1. Insertion Sort

Sorting Problem:

Input: a list of integers, $a_1, a_2, \dots, a_n$

Output: a permutation / reordering, $a_1', a_2', \dots, a_n'$

$$a_1' \le a_2' \le \cdots \le a_n'$$

Pseudo-code:

```
Insertion-Sort:
For j=2 to A.length:
    key = A[j]
    i = j-1
    While i>0 and A[i]>key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

"Place key into correct spot"

e.g. | 3 | 1 | 5 | 2 | 4 | → | 3 | 1 | 5 | 2 | 4 |

→ | 1 | 3 | 5 | 2 | 4 | → | 1 | 3 | 5 | 2 | 4 |

→ | 1 | 2 | 3 | 5 | 4 | → | 1 | 2 | 3 | 4 | 5 |

① Correctness:

Proof: We prove by showing the following loop invariant:

"At top of the iteration, $A[1], \dots, A[j-1]$ are sorted (and contain the original elements in $A[1:j-1]$)"

▷ Initialization:

We start with $j=2$, $A[1], \dots, A[j-1]$ is just $A[1]$. already sorted!

▷ Maintenance:

We know that $A[1:j-1]$ is sorted at the top of the itr., we want to argue $A[1:j]$ is sorted at the end of the itr., and hence also at the top of the next itr.

a) $A[i+1] = key$

b) $A[i+2:j]$ are originally $A[i+1:j-1]$

c) $A[1:i]$ are untouched (and already sorted)

$$\underbrace{A[1] \leq A[2] \leq \cdots \leq A[i]}_{(c)} \leq \overset{\substack{key \\ \downarrow (a)}}{A[i+1]} \leq \underbrace{A[i+2] \leq \cdots \leq A[j]}_{(b)}$$

"while" condition

▷ Termination:

At termination, $j = A.length+1$, so $A[1 : A.length]$ is sorted. #

② Runtime Analysis:

```
Insertion-Sort:
For j=2 to A.length:
    key = A[j]
    i = j-1
    While i>0 and A[i]>key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

c c' t_j

Worst Case: $\sum_{j=2}^{n}(c+t_j) = c\cdot(n-1)+\sum_{j=2}^{n} t_j$

$$\sum_{j=1}^{n-1} j = \frac{n(n-1)}{2}$$

$$= c\cdot(n-1)+\sum_{j=2}^{n} c'\cdot(j-1) = c\cdot(n-1)+c'\cdot\boxed{\sum_{j=2}^{n}(j-1)}$$

$$= c\cdot(n-1)+c'\cdot\frac{n(n-1)}{2}$$

The n^2 term dominates the other terms pretty fast, so we pretty much only care about it.

How do we formalize this?

2. Asymptotics

Def: Big-O

$$O(g(n)) = \{ f : \exists c, n_0 > 0, \text{ s.t. } \forall n > n_0, \; 0 \le f(n) \le c \cdot g(n) \}$$

Intuition: for sufficiently large n, $f(n)$ is "$\le$" $g(n)$, up to a constant factor.

We write $f(n) \in O(g(n))$, or simply as $f(n) = O(g(n))$.

Ex: $f(n) = \frac{1}{9999} n^2 + 10000n$

$g(n) = n^2$, show that $f(n) = O(g(n))$.

Proof: let $c = \frac{2}{9999}$, and $n_0 = 9999 \times 10001$

$$\begin{aligned} f(n) - c \cdot g(n) &= \frac{1}{9999} n^2 + 10000n - \frac{2}{9999} n^2 \\ &= 10000n - \frac{1}{9999} n^2 \\ &\le 10000n - \frac{1}{9999} \cdot (9999 \cdot 10001) \cdot n \\ &= -n < 0 \end{aligned}$$

※

e.g. $f(n) = \begin{cases} n & \text{if } n \text{ is odd} \\ -n^3 & \text{if } n \text{ is even} \end{cases}$, $f(n) = O(n)$?

No. We need $f(n) \geq 0$. Generally, we assume $f(n), g(n) \geq 0$.

small-o: $o(g(n)) = \{f : \forall c > 0, \exists n_0 > 0, \text{ s.t. } \forall n > n_0,$
$0 \leq f(n) \leq c \cdot g(n)\}$

e.g. $n = o(n^2)$: say fix some arbitrary c,
we can set $n_0 = \frac{1}{c}$. then we have
$c \cdot n^2 - n > c \cdot \frac{1}{c} \cdot n - n = 0$.

$n^2 \neq o(n^2)$: say $c = \frac{1}{2}$, then, regardless of n_0,
$c \cdot n^2 - n^2 = \frac{1}{2} n^2 - n^2 = -\frac{1}{2} n^2 < 0$

Big-Omega: $\Omega(g(n)) = \{f: \exists c, n_0 > 0, \text{ s.t. } \forall n > n_0, f(n) \geq c \cdot g(n)\}$

small-omega: $\omega(g(n)) = \{f: \forall c > 0, \exists n_0 > 0, \text{ s.t. } \forall n > n_0, f(n) \geq c \cdot g(n)\}$

Big-Theta: $\Theta(g(n)) = \{f: \exists c_1, c_2, n_0 > 0, \text{ s.t. } \forall n > n_0, c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$

Cheatsheet: as $n \to \infty$:	
O	$\leq$
o	$<$
Θ	$=$
ω	$>$
Ω	$\geq$

Ex: $n^2 + n^2 \log n + \frac{n^2}{\log n} + n^2 \log^2 n$

$= O(?)$

$= O(n^2 \log^2 n)$

Ex:

$3^n = O(2^n)$? ✗

$3^n = o(2^n)$? ✗

$3^n = \Theta(2^n)$? ✗

$3^n = \Omega(2^n)$? ✓

$3^n = \omega(2^n)$? ✓

So... given $f(n), g(n)$, how to check if $f(n)=O(g(n))$? Ω? ω? o? Θ?

- Go through definitions
- Consider $\lim_{n\to\infty} \frac{f(n)}{g(n)}$:

Let $f(n), g(n)$ be non-negative functions (for sufficiently large n):

If $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists, denoted as x, then

$x=\infty \Rightarrow f(n)=\omega(g(n))$

$x=0 \Rightarrow f(n)=o(g(n))$

$x=c'>0 \Rightarrow f(n)=\Theta(g(n))$

$x\neq\infty \Rightarrow f(n)=O(g(n))$

$x\neq 0 \Rightarrow f(n)=\Omega(g(n))$

Let's prove the one for $o(g(n))$:

Proof: $\lim_{n\to\infty} \frac{f(n)}{g(n)}=0 \Rightarrow \forall\, \varepsilon>0, \exists\, n_0>0$, s.t. $\forall n>n_0$, $-\varepsilon\leq \frac{f(n)}{g(n)}\leq\varepsilon$

$\Rightarrow \forall\, \varepsilon>0, \exists\, n_0>0$, s.t. $\forall n>n_0$, $\boxed{0\leq f(n)}\leq \varepsilon\cdot g(n)$ #

3. Summary

① Insertion Sort

- Take card (elements) one by one
  Insert each one into the right place in the sorted pile
- Proof of correctness: Loop Invariant
  - ▷ Initialization ▷ Maintenance ▷ Termination
- Runtime Analysis
  - ▷ Worst Case: $c \cdot (n-1) + c' \cdot \frac{n(n-1)}{2} = O(n^2)$
  - ▷ Best Case: $O(n)$
  - ▷ Average Case: $O(n^2)$

$\sum_{j=2}^{n} (j-1) \downarrow \sum_{j=2}^{n} \frac{(j-1)}{2}$

- In-place Algorithm
  - ▷ "operates on the original array"
  - ▷ "does not require additional space (apart from a small constant extra space)"

② Asymptotics

- Memorize the definitions (might need them for formal proofs)
- Use intuition ("$\leq / < / \geq / > / =$", highest-order term)
- Consider $f(n)/g(n)$ as $n \to \infty$ (only if limit exists!!!)