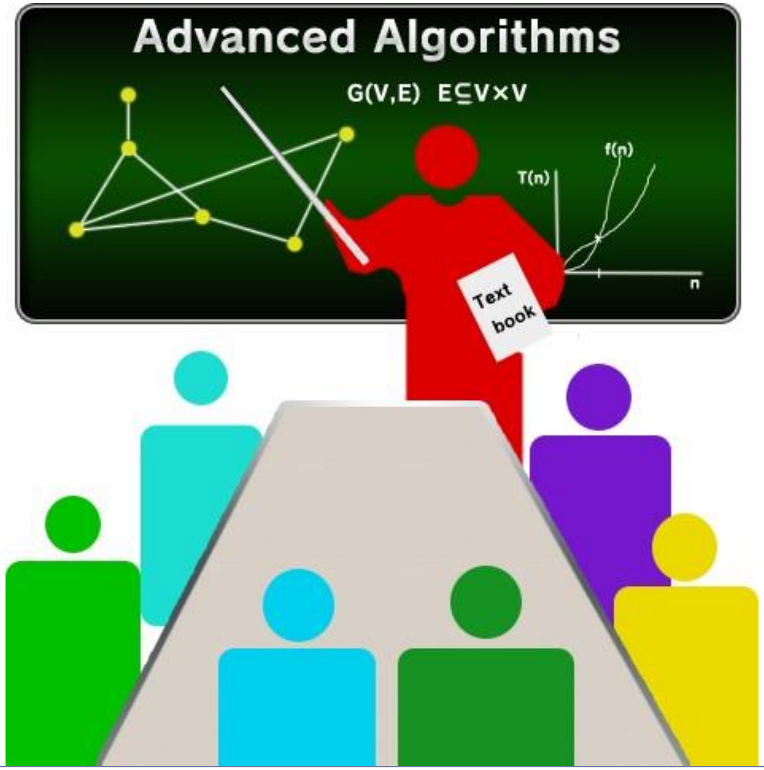

アルゴリズム特論 [AA2016]
Advanced Algorithms

Exercise 13. Randomized Algorithm

乱択アルゴリズム

- 乱数の性質を利用したアルゴリズム
- 処理の流れを乱数に任せる

- 乱数を使用することで、既存のアルゴリズムの高速化が実現する場合もある（Quicksort)

- 乱数を用いた素数判定アルゴリズム（フェルマーテスト）

【前提】フェルマーの小定理

フェルマー (Fermat) の小定理
p を素数とする.

1. $x \not\equiv 0 \pmod{p}$ のとき，等式 (1) が成り立つ.

$$x^{p-1} \equiv 1 \pmod{p} \tag{1}$$

2. 任意の x に対して等式 (2) が成り立つ.

$$x^{p} \equiv x \pmod{p} \tag{2}$$

- **p が素数であれば、 x^{p-1} を p で割った余りは 1**

【前提】フェルマーの小定理

フェルマー (Fermat) の小定理

p を素数とする.

1. $x \not\equiv 0 \pmod{p}$ のとき，等式 (1) が成り立つ.

$$x^{p-1} \equiv 1 \pmod{p} \tag{1}$$

2. 任意の x に対して等式 (2) が成り立つ.

$$x^p \equiv x \pmod{p} \tag{2}$$

逆

- x^{p-1}を p で割った余りは 1　ならば、 p は素数？

⇒ 必ず成り立つとは言えない。（反例は自分で示せ）

⇒ これを利用した素数判定アルゴリズム

【前提】フェルマーの小定理

フェルマー (Fermat) の小定理

p を素数とする.

1. $x \not\equiv 0 \pmod{p}$ のとき，等式 (1) が成り立つ.

$$x^{p-1} \equiv 1 \pmod{p} \tag{1}$$

2. 任意の x に対して等式 (2) が成り立つ.

$$x^p \equiv x \pmod{p} \tag{2}$$

- x を適当に（乱数で）決めて
- p が素数になりそうかどうかを調べる

【前提】フェルマーの小定理

フェルマー (Fermat) の小定理

p を素数とする.

1. $x \not\equiv 0 \pmod{p}$ のとき，等式 (1) が成り立つ.

$$x^{p-1} \equiv 1 \pmod{p} \tag{1}$$

2. 任意の x に対して等式 (2) が成り立つ.

$$x^p \equiv x \pmod{p} \tag{2}$$

- x^{p-1} (mod. p) **を計算して、１になれば素数** かなあ？

⇒ **こういうのはプログラムが得意、… だが実装で様々な問題に出くわす**

x^{p-1} (mod. p)

- プログラムで素数判定を行う

⇒ 一般に、p は大きい数になることが想定される

（小さい p ではプログラムを使う意味も無いし…）

- x^{p-1} の時点であっという間にオーバーフローしそうだ

- x^{p-1} (mod. p) = x (mod. p) ・ x (mod. p) ・ x (mod. p) ・…

という変形をしてはどうか？

⇒　それ、本当に x に関わらず成り立つの？（剰余と乗法の両立）

証明して、確認してから使うこと（演習課題）

フェルマーテスト　（pは素数か？）

- xを（2からp-2の範囲で）乱数を用いて決定する

  ⇒ x = 1 や p-1 でフェルマーの小定理が成立するのは自明だから
- x^{p-1} (mod. p) を計算する
- 結果が１になるか？

⇒　１ならｐは素数かもしれない

⇒　１じゃないならｐは絶対に合成数だ　→ 処理は即終了でOK

- かもしれない

このアルゴリズムは「フェルマーの小定理の逆」を行っているので、結果が１になったからといって「素数である」と即確定するわけではない。

いろんなｘで実験してみて、あらゆる場合で１になるなら「素数である」と認定することにしている。

フェルマーテストをすり抜けて来るイヤな奴ら

- **偽素数**

乱択素数判定アルゴリズムを通過してくる、
「合成数のくせに素数のふりをする」数の総称

- **カーマイケル数**

フェルマーテストの大敵。乱数で選択されたあらゆるｘに対して、 x^{p-1} (mod. p) ≡ 1 をクリアしてくる油断ならない偽素数ｐ

・あらゆるｘがｐと互いに素となる ⇒ gcd(x, p)=1

カーマイケル数について、手で素因数を探そうとすると実際難しくて、素因数分解をすることが大変であることがわかる

⇒ 乱数でｘを求めても、 x^{p-1} (mod. p) ≡ 1 にならないケースがなかなか引っかからないので、 誤判定を招く要因となる

乱択素数判定アルゴリズムのまとめ

フェルマーテストは、pが素数であるかどうかを確率的に判定する

- ｘを乱数で決める
- x^{p-1} (mod. p) を計算する
- 結果が１になるか？

⇒　１ならｐは素数かもしれない

⇒　１じゃないならｐは絶対に合成数だ

- ｐにカーマイケル数を入れると、合成数が誤判定をされる
- 素数である確信が持てるまで、十分に大きな試行回数をとって、様々なｘで検証を行う
- 乱数アルゴリズムの精度を上げるには、試行回数を上げることが重要（その分、実行時間はかかっちゃうけれど…）