// Constant runtime - Big O Notation: "O (1)"

function log(array) {

console.log(array[0]);

console.log(array[1]);

}

log([1, 2, 3, 4]);

log([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]);

// Linear runtime - Big O Notation: "O (n)"

function logAll(array) {

for (var i = 0; i < array.length; i++) {

console.log(array[i]);

}

}

logAll([1, 2, 3, 4, 5]);

logAll([1, 2, 3, 4, 5, 6]);

logAll([1, 2, 3, 4, 5, 6, 7]);

// Exponential runtime - Big O Notation: "O (n^2)"

function addAndLog(array) {

for (var i = 0; i < array.length; i++) {

for (var j = 0; j < array.length; j++) {

console.log(array[i] + array[j]);

}

}

}

addAndLog(['A', 'B', 'C']); // 9 pairs logged out

addAndLog(['A', 'B', 'C', 'D']); // 16 pairs logged out

addAndLog(['A', 'B', 'C', 'D', 'E']); // 25 pairs logged out

// Logarithmic runtime - Big O Notation: O (log n)

function binarySearch(array, key) {

var low = 0;

var high = array.length - 1;

var mid;

var element;

while (low <= high) {

mid = Math.floor((low + high) / 2, 10);

element = array[mid];

if (element < key) {

low = mid + 1;

} else if (element > key) {

high = mid - 1;

} else {

return mid;

}

}

return -1;

}