The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Analysis of Algorithms
CSC 402

2023-24

Subject Incharge

Dr. Bidisha Roy

Associate Professor

Room No. 401

email: bidisharoy@sfit.ac.in

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Module 2 – Divide and Conquer

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

DIVIDE AND CONQUER APPROACH

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

General Method

- Works on the approach of dividing a given problem into smaller sub problems (ideally of same size). ← ***Divide***
- Sub problems are solved independently using recursion. ← ***Conquer***
- Solutions for sub problems then combined to get solution for the original problem. ← ***Combine***

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Examples to be considered

- Merge Sort
- Quick Sort
- Binary search
- Finding Minimum and Maximum

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort

- **Divide:** If A has at least two elements (nothing needs to be done if A has zero or one elements), remove all the elements from A and put them into two sequences, L and R , each containing about half of the elements of A. (i.e. L contains the first n/2 elements and R contains the remaining n/2 elements).
- **Conquer:** Sort sequences L and R using Merge Sort.
- **Combine:** Put back the elements into A by merging the sorted sequences L and R into one sorted sequence

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort … contd

```
Merge_Sort (A, low, high)
    if low < high then
        mid ← (low+high)/2
        Merge_Sort(A, low, mid)
        Merge_Sort(A, mid+1, high)
        Merge (A, low, mid, high)
```

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

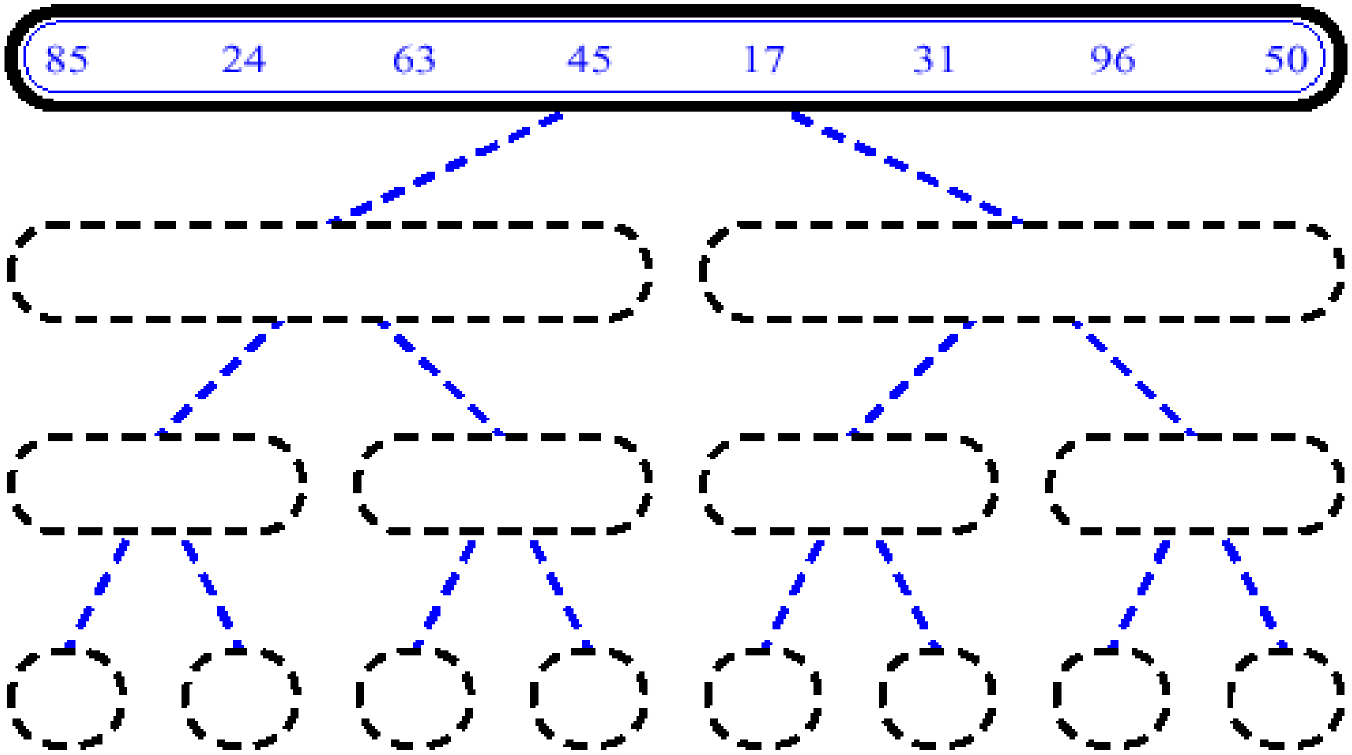
Merge(A, low, mid, high)

```
h ← low, i ← low, j ← mid + 1
Create auxiliary array B[ ]
while (h ≤ mid and j ≤ high) do
    if (A[h] ≤ A[j]) then
        B[i] ← A[h]
        h ← h+1
    Else
        B[i] ← A[j]
        j ← j+1
    i ← i+1
if (h>mid) then
    for k  ← j to high
        B[i] ← A[k]
        i ← i+1
else
    for k ← h to mid
        B[i] ← A[k]
        i ← i+1
for k ← low to high
    A[k] ← B[k]
```

Merge(A, low, mid , high)
Take the smallest of the two topmost elements of sequences A[low..mid] and A[mid+1..high] and put into the resulting sequence. Repeat this, until both sequences are empty. Copy the resulting sequence into A[low..high].

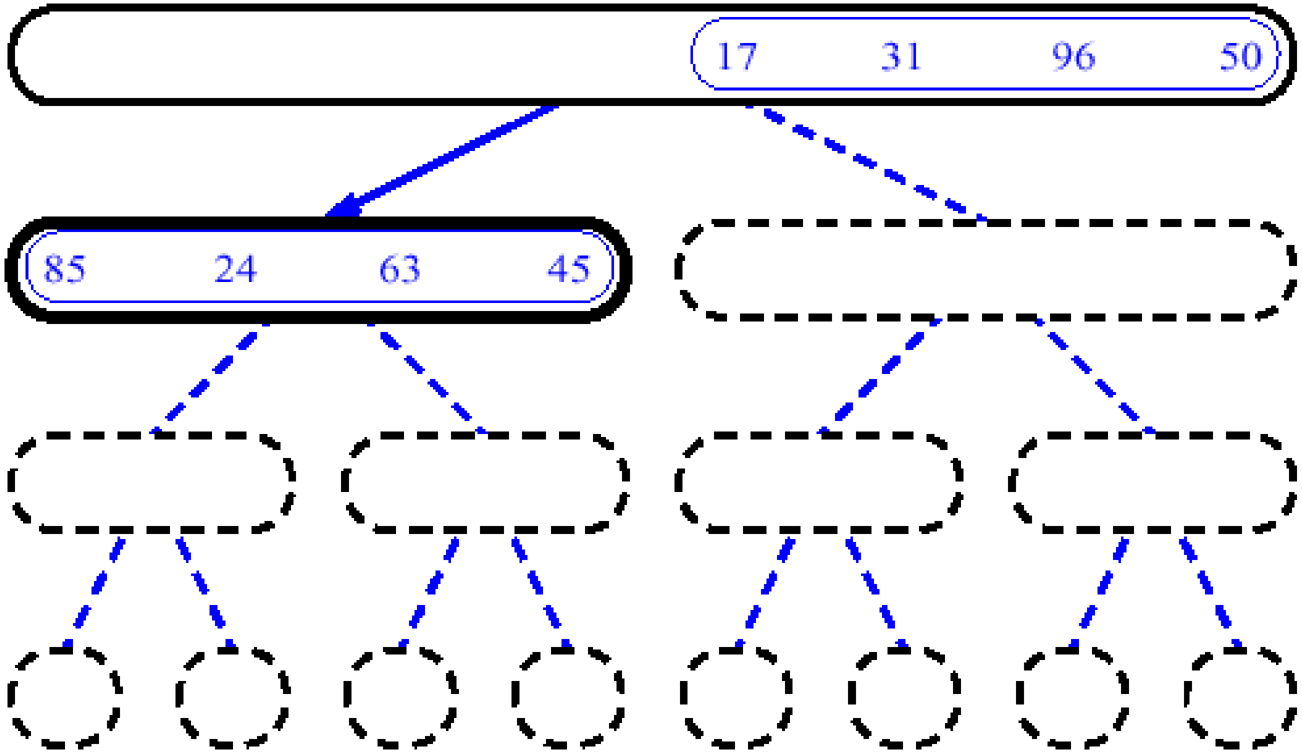
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 1



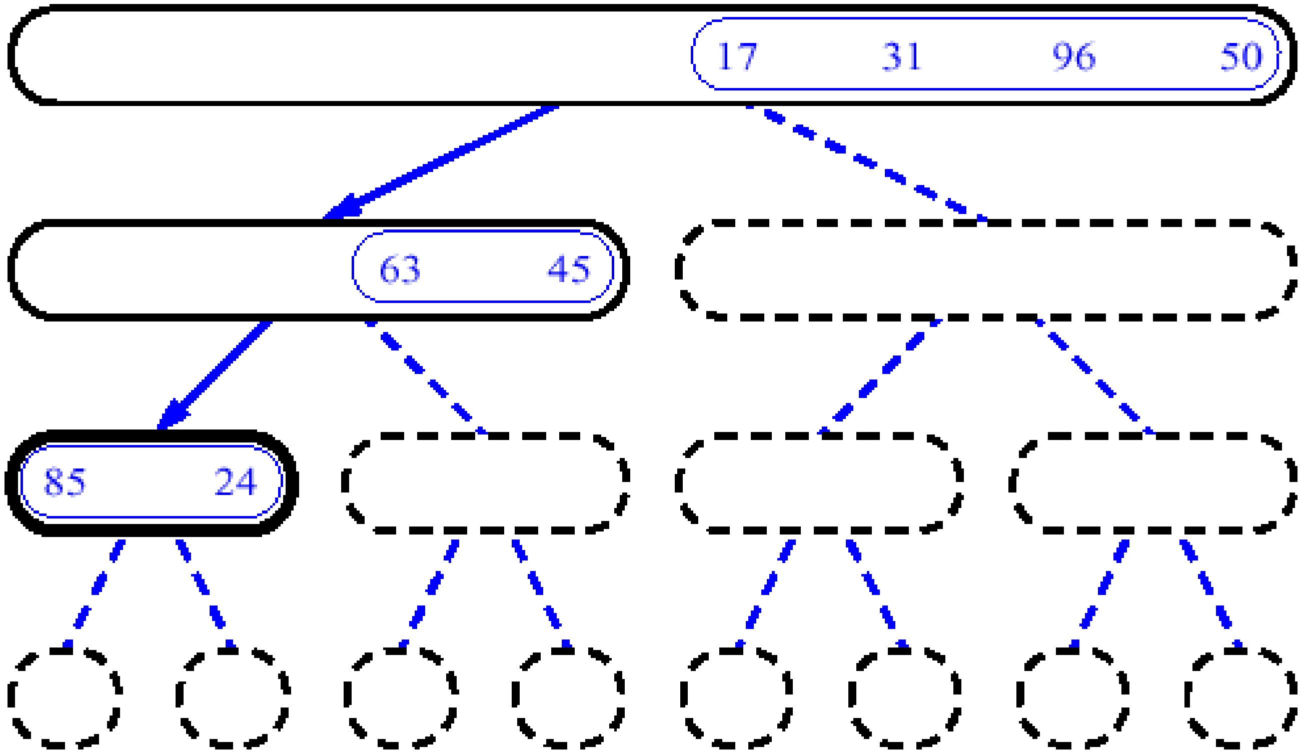
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 2



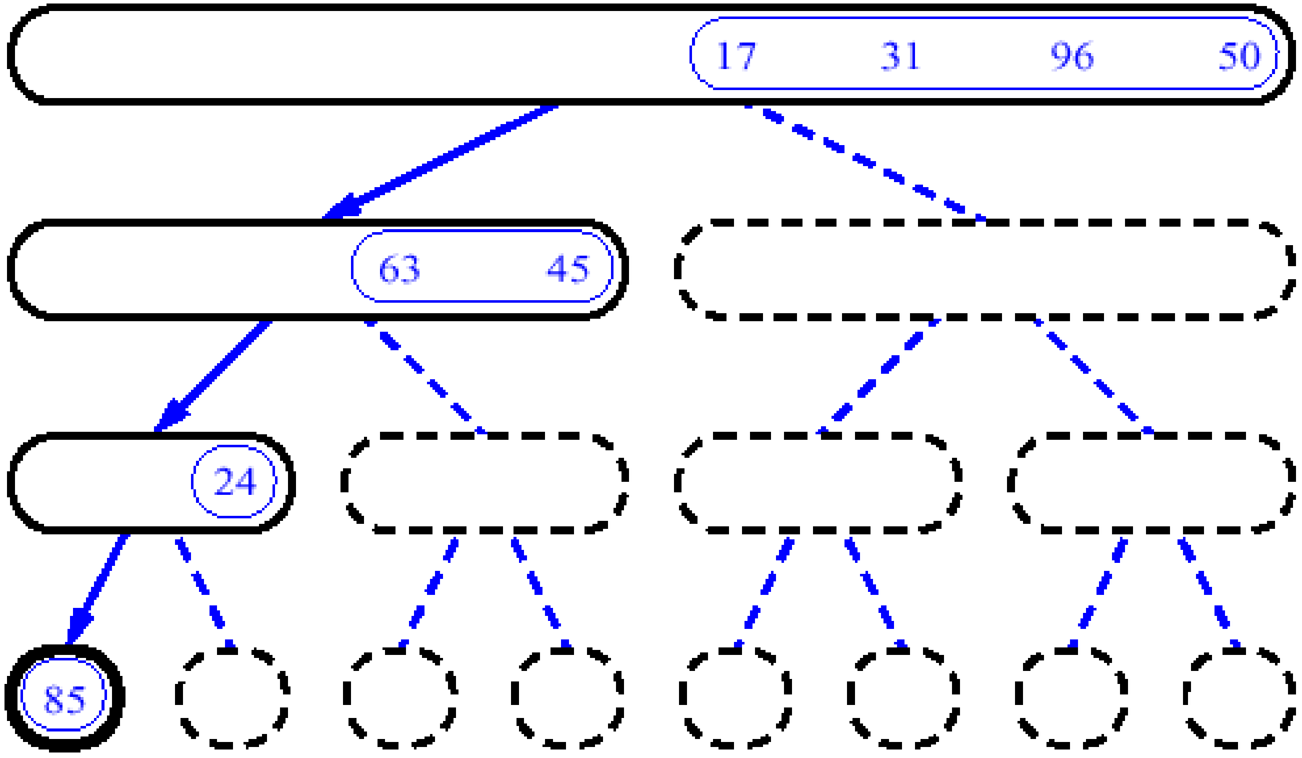
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 3



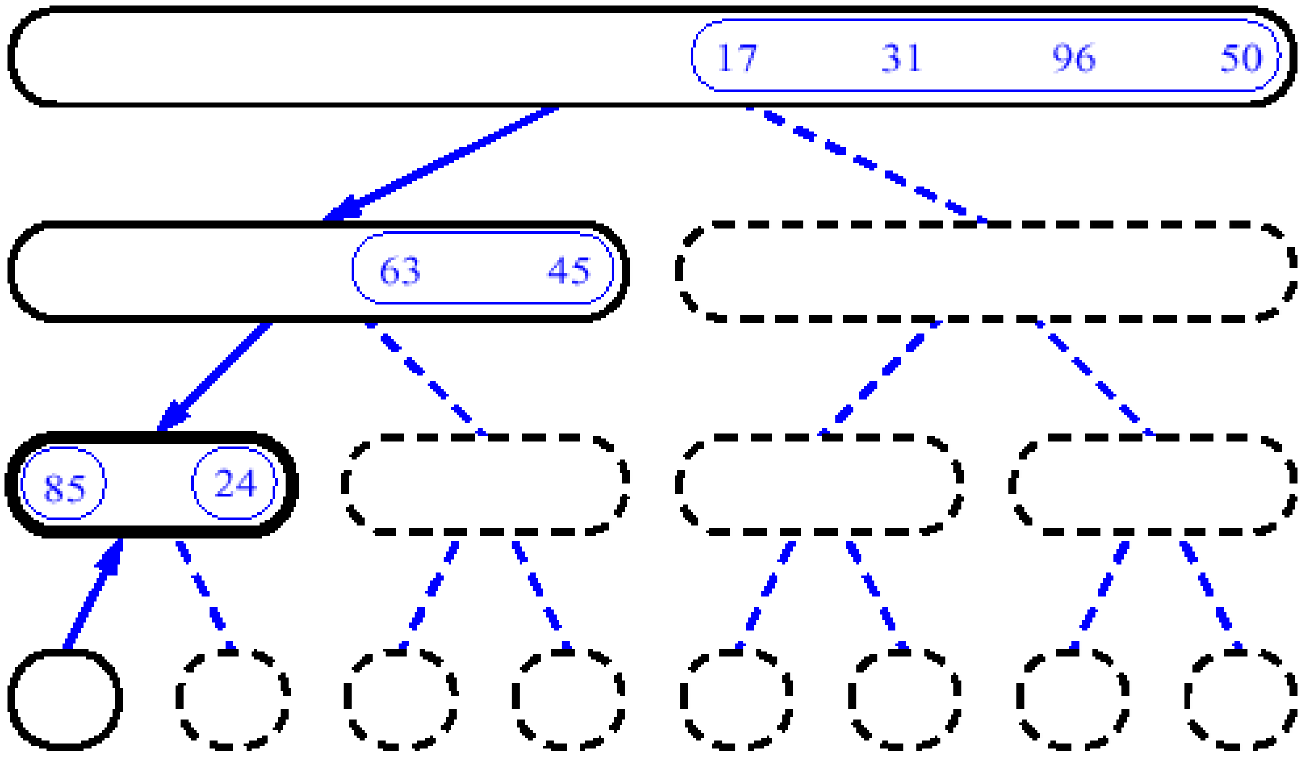
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 4



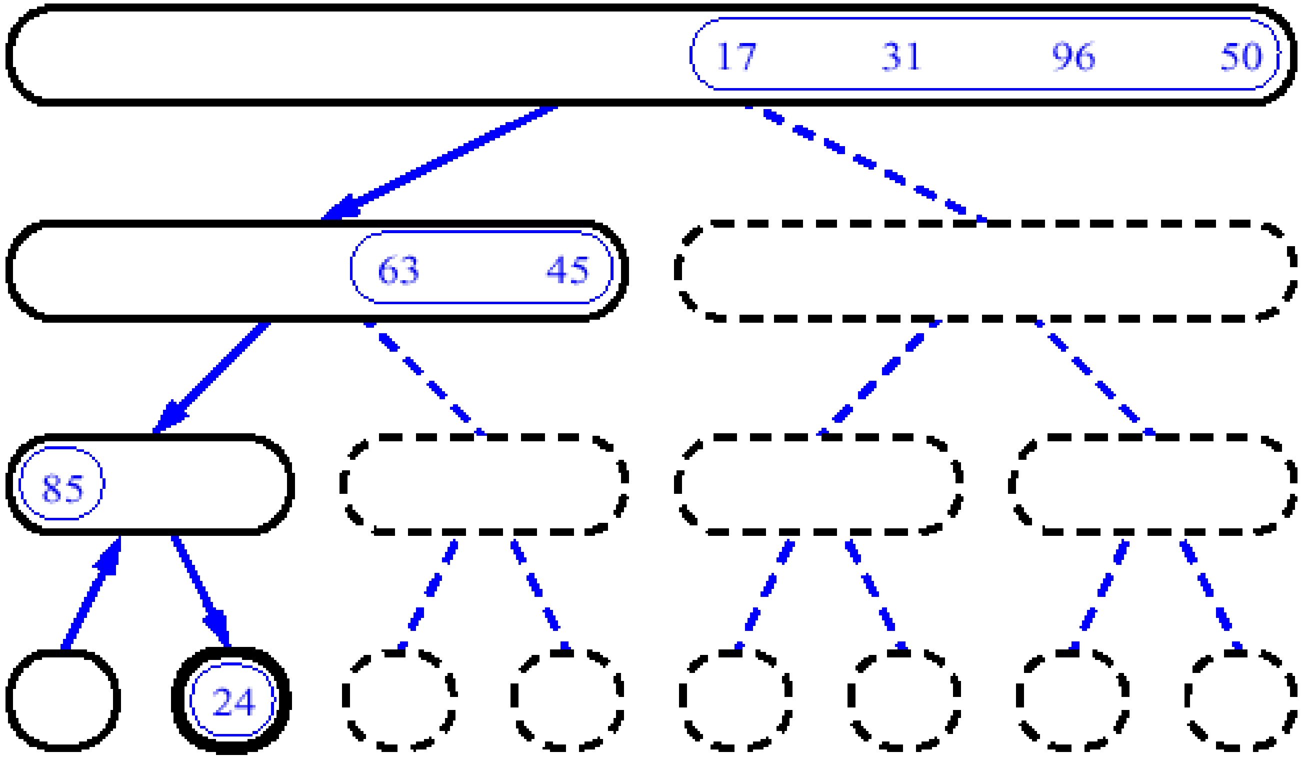
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 5



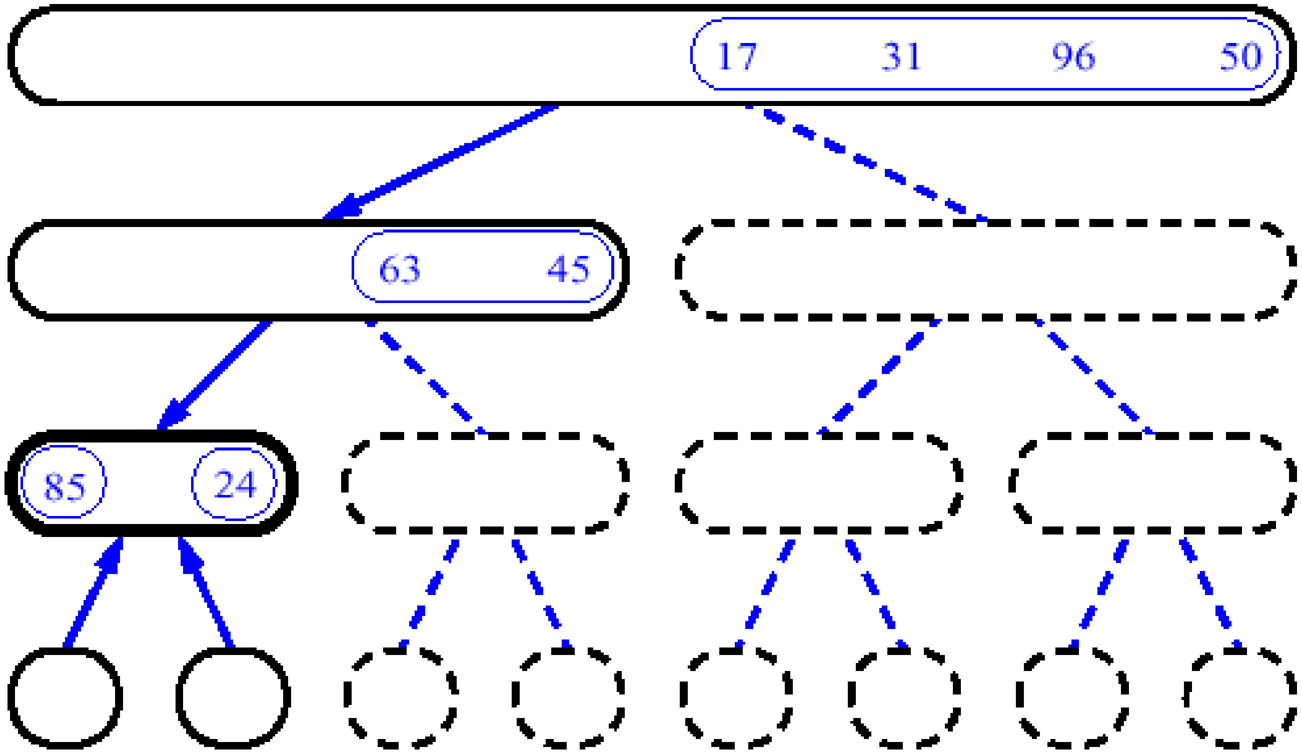
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 6



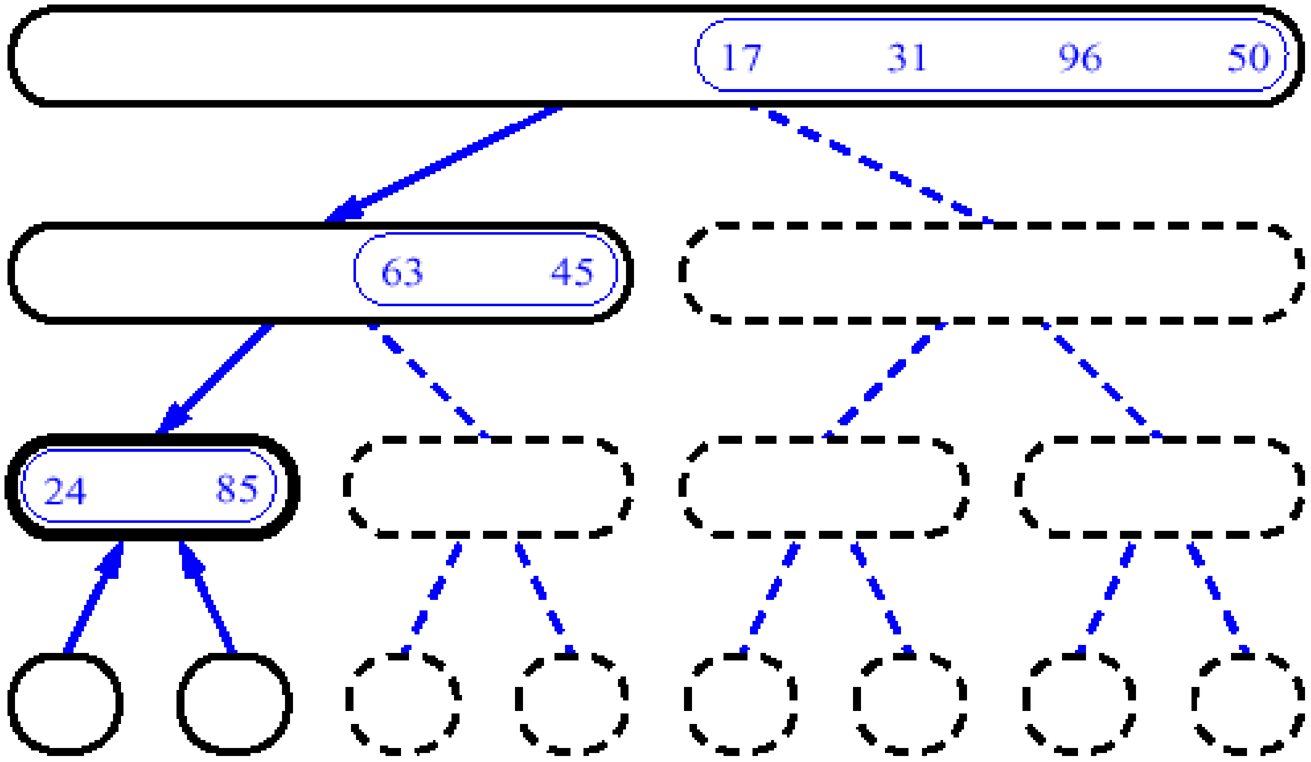
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 7



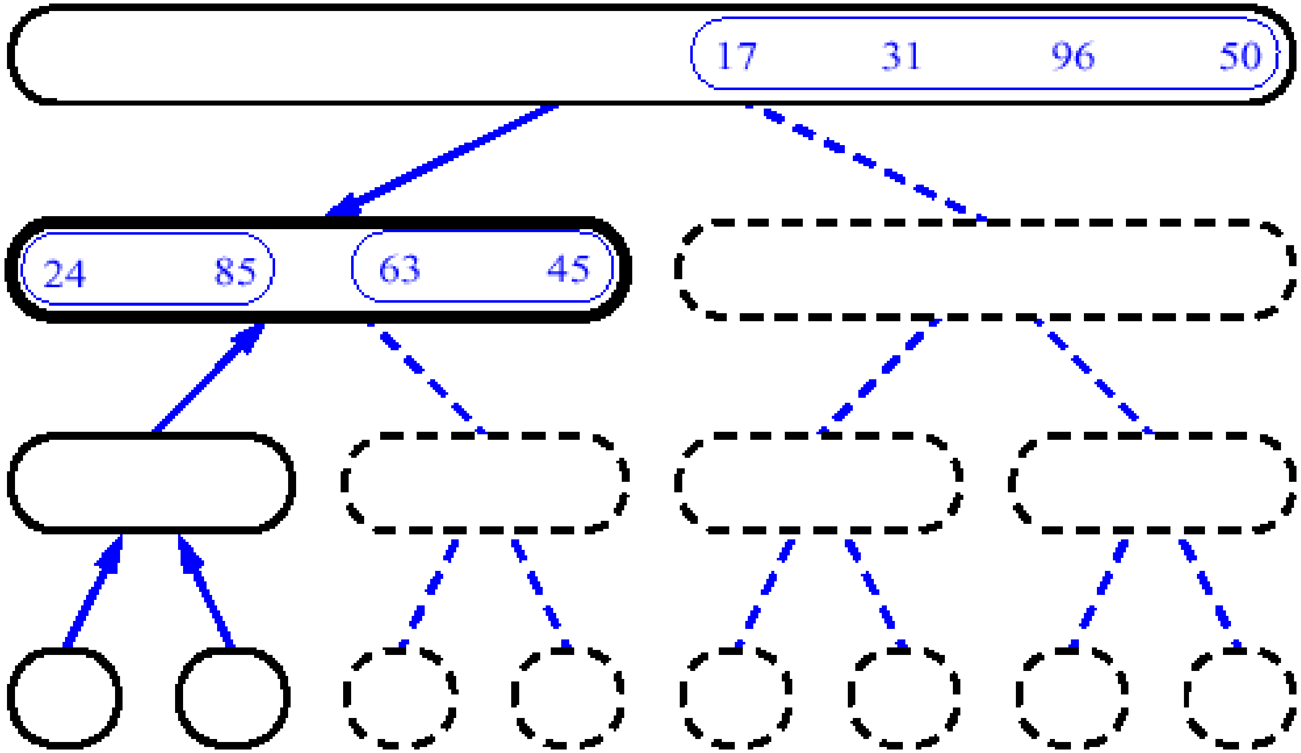
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 8



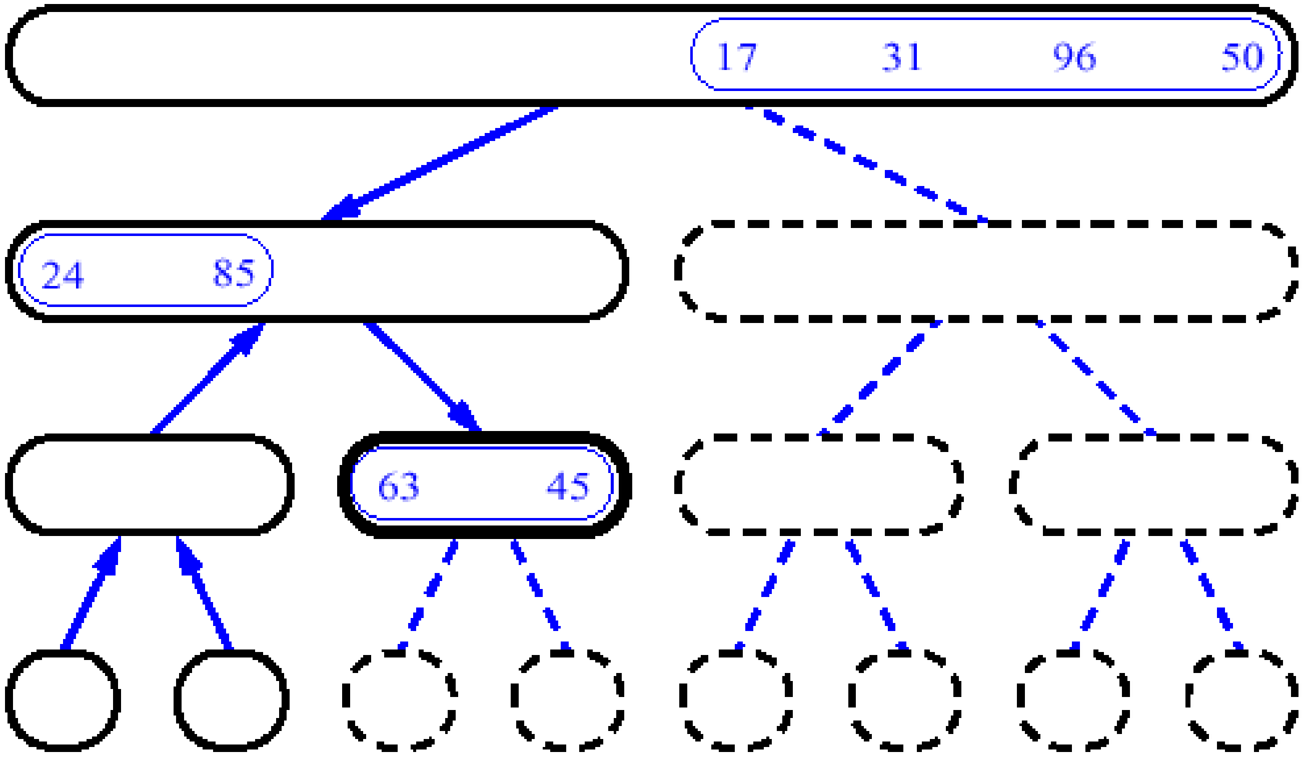
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 9



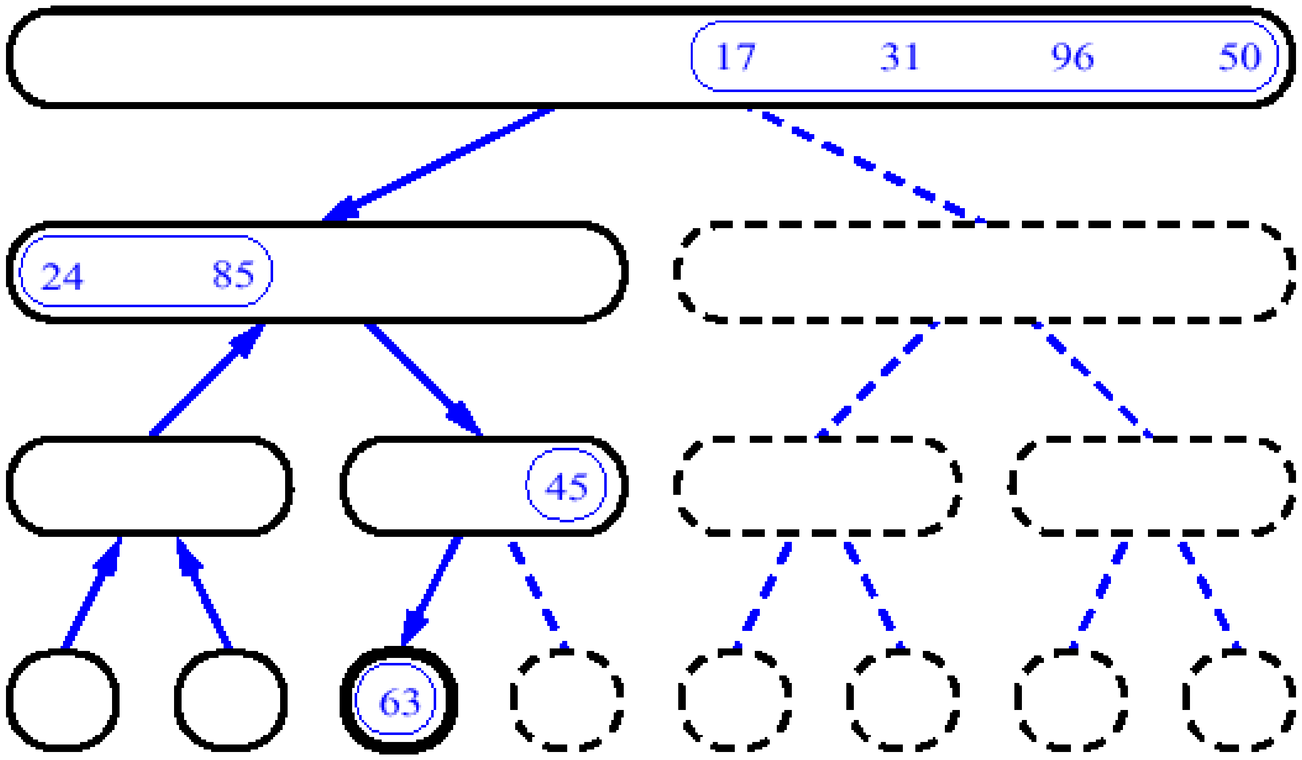
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 10



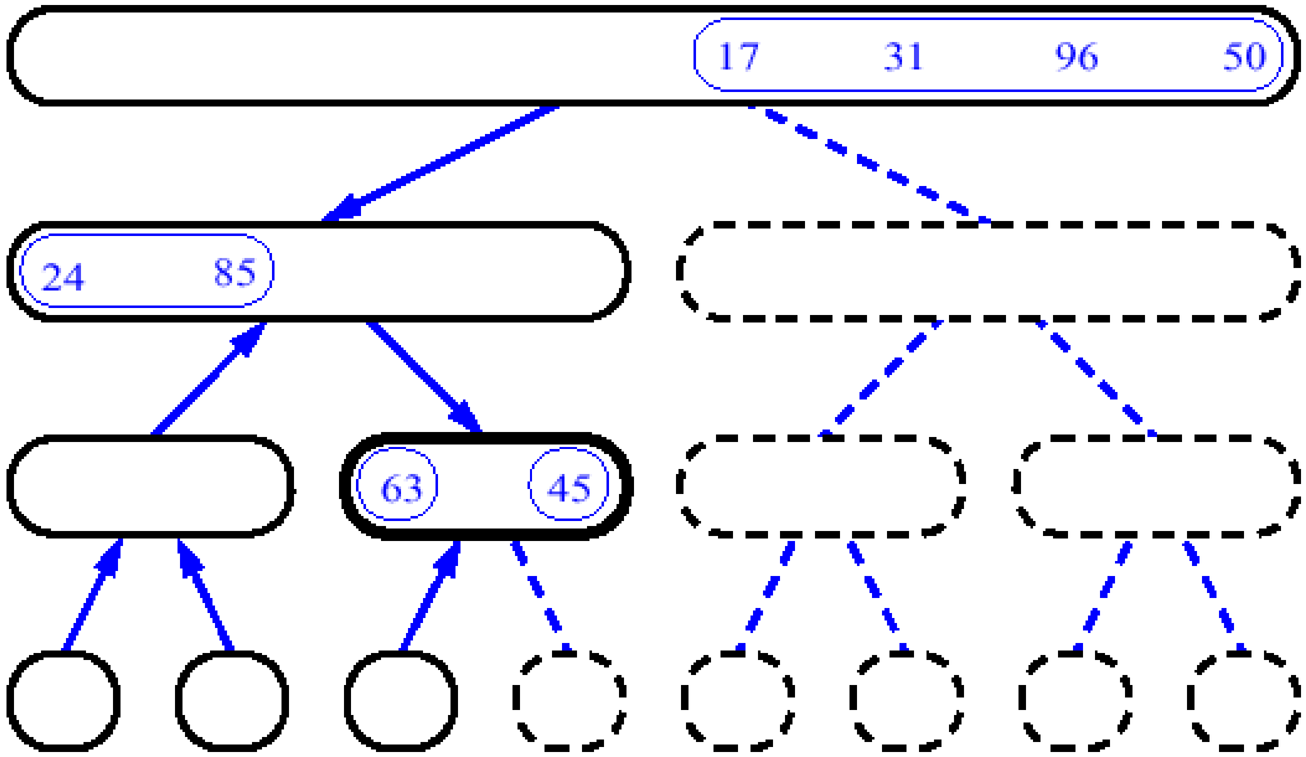
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 11



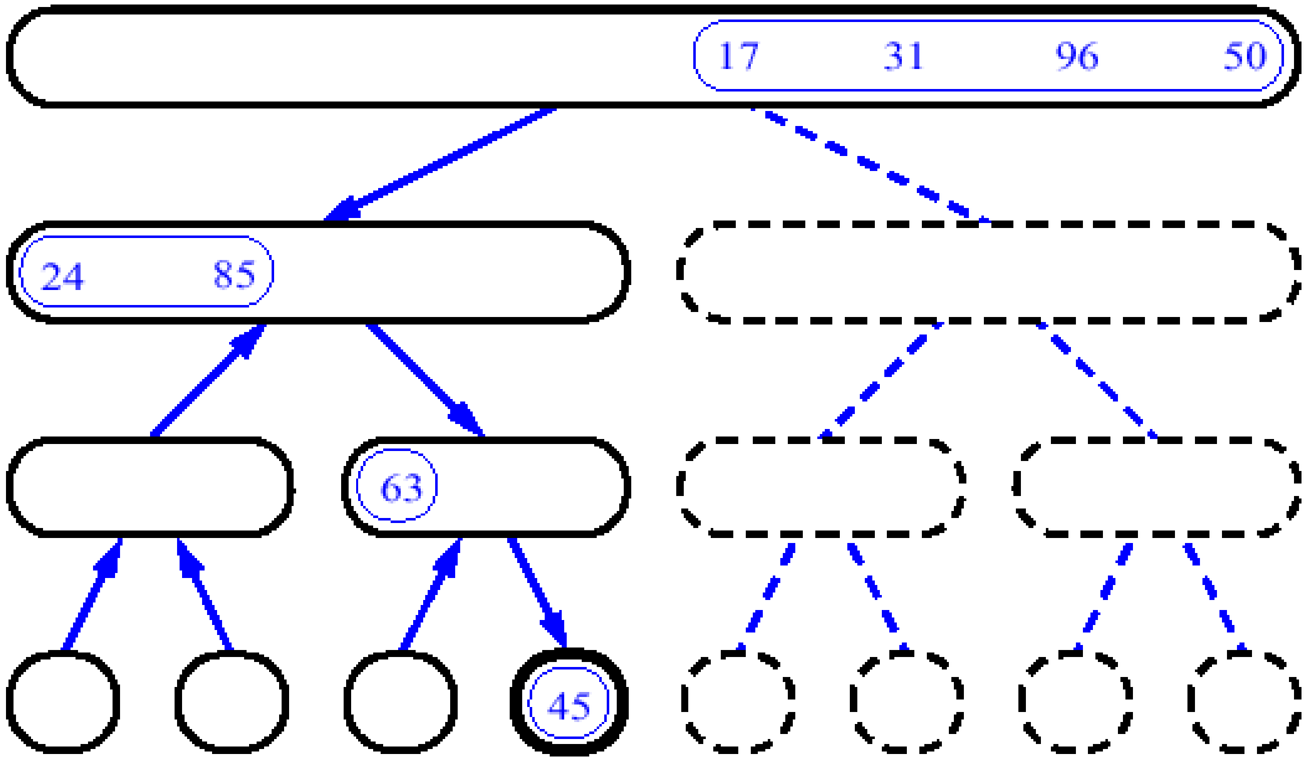
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 12



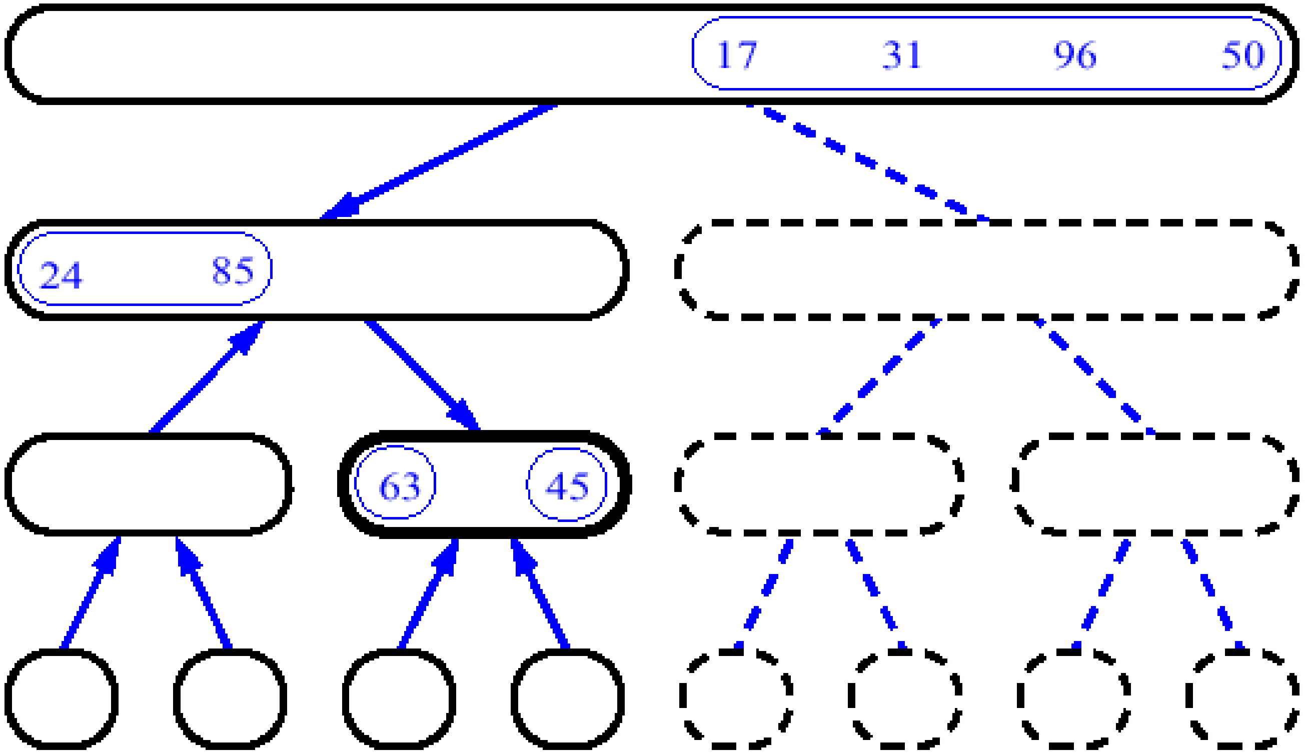
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 13



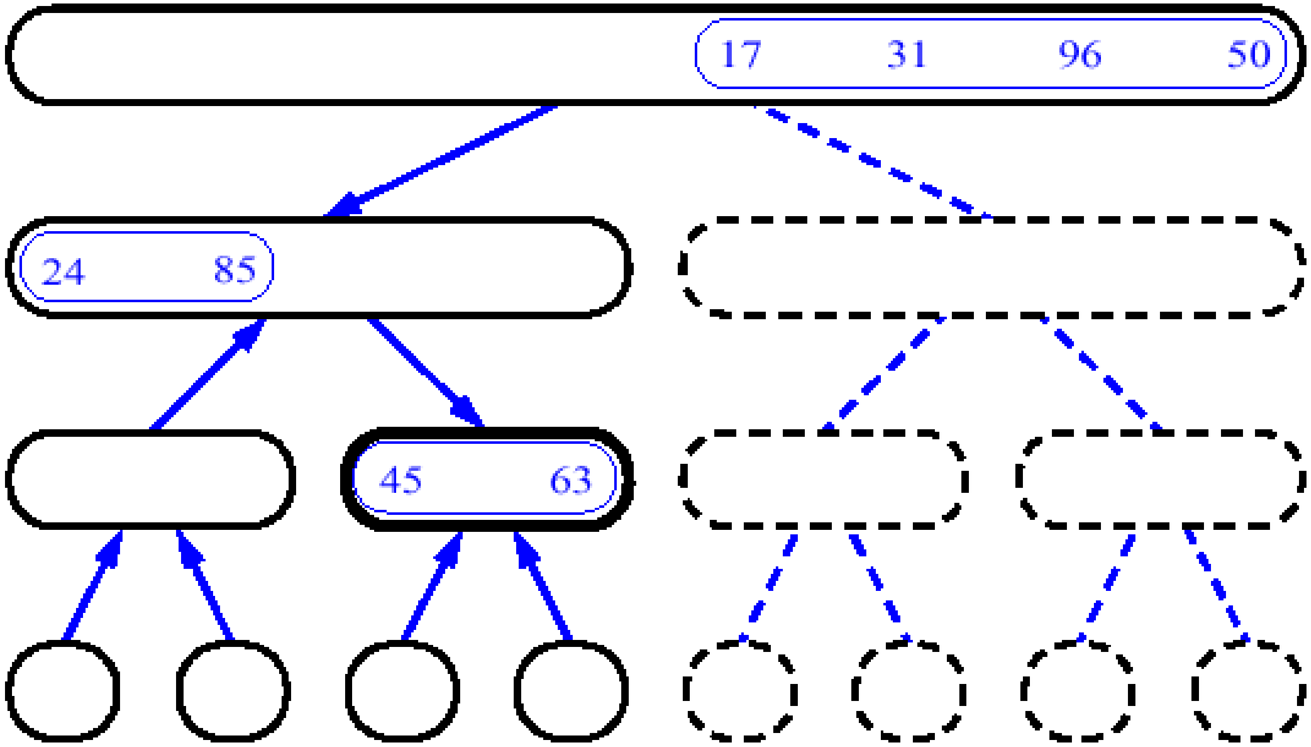
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 14



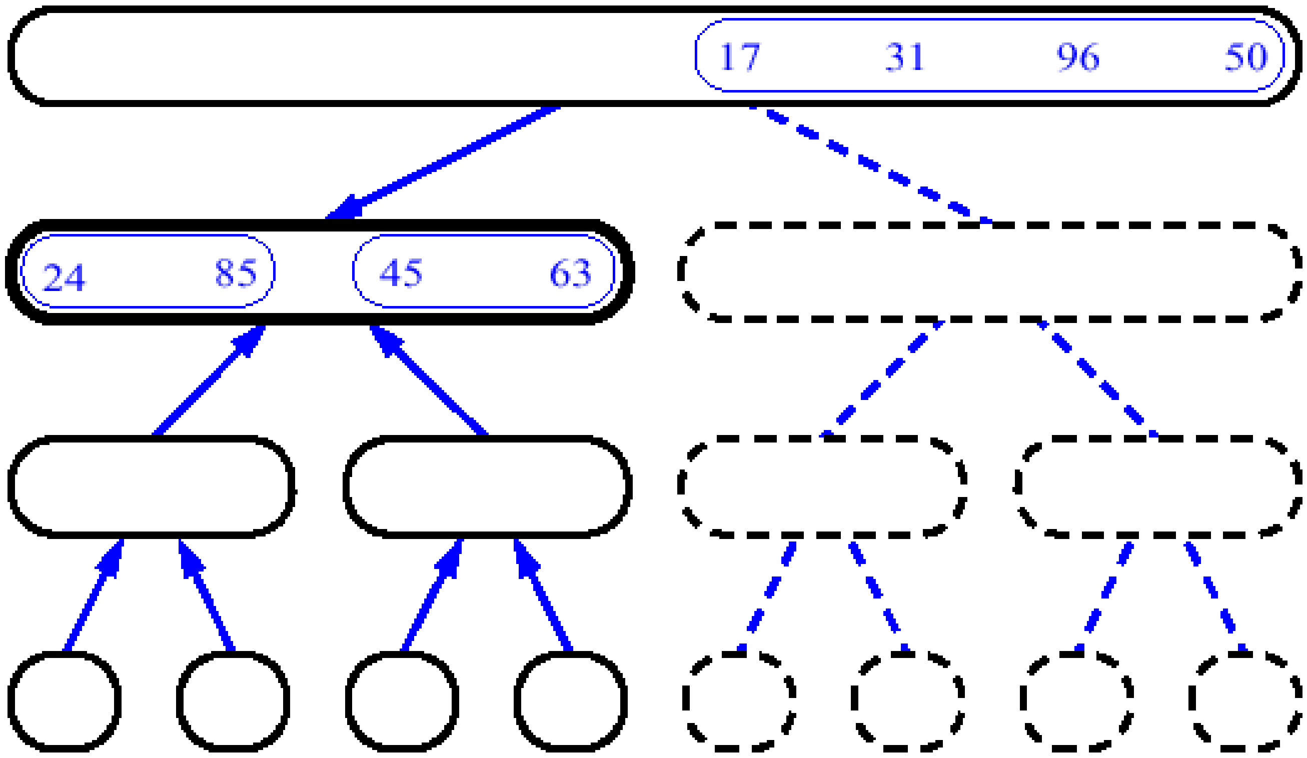
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 15



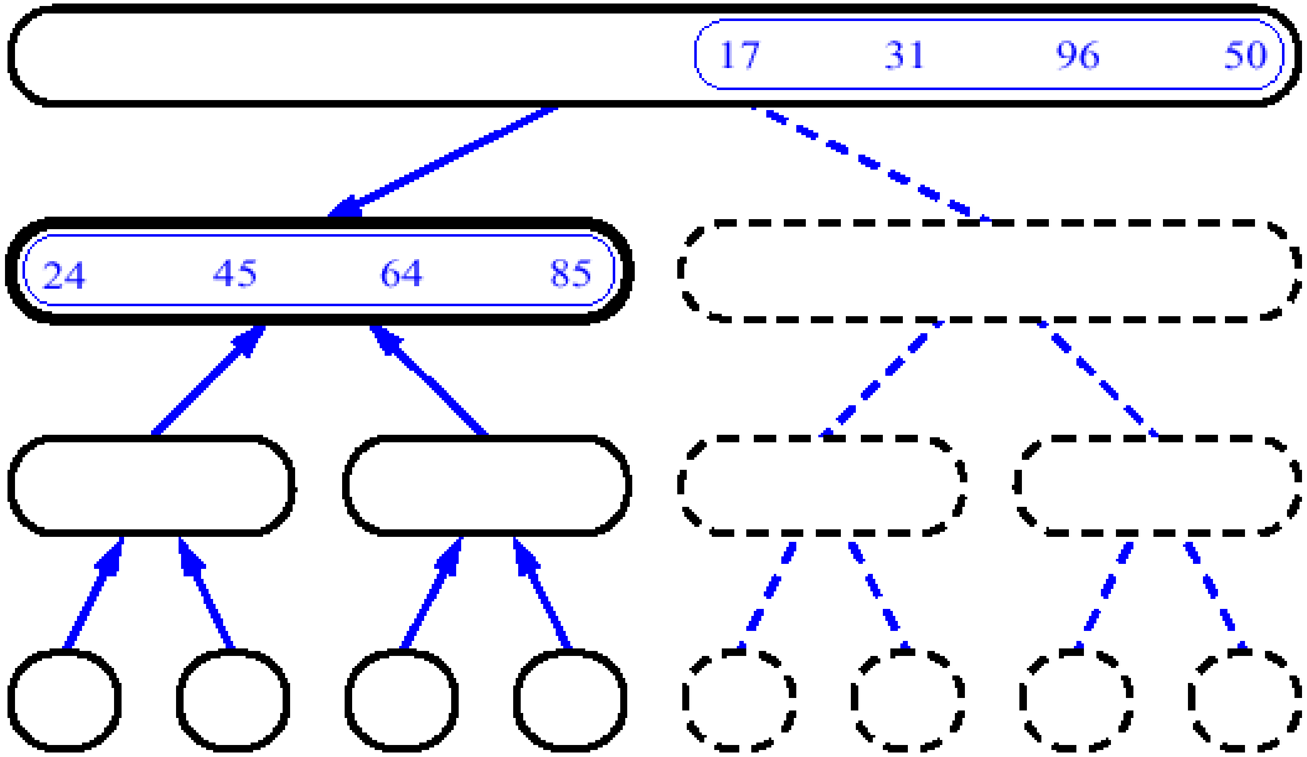
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 16



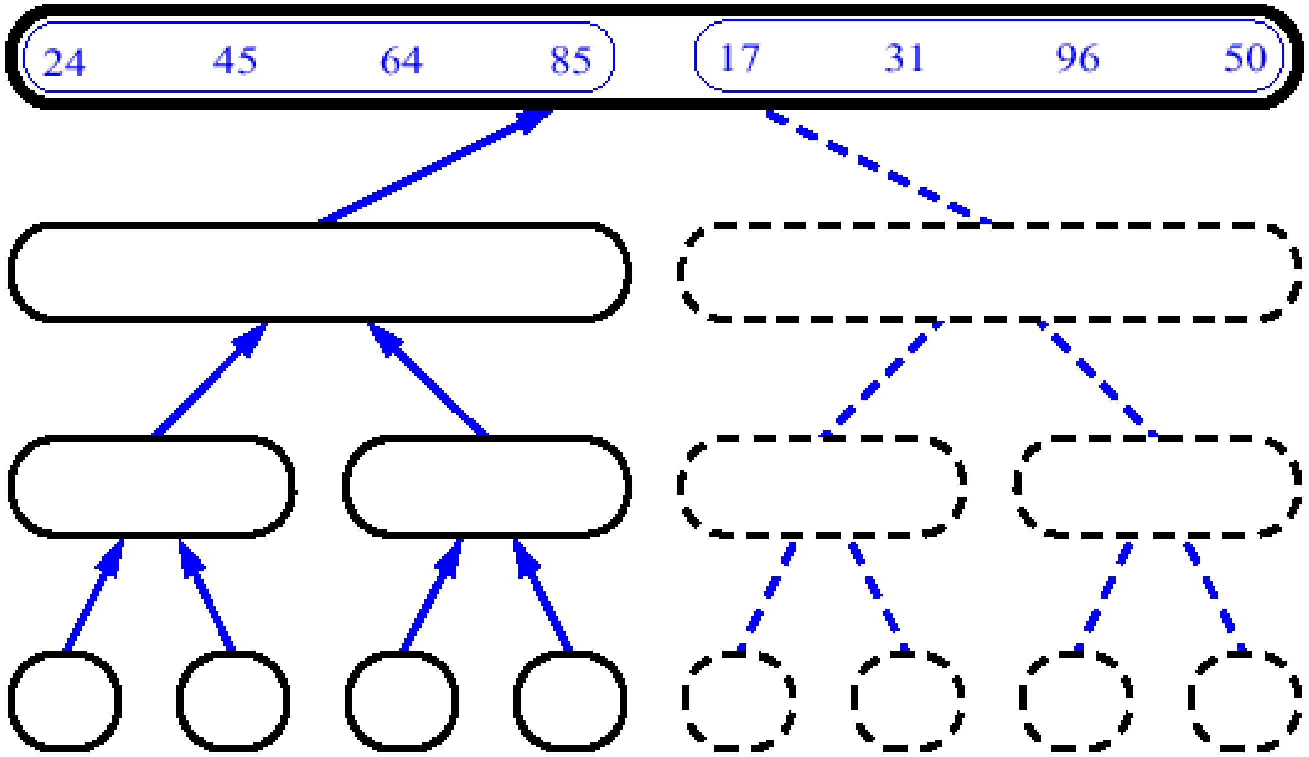
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 17



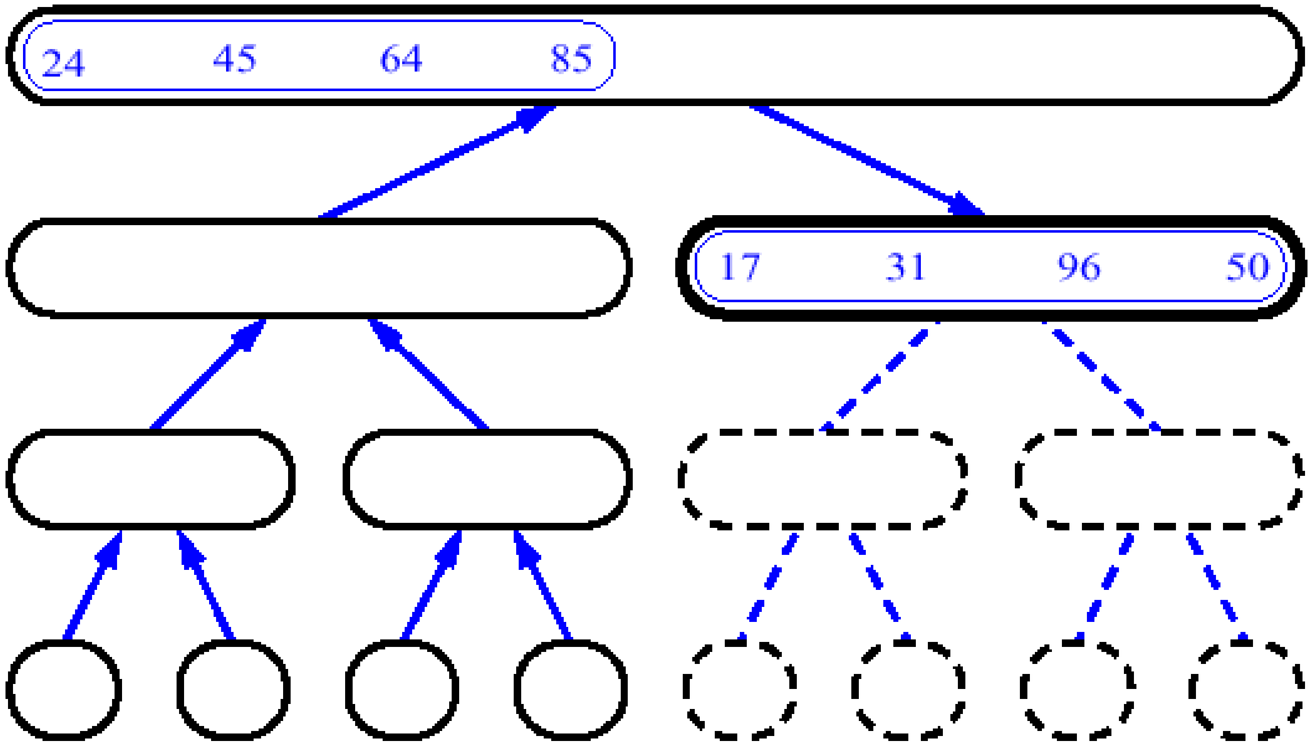
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 18



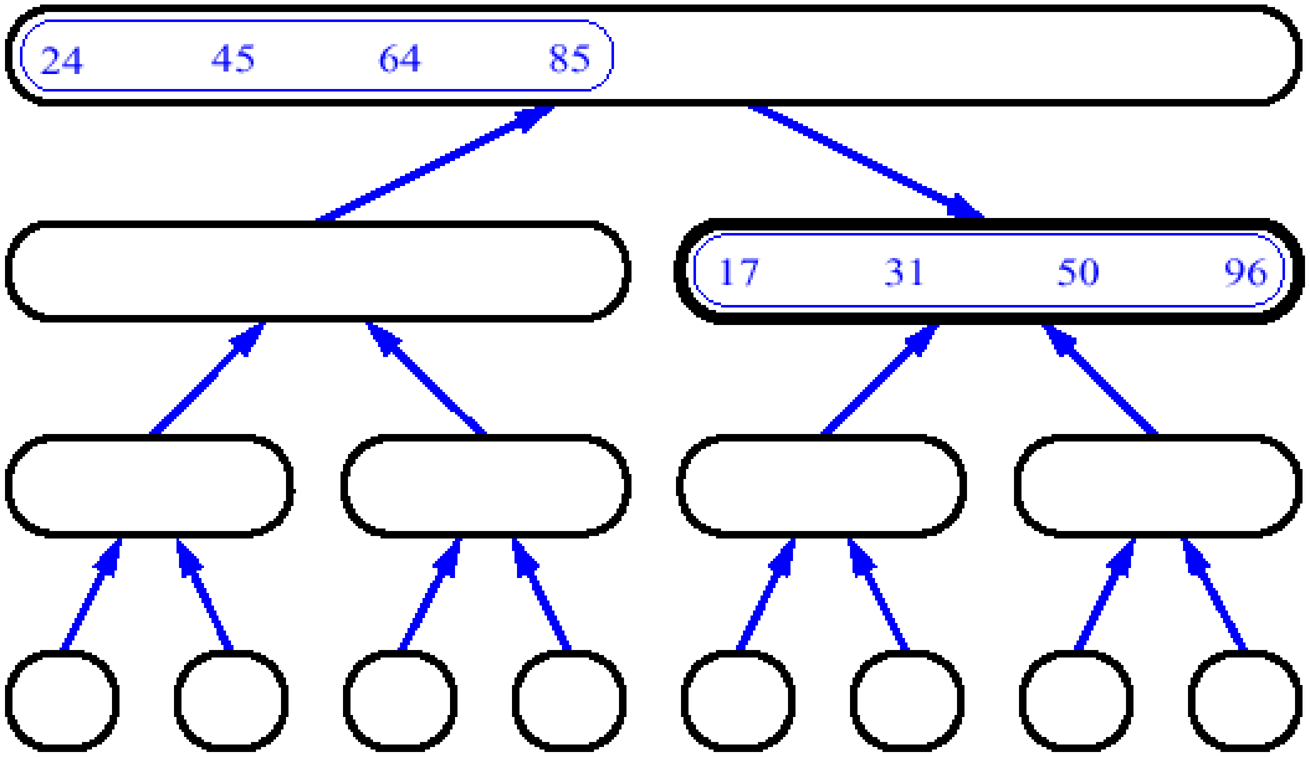
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 19



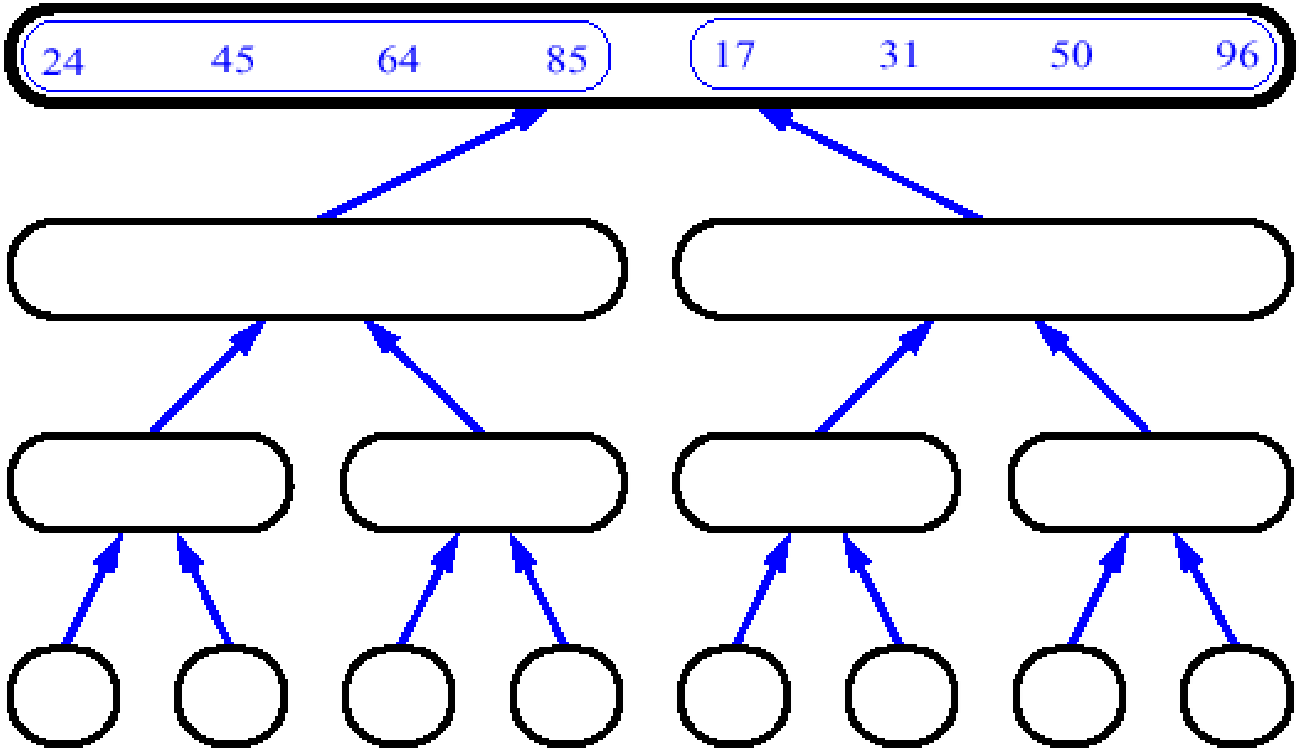
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 20



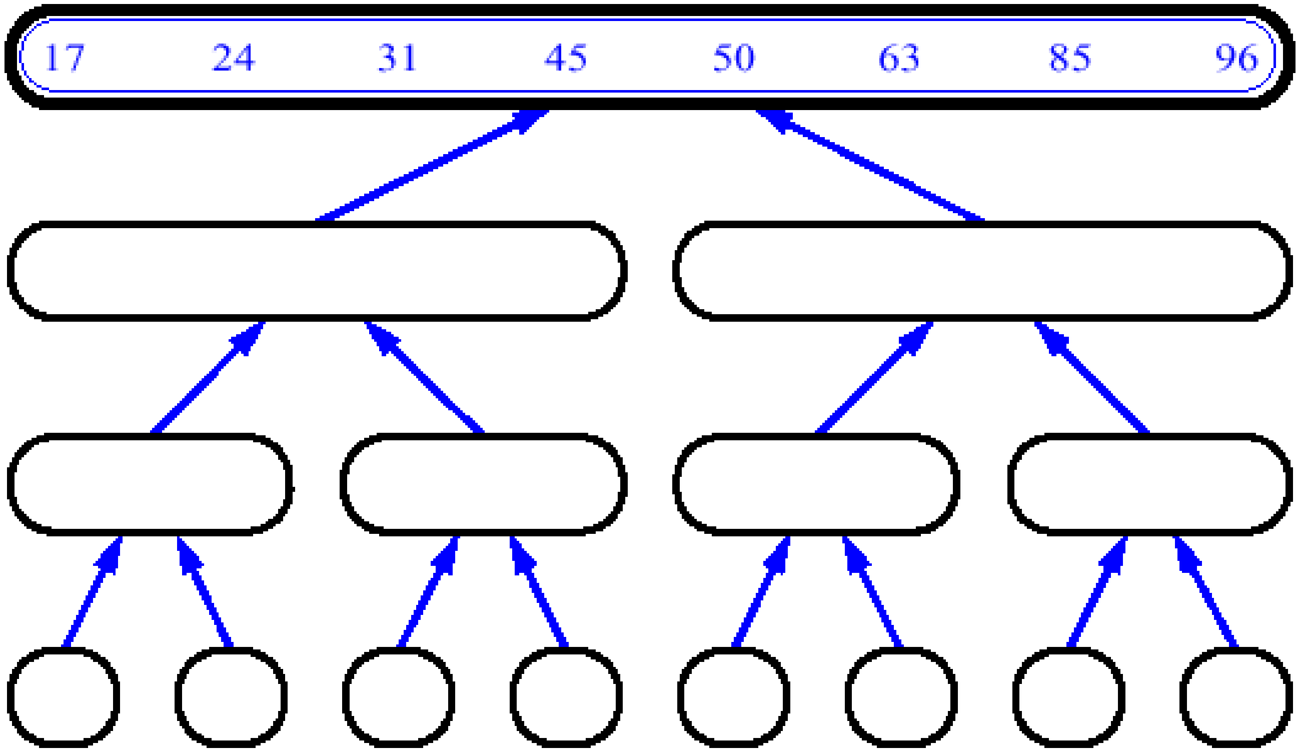
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 21



The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Merge Sort (Example) - 22



https://liveexample.pearsoncmg.com/dsanimation13ejava/MergeSorteBook.html

Running time of Merge Sort

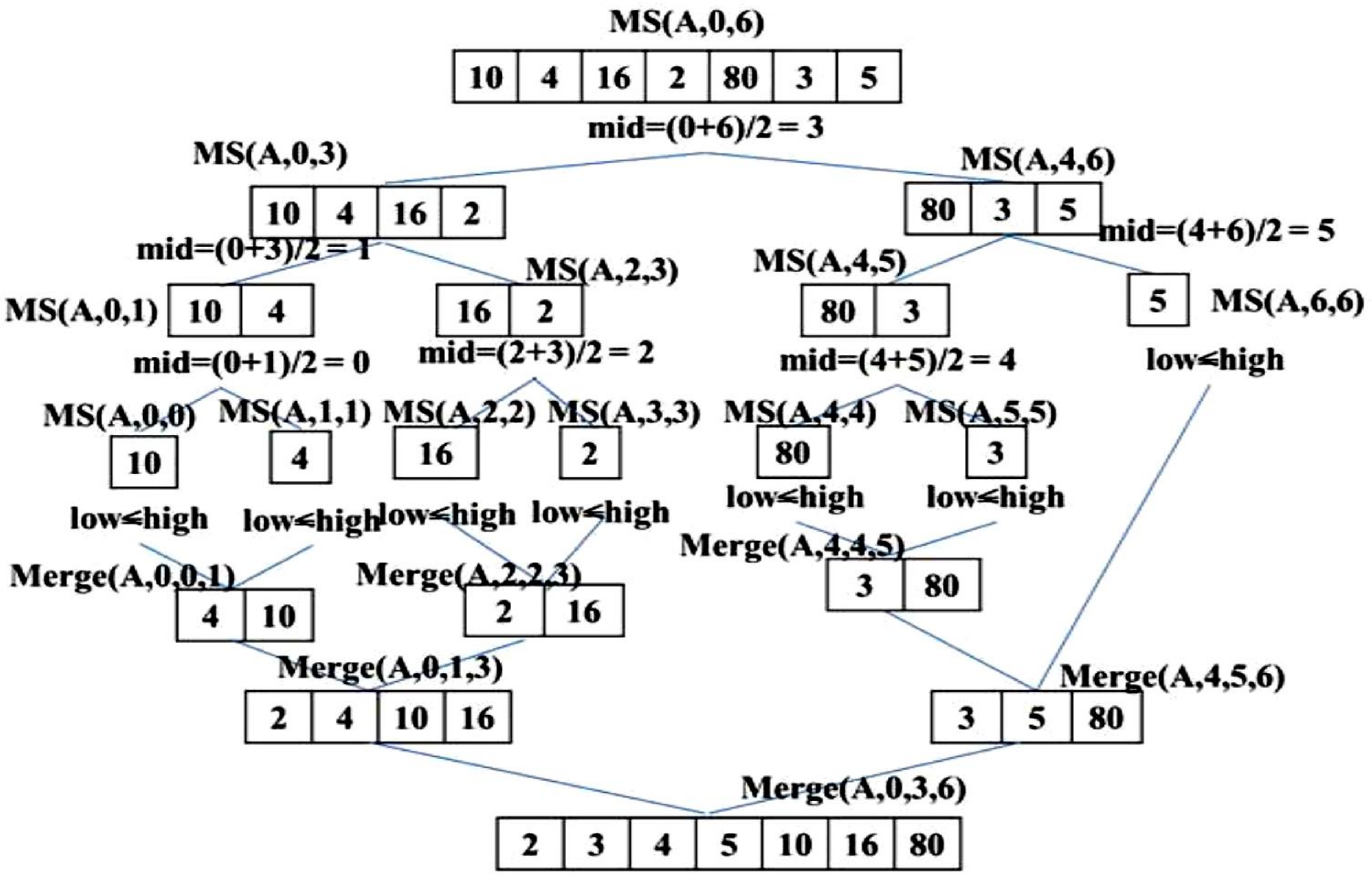
- To sort n numbers
  - if n=1…. done
  - recursively sort 2 lists of numbers (n/2) and (n/2) elements
  - Merge 2 sorted lists in O(n) time
- Again the running time can be expressed as a recurrence:

$$T(n) = \begin{cases} \text{solving\_trivial\_problem} & \text{if } n = 1 \\ \text{num\_pieces } T(n / \text{subproblem\_size\_factor}) + \text{dividing} + \text{combining} & \text{if } n > 1 \end{cases}$$

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Running time of Merge Sort

- Worst case and best case complexity same as average case complexity.
- Applications
  - Sorting linked list


MS(A,0,6)
10 4 16 2 80 3 5
mid=(0+6)/2 = 3
MS(A,0,3)
10 4 16 2
MS(A,4,6)
80 3 5
mid=(4+6)/2 = 5
mid=(0+3)/2 = 1
MS(A,2,3)
MS(A,4,5)
MS(A,0,1)
10 4
16 2
80 3
5
MS(A,6,6)
mid=(0+1)/2 = 0
mid=(2+3)/2 = 2
mid=(4+5)/2 = 4
low≤high
MS(A,0,0)
MS(A,1,1)
MS(A,2,2)
MS(A,3,3)
MS(A,4,4)
MS(A,5,5)
10
4
16
2
80
3
low≤high
low≤high
low≤high
low≤high
low≤high
low≤high
Merge(A,4,4,5)
Merge(A,0,0,1)
Merge(A,2,2,3)
3 80
4 10
2 16
Merge(A,0,1,3)
Merge(A,4,5,6)
2 4 10 16
3 5 80
Merge(A,0,3,6)
2 3 4 5 10 16 80

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Quicksort

- Also called ***partition exchange sort.***
- Fastest known sorting algorithm in practice.
- The array A[p..r] is partitioned into two non-empty subarrays A[p..q] and A[q+1..r]
- Unlike merge sort, no combining step

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Quicksort … contd

```
Quicksort(A, lb, ub)
    if (lb < ub) then
        pivot ← Partition(A, lb, ub)
        Quicksort(A, lb, pivot-1)
        Quicksort(A, pivot+1, ub)
```

- To sort an entire array A, initially lb is 0/1 and ub is n-1/n.

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Quicksort … contd

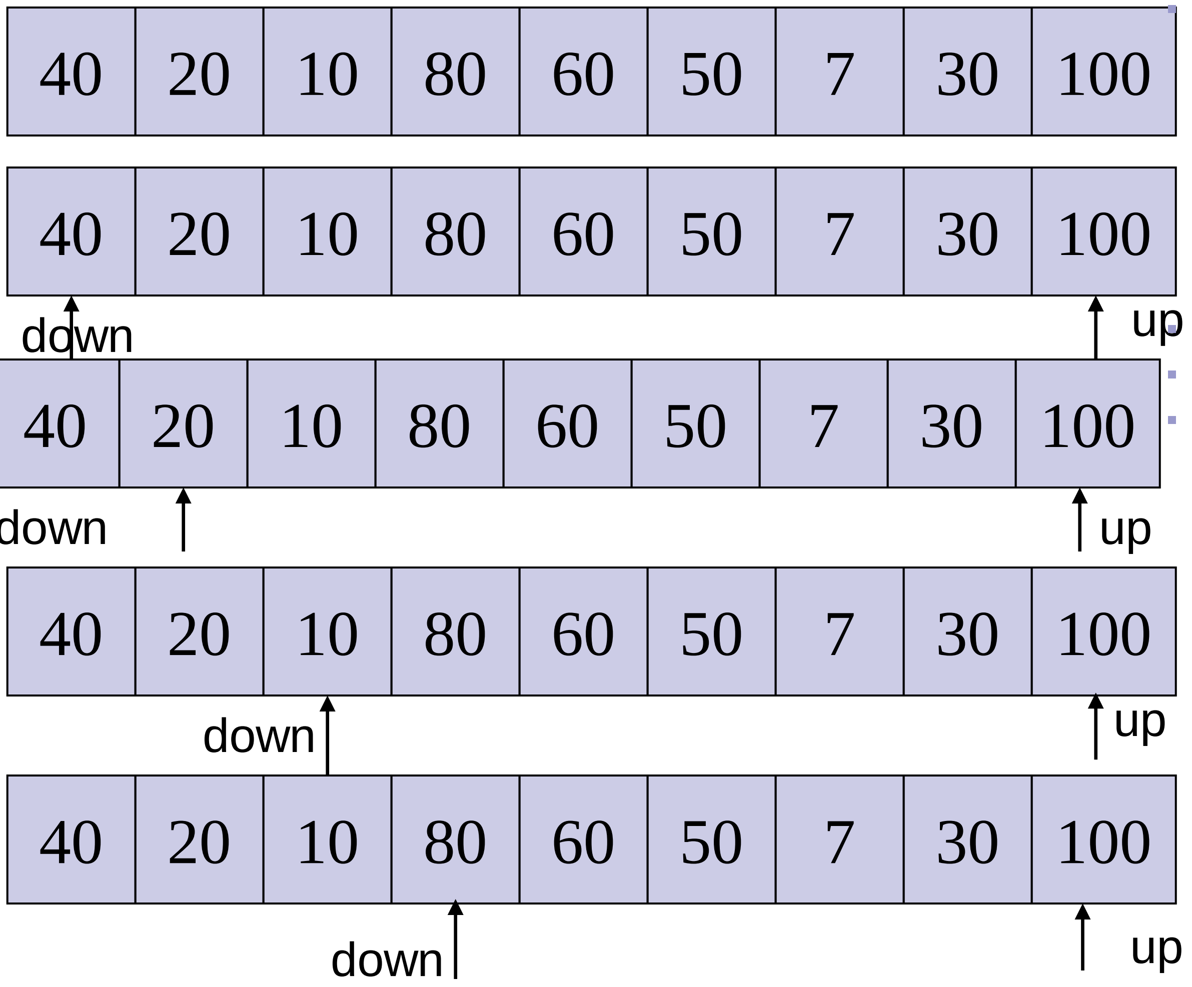
- Partition(A, lb, ub)

```
x ← A[lb]
up ← ub
down ← lb
while (down < up) do
    while (A[down] ≤ x and down < ub)
        down ← down +1
    while (A[up] > x)
        up ← up – 1
    if (down < up)
        Swap (A[down] , A[up])
A[lb] ← A[up]
A[up] ← x
Return up
```

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

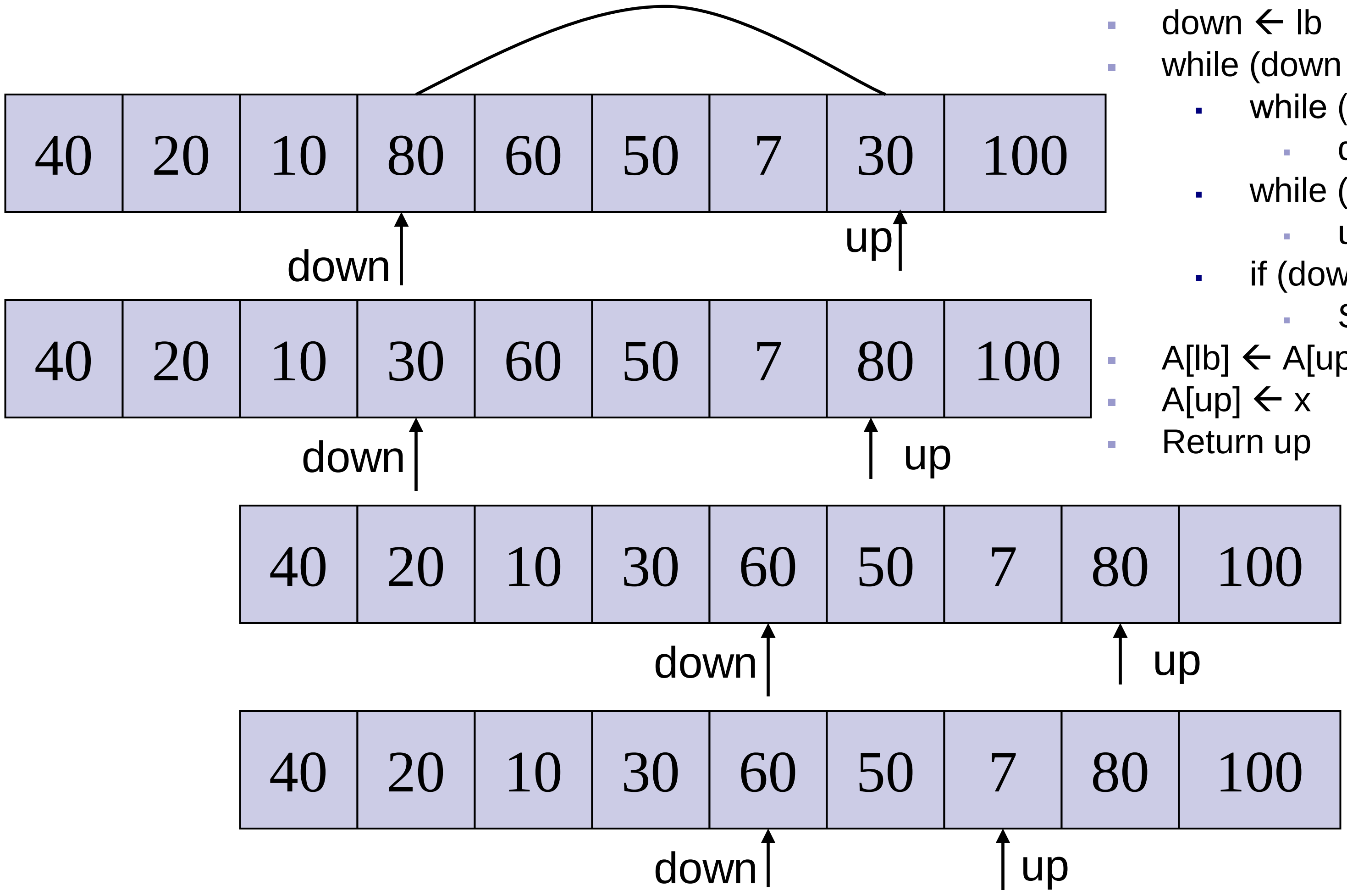
Quicksort … contd

```
Partition(A, lb, ub)
    x ← A[lb]
    up ← ub
    down ← lb
    while (down < up) do
        while (A[down] ≤ x and down < ub)
            down ← down +1
        while (A[up] > x)
            up ← up – 1
        if (down < up)
            Swap (A[down], A[up])
    A[lb] ← A[up]
    A[up] ← x
    Return up
```



The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

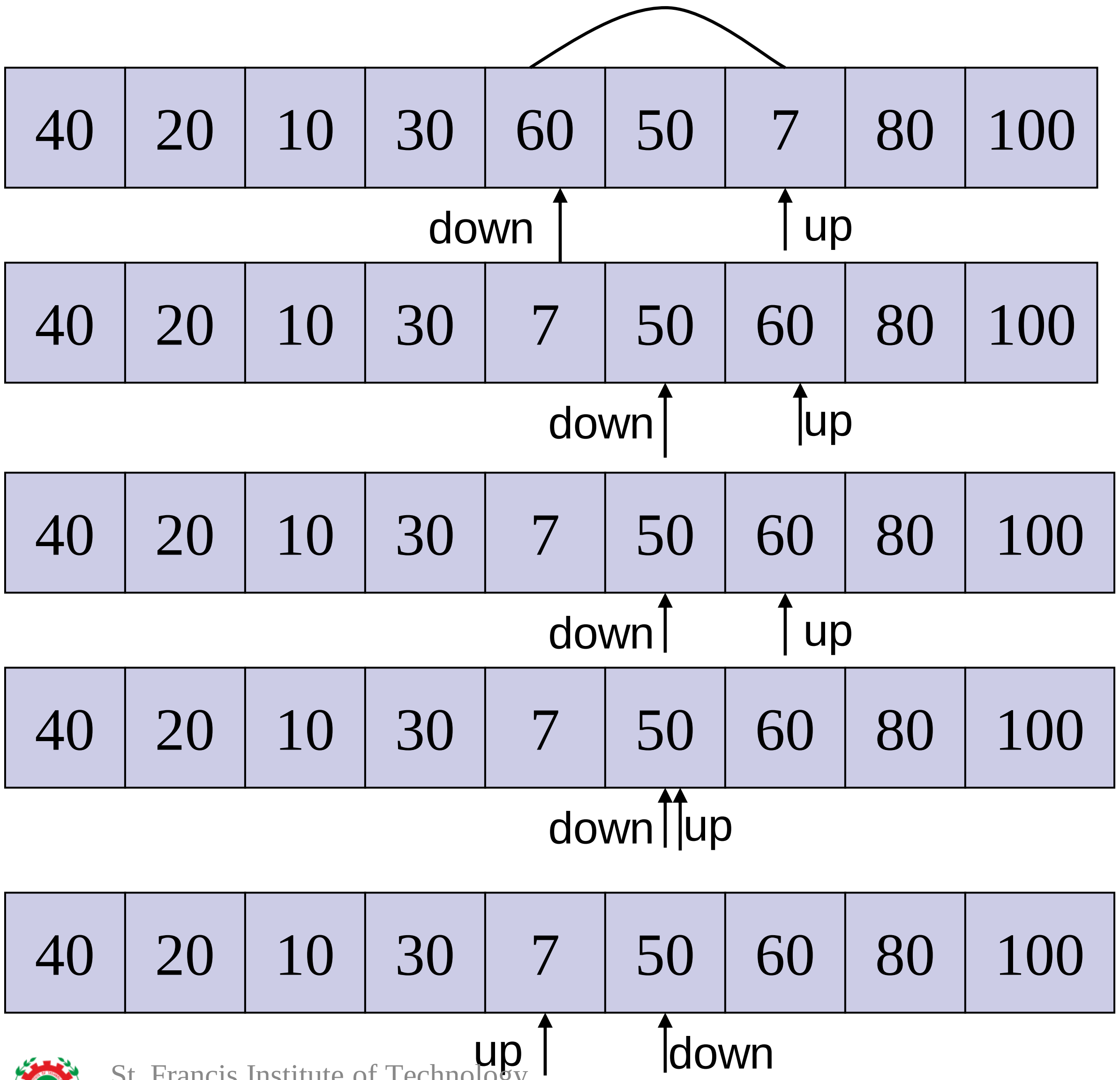
Quicksort …contd



```
Partition(A, lb, ub)
    x ← A[lb]
    up ← ub
    down ← lb
    while (down < up) do
        while (A[down] ≤ x and down < ub
            down ← down +1
        while (A[up] > x)
            up ← up – 1
        if (down < up)
            Swap (A[down], A[up])
    A[lb] ← A[up]
    A[up] ← x
    Return up
```

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

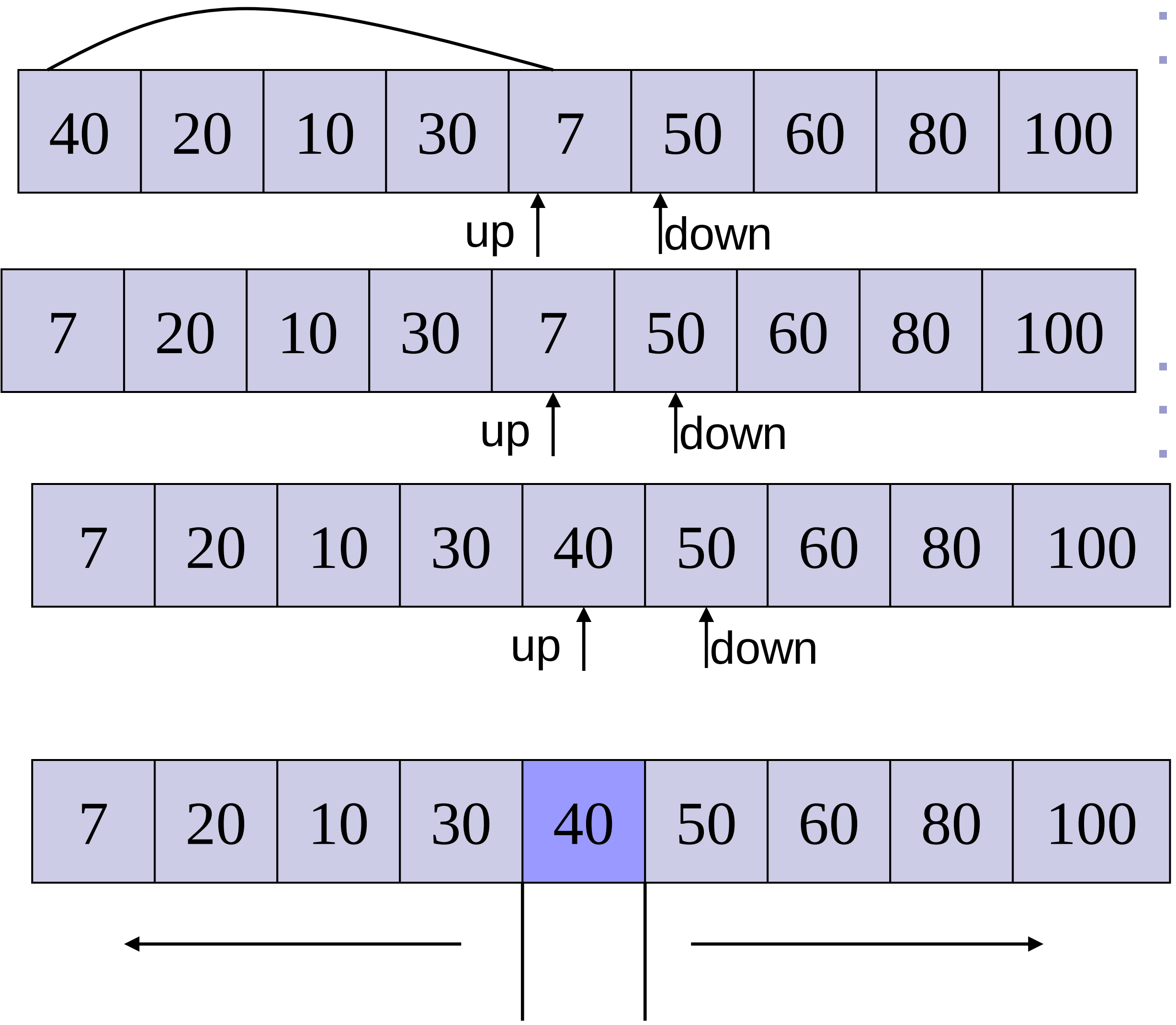
Quicksort …contd



```
Partition(A, lb, ub)
    x ← A[lb]
    up ← ub
    down ← lb
    while (down < up) do
        while (A[down] ≤ x and down < ub
            down ← down +1
        while (A[up] > x)
            up ← up – 1
        if (down < up)
            Swap (A[down], A[up])
    A[lb] ← A[up]
    A[up] ← x
    Return up
```

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Quicksort … contd



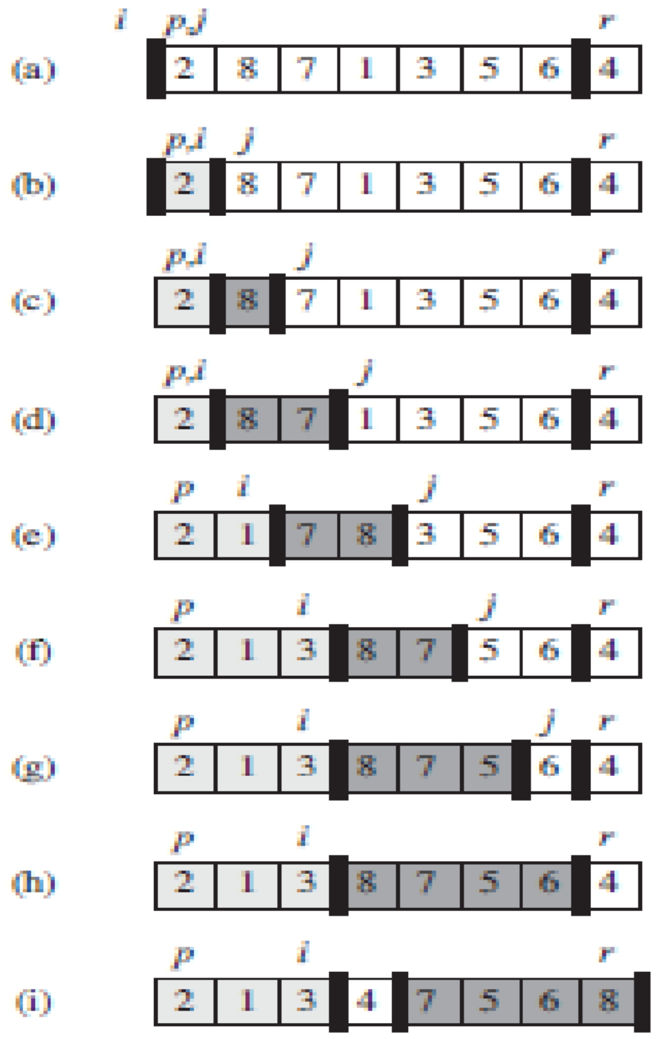
```
Partition(A, lb, ub)
    x ← A[lb]
    up ← ub
    down ← lb
    while (down < up) do
        while (A[down] ≤ x and down < ub
            down ← down +1
        while (A[up] > x)
            up ← up – 1
        if (down < up)
            Swap (A[down], A[up])
    A[lb] ← A[up]
    A[up] ← x
    Return up
```

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Quicksort… alternate algorithm

```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```


(a) i p,j r 2 8 7 1 3 5 6 4
(b) p,i j r 2 8 7 1 3 5 6 4
(c) p,i j r 2 8 7 1 3 5 6 4
(d) p,i j r 2 8 7 1 3 5 6 4
(e) p i j r 2 1 7 8 3 5 6 4
(f) p i j r 2 1 3 8 7 5 6 4
(g) p i j r 2 1 3 8 7 5 6 4
(h) p i r 2 1 3 8 7 5 6 4
(i) p i r 2 1 3 4 7 5 6 8

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Quicksort … Analysis

- For analyzing quicksort, we count only number of element comparisons.
- Running time
  - pivot selection: constant time, i.e. O(1)
  - partitioning: linear time, i.e. O(N)
  - running time of the two recursive calls
- $T(n) = T(i)+T(n-i-1)+cn$
  - *c* is a constant
  - *i* is the number of elements in the partition.

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Quicksort… Analysis

- Worst case
  - Complexity O(n^2)
- Average Case
  - Complexity O(nlogn)
- Applications
  - Commercial Applications

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Randomized Quicksort

- *Choose the pivot randomly* (or randomly permute the input array before sorting).
- The running time of the algorithm is independent of input ordering.
- No specific input elicits worst case behavior.
  - The worst case depends on the random number generator.
- Helps modify Quicksort so that it performs well on every input.

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Randomized Quicksort

```
R_Quicksort(A, lb, ub)
    if (lb < ub) then
        i ← Random(lb, ub)
        Swap A[up] ↔A[i]
        pivot ← Partition(A, lb, ub)
        Quicksort(A, lb, pivot-1)
        Quicksort(A, pivot+1, ub)
```

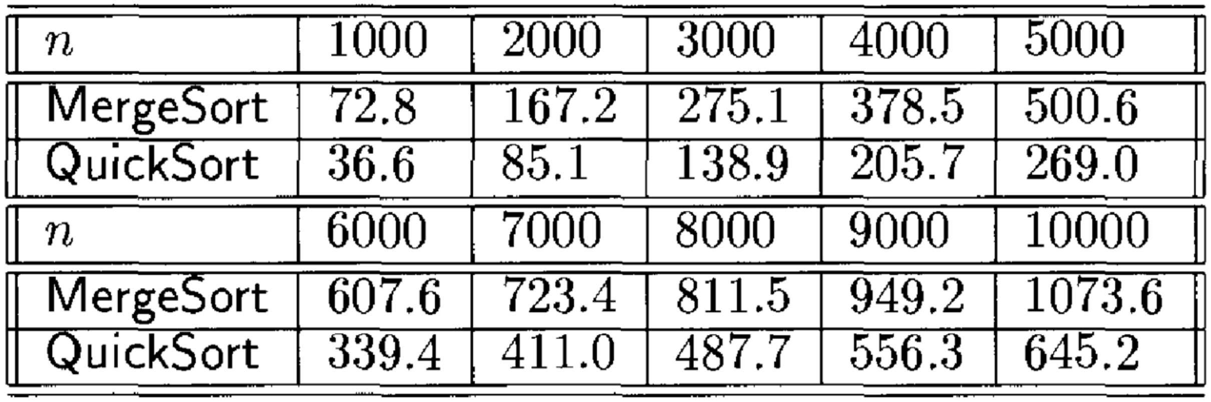
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Randomized Quicksort … Analysis

- Assumptions: all elements are distinct
- All partitions from 0:n-1 and n-1:0 equally likely
- Probability of each partition is 1/n
- Average case complexity is O(nlogn).

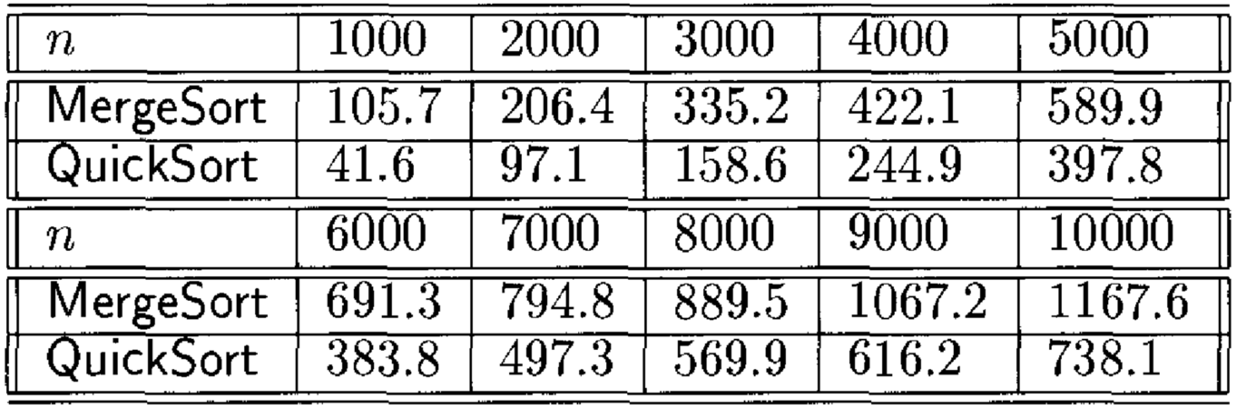
The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Posteriori Analysis



n	1000	2000	3000	4000	5000
MergeSort	72.8	167.2	275.1	378.5	500.6
QuickSort	36.6	85.1	138.9	205.7	269.0
n	6000	7000	8000	9000	10000
MergeSort	607.6	723.4	811.5	949.2	1073.6
QuickSort	339.4	411.0	487.7	556.3	645.2

Average Sorting time on Random inputs



n	1000	2000	3000	4000	5000
MergeSort	105.7	206.4	335.2	422.1	589.9
QuickSort	41.6	97.1	158.6	244.9	397.8
n	6000	7000	8000	9000	10000
MergeSort	691.3	794.8	889.5	1067.2	1167.6
QuickSort	383.8	497.3	569.9	616.2	738.1

Average Sorting time on Worst inputs

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Why Priori Analysis???

- Telephone directory for mobile users in India
  - 1 billion ≈ 10^9 phones
- n^2 sorting algorithm requires 10^{18} operations
- CPU cycle: 10^8 operations per second i.e. 10^{10} seconds
  - 2778000 hours → 11570 days → 300 years
- $n\log_2 n$ algorithm takes only 3 x 10^{10} operations
  - 300 seconds or 5 minutes

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Binary Search

- Basics
  - Sequential search not very efficient for large lists.
  - Binary search faster
  - Works only on sorted list
  - E.g. searching for some number in a phone book based on person’s name.
- Logic: compares the target (search key) with the middle element of the array and terminates if it is the same, else it divides the array into two sub arrays and continues the search in left/right sub array if the target is less/greater than the middle element of the array.

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Binary Search … Recursive

```
Bin_Search(A, target, low, high)
    If (high < low)
        return not found
    mid ← (low + high)/ 2
    if (A[mid] > target)
        return Bin_Search(A, target, low, mid-1)
    if (A[mid] < target)
        return Bin_Search(A, target, mid+1, high)
    else
        return mid
```

−15, −6, 0, 7, 9, 23, 54, 82, 101, 112, 125, 131, 142, 151

$x = 151$	*low*	*high*	*mid*
	1	14	7
	8	14	11
	12	14	13
	14	14	14
			found

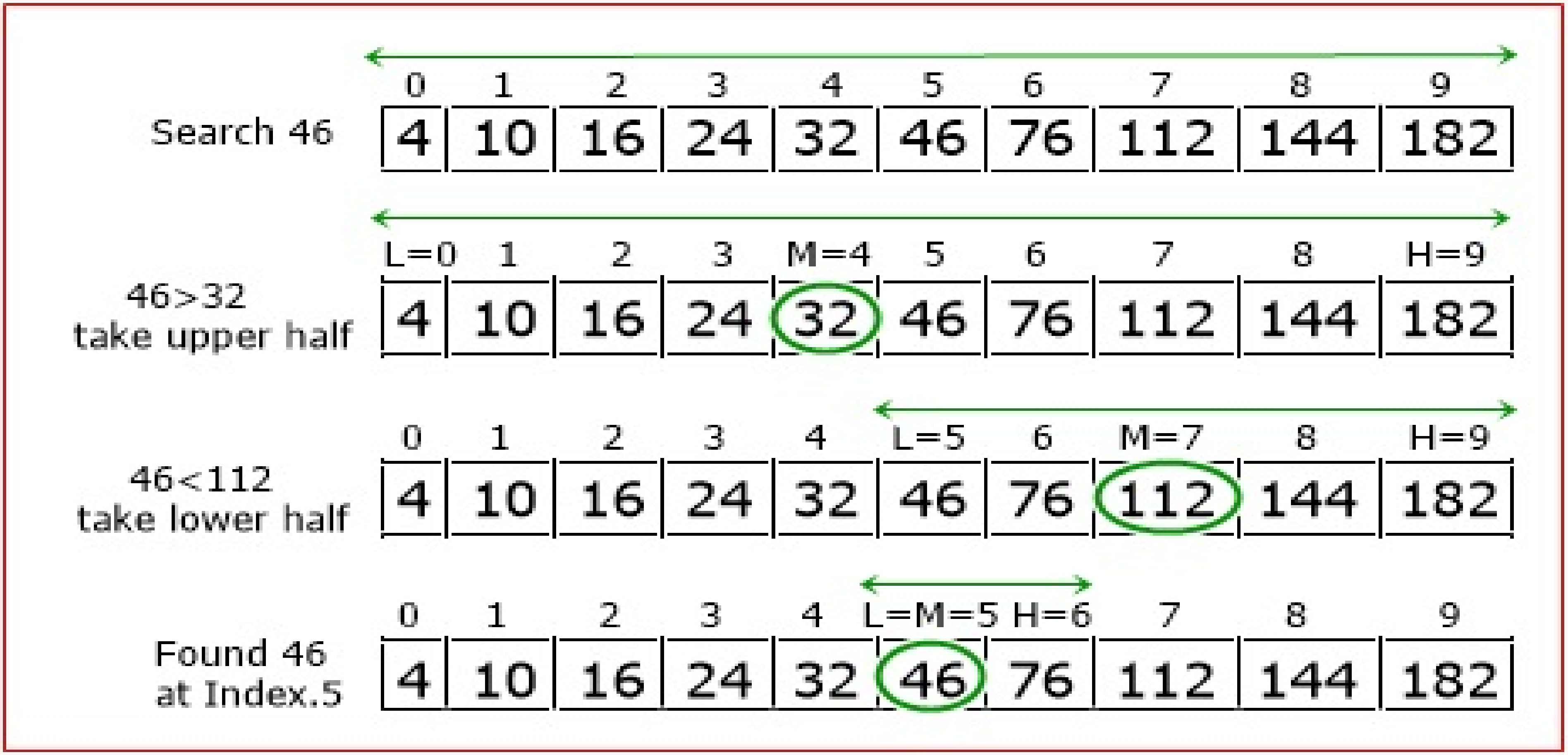
$x = -14$	*low*	*high*	*mid*
	1	14	7
	1	6	3
	1	2	1
	2	2	2
	2	1	not found

$x = 9$	*low*	*high*	*mid*
	1	14	7
	1	6	3
	4	6	5
			found

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Binary Search … Recursive

https://yongdanielliang.github.io/animation/web/BinarySearchNew.html



The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Binary Search …recursive

- Average and worst case complexity is O(log n)
- Easier to implement.
- Note:
  - Successful searches: Best: Θ(1), Worst and Average: $\Theta(\log_2 n)$
  - Unsuccessful searches: Best, Worst and Average: $\Theta(\log_2 n)$

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Binary Search … non recursive

```
Bin_search (A, n, target)
    left ← 1
    right ← n
    While (left ≤ right) do
        mid ← ( left + right )/2
        If (target = A[mid])
            then return mid
        If (target < A[mid])
            then right ← mid -1
        If (target > A[mid])
            then left ← mid+1
    Return element not found
```

$-15, -6, 0, 7, 9, 23, 54, 82, 101, 112, 125, 131, 142, 151$

$x = 151$	*low*	*high*	*mid*
	1	14	7
	8	14	11
	12	14	13
	14	14	14
			found

$x = -14$	*low*	*high*	*mid*
	1	14	7
	1	6	3
	1	2	1
	2	2	2
	2	1	not found

$x = 9$	*low*	*high*	*mid*
	1	14	7
	1	6	3
	4	6	5
			found

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Binary Search … non recursive

- Complexity order same as recursive search i.e. O(log n)
- Vs. Linear Search
  - Linear search is of order O(n).
  - Benefit of binary search over linear search becomes significant for lists over about 100 elements.
  - For smaller lists linear search may be faster because of the speed of the simple increment compared with the divisions needed in binary search.

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Minimum and Maximum in a list

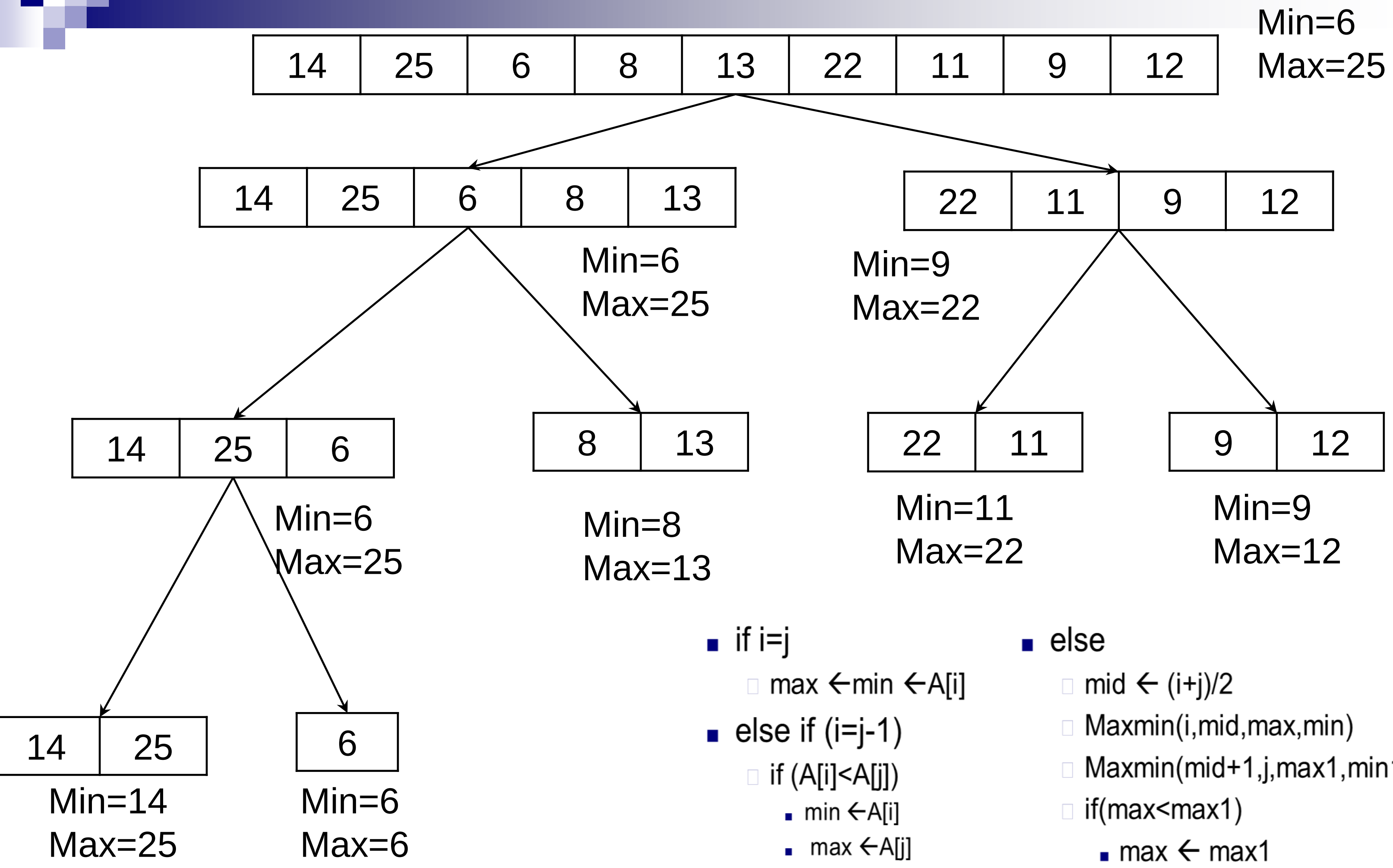
- Find the minimum and maximum in a list of *n* elements
- Divide the problem into n/2 subproblems and find their minimum and maximum recursively
- Combine the solutions

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Maxmin(i,j,max,min)

```
if i=j
    max ← min ← A[i]
else if (i=j-1)
    if (A[i]<A[j])
        min ← A[i]
        max ← A[j]
    else
        max ← A[i]
        min ← A[j]
else
    mid ← (i+j)/2
    Maxmin(i,mid,max,min)
    Maxmin(mid+1,j,max1,min1)
    if(max<max1)
        max ← max1
    if(min>min1)
        min ← min1
```

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.



```
if i=j
    max ←min ←A[i]
else if (i=j-1)
    if (A[i]<A[j])
        min ←A[i]
        max ←A[j]
    else
        max ← A[i]
        min ← A[j]
else
    mid ← (i+j)/2
    Maxmin(i,mid,max,min)
    Maxmin(mid+1,j,max1,min1)
    if(max<max1)
        max ← max1
    if(min>min1)
        min ← min1
```

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Time Complexity

- $T(n) = 2T(n/2) + d \qquad ,n>2$

  $\qquad =2 \text{ or } d \qquad , n=2$

  $\qquad =1 \text{ or } c \qquad ,n=1$
- $T(n) = O(n)$

The material in this presentation belongs to St. Francis Institute of Technology and is solely for educational purposes. Distribution and modifications of the content is prohibited.

Next

- Greedy Method