



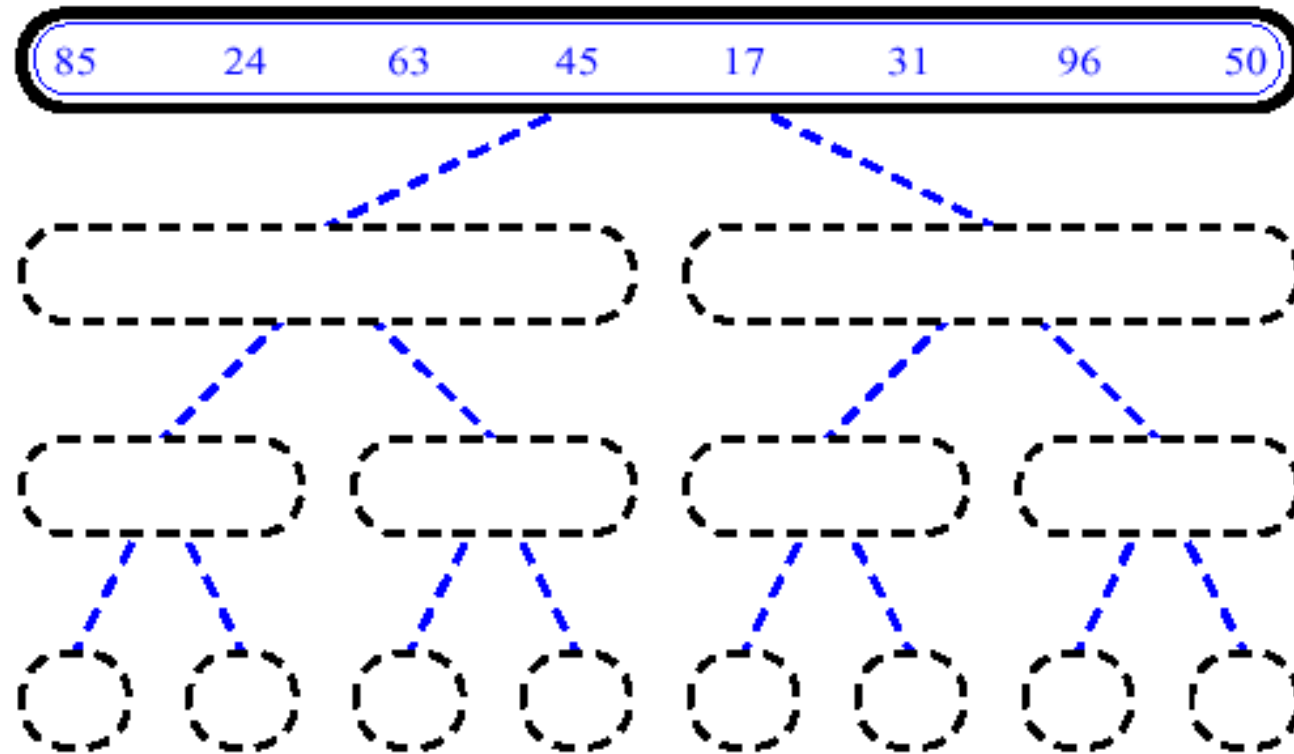
**Prepared by,**

Chetan Shetty  
Assistant Professor  
Department of CSE  
MSRIT  
Bangalore

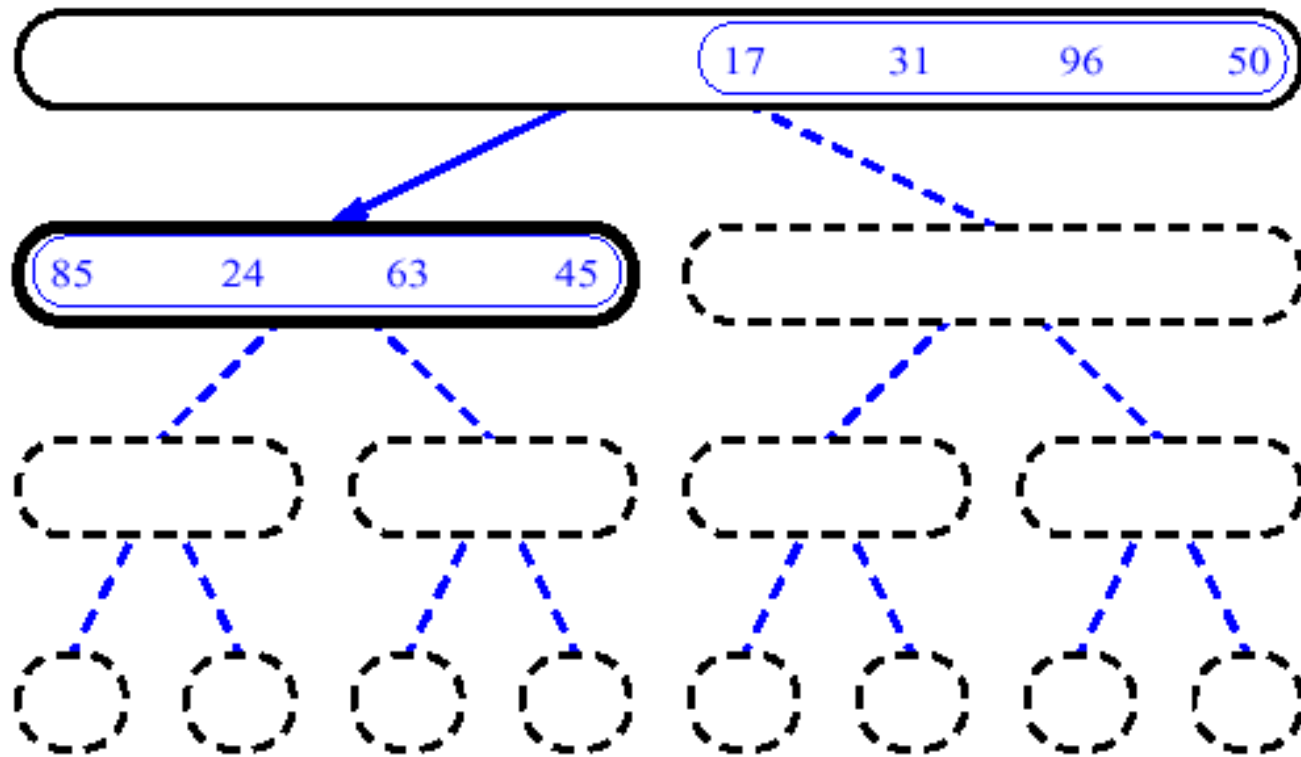
# DIVIDE AND CONQUER

- **Divide** the problem into sub-problems that are similar to the original but smaller in size.
- **Conquer** the sub-problems by solving them **recursively**. If they are small enough, just solve them in a straightforward manner.
- **Combine** the solutions to create a solution to the original problem

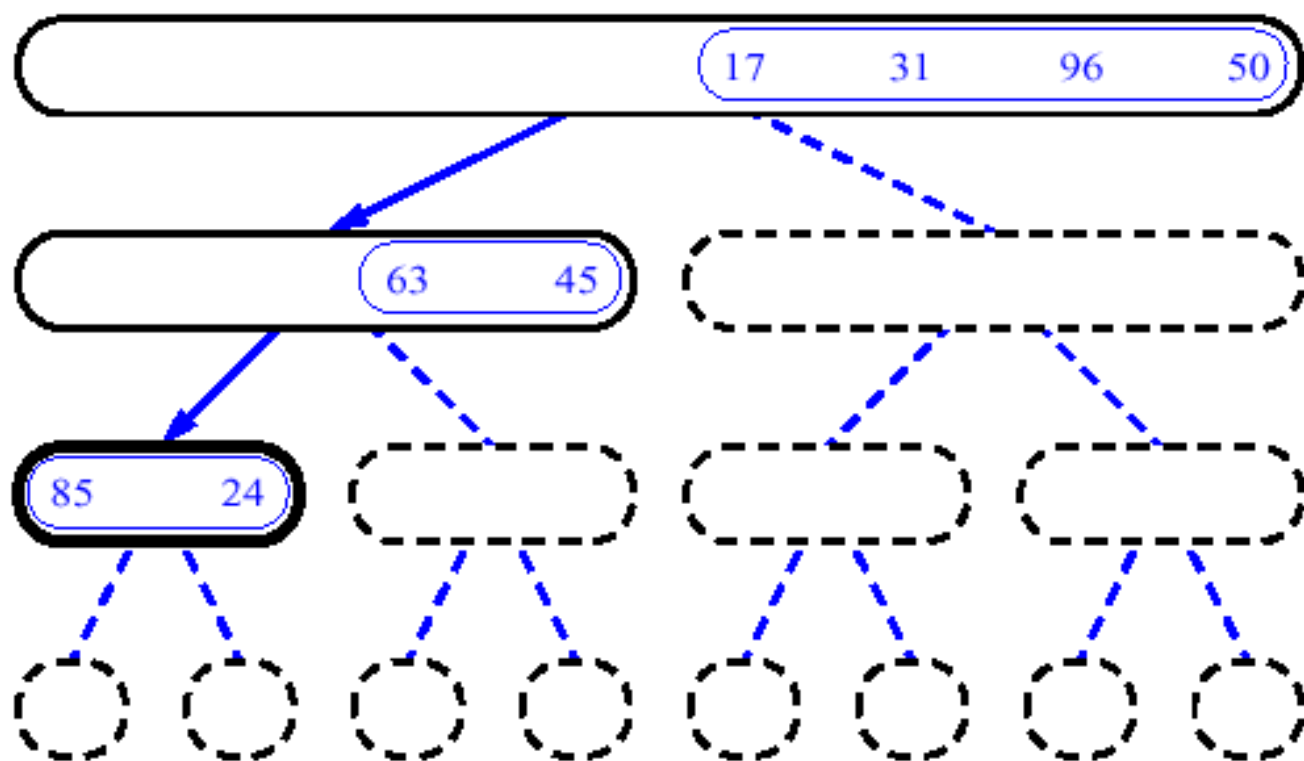
# MERGESORT



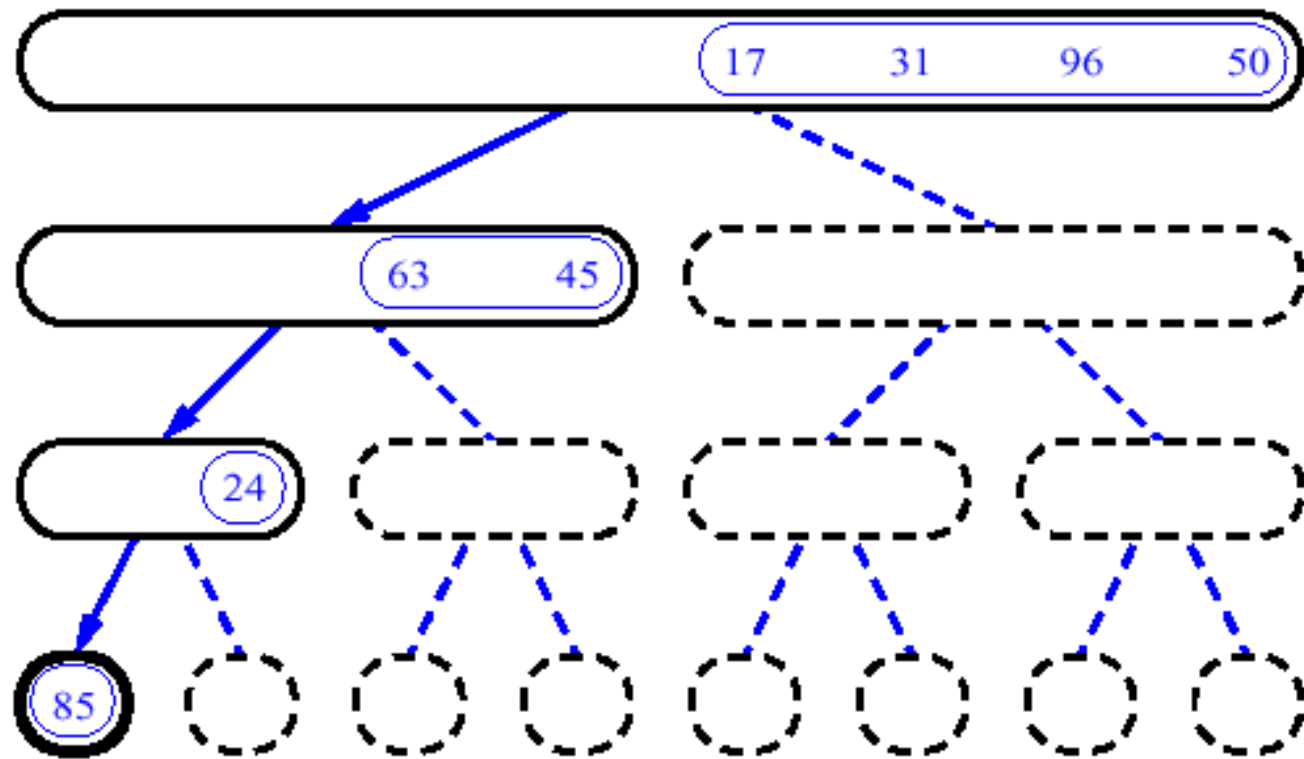
CONT...



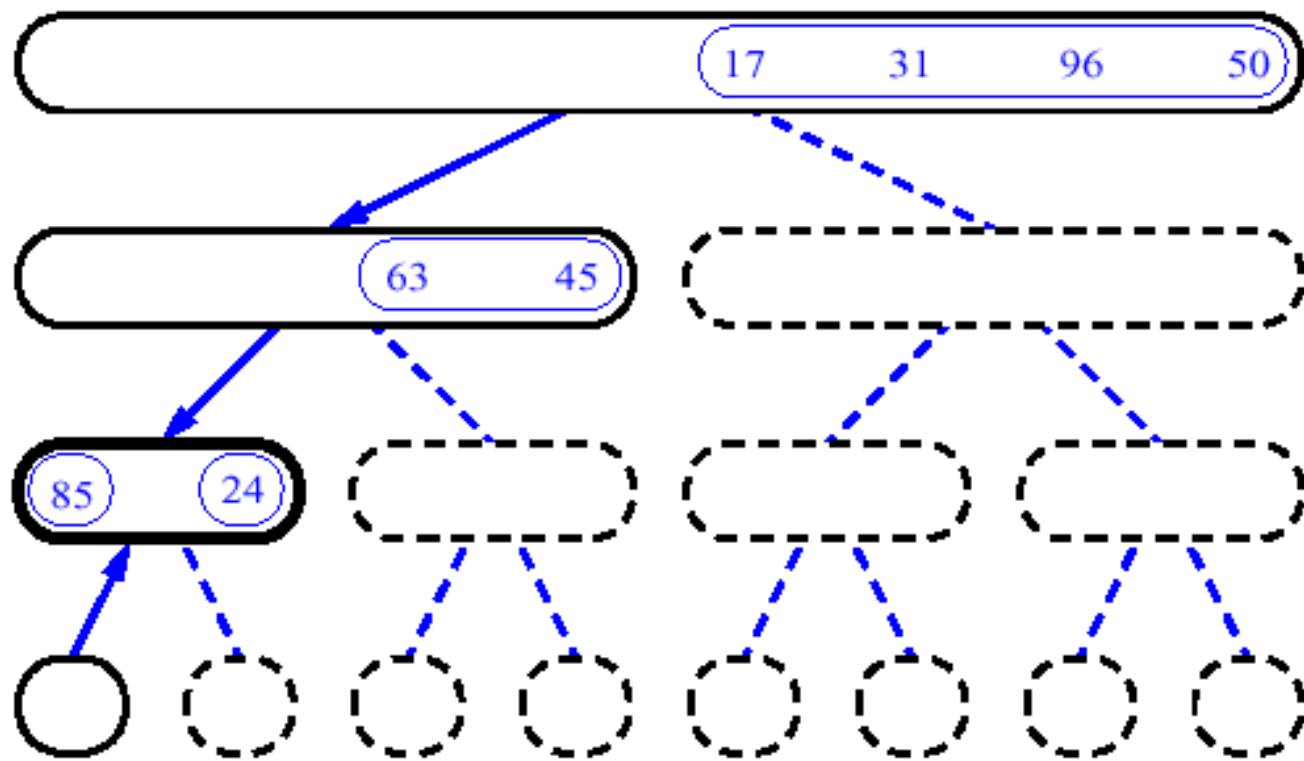
CONT...



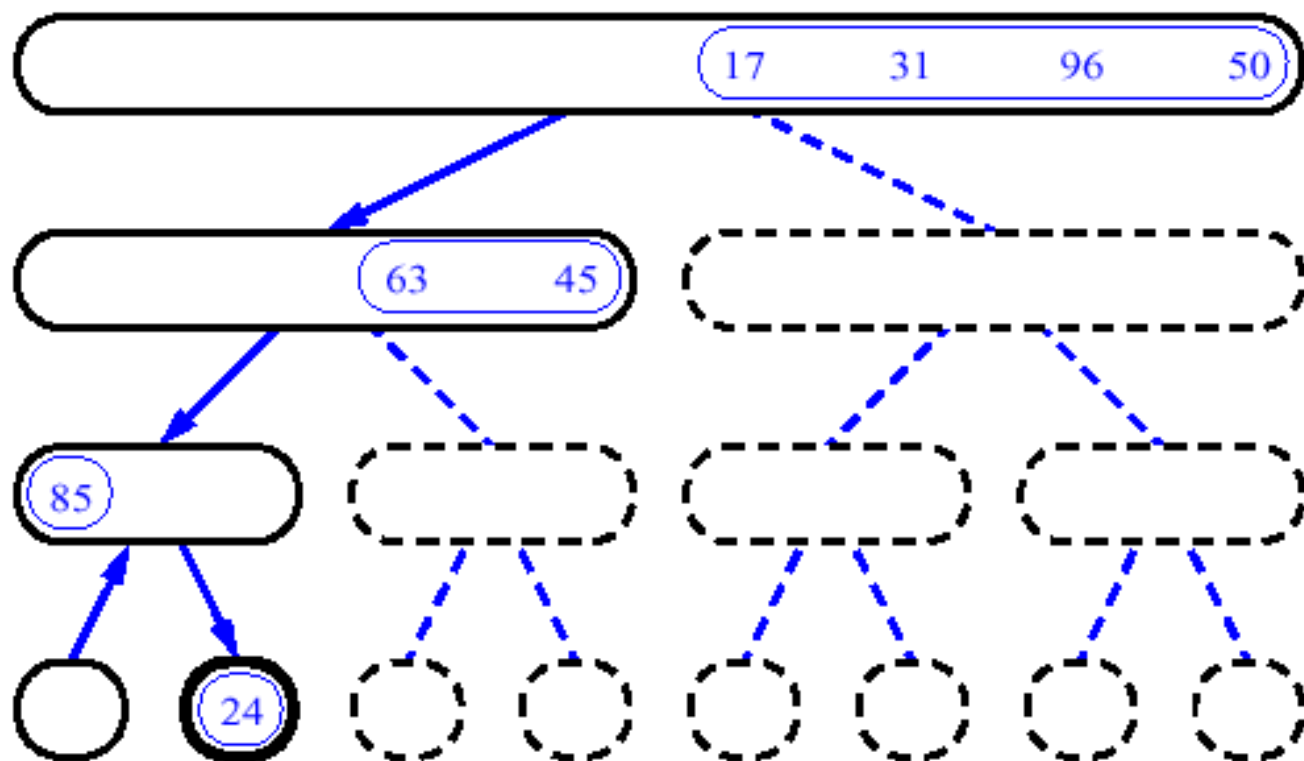
CONT...



CONT...

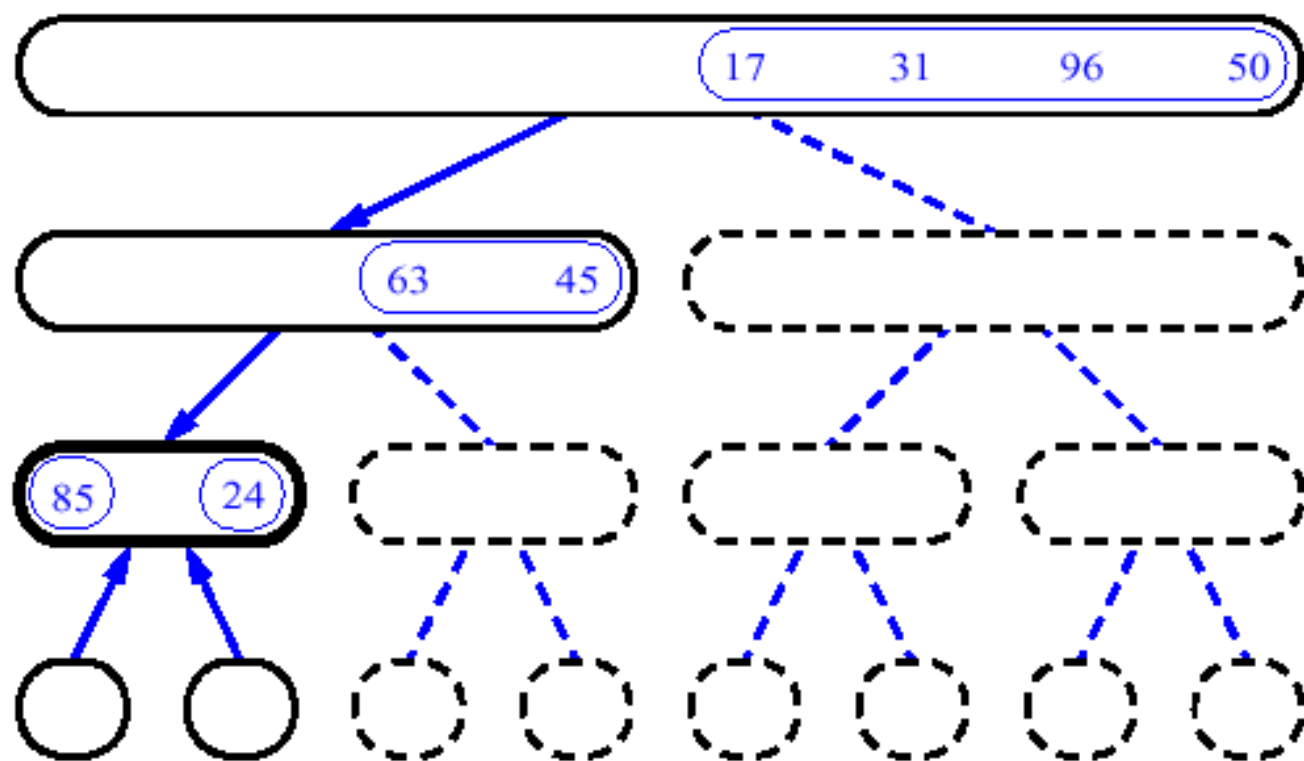


CONT...

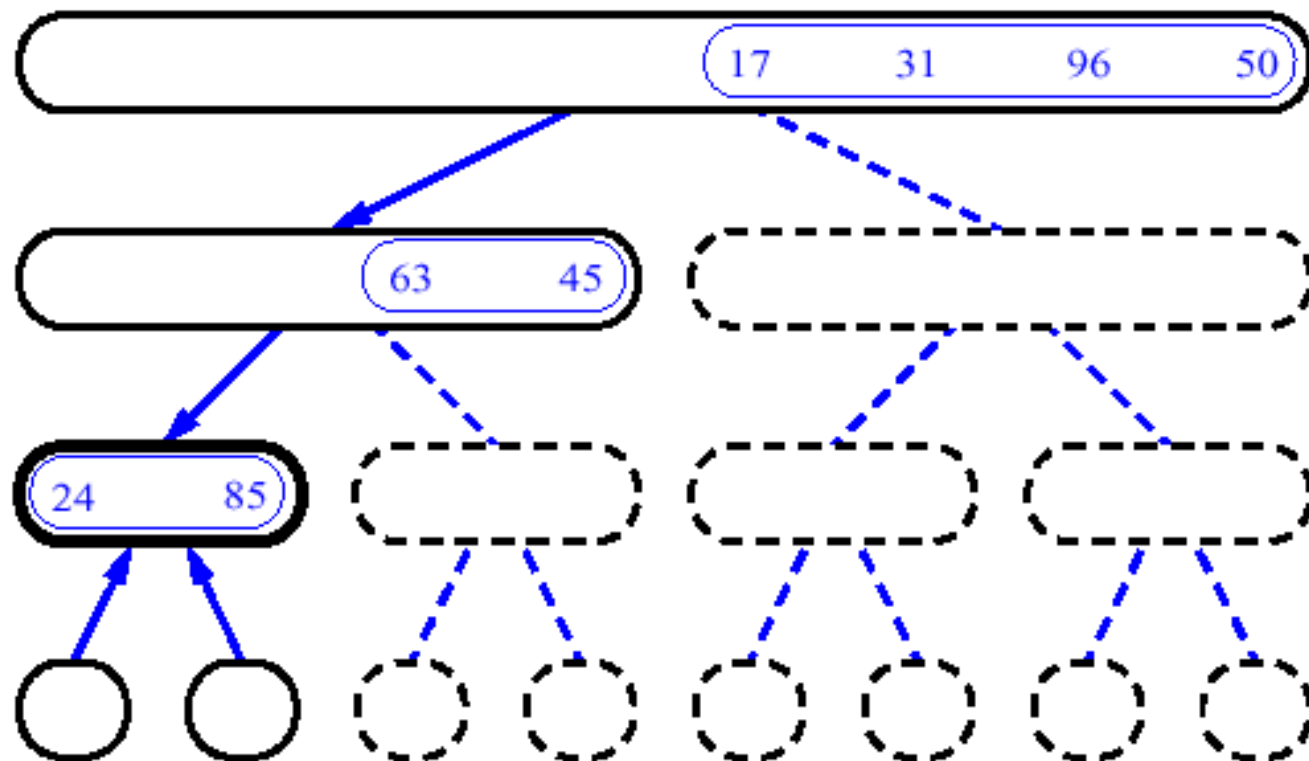




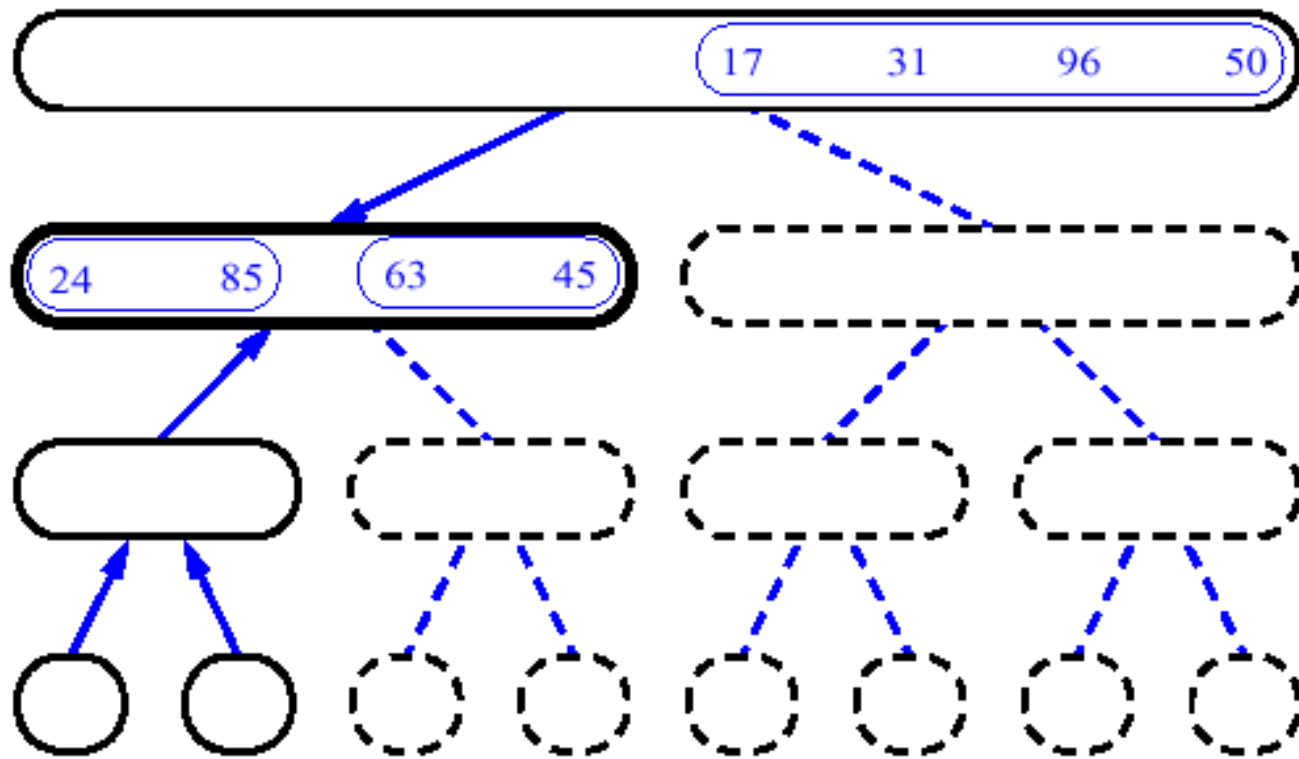
CONT...



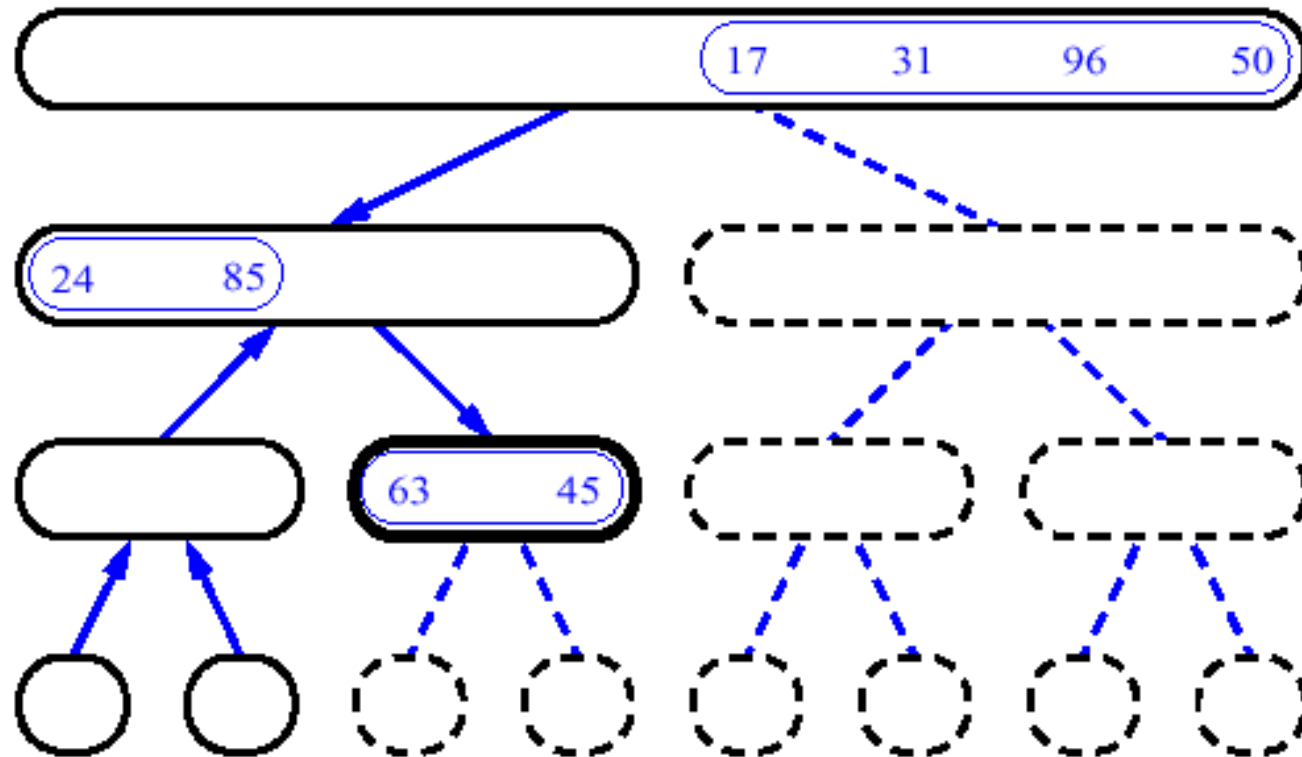
# CONT...



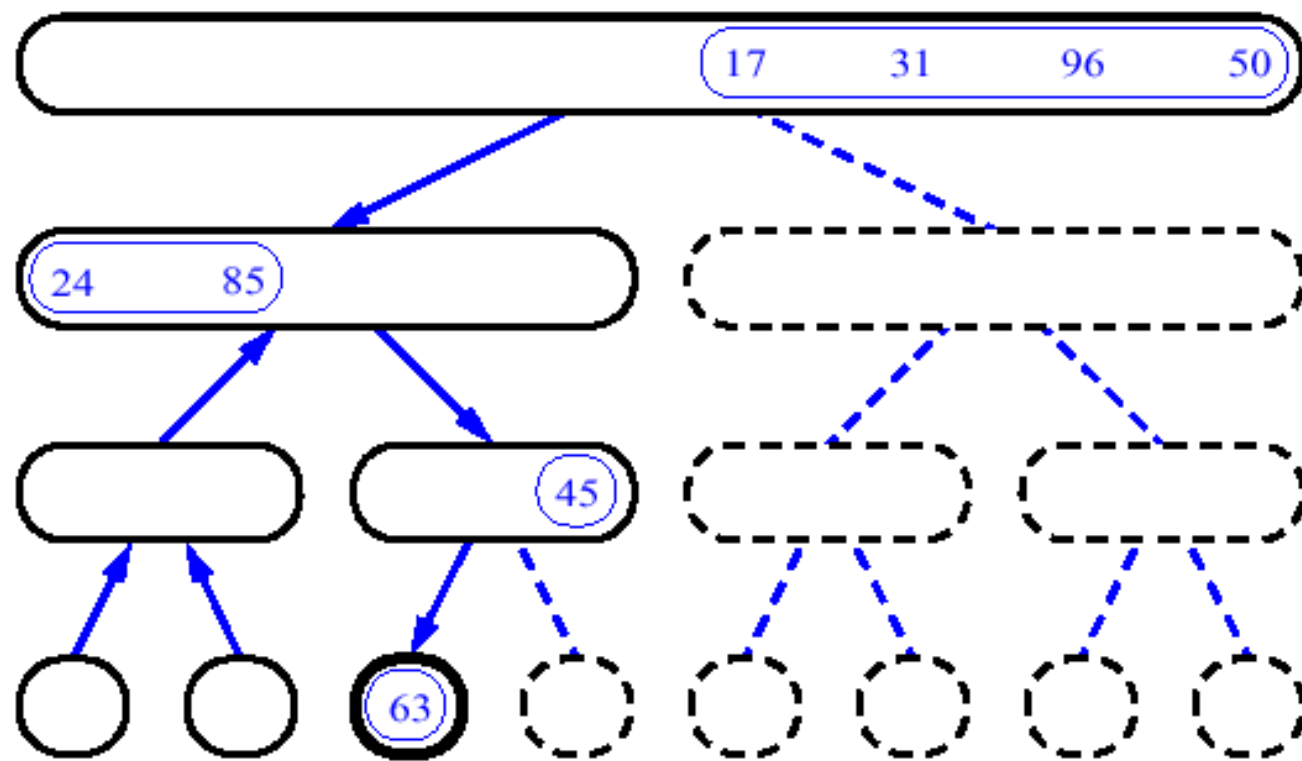
CONT...



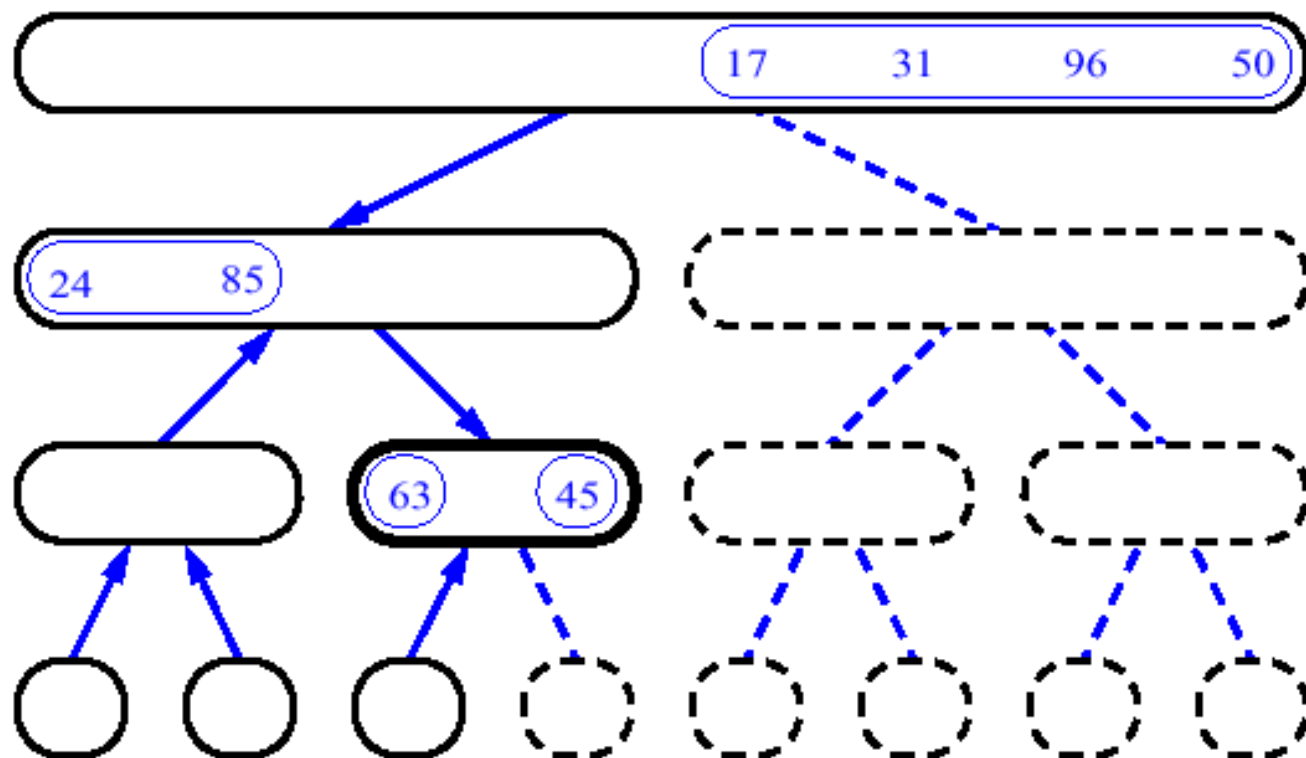
CONT...



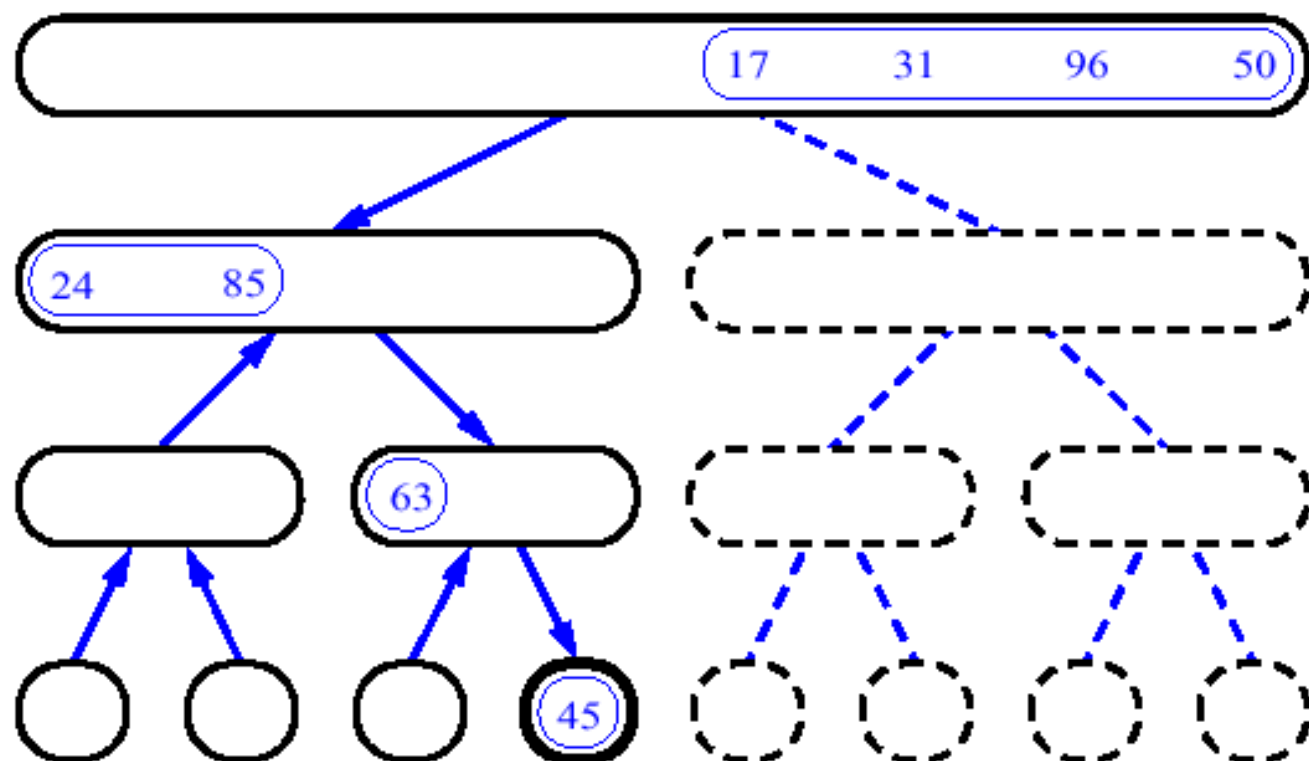
CONT...



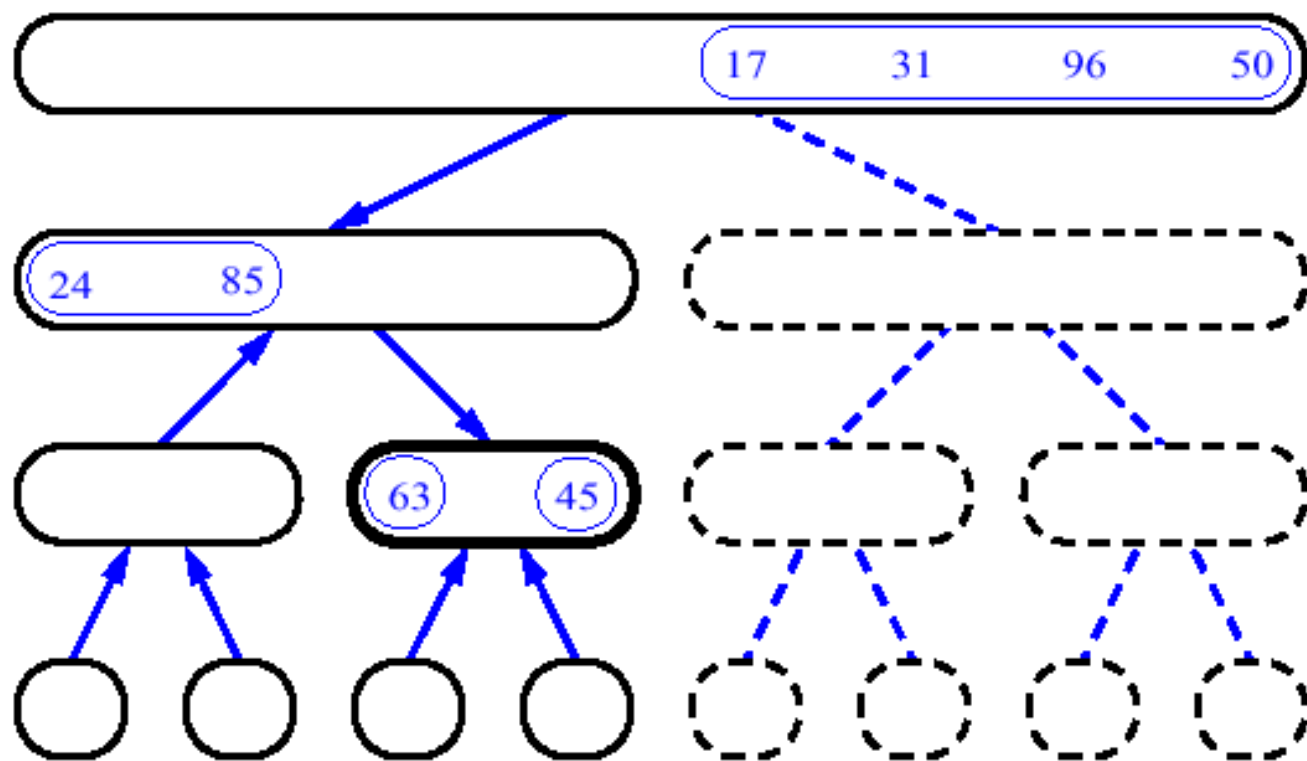
CONT...



CONT...

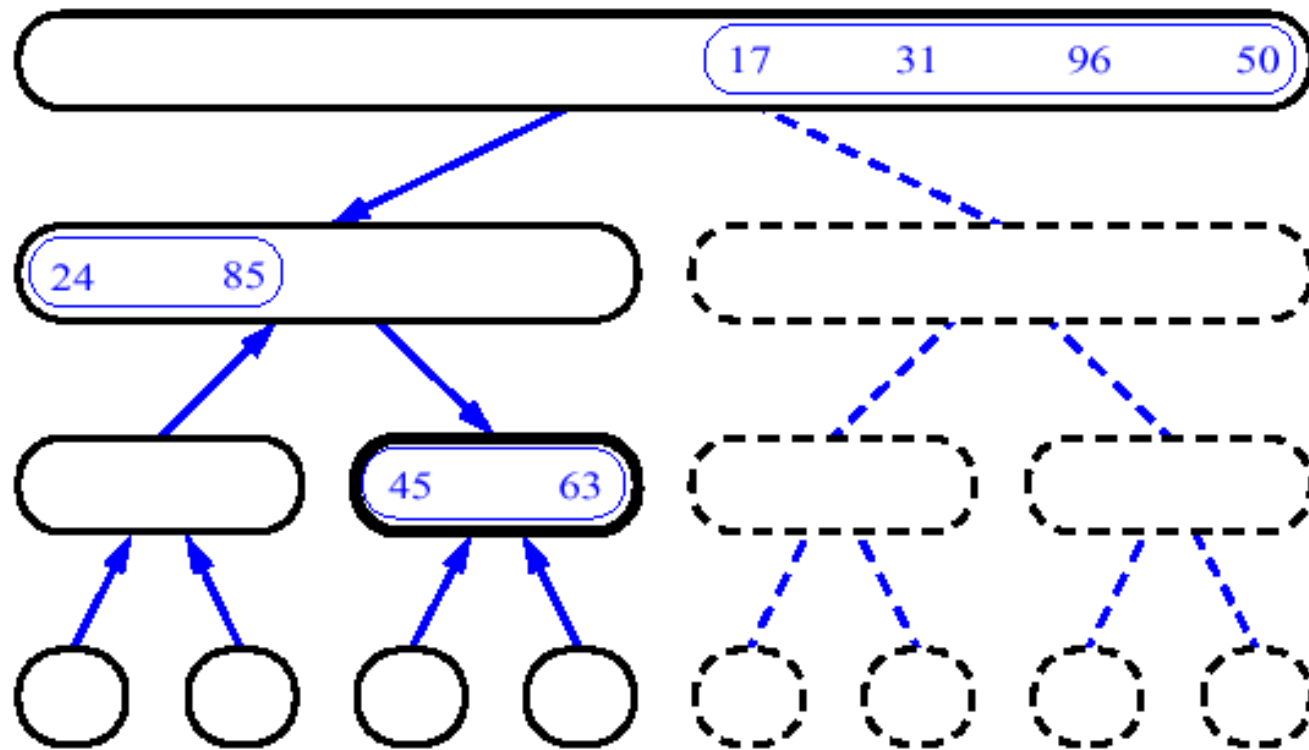


CONT...

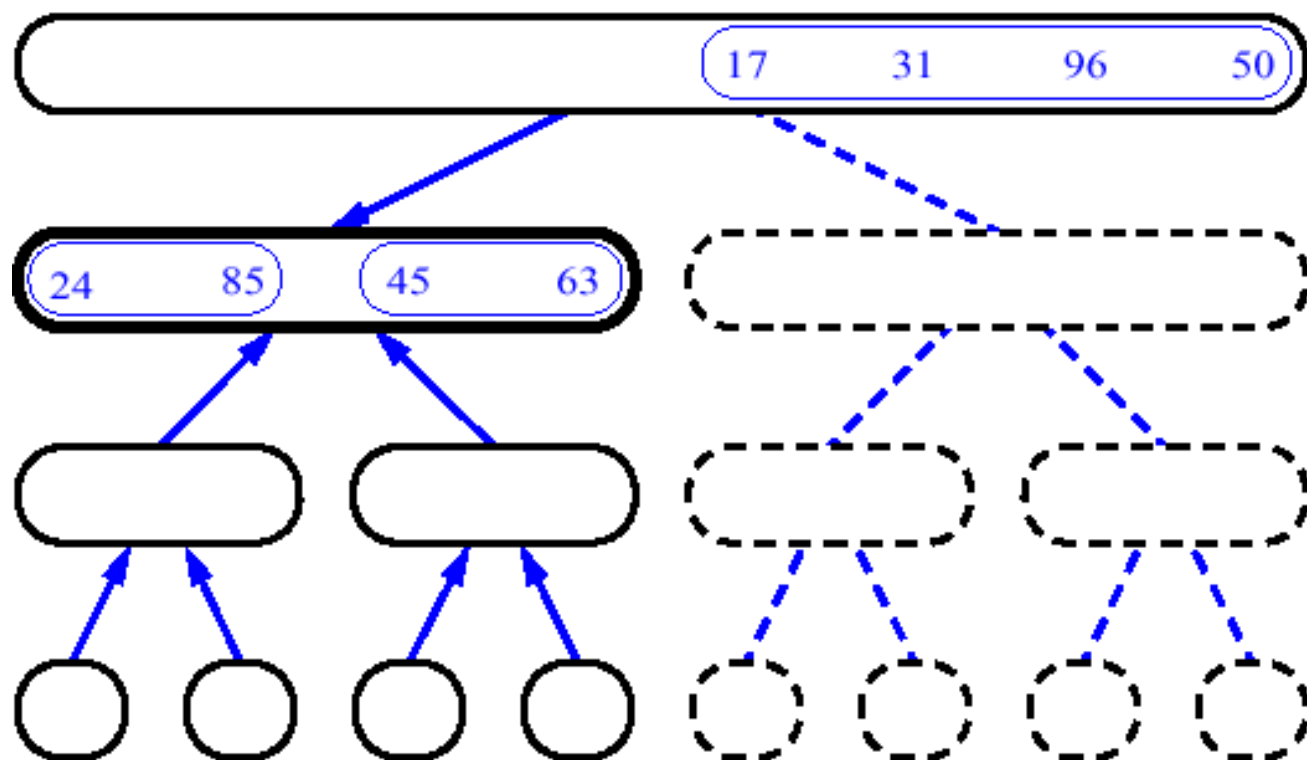




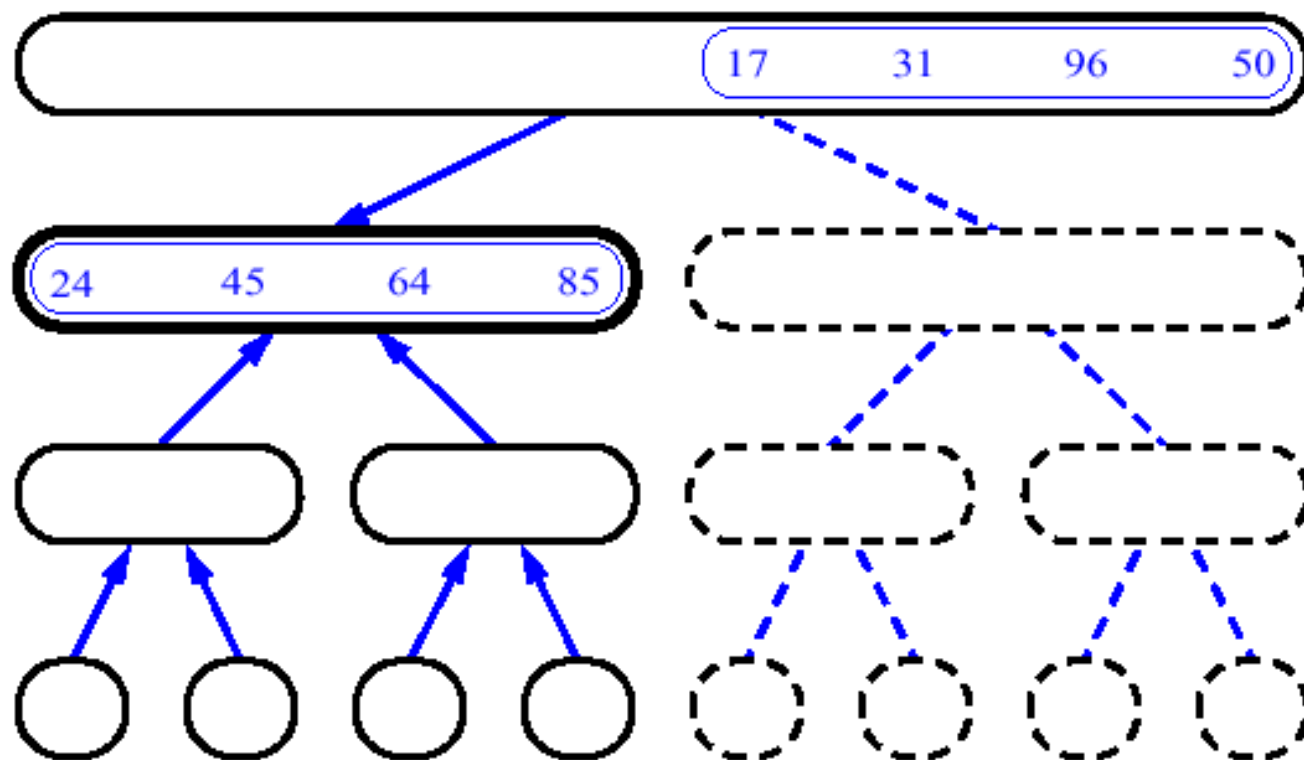
CONT...



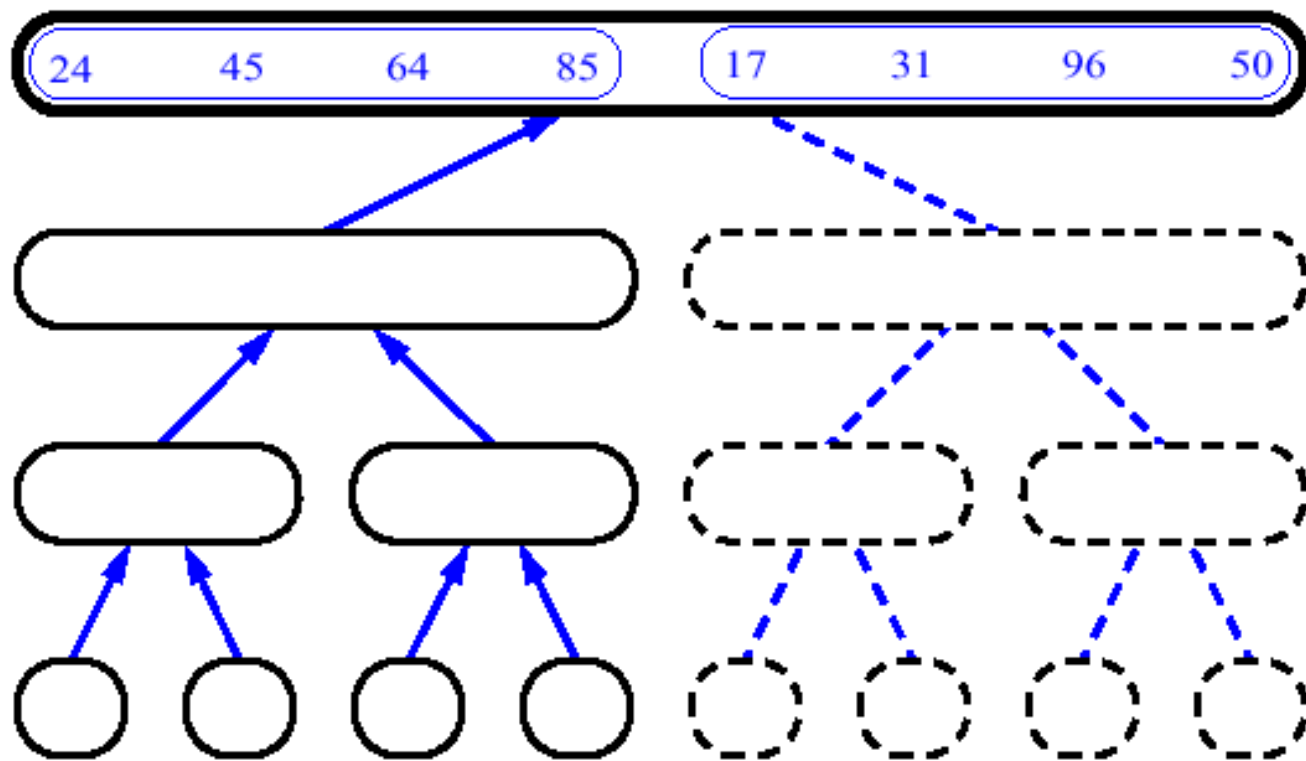
CONT...



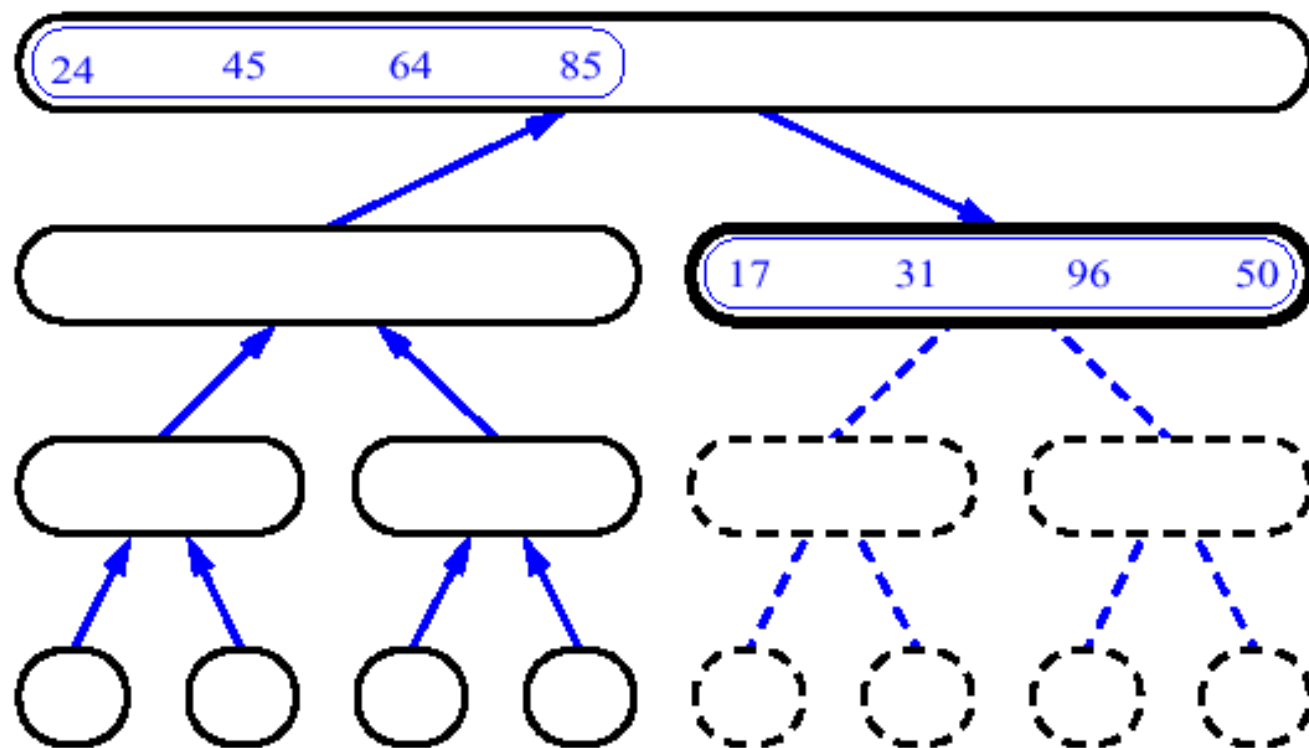
CONT...



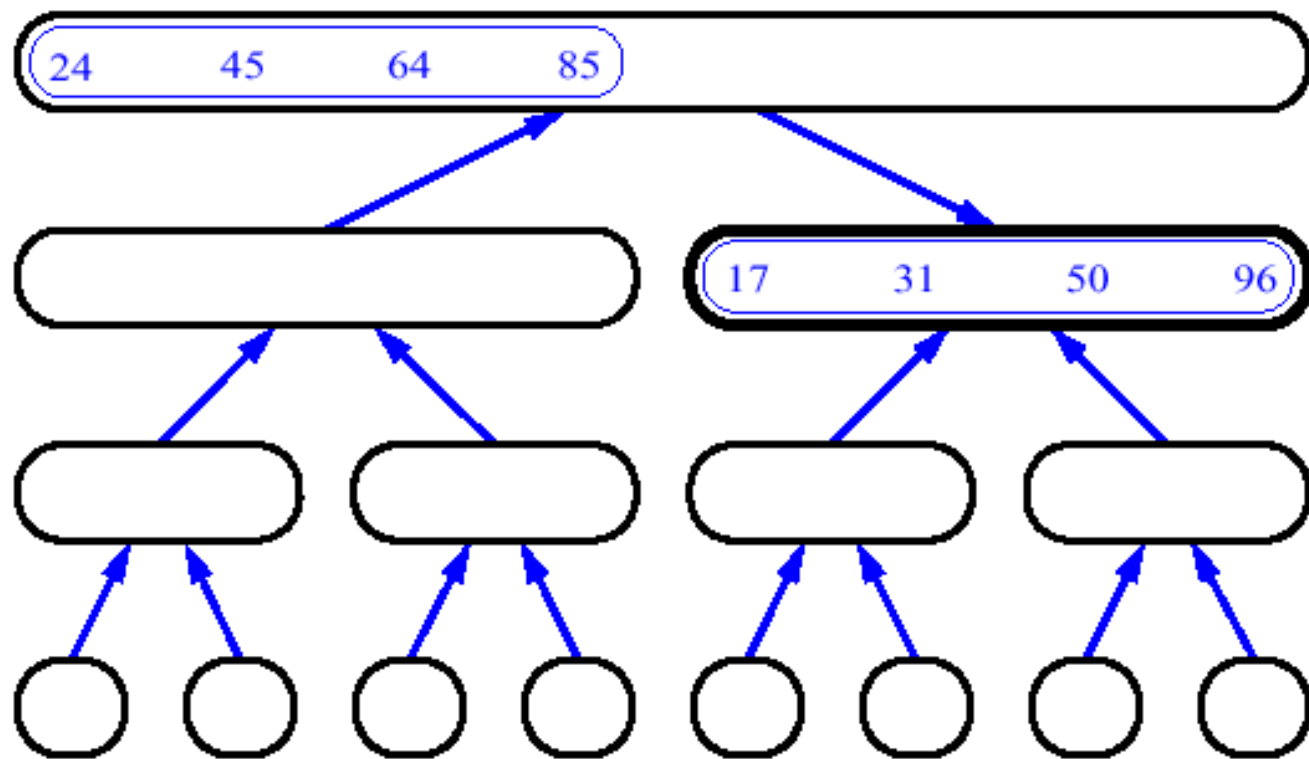
CONT...



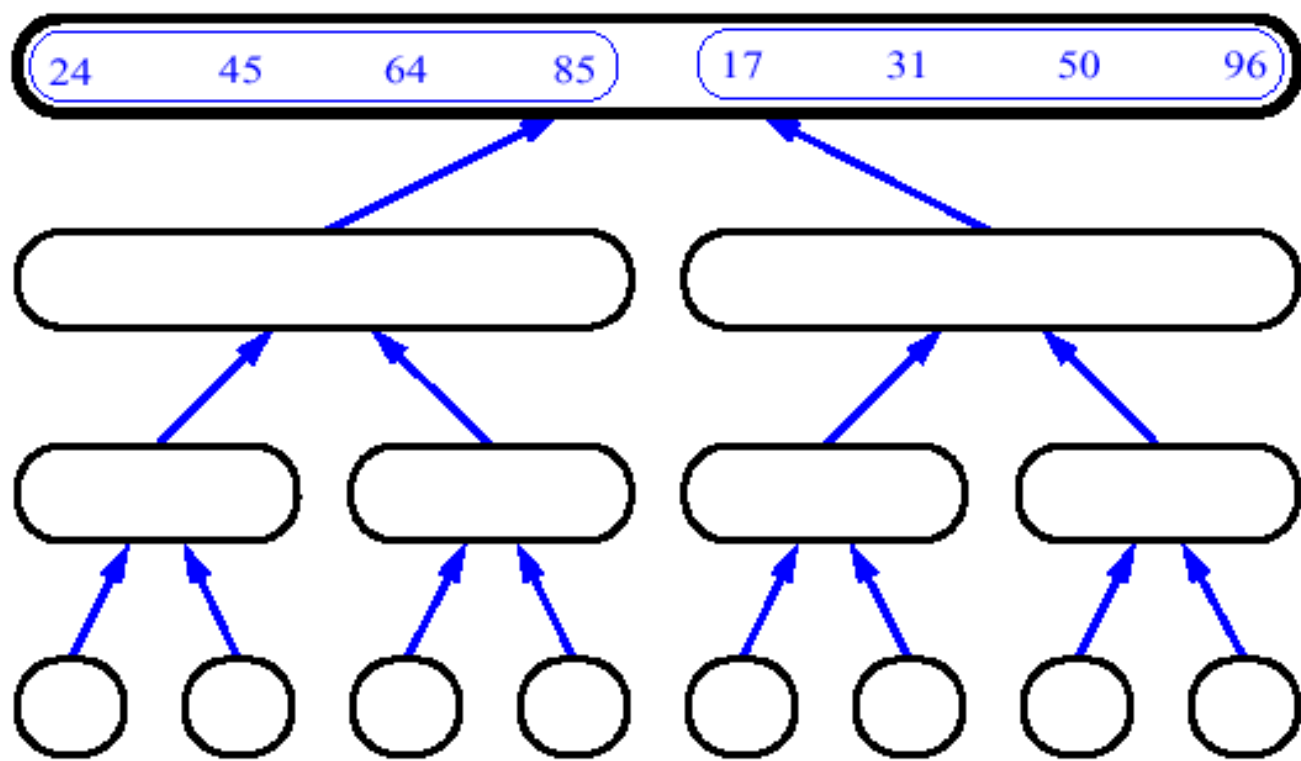
CONT...



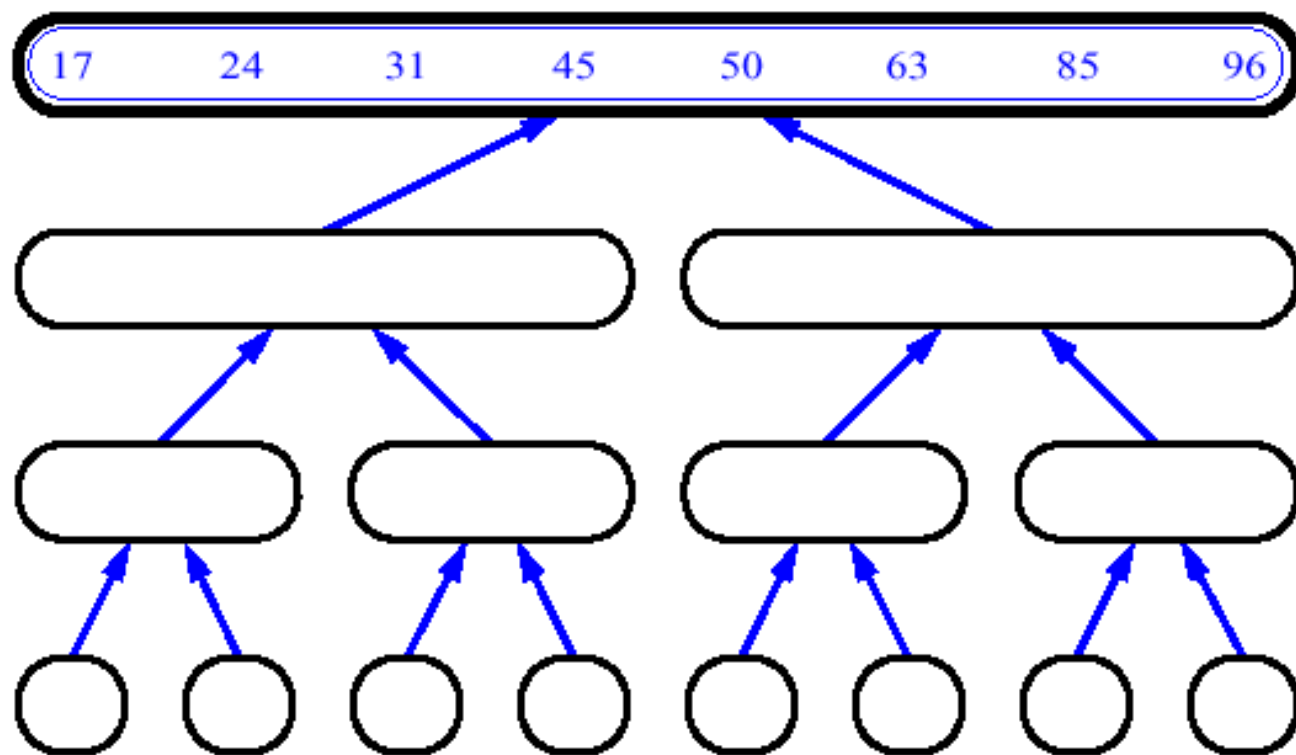
CONT...



CONT...



CONT...





14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

Merge

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6
---

Merge

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14
---	----

Merge

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23
---	----	----

Merge

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33
---	----	----	----

Merge

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42
---	----	----	----	----

Merge

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42	45
---	----	----	----	----	----

Merge



14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42	45	67
---	----	----	----	----	----	----

Merge

14	23	45	98
----	----	----	----

6	33	42	67
---	----	----	----

6	14	23	33	42	45	67	98
---	----	----	----	----	----	----	----

Merge

# ALGORITHM

MergeSort( $A[0 \dots n-1]$ )

if  $n > 1$

Copy  $A[0 \dots (n/2) - 1]$  to  $B[0 \dots (n/2) - 1]$

Copy  $A[(n/2) \dots n-1]$  to  $C[0 \dots (n/2) - 1]$

MergeSort( $B[0 \dots n-1]$ )

MergeSort( $C[0 \dots n-1]$ )

Merge( $B, C, A$ )

Merge( $B[0 \dots p-1], C[0 \dots q-1], A[0 \dots p+q-1]$ )

$i \leftarrow 0, j \leftarrow 0, k \leftarrow 0$

while  $i < p$  &  $j < q$  do

if  $B[i] \leq C[j]$

$A[k] \leftarrow B[i]; i++;$

else

$A[k] \leftarrow C[j]; j++;$

$k \leftarrow k+1;$

//Copy left over elements

if  $i == p$

copy  $C[j \dots q-1]$  to  $A[k \dots p+q-1]$

else

copy  $B[j \dots p-1]$  to  $A[k \dots p+q+1]$

8

3

2

9

7

1

5

4

MergeSort(A[0....n-1]) → T(n)

if n>1

Copy A[0.... (n/2) -1] to B[ 0.... (n/2) -1]

Copy A[(n/2) ... n-1] to C[ (n/2) ... n-1]

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ T(n/2) + T(n/2) + n & \text{otherwise} \end{cases}$$

(The terms  $T(n/2)$  and  $T(n/2)$  are labeled "solve left half" and "solve right half" respectively, and the term  $n$  is labeled "merging".)

MergeSort(B[0....n-1]) → T(n/2)

MergeSort(C[0....n-1]) → T(n/2)

Merge(B,C,A) → T(n)

$$T(n) = 2T(n/2) + cn$$

# APPROACHES TO SOLVE RECURSION

## Approach 1:

1. Intuitive solution to recurrence is to “unroll” the recursion, accounting for the running time of first few levels.
2. Identify a pattern that can be continued as the recursion expands.
3. Sum the running times over all levels of the recursion and thereby arrives at a total running time.

## Step1: Analyze the first few levels.

- 1<sup>st</sup> level of recursion  $\rightarrow$  Single problem of size  $n \rightarrow O(n)$
- 2<sup>nd</sup> level of recursion  $\rightarrow$  2 problems each of size  $n/2 \rightarrow O(n/2)$
- 3<sup>rd</sup> level of recursion  $\rightarrow$  4 problems each of size  $n/4 \rightarrow O(n/4)$

## Step 2: Identifying the pattern.

- At level  $j$  of the recursion, the number of subproblems are now a total of  $2^j$
- Each problem has shrunk in size by factor of 2 “j” time  $\rightarrow n/2^j$

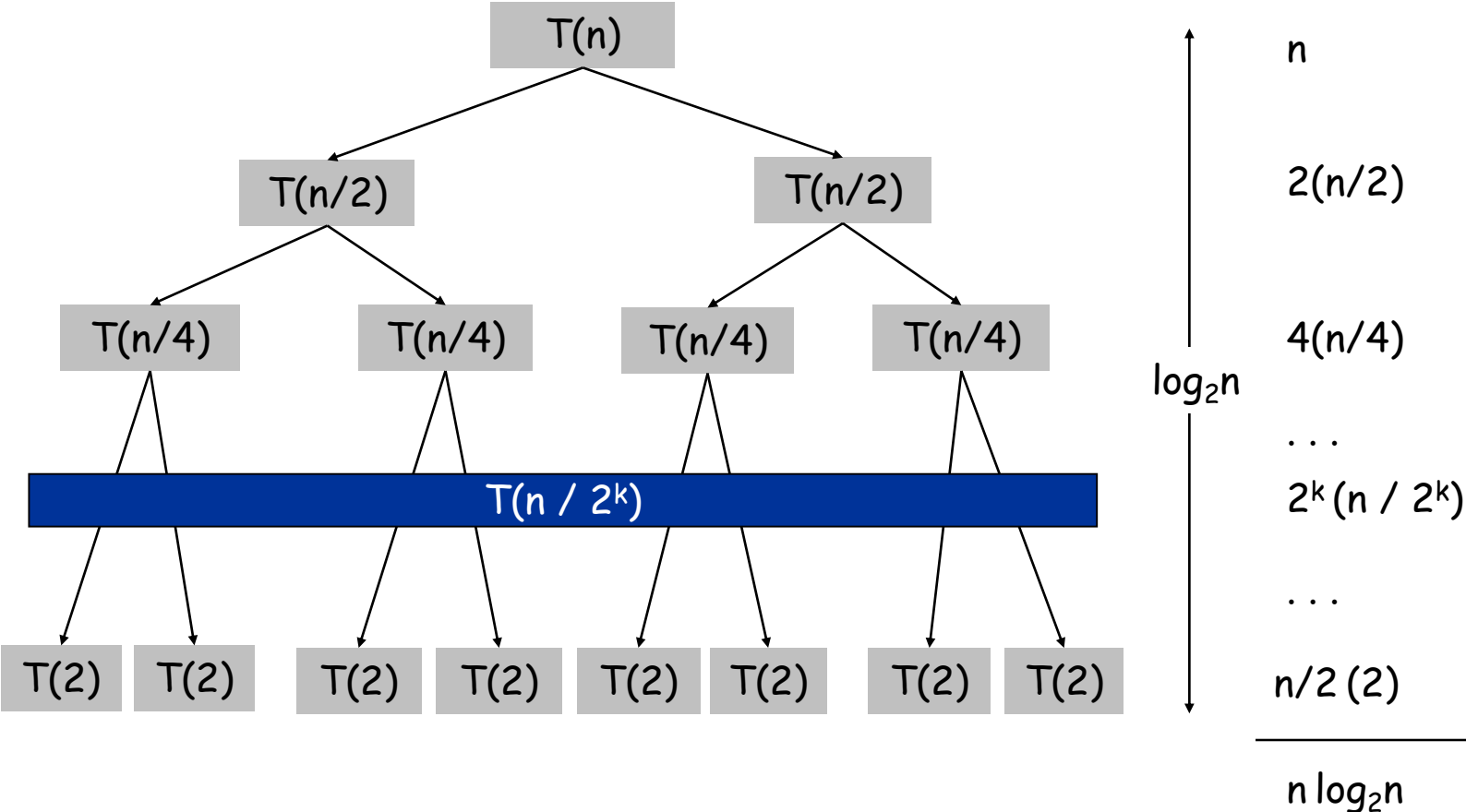
## Step 3: Summing overall levels of recursion.

- The number of times the input must be halved to reduce the size of  $n$  to 2 is  $\log n$
- There are totally “n” levels of recursion  $\rightarrow O(n \log n)$

# ANALYSIS



$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{2T(n/2)}_{\text{sorting both halves}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$



# SUBSTITUTING A SOLUTION INTO THE MERGESORT RECURSION

$$T(n) = O(n \log n)$$

$$T(n) = c \cdot n \log n$$

$$T(n/2) = c \cdot n/2 \log n/2$$

$$T(n) = 2T(n/2) + O(n)$$

$$T(n) = 2T(n/2) + cn$$

$$T(n) = 2 \cdot c \cdot n/2 \log n/2 + cn$$

$$T(n) = cn \cdot [\log n - 1] + cn$$

$$T(n) = cn \cdot \log n - cn + cn$$

$$T(n) = cn \cdot \log n$$

$$T(n) = O(n \log n)$$