Let f and g be functions $\mathbb N \rightarrow \mathbb R_{\geq 0}$ for the remainder of this summary.

$\newcommand\bigO{\mathcal O}$ $\newcommand\bigTh{\Theta}$ $\newcommand\bigOm{\Omega}$ $\newcommand\smallo{\mathcal o}$ $\newcommand\smallth{\theta}$ $\newcommand\smallom{\omega}$ $\newcommand\nat{\mathbb N}$ $\newcommand\rpos{\mathbb R_{\geq 0}}$

Big O

> Definition
>
> $f \in \bigO (g)$ iff:
>
> $\exists n_0 \in \nat : \exists c \in \rpos : \forall n \geq n_0 : f(n) \leq cg(n)$

Big Omega

> Definition
>
> $f \in \bigOm (g)$ iff:
>
> $\exists n_0 \in \nat : \exists c \in \rpos : \forall n \geq n_0 : c > 0 \text { and } f(n) \geq c g(n)$

We need $c > 0$ otherwise the relation might be $f(n) \geq 0$ which is too easy to satisfy. Also note that $1/c$ exists and $1/c > 0$, so that we can also write $g(n) \leq 1/c \cdot f(n)$ and we have:

> Lemma
>
> $f \in \bigO (g)$ iff. $g \in \bigOm (f)$

Big Theta

> Definition $f \in \bigTh (g)$ iff:
>
> $f \in \bigO (g)$ and $f \in \bigOm (g)$

In other words,

$\exists n_0 \in \nat : \exists b,c \in \rpos : \forall n \geq n_0 : c > 0 \text { and } bg(n) \geq f(n) \geq c g(n)$

Some properties

> Proposition If $f \in \bigO (g)$ and $g \in \bigO (h)$, then $f \in \bigO (h)$

> Proof: We are given that there are natural numbers n_0 and m_0 and two nonnegative constants c and d s.t. $$\forall n \geq n_0 : f(n) \leq cg(n)$$ and $$\forall n \geq m_0 : g(n) \leq dh(n)$$ This implies that if we set $M = \max {n_0,m_0}$, we have for all $n\geq M$ that $f(n) \leq cdh(n)$, so that $f \in \bigO (h)$.

How to calculate runtime complexity

The following steps:

- denote cost and repetition count in every singleton statement.
- add all costs multiplied by repetition count.

Example:

```
bool isIn (int xs [] , int x) {
    int i = 0;                  //c_1, 1
    while (i < xs.length()) {   //c_2, n
        if(x == xs[i]) {        //c_3, n
            return true;        //c_4, 1
        }
    i++;                        //c_5, n
    }
    return false;               //c_6, n
}
```

total cost is $T(n) = (c_1 + c_4 + c_6) + (c_2 + c_3 + c_5)n \in \bigO (n)$, clearly.

Worst case: gives an upper bound that holds for *every input*.