

コンピュータサイエンス基礎演習Ⅱ

第3回

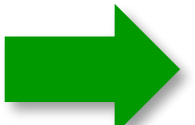

計算量のグラフを書いてみよう

本日の内容

- 関数の実行時間をグラフ化して、計算量の特徴を理解する
 - 問題のサイズ(データの数)を変えながら関数の実行時間を計測する
- 再帰呼出しを1行ずつ実行し、その様子を視覚的に理解する

1. 計算量について 復習

(再掲) 計算量とは

- アルゴリズムが、問題を解くのにどの程度の計算ステップ(計算時間)を要するかを表す指標のことを「計算量」と呼びます
- 一般には、入力の大きさに応じて計算時間がどのように変化するかに注目した「 O (オーダー)記法」を用いて、表現します
 - 計算量が $O(n^2)$ であれば、計算時間が入力の大きさの2乗に比例するということを意味します
 - 例) 線形探索の計算量  $O(n)$
 - 例) 二分探索の計算量  $O(\log n)$

(再掲)覚えておくべき計算量

オーダー記法	一般的な呼称	意味合い
$O(1)$	定数時間	n に関わらず時間は一定。
$O(\log n)$	対数時間	n の対数に比例して、時間がかかる。 (例：二分探索)
$O(n)$	線形時間	n に比例して、時間がかかる。 (例：線形探索)
$O(n \log n)$	準線形時間	n と n の対数の積に比例して、時間がかかる。 (例：実用的なソート)
$O(n^2)$	二乗時間	n の2乗に比例して、時間がかかる。 (例：遅いソート)
$O(C^n)$	指数時間	定数(2など)の n 乗に比例して、時間がかかる。 現実的に解くことが難しいと見なされる。

(再掲) 計算量の見積もり方の基本

- ここでは n を引数とした関数の計算時間をオーダー記法で見積もることを考えます
- この時の基本は、関数内の決まった処理の繰り返しの回数に注目します
 - 例) n によらず繰り返し回数は一定 $\rightarrow O(1)$
 - 例) 繰り返し回数は n に比例 $\rightarrow O(n)$

(再掲)for文を含む関数の実行時間①

- 以下のように、決まった処理がfor文で繰り返される場合は、それが何回繰り返されるかに着目し、計算量を見積もります
 - 二重ループの場合は、 n^2 回繰り返されます

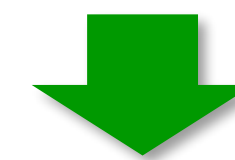
```
def func1(n):  
    s = 0  
    for i in range(n):  
        # Something to do  
        s += 1  
    return s
```

 $O(n)$

```
def func2(n):  
    s = 0  
    for i in range(n):  
        for j in range(n):  
            s += 1  
    return s
```

 $O(n^2)$

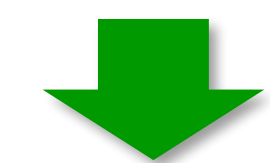
```
def func3(n):  
    s = 0  
    for i in range(2 ** n):  
        s += 1  
    return s
```

 $O(2^n)$

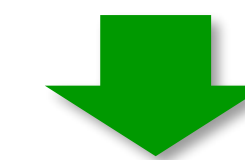
(再掲)for文を含む関数の実行時間②

- 以下のように繰り返す回数が定数である場合には、その部分の処理は一定時間で終わることに注意しましょう
 - 計算量の意味合いを考えれば、明らかですよネ？

```
def func4(n):  
    s = 0  
    for i in range(n):  
        for j in range(100):  
            s += 1  
    return s
```

 $O(n)$

```
def func5(n):  
    s = 0  
    for i in range(100):  
        for j in range(100):  
            s += 1  
    return s
```

 $O(1)$