CS 311: Algorithm Design and Analysis

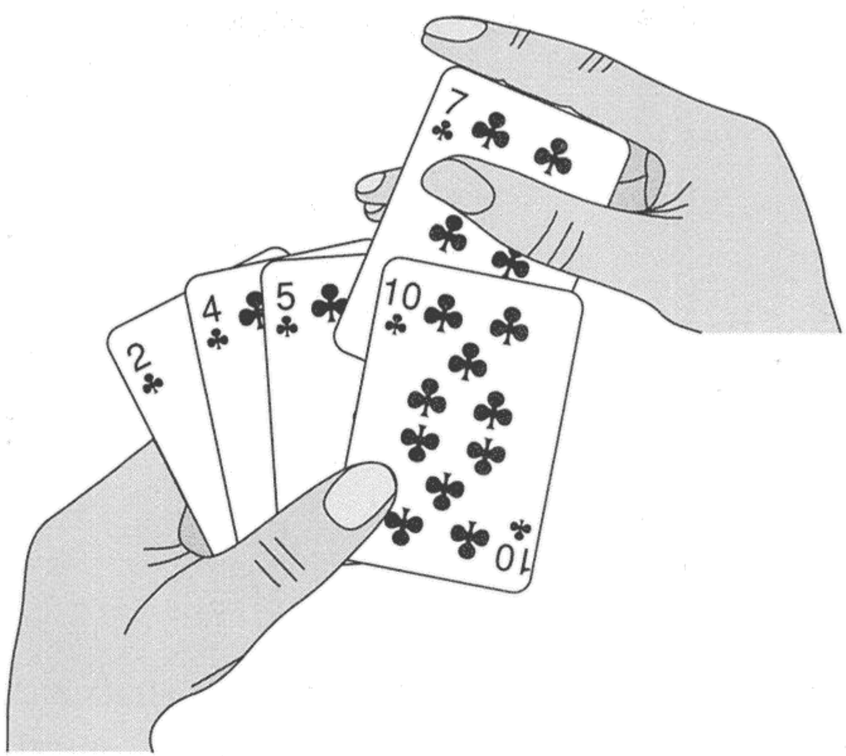
Lecture 3

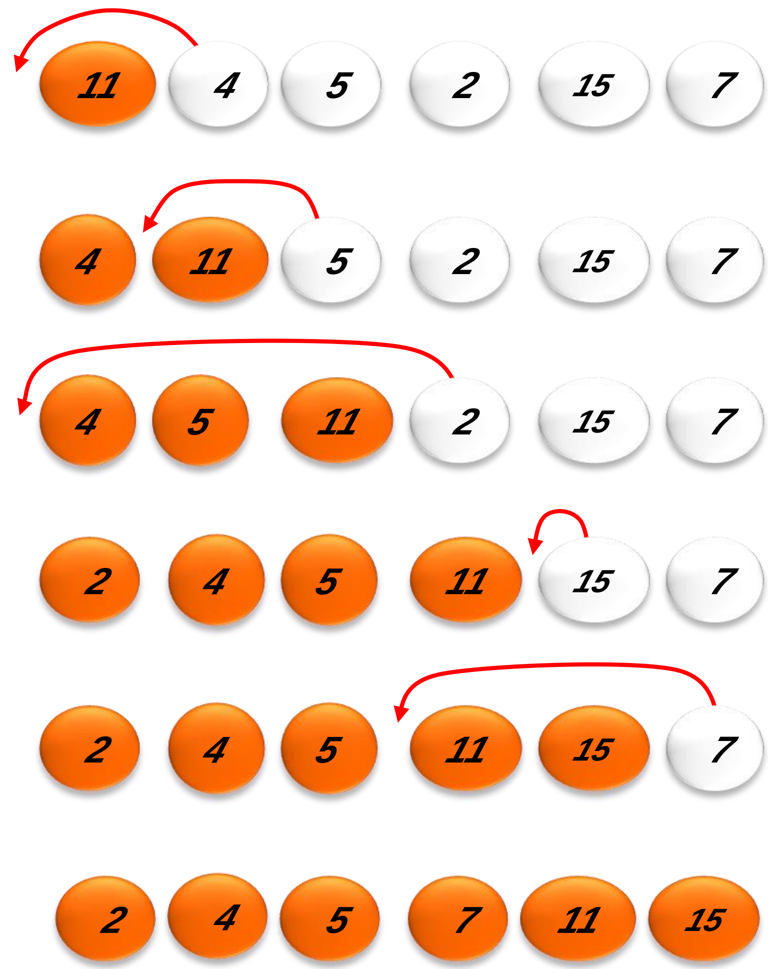
Last Lecture we have

- Analysis of Algorithms
- Asymptotic notation
- Insertion sort

Insertion Sort

an incremental algorithm





An Example: Insertion Sort

```
InsertionSort(A, n) {
  for i = 2 to n {
     key = A[i]
     j = i - 1;
     while (j > 0) and (A[j] > key) {
          A[j+1] = A[j]
          j = j - 1
     }
     A[j+1] = key
  }
}
```

From lecture by David Luebke

An Example: Insertion Sort

30	*10*	*40*	*20*
1	2	3	4

$i = \varnothing$ $\quad j = \varnothing$ $\quad key = \varnothing$

$A[j] = \varnothing$ $\quad A[j+1] = \varnothing$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

30	10	40	20
1	2	3	4

$i = 2$ $j = 1$ $key = 10$

$A[j] = 30$ $A[j+1] = 10$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
                A[j+1] = A[j]
                j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

1	2	3	4
30	*30*	*40*	*20*

$i = 2 \quad j = 1 \quad key = 10$

$A[j] = 30 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

An Example: Insertion Sort

1	2	3	4
30	*30*	*40*	*20*

i = 2   j = 1   key = 10
A[j] = 30   A[j+1] = 30

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

30	30	40	20
1	2	3	4

$i = 2$ $j = 0$ $key = 10$

$A[j] = \varnothing$ $A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

1	2	3	4
30	*30*	*40*	*20*

i = 2   j = 0   key = 10
A[j] = ∅   A[j+1] = 30

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

1	2	3	4
10	*30*	*40*	*20*

$i = 2 \quad j = 0 \quad key = 10$

$A[j] = \varnothing \quad A[j+1] = 10$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

An Example: Insertion Sort

10	30	40	20
1	2	3	4

$i = 3 \quad j = 0 \quad key = 10$

$A[j] = \emptyset \quad A[j+1] = 10$

```
InsertionSort(A, n) {
  for i = 2 to n {
      key = A[i]
      j = i - 1;
      while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
      }
      A[j+1] = key
  }
}
```

An Example: Insertion Sort

1	2	3	4
10	*30*	*40*	*20*

$i = 3 \quad j = 0 \quad key = 40$

$A[j] = \varnothing \quad A[j+1] = 10$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
                A[j+1] = A[j]
                j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	20
1	2	3	4

i = 3 j = 0 key = 40

A[j] = ∅ A[j+1] = 10

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	20
1	2	3	4

i = 3 j = 2	key = 40
A[j] = 30	A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	*30*	*40*	*20*
1	2	3	4

i = 3 j = 2 key = 40
A[j] = 30 A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	*30*	*40*	*20*
1	2	3	4

i = 3 j = 2	key = 40
A[j] = 30	A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
               A[j+1] = A[j]
               j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

1	2	3	4
10	*30*	*40*	*20*

$i = 4$ $j = 2$ $key = 40$

$A[j] = 30$ $A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	20
1	2	3	4

i = 4 j = 2 key = 20
A[j] = 30 A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
             A[j+1] = A[j]
             j = j - 1
        }
        A[j+1] = key
   }
}
```

An Example: Insertion Sort

10	30	40	20
1	2	3	4

i = 4 j = 2 key = 20
A[j] = 30 A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	20
1	2	3	4

i = 4 j = 3 key = 20
A[j] = 40 A[j+1] = 20

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	20
1	2	3	4

i = 4 j = 3 key = 20
A[j] = 40 A[j+1] = 20

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	40
1	2	3	4

$i = 4$ $j = 3$ $key = 20$

$A[j] = 40$ $A[j+1] = 40$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	*30*	*40*	*40*
1	2	3	4

i = 4	j = 3	key = 20
A[j] = 40		A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
               A[j+1] = A[j]
               j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	40
1	2	3	4

i = 4   j = 3   key = 20
A[j] = 40   A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

1	2	3	4
10	*30*	*40*	*40*

i = 4   j = 2   key = 20
A[j] = 30   A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
               A[j+1] = A[j]
               j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	40	40
1	2	3	4

i = 4   j = 2   key = 20
A[j] = 30   A[j+1] = 40

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

1	2	3	4
10	*30*	*30*	*40*

$i = 4 \quad j = 2 \quad key = 20$

$A[j] = 30 \quad A[j+1] = 30$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
                A[j+1] = A[j]
                j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	*30*	*30*	*40*
1	2	3	4

i = 4 j = 2 key = 20
A[j] = 30 A[j+1] = 30

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	30	40
1	2	3	4

i = 4   j = 1   key = 20
A[j] = 10   A[j+1] = 30

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	30	30	40
1	2	3	4

i = 4   j = 1   key = 20
A[j] = 10   A[j+1] = 30

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	20	30	40
1	2	3	4

i = 4 j = 1 key = 20
A[j] = 10 A[j+1] = 20

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

From lecture by David Luebke

An Example: Insertion Sort

10	20	30	40
1	2	3	4

$i = 4$ $j = 1$ $key = 20$
$A[j] = 10$ $A[j+1] = 20$

```
InsertionSort(A, n) {
   for i = 2 to n {
        key = A[i]
        j = i - 1;
        while (j > 0) and (A[j] > key) {
              A[j+1] = A[j]
              j = j - 1
        }
        A[j+1] = key
   }
}
```

Done!

From lecture by David Luebke

Insertion Sort: Time Complexity

```
Algorithm InsertionSort(A[1..n])
for i ← 2 .. n do
        LI: A[1..i –1] is sorted, A[i..n] is untouched.
    § insert A[i] into sorted prefix A[1..i–1] by right-cyclic-shift:
        key ← A[i]
        j ← i –1
        while j > 0 and A[j] > key do
            A[j+1] ← A[j]
            j ← j –1
        end-while
        A[j+1] ← key
end-for
end
```

$$T(n) = \Theta\left(\sum_{i=2}^{n}(1 + t_i + 1)\right) = \Theta\left(n + \sum_{i=2}^{n} t_i\right) = \Theta\left(n + \sum_{i=2}^{n} i\right) = \Theta\left(n + n^2\right) = \Theta\left(n^2\right).$$

Worst-case: $t_i = i$ iterations (reverse sorted).

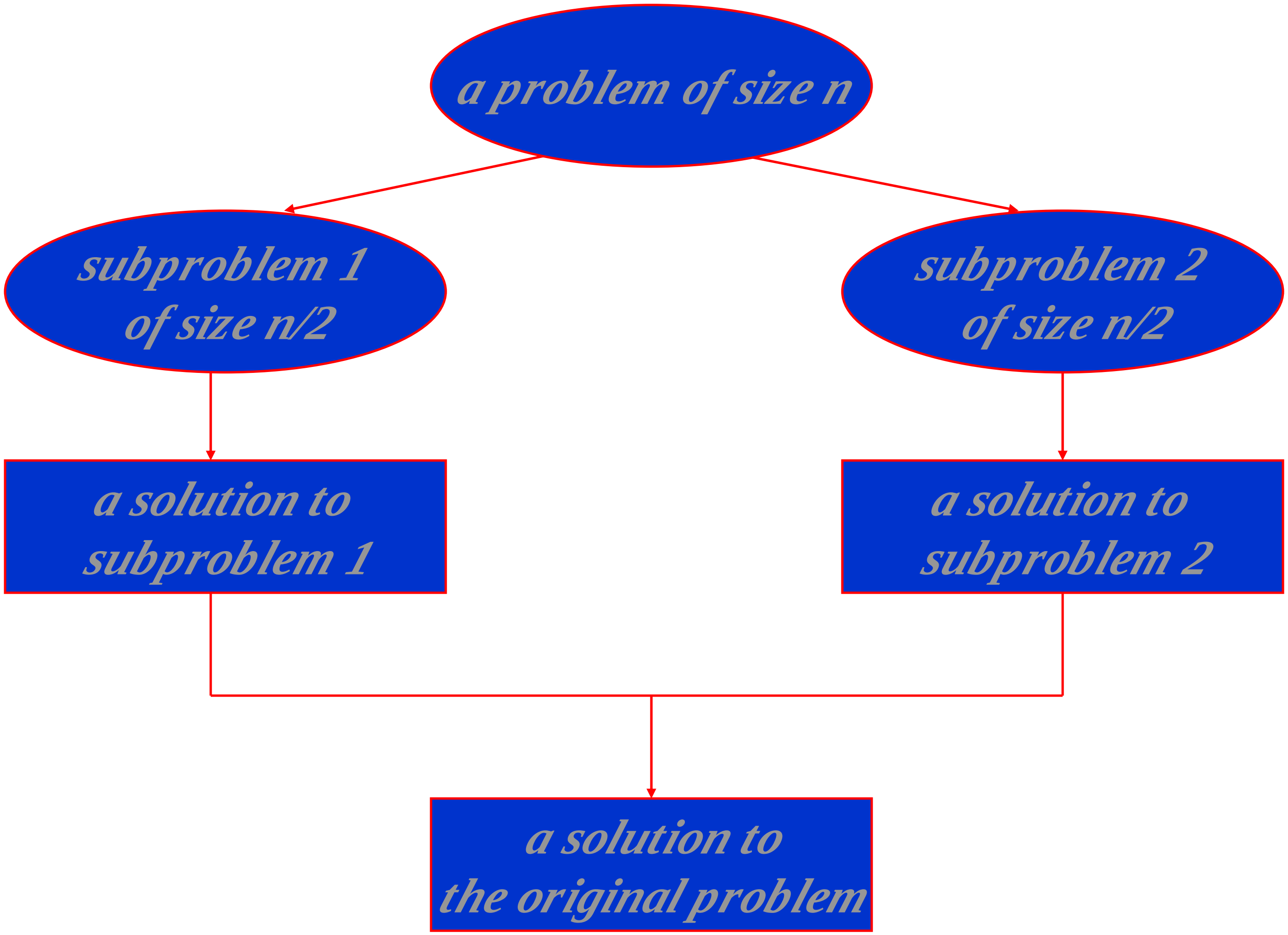
$$\sum_{i=2}^{n} i = \frac{n(n+1)}{2} - 1 = \Theta\left(n^2\right).$$

The divide-and-conquer Design Paradigm

- ***Divide** the problem into subproblems.*
- ***Conquer** the subproblems by solving them recursively.*
- ***Combine** subproblem solutions.*

- *Many algorithms use this paradigm.*

Divide-and-conquer Technique



Divide and Conquer Examples

- *Sorting: mergesort and quicksort*
- *Matrix multiplication-Strassen's algorithm*
- *Binary search*
- *Powering a Number*
- *Closest pair problem*
- *….etc.*

Binary search

- Find an element in a sorted array:
  - Divide: Check middle element.
  - Conquer: Recursively search 1 sub array.
  - Combine: Trivial.
  - Example: Find 9

    3 5 7 8 9 12 15

Binary search

- Find an element in a sorted array:
  - Divide: Check middle element.
  - Conquer: Recursively search 1 sub array.
  - Combine: Trivial.
  - Example: Find 9

3	5	7	8	9	12	15

Binary search

- Find an element in a sorted array:
  - Divide: Check middle element.
  - Conquer: Recursively search 1 sub array.
  - Combine: Trivial.
  - Example: Find 9

3	5	7	8	9	12	15

Binary search

- Find an element in a sorted array:
  - Divide: Check middle element.
  - Conquer: Recursively search 1 sub array.
  - Combine: Trivial.
  - Example: Find 9

3	5	7	8	9	12	15

Binary search

- Find an element in a sorted array:
  - Divide: Check middle element.
  - Conquer: Recursively search 1 sub array.
  - Combine: Trivial.
  - Example: Find 9

3 5 7 8 9 12 15

Binary search

- Find an element in a sorted array:
  - Divide: Check middle element.
  - Conquer: Recursively search 1 sub array.
  - Combine: Trivial.
  - Example: Find 9

3	5	7	8	9	12	15

Binary search

- Find an element in a sorted array:
  - Divide: Check middle element.
  - Conquer: Recursively search 1 sub array.
  - Combine: Trivial.
  - Example: Find 9

3 5 7 8 9 12 15

Binary Search

```
int binarySearch(int a[], int size, int x) {
   int low =0;
   int high = size -1;
   int mid;              // mid will be the index of
                         // target when it’s found.
   while (low <= high) {
     mid = (low + high)/2;
     if (a[mid] < x)
        low = mid + 1;
     else if (a[mid] > x)
         high  = mid - 1;
     else
         return mid;
   }
   return -1;
}
```

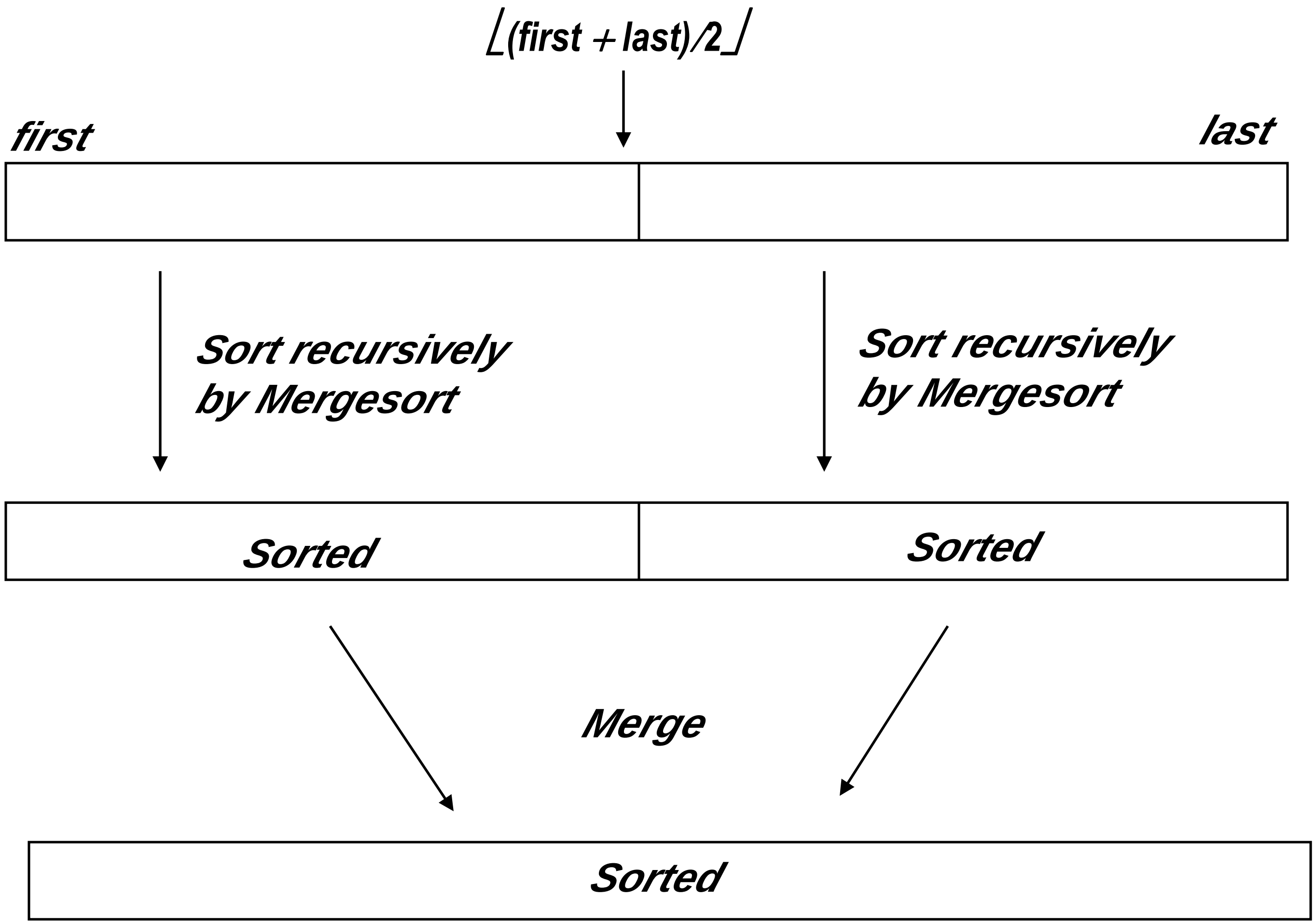
Mergesort

Algorithm:

- Split array A[1..n] in two and make copies of each half in arrays $B[1 \ldots \lfloor \frac{n}{2} \rfloor]$ and C[1 ... $\lceil \frac{n}{2} \rceil$]
- *Sort arrays B and C*
- *Merge sorted arrays B and C into array A*

Using Divide and Conquer: Mergesort

- Mergesort Strategy



Mergesort

Algorithm:

- *Split array A[1..n] in two and make copies of each half in arrays* $B[1 \ldots \lfloor \frac{n}{2} \rfloor]$ *and* $C[1 \ldots \lceil \frac{n}{2} \rceil]$
- *Sort arrays B and C*
- *Merge sorted arrays B and C into array A as follows:*
  - *Repeat the following until no elements remain in one of the arrays:*
    - *compare the first elements in the remaining unprocessed portions of the arrays*
    - *copy the smaller of the two into A, while incrementing the index indicating the unprocessed portion of that array*
  - *Once all elements in one of the arrays are processed, copy the remaining unprocessed elements from the other array into A.*

Algorithm: Mergesort

Input: Array E and indices first and last, such that the elements E[i] are defined for first $\leq$ i $\leq$ last.

Output: E[first], …, E[last] is a sorted rearrangement of the same elements

```
void mergeSort(Element[] E, int first, int last)
    if (first < last)
        int mid = ⌊(first+last)/2⌋;
        mergeSort(E, first, mid);
        mergeSort(E, mid+1, last);
        merge(E, first, mid, last);
    return;
```

Merge Sort

MERGE-SORT $A[1\,..\,n]$

1. If $n = 1$, done.
2. Recursively sort $A[\,1\,..\,\lceil n/2\rceil\,]$ and $A[\,\lceil n/2\rceil+1\,..\,n\,]$.
3. "***Merge***" the 2 sorted lists.

- How to express the cost of merge sort?

$$T(n) = 2T(n/2) + \Theta(n) \text{ for } n>1,\ T(1)=0 \Rightarrow \Theta(n \lg n)$$

Merge Sort

1. ***Divide*: Trivial.**
2. ***Conquer*: Recursively sort subarrays.**
3. ***Combine*: Linear-time merge.**

$$T(n) = \underbrace{2}_{\text{\# subproblems}}T(\underbrace{n/2}_{\text{subproblem size}}) + \underbrace{\Theta(n)}_{\text{cost of dividing and combining}}$$

Efficiency of Mergesort

- *All cases have same efficiency: Θ(n log n)*

- Number of comparisons is close to theoretical minimum for comparison-based sorting:
  - $\lceil \lg n! \rceil \approx \lceil n \lg n - 1.44 n \rceil$

- *Space requirement: Θ(n) (<u>NOT</u> in-place)*

- *Can be implemented without recursion (bottom-up)*

Master theorem

- If $T(n) = aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + O\left(n^d\right)$ for some constants a > 0, b> 1, and d $\geq$ 0, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a. \end{cases}$$

1. $a < b^d$ $\quad T(n) \in \Theta(n^d)$
2. $a = b^d$ $\quad T(n) \in \Theta(n^d \lg n)$
3. $a > b^d$ $\quad T(n) \in \Theta(n^{\log_b a})$

The End