# 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;  } |

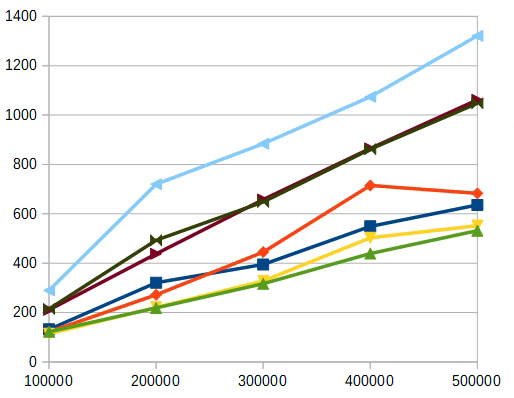
## 関数6

|  |  |
| --- | --- |
| 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、性能を比較調査するために、要素数300の配列から重複を取り除く操作を1万回、2万回、3万回、4万回、5万回……と、1万回ずつ増やしながら10万回まで実行して、以下のようなグラフで比較するのに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。

|  |  |
| --- | --- |
| 実行回数 | 所要時間（ミリ秒） |
| 10000 | 200 |
| 20000 | 400 |
| 30000 | 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;  } |

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つの関数について、要素数が増加し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実際に確認するために、必要な情報を得る方法を考えて、実装してみてください。

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。

# 練習課題4

課題1の6つの関数よりも計算量が少なくなるアルゴリズムがあるかどうかを考えて、実装してみてください。

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。