科目名：プログラミング技法 II　　担当教員名：新田　直子
学籍番号：01Z18032　　名前：トゥチェク・トム

2019 年 07 月 15 日

課題１１：　アフィン暗号

課題１１－１～１１－１０：

課題１１－１では、指示通り、ユークリッドの互除法で、ｇｃｄを計算する関数を作った。ウィキペディアからの疑似コードに基づいた。

表 1　ユークリッドの互除法の疑似コード
出展：https://en.wikipedia.org/wiki/Euclidean_algorithm#Implementations

```
function gcd(a, b)
        while b ≠ 0
       t := b;
       b := a mod b;
       a := t;
    return a;
```

課題１１－２では、ランダムな二つの鍵を生成する。aとNは互いに素であるという条件を満たすまで、繰り返す。

課題１１－３では、「i = (i * a + b) % N」で、テキストを暗号化する関数を作った。

課題１１－４では、encrypt_file という関数を作った。ファイルから読み込み、別のファイルに、暗号化されたテキストを保存する。使われた鍵も、また別のファイルに保存する。

課題１１－５では、拡張ユークリッドの互除法を実装した。またウィキペディアからの疑似コードに基づいて、作った。

表 2　拡張ユークリッドの互除法の疑似コード
出展：https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm#Modular_integers

```
function inverse(a, n)
    t := 0;     newt := 1;
    r := n;     newr := a;
    while newr ≠ 0
        quotient := r div newr
        (t, newt) := (newt, t - quotient * newt)
        (r, newr) := (newr, r - quotient * newr)
    if r > 1 then return "a is not invertible"
    if t < 0 then t := t + n
    return t
```

課題１１－６では、前に作った関数を用いて、暗号化されたファイルを解読する。鍵は別のファイルに保存されたので、そのファイルから読み込み、解読のために鍵を使う。

課題１１－７では、Regex を用いて、もらったキストの英語単語率を計算する関数を作った。Regex の代わりに、Regex より速い方法も試したが、アルファベットのみの単語が求められたら、Regex のほうは一番いいと思う。

課題１１－８では、decrypt_brute_force という関数を作った。あり得るすべての鍵を試して、前の英語単語率の関数で、正しい結果を決定する。結構長い時間がかかるが、働く。

課題１１－９では、Brute force ではなく、洗練された方法を使う。まず英語の文字の頻度をランダムなテキストで計算し、それで辞書を作る。その辞書の最高の頻度の文字と、最低の頻度の文字で、暗号化されたテキストの鍵を推定できる。だが、実装するときに、二つの問題が出てきた。一つ目は、Modulo を使う公式を逆算することは思ったより難しかった。結局、https://www.expii.com/t/solving-linear-congruence-ax-b-mod-n-3389 からの解説でできた。二つ目は、一番低い頻度の文字を使うことは無理だということがすぐに明らかになった。元のテキストに特殊文字があれば、それが前に作った辞書に当たる文字であるチャンスが低い。一番低い頻度の文字だけではなく、最もの３つの文字などを使っても、正しい結果が出てこなかった。そのため、その代わりに、５つ高い頻度の文字を取り、そのあらゆる組み合わせ、それで鍵を推定できた。最低頻度文字を使う古いコードはコメントされ、まだ残っている。

課題１１－１０では、前の二つのアルゴリズムを比較する。Brute force のほうは、約３０倍遅くて、候補となった鍵対の数は４０倍以上。別のテキストでも実行し、その結果は下記の表に示す。

表３　課題 11－10 の結果

Brute Force method successful. --Result:-- [Emma by Jane Austen 1816] VOLUME I CHAPTER I

Emma Woodhouse, handsome, clever, and rich, with a comfortable home
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.

[…]

"With a great deal of pleasure, sir, at any time," said Mr. Knightley,
laughing, "and I agree with you entirely, that it will be a much
better thing. Invite him to dinner, Emma, and help him to the best
of the fish and the chicken, but leave him to chuse his own wife.
Depend upon it, a man of six or seven-and-twenty can take care
of himself."
 --End Result--
Keys found: 71 56
Number of key candidates: 4240
Time needed: 1078.8317229747772 s

Character Frequency Analysis successful.
--Result:--
 [Emma by Jane Austen 1816]

VOLUME I

CHAPTER I

Emma Woodhouse, handsome, clever, and rich, with a comfortable home
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.

[…]

"With a great deal of pleasure, sir, at any time," said Mr. Knightley,
laughing, "and I agree with you entirely, that it will be a much
better thing. Invite him to dinner, Emma, and help him to the best
of the fish and the chicken, but leave him to chuse his own wife.
Depend upon it, a man of six or seven-and-twenty can take care
of himself."
 --End Result--
Keys found: 71 56
Number of key candidates: 100
Time needed: 35.756192684173584 s

表 4　課題 11－10 の結果、別の暗号化されたテキストで

Brute Force method successful.
--Result:--
 Hanlon's razor
From Wikipedia, the free encyclopedia

Hanlon's razor is an aphorism expressed in various ways, including:

   "Never attribute to malice that which is adequately explained by stupidity."[1]

An eponymous law, probably named after a Robert J. Hanlon, it is a philosophical razor which suggests a way of eliminating unlikely explanations for human behavior.

Inspired by Occam's razor,[2] the aphorism became known in this form and under this name by the Jargon File, a glossary of computer

programmer slang.[3][1] Later that same year, the Jargon File editors noted lack of knowledge about the term's derivation and the existence of a similar epigram by William James.[4] In 1996, the Jargon File entry on Hanlon's Razor noted the existence of a similar quotation in Robert A. Heinlein's novella Logic of Empire (1941), with speculation that Hanlon's Razor might be a corruption of "Heinlein's Razor".[5] (The character "Doc" in Heinlein's story described the "devil theory" fallacy, explaining, "You have attributed conditions to villainy that simply result from stupidity.")[6]

--End Result--
Keys found: 38 59
Number of key candidates: 4240
Time needed: 70.81352376937866 s

Character Frequency Analysis successful.
--Result:--
Hanlon's razor
From Wikipedia, the free encyclopedia

Hanlon's razor is an aphorism expressed in various ways, including:

"Never attribute to malice that which is adequately explained by stupidity."[1]

An eponymous law, probably named after a Robert J. Hanlon, it is a philosophical razor which suggests a way of eliminating unlikely explanations for human behavior.

Inspired by Occam's razor,[2] the aphorism became known in this form and under this name by the Jargon File, a glossary of computer programmer slang.[3][1] Later that same year, the Jargon File editors noted lack of knowledge about the term's derivation and the existence of a similar epigram by William James.[4] In 1996, the Jargon File entry on Hanlon's Razor noted the existence of a similar quotation in Robert A. Heinlein's novella Logic of Empire (1941), with speculation that Hanlon's Razor might be a corruption of "Heinlein's Razor".[5] (The character "Doc" in Heinlein's story described the "devil theory" fallacy, explaining, "You have attributed conditions to villainy that simply result from stupidity.")[6]

--End Result--
Keys found: 38 59
Number of key candidates: 100
Time needed: 2.4813530445098877 s

課題１１－１１：

課題１１－１１では、条件を満たさない鍵で暗号化してみる。下記の表に示した通り、「a,b = 1,0」や「a,b = N+1,N+3」の鍵対で、別に問題はなさそう。なぜかというと、私もよくわからない。もしかして、ある文字があれば、働かないかもしれないが、私のテストケースにはその文字がなかったので、普通のようにできた。「a,b = N/3,1」の鍵対の場合には、もう解読できなかった。なぜなら、下記の表に示した通り、暗号化されたテキストは三つの文字だけを使いになってしまったから。

表５　課題１１－１１の結果

a,b = 1,0 decryption using keys, Success: True
a,b = 1,0 decryption character frequency analysis, Success: True
---
a,b = N+1,N+3 decryption using keys, Success: True
a,b = N+1,N+3 decryption character frequency analysis, Success: True
---
a,b = N/3,1 decryption using keys, Success: False
a,b = N/3,1 decryption character frequency analysis, Success: False

--Encoded result--

c3BccBc33c3Bcc33cc33ccBc3B3c333BBB3ccBB3BBc3ccc3B3c333c3BccBc33c3Bcc3c333B333Bccc333Bc3cB33B33cB333cccc33B33333cBccc3cB3
c3333333cB3Bc33BBccBcBB3Bc333cccB3BB3B3BBccB3c3333BBc3BBc33Bc3c3cBB33B333Bc3c3cB3B3B3c33BB3B3cB33cc33c3B333ccB3Bc33B3
3B333cBBc3333cBBcB3BB3c3BccB33cB3c33333Bccc3c3Bcc3c3c3Bcc3BBccB33c33B3B3333B333cc3Bcc3cB3BcB33cBccBBc33Bc3c3B3BccB33ccc
3Bc33B3BBB33cccB3333B33ccB33B333cc33c33c3Bcc3BBc3BBB333Bccc333BBc33B3BBcBB3cB3BBc33ccc333B33cB3Bc3BBc33B33B3B33BBB3B
3c3cB33ccB33333cc333c33cc3cc33cBBc33cc3c333Bc33c3B3BBccB3c333BBc3BB3B3333B33B3c33BBB3B3c3cB33ccB3B3cBcc33BcBB33c3cB3cc3
BBcBcB33B33BccB3BBB3BBc3c333Bcc33BccB33B33BBB3Bcc3BBBcB3cc3333c3cc3c3B3c3c333B33ccccc333B33B3BB3c33B33ccc33BBB3B3c3cB3
3ccB3BBBc33cB3c3BccBc3333Bcc3BcBB33BBB3Bcc3BBBcB3cc3333c3cc3c3BccB3BccB3cB33cBBcB3BB3cBcBcBcBc33Bc3Bcc333c3cc3cc3c33ccB
333c33B33BcBB333Bccc3BccB3BB3B3c3BccBc3333Bcc33c3BB3BB333ccccc3BccB3cc33cBcBcBcBc3333Bcc3BBBc33cBB3cB3c3cBBc33Bcc33cB3c
BcBcBcBc333Bcc333B3cccBB33BBB333B3cc3BBBcc333c3cc3c333Bc3c3cBcB3333Bcc3B33B33BBccBcBB33ccB3cBccB33Bc33ccc3cB33BB3B33c33
c33cB3ccB3ccc333Bc3c3cB3B3BBcc3