Analysis of Algorithm

Definition Step: chosen operations performed by the algorithm such that the total number of these operations is the same as the number of all operations performed by the algorithm to within a constant factor

Definition $t_A(I)$ denote the number of steps the algorithm A taks for input I
Example linear search

```
LS(L, x)
  i ← 1
  while i ≤ len(L)
     if L[i] = x then return I
     i ← i + 1
  return 0
```

Steps: number of comparisons with x

x	$t_{LS}([2,4,6,8], x)$
2	1
4	2
8	4
1	4

Definition $T(n)$ the running time of one algorithm on input of size n, because running time of a program usually increases as the size of the input increases
$T_W: \mathbb{Z}^+ \to \mathbb{N} \coloneqq T_W(n) = \max\{\, t_n(I) \mid size(I) = n \,\}$ the worst time complexity
$T_A: \mathbb{Z}^+ \to \mathbb{R}^+ \coloneqq T_A(n) = \frac{\sum_i i \in \{t_n(I) \mid size(I)=n\}}{|\{t_n(I) \mid size(I)=n\}|}$ the average time complexity
Sometimes algorithm runtime functions has more than 2 parameters, such as the graph algorithm
$T_G(v, e), v = \#vertices, e = \#edges$
Breaking point: The point which one algorithm starts to takes less time than another
$A, B \in Alg.\, T_A(n) = n^3, T_B(n) = 8n + 3$ When $n > 3 \to T_A(n) > T_B(n), n < 3 \to T_A(n) < T_B(n), 3$ is the breaking point
Leading coefficients and exponential terms determine algorithm's running time
Upper bound of the algorithm u: $\forall n \in \mathbb{Z}^+. \forall i \in Input. |I| = n \to t_A(I) \leq u(n)$
Lower bound on the worst cases of the algorithm l: $\forall n \in \mathbb{Z}^+. \exists i \in Input. |i| = n$ AND $t_A(i) \geq l(n)$

Big-Oh $O(f) = \{g \mid \exists c \in \mathbb{R}^+. \exists n_0 \in \mathbb{N}. \forall n \in \mathbb{N}. n \geq n_0 \to g(n) \geq cf(n)\}$
Big-Omega $\Omega(f) = \{g \mid \exists c \in \mathbb{R}^+. \exists n_0 \in \mathbb{N}. \forall n \in \mathbb{N}. n \geq n_0 \to g(n) \leq cf(n)\}$
Big-Theta $\Theta(f) = \{g \mid \exists c \in \mathbb{R}^+ \exists d \in \mathbb{R}^+. \exists n_0 \in \mathbb{N}. \forall n \in \mathbb{N}. n \geq n_0 \to cf(n) \leq g(n) \leq df(n)\}$

Iterative Algorithm Time Complexity

Code with loops and procedures calls $O(1)$

If statement: $\in O(h(n) + \max\{f(n), g(n)\})$

    if C | $O(h(n))$
        then A | $O(f(n))$
    else B | $O(g(n))$

For/while loop: if a loop is executed $O(f(n))$ times and each iteration is $O(g(n))$, then total is $O(f(n)g(n))$

Suppose we call with an input of size $g(n)$, the the call takes $O\big(f(g(n))\big)$ times

IS(A)

```
1. i ← 1
2. while i ≤ len(A)
3.    j ← i
4.    while j > 1 && A[i] < B[j − i]
5.        B[j] ← B[j − i]
6.        j ← j − 1
7.    B[j] ← A[i]
8.    i ← i + 1
9. return B
```

Example Insertion Sort

Let $T: \mathbb{Z}^+ \to \mathbb{N} := T(n) =$ the max number of steps taken by insertion sort on arrays of size n

Let step = comparisons assignments

Lemma1 $T(n) \in O(n^2)$

Proof Let $n \in \mathbb{N}^+$ be arbitrary and consider an arbitrary array A of length n

There are n complete iterations of the outer while loop (2-8). Each iterations consists of 4 steps and a 1 execution of an inner while loop (4-6)

> Each iteration of the inner loop takes at most 4 steps, and at most $i - 1$ complete iterations of the inner loop are performed during the iteration of the inner loop.
> The final complete iteration of the inner loop takes at most 2 steps (check the while statement)

So the total time taken by n complete iterations of the out while loop is at most

$\sum_{i=1}^{n}(4(i-1)+2+4) = \sum_{i=1}^{n} 4i + 2 = 2n^2 + 4n$

The final complete iteration of the outer while loop takes 1 more assignment and 1 while statement check, then $T(n) = 2n^2 + 4n + 2 \in O(n^2)$

By generalization, $\forall n \in \mathbb{Z}^+. T(n) \in O(n^2)$

Lemma2 $T \in \Omega(n^2)$

Proof Let $n \in \mathbb{Z}^+$ be arbitrary, consider the input $A = [n, n-1, \ldots, 1]$

During the 1st iteration, the outer while loop takes 5 steps (2,3,4(1),7,8), and afterwards, $j = i, B[i] = A[i] = n$

If $i < n$, then during the rest iterations of the outer while loop, 4 steps are performed and adding the inner while loop.

There are i complete iterations of the inner while loop since the first i elements of B are all larger than $A[i] = n - i$

In the final iteration of the inner while loop, only 2 step is performed For a total of $4i + 2$ steps

The outer loop completely iterates n times, the last incomplete iteration of the outer while loop takes 1 step, there's also one more step taken before the outer while loop

$$T(A) = 5 + 2 + \sum_{i=1}^{n-1} 4 + 4i + 1 = 7 + 5n + 4\sum_{i=1}^{n-1} i \in \Omega(n^2)$$

Corollary $T(A) \in \Theta(n^2)$

Running time of Recursive Algorithm

Square(n)

```
if n = 1
    return n
else return (2n − 1 + square(n − 1))
```

Proof Define $T: \mathbb{Z}^+ \to \mathbb{N} := T(n) =$ the number of arithmatic operations performed by square(n)

$$T(n) = \begin{cases} 0 \mid n = 1 \\ 4 + T(n-1) \mid n > 1 \end{cases}$$

Hence $T(n) = 4n - 4$ for $n > 1$

RBS(A, f, l, x)

1. if $f = 1$
2. if $A[f] = n$ then return f
3. else return 0
4. $m \leftarrow \frac{f + l}{2}$
5. if $A[m] \geq x$ then RBS(A, f, m, x)
6. else RBS(A, m+1, l, x)

Proof Define $B: \mathbb{Z}^+ \to \mathbb{N} \coloneqq B(n) =$ the worst case number of comparision with x perfromed by RBS(A, f, l, x),

$1 + f + l = n$

$$B = \begin{cases} 1 \mid n = 1 \\ 1 + \max\left\{B\left(\left\lfloor\frac{n}{2}\right\rfloor\right), B\left(\left\lceil\frac{n}{2}\right\rceil\right)\right\} \mid n > 1 \end{cases}$$