# Insertion/Bubble Sort
# Merge Sort
# Asymptotic Analysis

BLG335E Fall 2022 - Recitation 1

Caner Özer - ozerc@itu.edu.tr

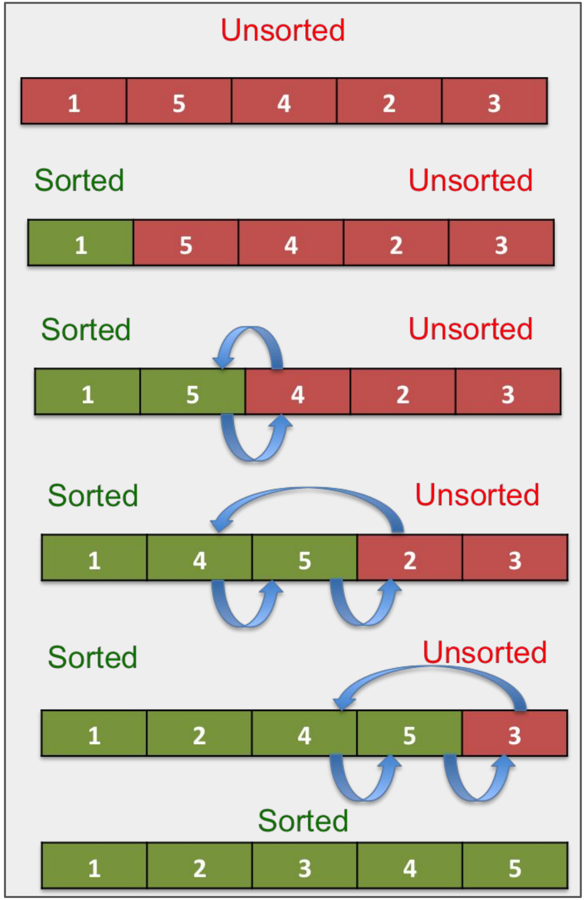
# Course Logistics

Docker information & guidelines to be submitted soon (hopefully tomorrow)

Crucial for your homeworks to be evaluated correctly

Please try installing them before the next week's recitation

# Insertion Sort



# Insertion Sort

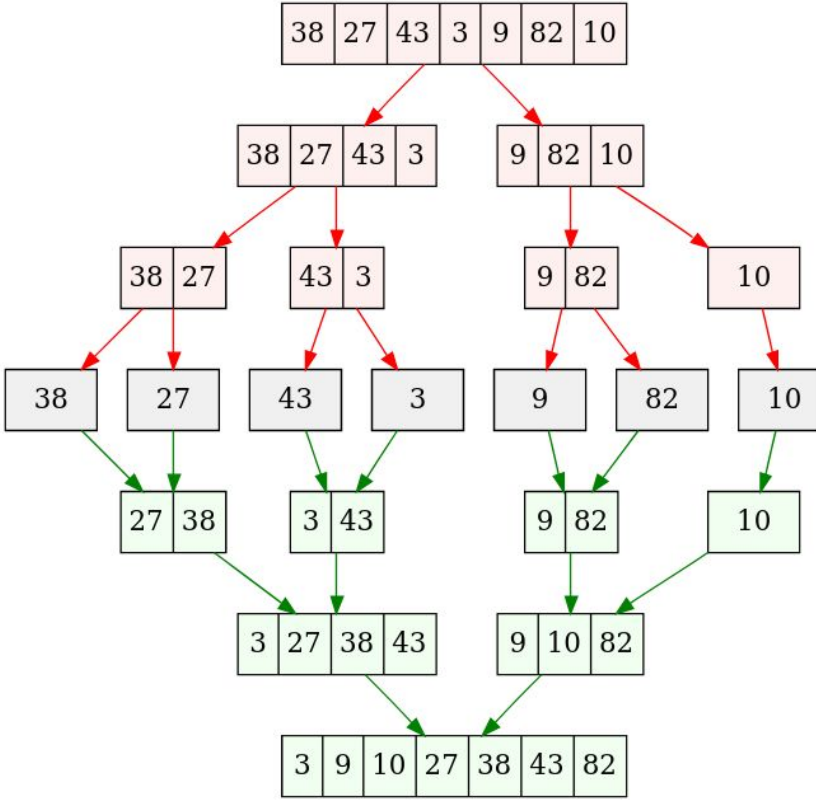
```
INSERTION-SORT (A, n)      ▷ A[1 . . n]
  for j ← 2 to n
       do key ← A[j]
          i ← j – 1
          while i > 0 and A[i] > key
               do A[i+1] ← A[i]
                  i ← i – 1
          A[i+1] = key
```

**Algorithm 1:** Bubble sort

```
Data: Input array A[]
Result: Sorted A[]
int i, j, k;
N = length(A);
for j = 1 to N do
    for i = 0 to N-1 do
        if A[i] > A[i+1] then
            temp = A[i];
            A[i] = A[i+1];
            A[i+1] = temp;
        end
    end
end
```

Less swap operations on average
comparing bubble-sort

# Merge Sort




```
Merge-Sort(A, p, r)
if p < r
then {q ← ⌊(p+r)/2⌋
        Merge-Sort(A, p, q)
        Merge-Sort(A, q+1, r)
        Merge(A, p, q, r)}
```

# Merge(A, p, q, r)

```
n1 ← (q – p) + 1
n2 ← (r – q)
create arrays L[1..n1+1] and R[1..n2+1]
for i ← 1 to n1 do
  L[i] ← A[(p + i) -1]
for j ← 1 to n2 do
  R[j] ← A[q + j]
L[n1 + 1] ← ∞
R[n2 + 1] ← ∞
i ← 1
j ← 1
for k ← p to r do
  if L[i] <= R[j]
      then A[k] ← L[i]
           i ←i + 1
      else A[k] ← R[j]
           j ← j + 1
```

# Merge Sort

For Merge-Sort definition, try to remember the postorder traversal from data structures, and while debugging, try to form a program stack while maintaining an order. Merge function should work well as expected as long as you set the last elements of the L and R arrays to infinity. Try sorting a more simpler example like [3, 4, 1, 7, 2] to see it clearly.

# Bubble Sort (Ascending)

| | | | | | |
|---|---|---|---|---|---|
| **7** | **8** | 2 | 4 | 9 | |
| 7 | **8** | **2** | 4 | 9 | The second and third elements will be swapped |
| 7 | 2 | **8** | **4** | 9 | The third and forth elements will be swapped |
| 7 | 2 | 4 | **8** | **9** | |

Iteration 1

| | | | | | |
|---|---|---|---|---|---|
| **7** | **2** | 4 | 8 | 9 | The first two elements will be swapped |
| 2 | **7** | **4** | 8 | 9 | The second and third elements will be swapped |
| 2 | 4 | **7** | **8** | 9 | |
| 2 | 4 | 7 | **8** | **9** | |

Iteration 2

| | | | | |
|---|---|---|---|---|
| **2** | **4** | 7 | 8 | 9 |
| 2 | **4** | **7** | 8 | 9 |
| 2 | 4 | **7** | **8** | 9 |
| 2 | 4 | 7 | **8** | **9** |

Iteration 3

# Bubble Sort

---

**Algorithm 1** Bubble Sort

---

**Input:** Input Array $A$
**Output:** Sorted Array $A$

```
N ← len(A)
for int j = 1 to N do
    for int i = 0 to N − 1 do
        if A[i] > A[i + 1] then
            temp ← A[i]
            A[i] ← A[i + 1]
            A[i + 1] ← temp
        end if
    end for
end for
return A
```

---

Something is missing...

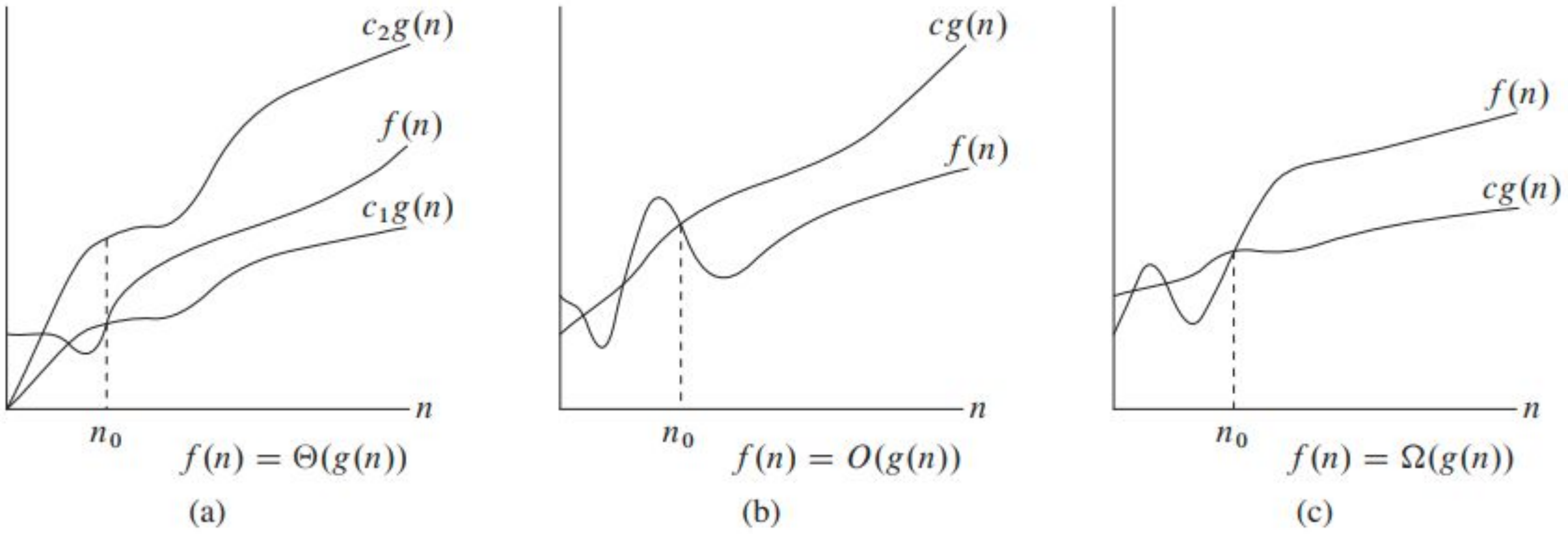
# Bubble Sort

**Algorithm 2** Corrected Bubble Sort

```
Input: Input Array A
Output: Sorted Array A
  N ← len(A)
  for int j = 1 to N do
      swapped ← false
      for int i = 0 to N − 1 do
          if A[i] > A[i + 1] then
              temp ← A[i]
              A[i] ← A[i + 1]
              A[i + 1] ← temp
              swapped ← true
          end if
      end for
      if not swapped then
          break
      end if
  end for
  return A
```

Extending the implementation to handle the best-case scenario

# Asymptotic Notations (Recall)




$$1 << \log\ n << n << n\ \log\ n << n^2 << 2^n << n! << n^n$$

# Asymptotic Notation - Q1

$\Theta(g(n)) = \{ f(n)$ : there exist positive constants $c_1$, $c_2$, and $n_0$ such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0 \}$ .[1]

Let f(n) and g(n) be asymptotically nonnegative functions. Using the basic definition of Θ notation, prove that max(f(n), g(n)) = Θ(f(n) + g(n)).

Asymptotically nonnegativity: $\forall n > n_0$, $f(n) \ge 0$, $g(n) \ge 0$

Hence, we see that $f(n) + g(n) \ge f(n) \ge 0$ and $f(n) + g(n) \ge g(n) \ge 0$.

Defining $h(n) = \max(f(n), g(n))$, we can simplify $f(n) + g(n) \ge h(n) \ge 0$.

<u>Upper bound</u>: $c_2(f(n) + g(n)) \ge f(n) + g(n) \ge h(n) \Rightarrow$ $\mathbf{c_2 = 1}$

# Asymptotic Notation - Q1

By using h(n) itself, we can also see that

$h(n) \geq f(n) \geq 0$ and $h(n) \geq g(n) \geq 0$.

Summing these 2 inequalities yields

<u>Lower-bound</u>: $h(n) \geq 0.5(f(n) + g(n)) = c_1(f(n) + g(n)) \geq 0 \Rightarrow$ $\mathbf{c_1 = 0.5}$

...

# Asymptotic Notation - Q2

$$\Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le c g(n) \le f(n) \text{ for all } n \ge n_0 \} .$$

Prove that $(\log n)^{\log n} = \Omega(n^{3/2})$.

$cn^{3/2} = c(2^{\log n})^{3/2} = c(2^{3/2})^{\log n} \le (\log n)^{\log n}$

$c = 1, n_0 = 8$

...

# Asymptotic Notation - Q3

$O(g(n)) = \{f(n) :$ there exist positive constants $c$ and $n_0$ such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$ .

1. $2^{n+1} \stackrel{?}{=} O(2^n)$
2. $2^{2n} \stackrel{?}{=} O(2^n)$

1. $c2^n \ge 2^{n+1} = 2.2^n \ge 0$. Hence, for $c = 2$ and $n_0 = 0$, this expression holds.

2. $c2^n \ge 2^{2n} = 2^n.2^n \ge 0$. Since c needs to be a constant, this expression does not hold.

...