

計算量を学ぼう！



計算量を学ぼう！

ぱうえる（けんた）

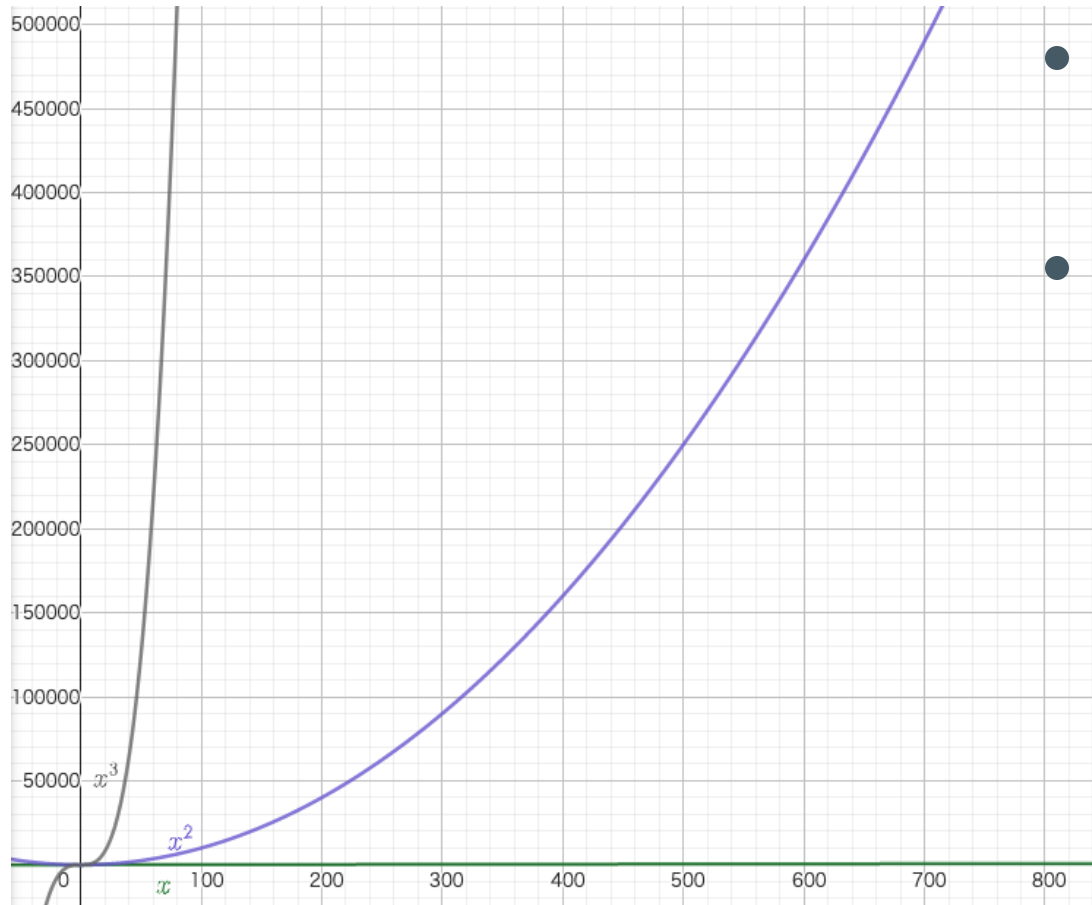
速いコードが書きたい！

でも速いコードってどうやって評価する??

- 「1,000,000個のデータに対して5秒で終了しました！」
 - データの個数が変わったらどうなる??
 - そもそもPythonで実行するかC言語で実行するかでも変わりそう

「データの大きさ」や「実行する環境」に依存しない評価方法が必要
→ 計算量の出番

オーダー記法 (1/2)



- n, n^2, n^3 では n が大きくなったとき値が大きく変化する
- 定数倍を考えないで、 n の項だけに注目すればいいのでは??

→ O (ランダウの記号) を用いる

オーダー記法 (2/2)

- 計算量は基本的にオーダー記法で書く
 1. 一番大きい項のみ残して表記する
$$n! > a^n > n^a > \log n > a \quad (a \text{ は定数})$$
 2. 定数倍は無視する

オーダー記法の例)

$$5n^3 + 4n^2 + 100n \longrightarrow O(n^3)$$

$$2^n + n^{100} + 10^9 n \longrightarrow O(2^n)$$

コードの計算量の調べ方

- n 回のループをする $\rightarrow O(n)$
- n 回のループの中で n 回のループをする（二重ループ）
 $\rightarrow O(n^2)$
- bit全探索（ n 個の要素についてある/ないの2通りを考える）
 $\rightarrow O(2^n)$
- n 個の順列を全て調べる $\rightarrow O(n!)$

ここまでの復習

このコードの計算量は??

```
# 1~n までの数の和を求める
n = int(input())

ans = 0
for i in range(1, n+1):
    ans += i

print(ans)
```

ここまでの復習（答え）

このコードの計算量は？？

```
# 1~n までの数の和を求める
n = int(input())

ans = 0
for i in range(1, n+1):
    ans += i

print(ans)
```

→ $O(n)$ (n までのループを1回している)

計算量の使い方

- 一般的なコンピュータが1秒間に計算できる回数は約 10^8 回
- n の大きさと実際の値は次の表のようになります

参考

- 計算量オーダーの求め方を総整理！～どこからlogが出て来るか～
<https://qiita.com/drken/items/872ebc3a2b5caaa4a0d0#1-3-計算量の使い方>

$\log n$	n	$n \log n$	n^2	n^3	2^n	$n!$
2	5	12	25	130	30	120
3	10	33	100	1000	1024	3628800
4	15	59	225	3375	32768	—
4	20	86	400	8000	1048576	—
5	25	116	625	15625	—	—
5	30	147	900	27000	—	—
7	100	664	10000	1000000	—	—
8	300	2468	90000	27000000	—	—
10	1000	9966	1000000	—	—	—
13	10000	132877	100000000	—	—	—
16	100000	1660964	—	—	—	—

計算量を落とすテクニック

今回は代表的なものを3つ紹介します。

- **公式を使う**
比較的単純な手法
- **累積和**
数列の区間の和を高速に求めるアルゴリズム
- **二分探索**
条件を満たす値があるかを高速に調べるアルゴリズム

公式を使う (1/3)

上のコードは、 $1 \sim n$ の和を求めるために $O(n)$ の計算をしています
($n = 100,000,000$ で2.6秒くらい必要)

```
In [6]: %%timeit
...: # 1~n までの数の和を求める
...: n = 100_000_000
...:
...: ans = 0
...: for i in range(1, n+1):
...:     ans += i
...:
2.61 s ± 3.82 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

公式を使う (2/3)

この公式を使えば、

$$\sum_{i=1}^n = \frac{1}{2}n(n+1)$$

```
# 1~n までの数の和を求める  
n = int(input())  
ans = n * (n + 1) // 2  
  
print(ans)
```

公式を使う (3/3)

```
In [9]: %%timeit
...: # 1~n までの数の和を求める
...: n = 100_000_000
...: ans = n * (n + 1) // 2
...:
...:
53.7 ns ± 3.87 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

なんと、53.7ナノ秒で終了！！

→ 約**5億倍**の高速化（ちょっと極端な例ではあるけど）

累積和 (/n)

あるたい焼き屋さんでは 毎日、売れた個数を記録しています。
営業開始から 7 日目までの売り上げは以下の通りでした。

1日目	2日目	3日目	4日目	5日目	6日目	7日目
20	50	30	10	30	0	40

2 日目から 5 日目までの売り上げの合計はいくらでしょうか？

$$\rightarrow 50 + 30 + 10 + 30 = 120 \text{ (個)}$$

累積和 (/n)

一般化してみると??

あるたい焼き屋さんでは、毎日売り上げを記録しています。

営業開始から i 日目の売り上げは A_i 円でした。

このとき、以下の問題に答えて下さい。

i 日目から j 日目までの売り上げの合計はいくらでしょうか？

累積和 (/n)

A_1	A_2	A_3	A_4	A_5	A_6	A_7
20	50	30	10	30	0	40

参考

- 計算量オーダーの求め方を総整理！ ～どこからlogが出て来るか～
<https://qiita.com/drken/items/872ebc3a2b5caaa4a0d0>