# 練習課題1

次のそれぞれの関数の計算量がどの程度になるか予想して、O記法で表してみてください。

## 関数1

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | function uniq1(array) {  const knownElements = {};  const uniquedArray = [];  for (const elem of array) {  if (elem in knownElements)  continue;  uniquedArray.push(elem);  knownElements[elem] = true;  }  return uniquedArray;  } |

## 関数2

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | function uniq2(array) {  const uniquedArray = [];  for (const elem of array) {  if (uniquedArray.indexOf(elem) < 0)  uniquedArray.push(elem);  }  return uniquedArray;  } |

## 関数3

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | function uniq3(array) {  const knownElements = new Set();  for (const elem of array) {  knownElements.add(elem);  }  return Array.from(knownElements);  } |

## 関数4

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | function uniq4(array) {  const knownElements = {};  const uniquedArray = [];  for (let i = 0, maxi = array.length; i < maxi; i++) {  if (array[i] in knownElements)  continue;  uniquedArray.push(array[i]);  knownElements[array[i]] = true;  }  return uniquedArray;  }; |

## 関数5

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | function uniq5(array) {  const uniquedArray = [];  for (const elem of array) {  if (!uniquedArray.includes(elem))  uniquedArray.push(elem);  }  return uniquedArray;  } |

## 関数6

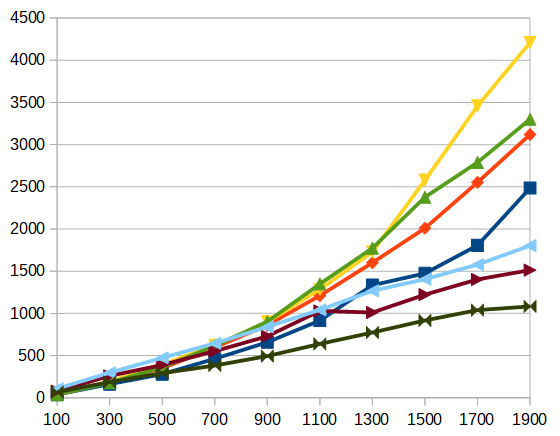
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | function uniq6(array) {  const knownElements = new Map();  const uniquedArray = [];  for (const elem of array) {  if (knownElements.has(elem))  continue;  uniquedArray.push(elem);  knownElements.set(elem, true);  }  return uniquedArray;  } |

# 練習課題2

課題1の6つの関数はすべて、与えた配列から重複を取り除くJavaScriptの関数です。

それぞれの関数について、配列の要素数が100、200、300、400……と増加していったときの計算量の増大の仕方が、予想したO記法の通りであるかどうかを確認したいです。

以下のようなグラフで確認するとして、必要な情報を得る方法を考えて、実装してみてください。



必要な情報を得る方法を考えて、実装してみてください。

グラフを作るためには、以下のような表形式で表現できる情報が必要です。

|  |  |
| --- | --- |
| 要素数 | 所要時間（ミリ秒） |
| 100 | 200 |
| 200 | 400 |
| 300 | 600 |
| … | … |

任意の処理の所要時間を計測する方法は、以下の要領です。

|  |  |
| --- | --- |
| 1  2  3  4 | let startAt = Date.now();  （何らかの処理）  let delta = Date.now() – startAt;  // この「delta」が所要時間をミリ秒で表した数値となる |

任意の個数の重複を含む配列を作成する処理は、以下の要領です。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | function prepareArray(length) {  const array = [];  for (let i = 0; i < length; i++) {  array.push(parseInt(Math.random() \* (length / 10)));  }  return array;  } |

実行環境の性能によっては、配列の要素数が小さいと、グラフで確認できるほどの所要時間がかからない場合があります。その場合、要素数の初期値や、1回ごとに増加させる要素数を、より大きな数字にするとよいでしょう。

JavaScriptが不得手な方は、別の言語で実装してもよいものとします。

また、近くの他の受講者の方と協力して実装してもよいものとします。

# 練習課題3

課題1の6つの関数よりも性能が良い（計算量が小さい・所要時間が短い）アルゴリズムがあるかどうかを考えて、実装し、性能を計測してみてください。

JavaScriptが不得手な方は、別の言語で実装してもよいものとします。

また、近くの他の受講者の方と協力して実装してもよいものとします。