Useful Formulas

In this lesson, we'll study some mathematical formulae that make calculating time complexity easier!

We'll cover the following

- Formulas
- General Tips

Formulas#

Here is a list of handy formulas which can be helpful when calculating the time complexity of an algorithm:

Summation	Equation
$(\sum_{i=1}^{n} c) = c + c + c + \cdots + c$	cn
$(\sum_{i=1}^{n} i) = 1 + 2 + 3 + \cdots + n$	$\frac{n(n+1)}{2}$
$(\sum_{i=1}^{n} i^2) = 1 + 4 + 9 + \cdots + n^2$	$\frac{n(n+1)(2n+1)}{6}$

Summation	Equation
$\left(\sum_{i=0}^{n} r^i\right) = r^0 + r^1 + r^2 + \cdots + r^n$	$\frac{(r^{n+1}-1)}{r-1}$
$\sum_{i=0}^{n} 2^i = 2^0 + 2^1 + ... + 2^n$	$2^{n+1} - 1$

Some of the formulas dealing with logarithmic expressions:

Logrithmtic expressions	Equivalent Expression
$log\ (a\ *\ b)$	$log\ (a) + log\ (b)$
$log\ (a\ /\ b)$	$log\ (a) - log\ (b)$
$log\ a^n$	$n\ log\ a$
$\sum_{i=1}^{n} log\ i = log\ 1 + log\ 2 + ... + log\ n$ $= log(1.2...n)$	$log\ n!$

General Tips#

1. Every time a list or array gets iterated over $c \times length$ times, it is most likely in $O(n)$ time.
2. When you see a problem where the number of elements in the problem space gets halved each time, that will most probably be in $O(logn)$

runtime.

3. Whenever you have a singly nested loop, the problem is most likely in quadratic time.

Back

Other Common Asymptotic Notations ...

Next

Common Complexity Scenarios

Completed

Report an Issue