Seminar 02

Thursday, March 17, 2022 2:58 PM

Complexity

Ω - omega - lowerbound
Θ - theta - constant
O - big O - upperbound

1) Compute $T(n)$ - number of steps

$\Omega(n)$

$\Theta(n^2)$

$O(n^2)$; $O(n^3)$

$T(n) = 2n^2 + n$

1) $n^2 \in O(n^3)$ - true
2) $n^3 \in O(n^2)$ - false
3) $2^{n+1} \in \Theta(2^n)$ - true
4) $2^{2n} \in \Theta(2^n)$ - false
5) $n^2 \in \Theta(n^3)$ - false
6) $2^n \in O(n!)$ - true
7) $\lg n \in \Theta(\log_2 n)$ - true

8) $O(n) + \Theta(n^2) = \Theta(n^2)$
9) $\Theta(n) + O(n^3) = O(n^2)$
10) $O(f) + O(g) = O(\max(f, g))$
11) $3^n \in O(2^n)$ - false
12) $\log_2 3^n \in \Theta(\log_2 2^n)$ - true

13)

	Runtime				Space Complexity
	B.C.	W.C.	Avg	Total	
Linear Search	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	$O(n)$	$\Theta(1)$
Binary Search	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(\log_2 n)$	$O(\log_2 n)$	$\Theta(1)$
Selection Sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(1)$ = in-place
Bubble Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(n^2)$	$\Theta(1)$
Quick Sort	$\Theta(n\log n)$	$\Theta(n^2)$	$\Theta(n\log n)$	$O(n^2)$	$\Theta(1)$
Merge Sort	$\Theta(n\log n)$	$\Theta(n\log n)$	$\Theta(n\log n)$	$\Theta(n\log n)$	$\Theta(n)$

14)

```
Subalgorithm S1(n) is:
    | for i ← 1, n execute:
```

```
[illegible] subalgorithm S(n) is:
      for i ← 1, n execute:
          j ← n
$n$       while $j \neq 0$ execute:        $n$          $\Theta(n \log n)$
              $j \leftarrow [\frac{j}{2}]$       $\Theta(\log_2 n)$
          end-while
      end-for
  end-subalgorithm
```

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

```
  subalgorithm S1(n) is:
      for i ← 1, n execute:
          j ← i
$n$       while $j \neq 0$ execute:
              $j \leftarrow [\frac{j}{2}]$       $\log_2 i$
          end-while
      end-for
  end-subalgorithm
```

$$\sum_{i=1}^{n} \log_2 i = \log_2 1 + \log_2 2 + \dots + \log_2 n =$$

$$= \log_2 n!$$

$$\Theta(\log_2 n!)$$

$T(n) = \Theta(\log(n!)) = \Theta(n \log n)$ (Sterling approximation)

```
  subalgorithm (x, n, a) is:
      found ← false
      for i ← 1, n execute:
$n$          if $x_i = a$ then:                 $\Theta(n)$
                found ← true
            end-if
      end-for
  end-subalgorithm
```

```
subalgorithm (x, n, a) is:
    found ← false
    i ← 1
    while $i \neq n$ execute:
        if $x_i = a$ then:
            found ← true
        end-if
    end-for
end-subalgorithm
```

BC: $\Theta(1)$ Total: $O(n)$

WC: $\Theta(n)$

AC: $\Theta(n)$

<u>AC:</u> $T(n) = C_1 P_{C_1} + C_2 P_{C_2} + \dots + C_n P_{C_n} =$

$= \frac{1}{n}(1 + 2 + \dots + n) = \frac{n(n+1)}{2n} = \frac{n+1}{2}$

$\Rightarrow \Theta(n)$