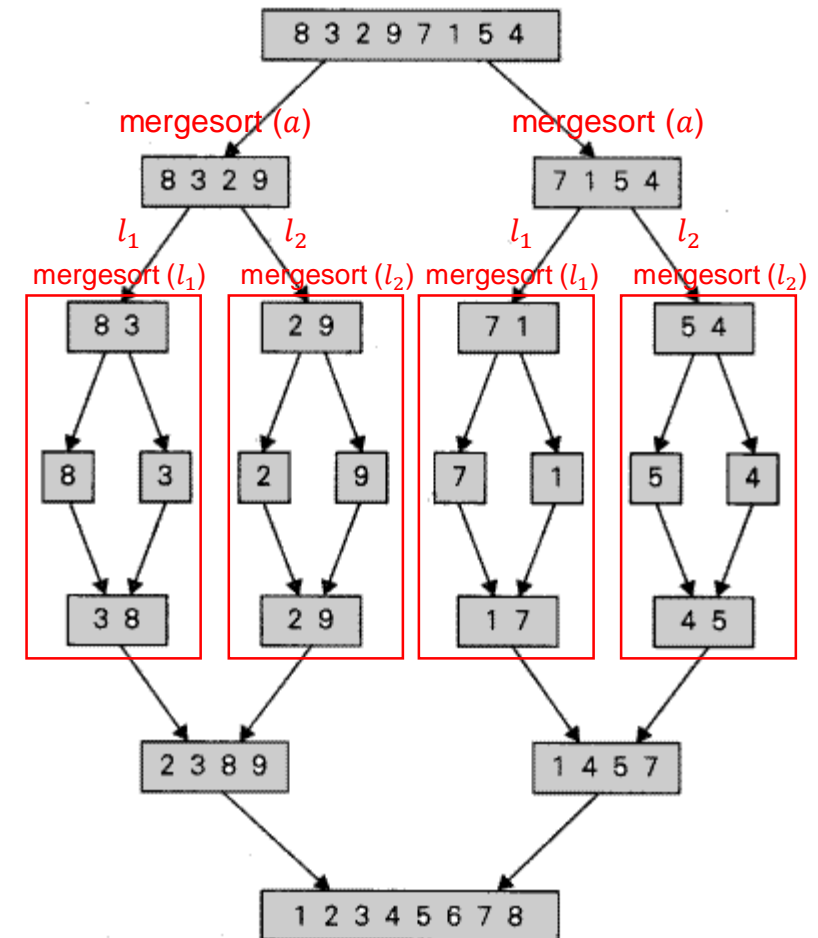
Merge Sort

```
func mergesort( var a as array )  
  if ( n == 1 ) return a  
  
  var l1 as array = a[0] ... a[n/2]  
  var l2 as array = a[n/2+1] ... a[n]  
  
  l1 = mergesort( l1 )  
  l2 = mergesort( l2 )  
  
  return merge( l1, l2 )  
end func
```



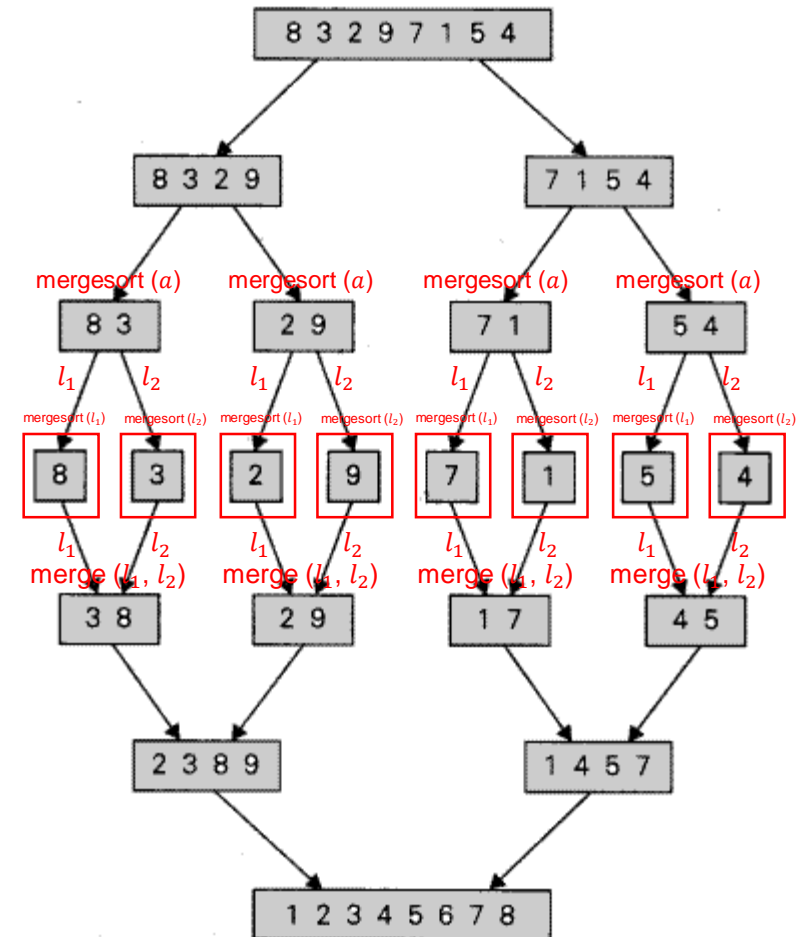
Merge Sort

```
func mergesort( var a as array )  
  if ( n == 1 ) return a  
  
  var l1 as array = a[0] ... a[n/2]  
  var l2 as array = a[n/2+1] ... a[n]  
  
  l1 = mergesort( l1 )  
  l2 = mergesort( l2 )  
  
  return merge( l1, l2 )  
end func
```



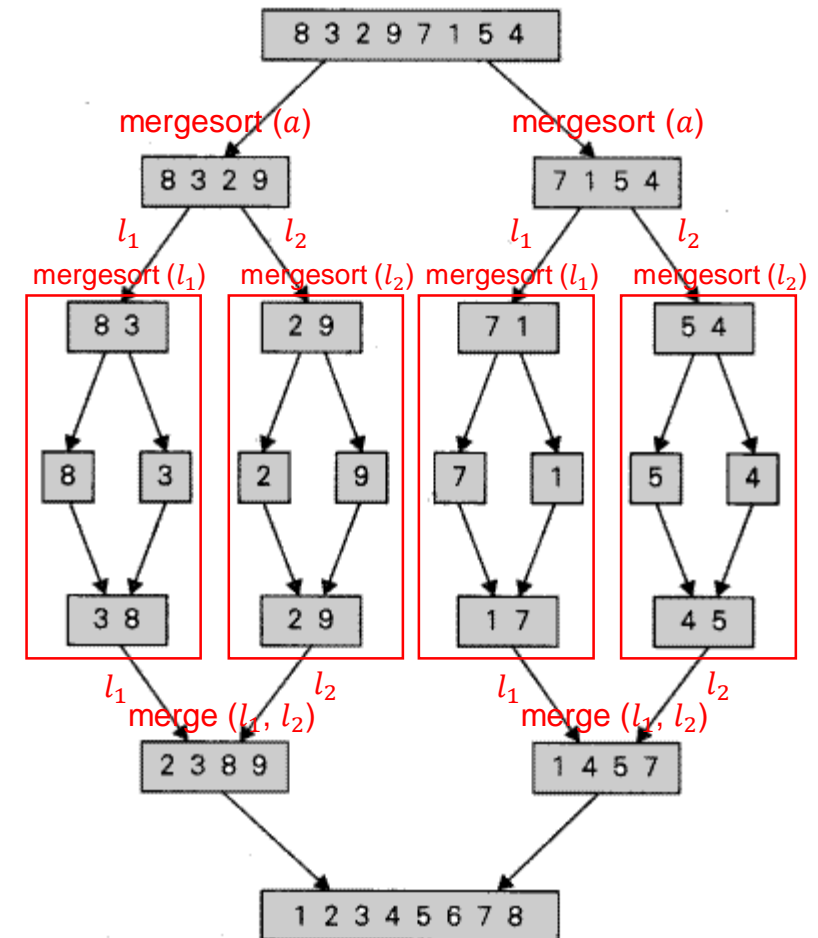
Merge Sort

```
func mergesort( var a as array )  
  if ( n == 1 ) return a  
  
  var l1 as array = a[0] ... a[n/2]  
  var l2 as array = a[n/2+1] ... a[n]  
  
  l1 = mergesort( l1 )  
  l2 = mergesort( l2 )  
  
  return merge( l1, l2 )  
end func
```



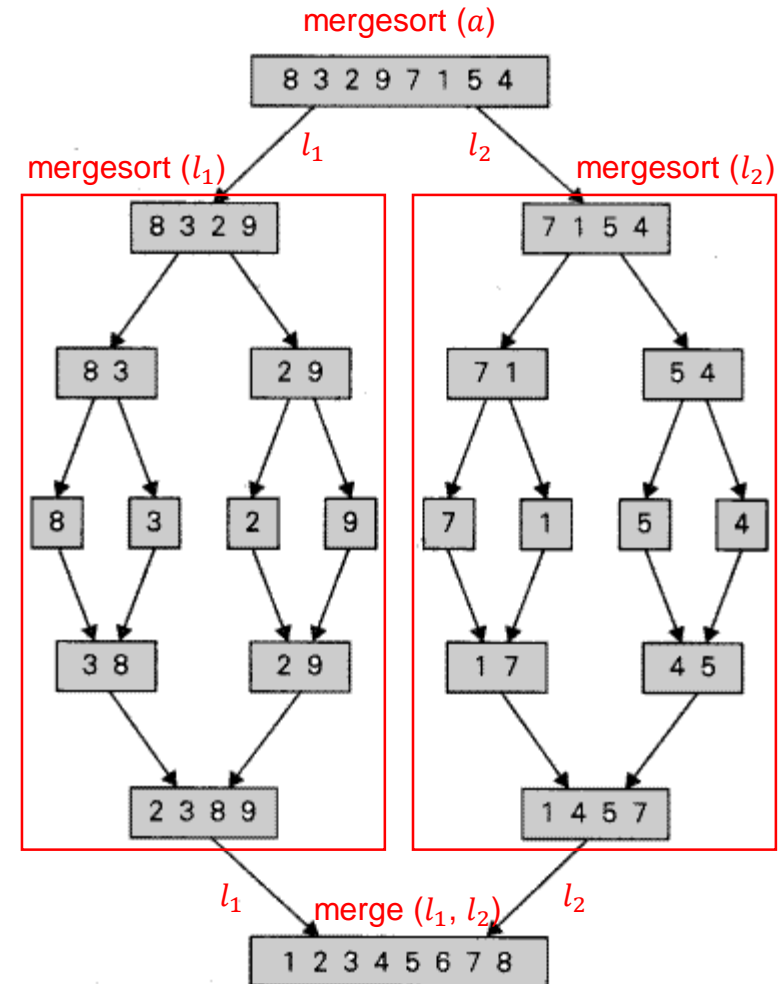
Merge Sort

```
func mergesort( var a as array )  
  if ( n == 1 ) return a  
  
  var l1 as array = a[0] ... a[n/2]  
  var l2 as array = a[n/2+1] ... a[n]  
  
  l1 = mergesort( l1 )  
  l2 = mergesort( l2 )  
  
  return merge( l1, l2 )  
end func
```



Merge Sort

```
func mergesort( var a as array )  
  if ( n == 1 ) return a  
  
  var l1 as array = a[0] ... a[n/2]  
  var l2 as array = a[n/2+1] ... a[n]  
  
  l1 = mergesort( l1 )  
  l2 = mergesort( l2 )  
  
  return merge( l1, l2 )  
end func
```



Merge Sort

```
func mergesort( var a as array )  
  if ( n == 1 ) return a
```

$n > 1$

$T(n)$

$n = 1$

$T(n)$

$\Theta(1)$

```
  var l1 as array = a[0] ... a[n/2]  
  var l2 as array = a[n/2+1] ... a[n]
```

```
  l1 = mergesort( l1 )
```

$T\left(\frac{n}{2}\right)$

```
  l2 = mergesort( l2 )
```

$T\left(\frac{n}{2}\right)$

```
  return merge( l1, l2 )  
end func
```

$\Theta(n)$

$$\Rightarrow T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases}$$

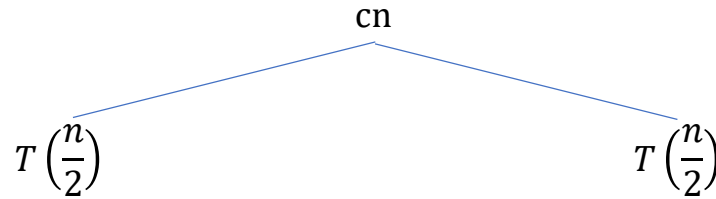
Merge Sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } \mathbf{n = 1} \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } \mathbf{n > 1} \end{cases} \quad \Theta(n) \approx cn$$

$$T(n)$$

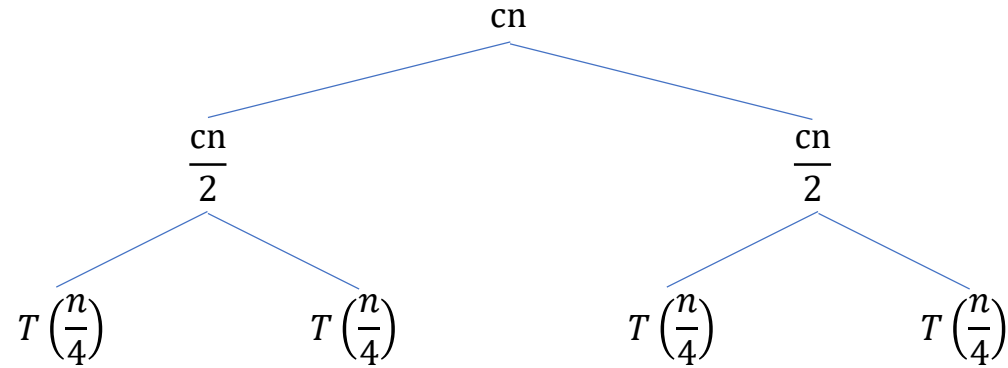
Merge Sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } \mathbf{n} = \mathbf{1} \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } \mathbf{n} > \mathbf{1} \end{cases} \quad \Theta(n) \approx cn$$



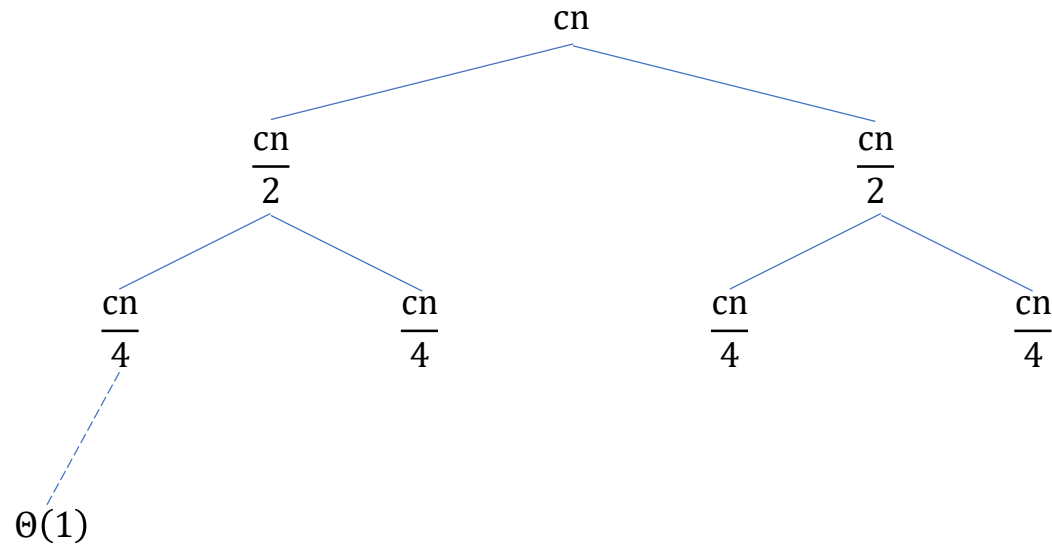
Merge Sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases} \quad \Theta(n) \approx cn$$



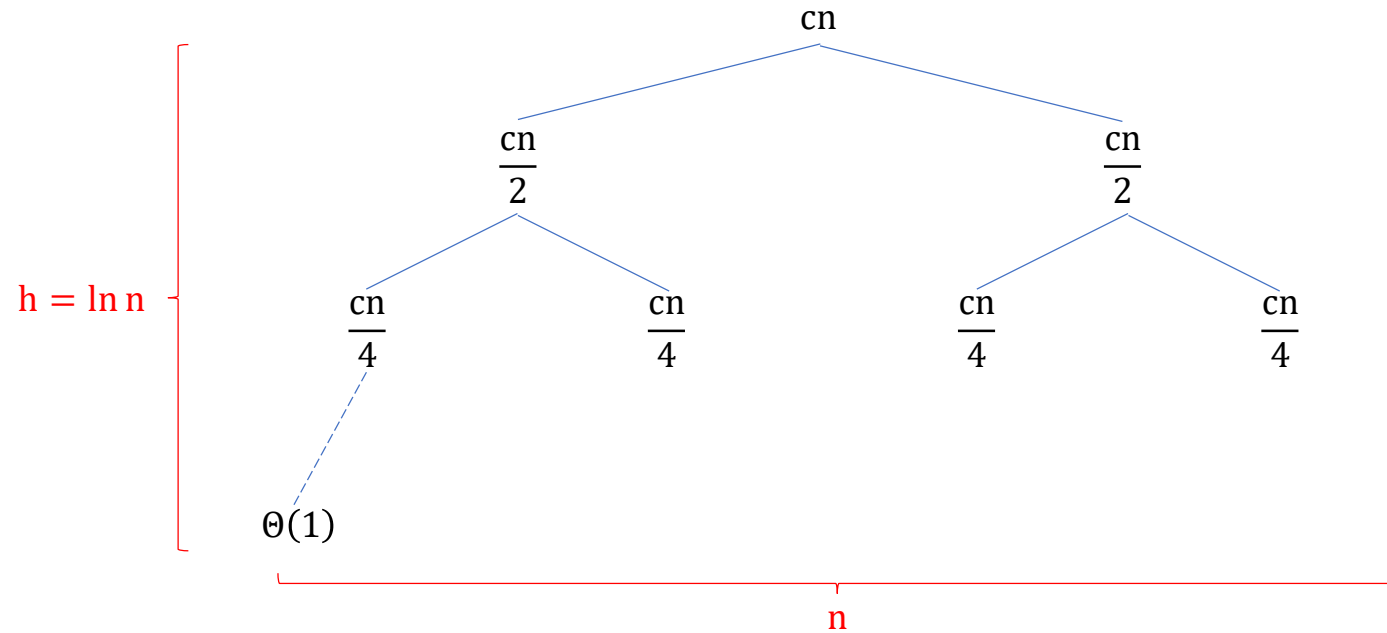
Merge Sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases} \quad \Theta(n) \approx cn$$



Merge Sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases} \quad \Theta(n) \approx cn$$



Merge Sort

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n) & \text{if } n > 1 \end{cases} \quad \Theta(n) \approx cn$$

