Constant-time
$O(1)$

→ Accessing an element of list

Log. Algorithms
$O(\log n)$

$$f(n) = \log_b n = \frac{1}{\log_2 b} \log_2 n = O(\log_2 n)$$

→ Binary Search

$n = 2^{10}$ → $\log n = 10$

$n = 2^{100}$ → $\log n = 100$

Linear Algorithms
$O(n)$

→ Linked List Access

→ Linear Search

→ Merge two Sorted Lists

$n = 100$ $O(n) = 100$

$n = 2^{100}$ $O(n) = 2^{100}$

Linearithmic Alg.
$O(n \log n)$

→ Merge Sort

Quadratic Alg.
$O(n^2)$

→ Insert Sort

→ Find Closest points on a plane

→ Stable Matching (Gale-Shapley) Alg.

Cubic Alg.
$O(n^3)$

→ Matrix Multiplication

→ Find non-overlapping Sets

Recurrent Algorithms

```
Sort ( A )
  if |A| = 0 then return
  j ← Find Min (A)
  Swap (A[0], A[j])
  Sort (A[1...n-1])
```

$$T(n) = n + T(n-1)$$

$$= n + (n-1 + T(n-2))$$

$$= n + (n-1) + (n-2) + T(n-3)$$

$$\vdots$$

$$= n + (n-1) + \cdots + T(0)$$

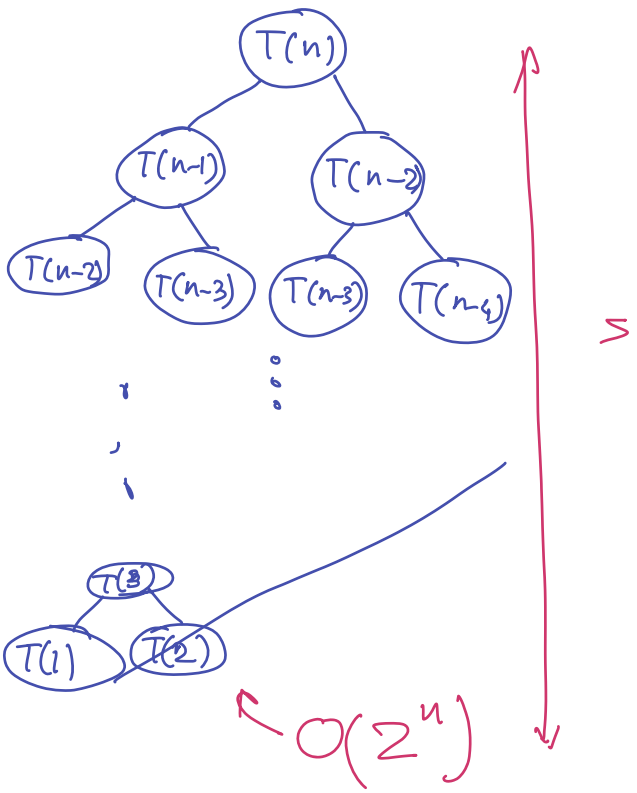
$$= \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

$$= \Theta(n^2)$$

```
Fib (n)
  F(1) = 1
  F(2) = 1
  F(i) = F(i-1) + F(i-2)
  F(3) = 2
```

```
Fib (n)
  if (n < 3) return 1
  return Fib(n-1) + Fib(n-2)
```

$$T(n) = 1 + T(n-1) + T(n-2)$$




```
Fib (n)
  if (n < 3) return 1
  a = 1
  b = 1 ; i = 3
  while (i ≤ n)
      F = a + b
      b = a
      a = F
  return F
```

$O(n)$

Binary Search Algorithm

Input: Sorted List, a key

Obj: to identify if key $\in$ List

```
B-Search(A, low, high, key)
    if low > high return false
    mid = ⌊(low+high)/2⌋
    if (key == A[mid])
        return True
    if (key < A[mid])
        return B-Search(A, low, mid-1, key)
    return B-Search(A, mid+1, high, key)
```

$$T(n) = 1 + T\left(\frac{n}{2}\right)$$

$$= 1 + \left(1 + T\left(\frac{n}{2^2}\right)\right) = 2 + T\left(\frac{n}{2^2}\right)$$

$$= 2 + \left(1 + T\left(\frac{n}{2^3}\right)\right) = 3 + T\left(\frac{n}{2^3}\right)$$

$$\vdots$$

$$= k + T\left(\frac{n}{2^k}\right)$$

$\frac{n}{2^k} = 1 \rightarrow k = \log n$

$$= \log n + T\left(\frac{n}{2^{\log n}}\right) = \Theta(\log n)$$

Merge-Sort Alg.

```
M-Sort(A, low, high)
    if (low = high) return A
    m = ⌊(low+high)/2⌋
    M-Sort(A, low, m)
    M-Sort(A, m+1, high)
    n ← Merge(A, low, m, high)
```

$$T(n) = n + 2T\left(\frac{n}{2}\right)$$

$$= n + 2\left(\frac{n}{2} + 2T\left(\frac{n}{2^2}\right)\right)$$

$$= 2n + 2^2 T\left(\frac{n}{2^2}\right)$$

$$= 2n + 2^2\left(\frac{n}{2^2} + T\left(\frac{n}{2^3}\right)\right)$$

$$= 3n + 2^3 T\left(\frac{n}{2^3}\right)$$

$$\vdots$$

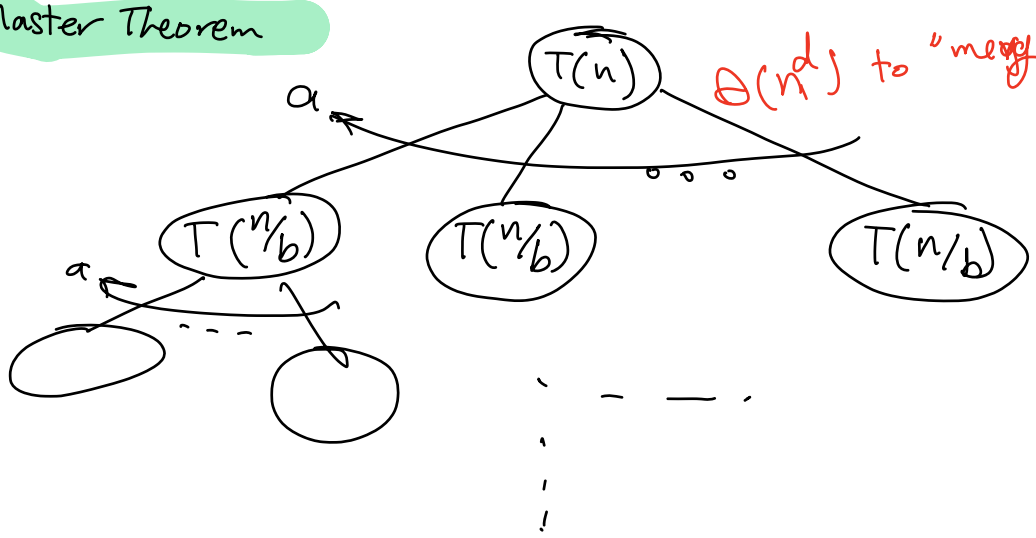
$$= kn + 2^k T\left(\frac{n}{2^k}\right)$$

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow k = \log n$

$$T(n) = n\log n + 2^{\log n} T(1)$$

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

Master Theorem



$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d)$$

if $a < b^d$: $T(n) = \Theta(n^d)$

if $a = b^d$: $T(n) = \Theta(n^d \log n)$

if $a > b^d$: $T(n) = \Theta(n^{\log_b a})$

$T(n) = 10\,T\left(\frac{n}{2}\right) + n^2$ $\quad a > b^d$

($a = 10$, $b = 2$, $d = 2$)

$\Rightarrow T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 10}) \simeq \Theta(n^{3.1})$