

Task 1

There are different ways of solving a problem in programming. However, each method may require different time to complete and different amount of memory for computation. Thus, measuring algorithm efficiency is important.

Some methods are better than other by requiring less time to complete and less memory for computation. At here, time refer to the amount of time require to complete, and space refer to the amount of memory require for computation.

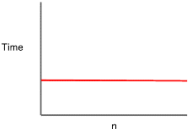
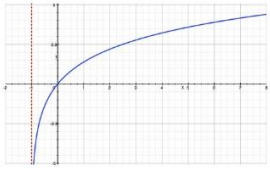
There must be a way to compare the performance of different types of algorithm without bias. The computation power of the computer should not be taken in as a factor since one algorithm could complete in different amount of time when perform on different type of computer. Computer scientist had proposed the usage of mathematical notation to describe the algorithm, instead of relying on the absolute value. The limiting behavior of a function when the argument approach towards infinity or a certain value could be described by using mathematical notation, which is big O notation.

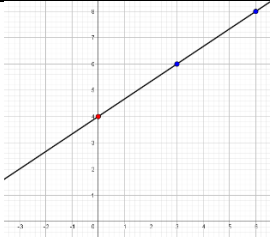
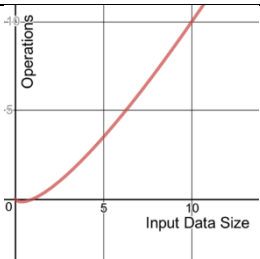
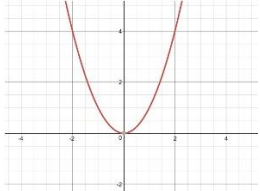
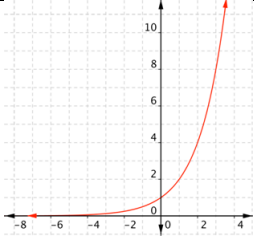
The runtime of the algorithm also depends on the input. A best-case scenario happens when optimal input is chosen and require least time to complete. A worst-case scenario happens when pessimal input is chosen and require longest time to complete.

Reference

CORMEN, T. H., & CORMEN, T. H. (2001). *Introduction to algorithms*. Cambridge, Mass, MIT Press.

Task 2

Time-Space Complexity	Graph	Description	Example
Constant Time $O(1)$	 A graph showing a horizontal red line on a coordinate system. The vertical axis is labeled 'Time' and the horizontal axis is labeled 'n'. The line indicates that the time taken is constant and does not change with the input size n.	No matter how many elements we are working with, the algorithm will always take the same amount of time	#1 <pre>int sum(int x, int y) { Return x + y; }</pre> #2 random access of array #3 pop element from stack
Logarithmic Time $O(\log n)$	 A graph showing a blue curve on a coordinate system. The curve starts at the origin and increases at a decreasing rate, characteristic of a logarithmic function. The horizontal axis is labeled 'n' and the vertical axis is labeled 'Time'.	An algorithm is said to take logarithmic time when $T(n) = O(\log n)$.	#1 binary search #2 search certain value on binary tree #3 dictionary search

Linear Time $O(n)$		Iterating through all elements in a collection of data.	<p>#1 for (int i = 0; i <= n; i++) { Cout << i; }</p> <p>#2 search certain element in linked list</p> <p>#3 check if there exist duplicate value in an array</p>
Quasilinear Time $O(n \log n)$		If perform algorithm with logarithm time complexity n time.	<p>#1 Fast Fourier Transforms</p> <p>#2 Mergesort</p> <p>#3 Heapsort</p>
Quadratic Time $O(n^2)$		Every element in a collection must be compared to every other element.	<p>#1 Bubble Sort</p> <p>#2 Insertion Sort</p> <p>#3 Selection Sort</p>
Exponential Time $O(2^n)$		an algorithm is exponential time if $T(n)$ is bounded by $O(2^{nk})$ for some constant k .	<p>#1 break a password by testing every possible combination</p> <p>#2 solve n-queens problem</p> <p>#3 brute force travelling sales man problem</p>