Big O

# Exercise 01:

What is the complexity of the below code ?

|  |
| --- |
| function sumAndProduct(array) {  var sum = 0;  var product = 1;  for (var i = 0; i < array.length; i++) {  sum += array[i];  }  for (var i = 0; i < array.length; i++) {  product \*= array[i];  }  console.log(sum + ', ' + product);  }  Time : O(n)  Space : O(1) |

# Exercise 02:

What is the complexity of below code?

|  |
| --- |
| function printPairs(array) {  for (var i = 0; i < array.length; i++) {  for (var j = 0; j < array.length; j++) {  console.log(array[i] + ',' + array[j]);  }  }  }  Time : O(n^2)  Space : O(1) |

# Exercise 03:

What is the complexity of below code?

|  |
| --- |
| function printUnorderedPairs(array) {  for (var i = 0; i < array.length; i++) {  for (var j = i + 1; j < array.length; j++) {  console.log(array[i] + ',' + array[j]);  }  }  }  Time : O(n^2)  Space : O(1) |

# Exercise 04:

What is the complexity of below code?

|  |
| --- |
| function printUnorderedPairs(arrayA, arrayB) {  for (var i = 0; i < arrayA.length; i++) {  for (var j = 0; j < arrayB.length; j++) {  if (arrayA[i] < arrayB[j]) {  console.log(arrayA[i] + ',' + arrayB[j]);  }  }  }  }  // O(n \* m)  // O(1) |

# Exercise 05:

What is the complexity of below code?

|  |
| --- |
| function printUnorderedPairs(arrayA, arrayB) {  for (var i = 0; i < arrayA.length; i++) {  for (var j = 0; j < arrayB.length; j++) {  for (var k = 0; k < 10000; k++) {  console.log(arrayA[i] + ',' + arrayB[j]);  }  }  }  }  // O(n \* m) |

# Exercise 06:

What is the complexity of below code and what it does ?

|  |
| --- |
| function r(array) {  for (var i = 0; i < array.length / 2; i++) {  var other = array.length - i - 1;  var temp = array[i];  array[i] = array[other];  array[other] = temp;  }  }  // I = 0; 4 operations  // I = 1; 4  // I = n / 2; 4  // 4 + 4 + … 4 => (n / 2 + 1) \* 4 => O(2n + 4) => O(n)  // O(1) |

# Exercise 07:

The following code sums the values of all nodes in a binary tree.

What is its complexity ?

|  |
| --- |
| function sum(node) {  if (node === undefined) {  return 0;  }  return sum(node.left) + node.value + sum(node.right);  }  // 1 => 2 lan tinh  // 1 + 1 left + 1 + 1 right  // 1 + 2 + 2 \* 2 + 2 \* 2 \* 2 + 2 ^ (n-1)=> 2 ^ n  // sum(1) => sum(2) => sum(4)  Sum(2) => sum(5) |

# Exercise 08:

The following code computes n! (n factorial). What is its complexity ?

|  |
| --- |
| function factorial(n) {  if (n < 0) {  return -1;  } else if (n == 0) {  return 1;  }  return n \* factorial(n - 1);  }  // f(n) => n \* f(n – 1)  f(n – 1) => n – 1 \* f(n – 2)  f(n – 2) => n – 2 \* f(n – 3)  …  N = 0 => 1  // O(n)  O(n) |

# Exercise 09:

The following code computes the Nth Fibonacci number

What is its complexity ?

|  |
| --- |
| function fib(n) {  if (n <= 0) {  return 0;  } else if (n == 1) {  return 1;  }  return fib(n - 1) + fib(n - 2);  }  Time : O(2^n)  Space : O(n^2) |

# Exercise 10:

The following code prints all Fibonacci numbers from 0 to n.

What is its complexity ?

|  |
| --- |
| function fib(n) {  if (n <= 0) {  return 0;  } else if (n == 1) {  return 1;  }  return fib(n - 1) + fib(n - 2);  }  function printAllFib(n) {  for (var i = 0; i < n; i++) {  console.log(i + ':' + fib(i));  }  }  Time : O(2^n)  Space : O(n^2) |