Laboratorium kryptograficzne dla licealistów 4

Projekt „Matematyka dla ciekawych świata”
Łukasz Mazurek

20.04.2017

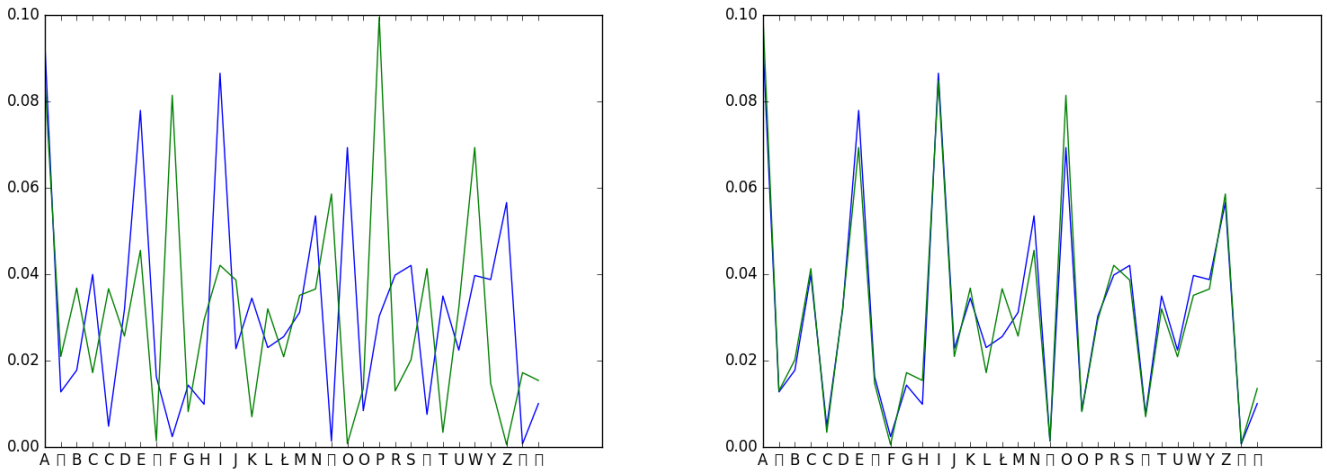
1 Poszukiwanie klucza

Szyfr Cezara udało nam się złamać już kilkukrotnie. Za każdym razem metoda postępowania była podobna: wiadomo, że są tylko 32 możliwe klucze, więc sprawdzaliśmy po kolei każdy z nich i „na oko” stwierdzaliśmy, który klucz jest dobry. Ocena „jakości” poszczególnych kluczy odbywała się na jeden z dwóch sposobów:

- roszyfrowywaliśmy tekst każdym potencjalnym kluczem i sprawdzaliśmy, czy otrzymane litery składają się w istniejące słowa,
- rysowaliśmy wykres częstości występowania liter w szyfrogramie i sprawdzaliśmy, przy jakim przesunięciu wykres „pokrywa się” z wykresem częstości występowania liter w naszym języku.

W obu przypadkach koniec końców musieliśmy „na oko” coś ocenić — czy to sens logiczny otrzymanych słów, czy „pokrywanie się” wykresów. Czy da się wyeliminować czynnik ludzki z tej procedury? Czy da się zautomatyzować proces łamania szyfru Cezara, tak aby komputer bez naszej pomocy potrafił rozszyfrować np. tysiąc różnych szyfrogramów w ułamku sekundy? Tym właśnie zajmiemy się na dzisiejszych zajęciach.

1.1 Odległość pomiędzy rozkładami



(a) (b)

Rysunek 1: Przykład rozkładów (a) niepokrywających się i (b) pokrywających się.

Na poprzednich zajęciach tworzyliśmy listy częstości występowania poszczególnych liter, a następnie rysowaliśmy odpowiadające im wykresy i sprawdzaliśmy, czy się pokrywają. Takie listy częstości są przez matematyków nazywane *rozkładami*. Gdy zobaczymy dwa takie rozkłady na jednym wykresie, bez trudu „na oko” ocenimy, czy rozkłady się pokrywają, czy nie. Jeżeli np. porównamy rys. 1a i rys. 1b, to chyba

każdy się zgodzi, że rozkłady na rys. 1a są zupełnie różne, a rozkłady na rys. 1b pokrywają się w dużym stopniu.

Czy istnieje jakaś wielkość matematyczna mierząca „pokrywanie się" rozkładów? Tak! Wielkością tą jest ***odległość między rozkładami***. Mówiąc nieformalnie, *odległość między rozkładami* to *suma pionowych odległości pomiędzy wykresami*. Formalnie, jeśli uznamy, że nasze rozkłady to funkcje $f(x)$ i $g(x)$, to odległość pomiędzy rozkładami f i g zdefiniowana jest jako suma:

$$odl(f,g) = |f(0) - g(0)| + |f(1) - g(1)| + |f(2) - g(2)| + \ldots + |f(31) - g(31)|$$

Policzmy najpierw na palcach, ile wynosi *odległość między rozkładami* na rys. 1a i rys. 1b, jeżeli ograniczymy się do pierwszych 5 liter alfabetu (A, Ą, B, C, Ć):

rys. 1					
	A	Ą	B	C	Ć
f	0,093	0,013	0,018	0,040	0,005
g	0,085	0,021	0,037	0,017	0,037
$\lvert f-g\rvert$	0,008	0,008	0,019	0,023	0,032
$odl(f,g)$	**0,090**				

rys. 2					
	A	Ą	B	C	Ć
f	0,093	0,013	0,018	0,040	0,005
g	0,099	0,013	0,020	0,041	0,003
$\lvert f-g\rvert$	0,006	0,000	0,002	0,001	0,002
$odl(f,g)$	**0,011**				

Zgodnie z intuicją, odległość między rozkładami na rys. 1a jest dużo większa niż odległość między rozkładami na rys. 1b. W ten sposób możemy zautomatyzować procedurę łamania szyfru Cezara: próbujemy rozszyfrować szyfrogram każdym z 32 możliwych kluczy i w każdym przypadku obliczamy rozkład częstości liter. Dla każdego rozkładu obliczamy jego *odległość* od rozkładu częstości liter w naszym języku. Prawidłowy klucz to ten, dla którego ta odległość jest najmniejsza.

Zadanie 1 *Napisz funkcję* `odl(f, g)`*, która obliczy odległość pomiędzy rozkładami* `f` *i* `g`*. Funkcja powinna przyjmować jako argumenty dwie listy* `f` *i* `g` *tej samej długości. Funkcja powinna zwracać jako wynik sumę wartości bezwzględnych różnic pomiędzy listami na poszczególnych pozycjach:*

$$|f(0) - g(0)| + |f(1) - g(1)| + \ldots$$

Do obliczania wartości bezwzględnej liczby możesz użyć wbudowanej funkcji Pythona **abs()**.

Używając napisanej funkcji, oblicz odległość pomiędzy rozkładami z rys. 1a dla pierwszych pięciu liter alfabetu. Wywołanie funkcji powinno wyglądać następująco:

```
f = [0.093, 0.013, 0.018, 0.040, 0.005]
g = [0.085, 0.021, 0.037, 0.017, 0.037]
print(odl(f, g))
```

```
0.09
```

1.2 Badanie rozkładów szyfru Cezara

Na poprzednich zajęciach obliczaliśmy rozkłady częstości liter dla różnych tekstów. Poniższy program oblicza rozkłady częstości liter w plikach `lalka.txt` i `tekst1.txt`:

```
alfabet = 'AĄBCĆDEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻ'

def zlicz(tekst):
  ile = [0] * 32
  suma = 0
  for znak in tekst:
    if znak in alfabet:
      poz = alfabet.index(znak)
      ile[poz] = ile[poz] + 1
      suma = suma + 1
```

```
  return [x / suma for x in ile]

plik = open('lalka.txt', 'r')
lalka = plik.read().upper()
plik.close()
ile_lalka = zlicz(lalka)

plik = open('tekst1.txt', 'r')
tekst1 = plik.read().upper()
plik.close()
ile_tekst1 = zlicz(tekst1)

print(ile_lalka)
print(ile_tekst1)
```

Zwróć uwagę na ostatnią linijkę definicji funkcji `zlicz`:

```
return [x / suma for x in ile]
```

Jest to niestandardowa postać pętli **for**, charakterystyczna dla Pythona. W liście `ile` mamy obliczone **bezwzględne** częstości występowania poszczególnych liter. Chcemy, aby funkcja `zlicz` zwracała **względne** częstości występowania liter, dlatego najpierw w zmiennej `suma` obliczamy łączną liczbę wszystkich liter w tekście, a następnie każdy element listy `ile` dzielimy przez tę sumę. Dokładnie to dzieje się w tej linijce. Linijkę tę możemy odczytać w następujący sposób: *zwróć listę, której kolejne elementy będą postaci* `x / suma` *dla* `x` *przyjmujących kolejne wartości z listy* `ile`.

Oczywiście moglibyśmy uzyskać ten sam efekt używając standardowej postaci pętli **for**:

```
for i in range(len(ile)):
  ile[i] = ile[i] / suma
return ile
```

Zostańmy jednak przy pierwotnej, krótszej konstrukcji.

Nie zamykając napisanego dotychczas kodu (funkcja `odl` jeszcze nam się przyda), otwórz kolejny pusty projekt na stronie `repl.it` w nowej karcie przeglądarki. Przepisz kod powyższego programu do otwartego właśnie projektu, a następnie dodaj do projektu pliki `lalka.txt` i `tekst1.txt` pobrane ze strony `ciekawi.icm.edu.pl`[1] Po uruchomieniu, program powinien wypisać dwie listy częstości dla obu plików, po 32 liczby każda:

```
[0.09311553008330047, 0.01275417469120047, 0.017781593459714907,
0.03994090849324392, 0.0048108530369476606, 0.031889303973207724,
0.07790952193114757, 0.016281102319573675, ... ]
[0.08480810604483309, 0.020959122770490664, 0.036782566451523356,
0.017211465056561872, 0.036643764313970435, 0.02567839544728989,
0.04552710111735721, 0.0013880213755291832, ... ]
```

Następnie, używając napisanej wcześniej funkcji `odl`, oblicz odległość pomiędzy otrzymanymi rozkładami `ile_lalka` i `ile_tekst1`.:

```
print(odl(ile_lalka, ile_tekst1))
```

```
0.826033617741158
```

[1] Przypomnienie: aby dodać do projektu w `repl.it` nowy plik tekstowy, należy kliknąć przycisk *Add new file* w lewym górnym rogu okna, zgodzić się na przejście do *Project mode*, a następnie jeszcze raz kliknąć przycisk *Add new file*, zmienić nazwę nowego pliku na pożądaną i skopiować do niego zawartość odpowiedniego pliku. Pliki o których mowa znajdują się na stronie `ciekawi.icm.edu.pl`, zakładka *Dla uczestników* → *Materiały - ćwiczenia* jako materiały do skryptu *Lab 3. Analiza częstościowa.*

Zadanie 2 *Napisz program, który w pętli obliczy wszystkie 32* rotacje *listy* `ile_tekst1` *i dla każdej rotacji wypisze linijkę zawierającą numer tej rotacji i po dwukropku odległość tej rotacji od listy* `ile_lalka`*:*

```
0: 0.826033617741158
1: 0.8483552668940029
2: 0.826649214255246
...
```

Przypomnienie: k-tą rotacją listy nazywamy jej cykliczne przesunięcie w lewo o k pozycji. Np. drugą rotacją listy [1, 2, 3, 4, 5, 6] jest lista [3, 4, 5, 6, 1, 2]. Zatem k-tą rotację listy możemy uzyskać poprzez sklejenie fragmentu listy od k-tej pozycji (włącznie) do końca oraz fragmentu listy od początku do k-tej pozycji (wyłącznie).

Jeśli poprawnie rozwiązałeś poprzednie zadanie, dla jednej z rotacji odległość powinna być zdecydowanie mniejsza niż dla pozostałych. Dzieje się tak dlatego, że tekst w pliku `tekst1.txt` powstał poprzez zaszyfrowanie pewnego tekstu szyfrem Cezara o pewnym kluczu k. Zatem jego rozkład częstości jest przesuniętym rozkładem częstości z pewnego tekstu napisanego w naszym języku, a więc odpowiednio przesunięty rozkład będzie się „pokrywał" z rozkładem z pliku `lalka.txt`.

Zadanie 3 *Napisz funkcję* `znajdz_przesuniecie(f, g)`*, która zwróci przesunięcie rozkładu* `g` *względem rozkładu* `f`*, czyli takie* `k`*, że* `k`*-ta rotacja rozkładu* `g` *jest* najbliżej *rozkładu* `f`*.*

Funkcja powinna używać dwóch zmiennych: `min_odleglosc` *oraz* `najlepsze_przesuniecie`*. W każdym kroku zmienna* `min_odleglosc` *powinna przechowywać najmniejszą odległość, jaką udało nam się do tej pory uzyskać, a zmienna* `najlepsze_przesuniecie` *powinna przechowywać numer rotacji, dla której ta najmniejsza odległość wystąpiła. Funkcja powinna w pętli sprawdzać kolejne przesunięcia* `k` *i w przypadku, gdy dane przesunięcie daje mniejszą odległość niż* `min_odleglosc`*, aktualizować obie zmienne.*

Użyj funkcji `znajdz_przesuniecie`*, aby znaleźć przesunięcie rozkładu* `ile_tekst1` *względem rozkładu* `ile_lalka`*:*

```
print(znajdz_przesuniecie(ile_lalka, ile_tekst1))
```

```
21
```

2 Szyfr Vigenere'a o nieznanej długości klucza

Podczas ostatnich zajęć (a dokładniej, w pracy domowej) udało nam się rozszyfrować szyfr Vigenere'a z kluczem o długości 5. Rozszyfrowanie polegało na tym, że patrzyliśmy na co piątą literę i traktowaliśmy te litery jako tekst zaszyfrowany szyfrem Cezara, a następnie rozszyfrowywaliśmy go, szukając najlepszego przesunięcia rozkładu częstości liter.

Co w przypadku, gdy nie znamy z góry długości klucza? Oczywiście moglibyśmy sprawdzać po kolei klucze długości 1, 2, itd. i „na oko" oceniać, przy jakiej długości klucza odpowiednie litery rzeczywiście układają się w szyfr Cezara. Spróbujmy zautomatyzować również ten etap.

Poniższa, zmodyfikowana funkcja `zlicz` oblicza rozkład liter, za każdym razem przeskakując o kilka liter w tekście. Dokładniej, wywołanie funkcji `zlicz(tekst, poczatek, krok)` zlicza litery rozpoczynając od litery na pozycji `poczatek` i za każdym razem przeskakując o `krok` liter do przodu. Np. wywołanie `zlicz(tekst, 0, 5)` obliczy rozkład w zbiorze liter na pozycjach 0, 5, 10, 15, 20, itd.

```
def zlicz(tekst, poczatek, krok):
  ile = [0] * 32
  suma = 0
  licznik = 0
  for znak in tekst:
    if licznik % krok == poczatek and znak in alfabet:
      poz = alfabet.index(znak)
      ile[poz] = ile[poz] + 1
```

```
      suma = suma + 1
    licznik = licznik + 1
  return [x / suma for x in ile]
```

Zwróć uwagę na dodane linijki: w każdym kroku pętli zwiększamy wartość licznika o 1 i zliczamy litery tylko w momentach, gdy reszta z dzielenia wartości licznika przez `krok` wynosi `poczatek`. Czyli dla `krok = 5` i `poczatek = 0` będziemy zliczać tylko litery z pozycji 0, 5, 10, 15, itd.

Zadanie 4 *Używając powyższej funkcji* ***zlicz*** *oraz napisanych wcześniej funkcji, oblicz rozkład częstości liter w co piątej literze tekstu z pliku* ***tekst2.txt***[2]*. Następnie oblicz najlepsze przesunięcie tego rozkładu względem rozkładu wszystkich liter z pliku* ***lalka.txt****. Na koniec oblicz odległość otrzymanego rozkładu przesuniętego o to najlepsze przesunięcie od rozkładu z Lalki.*

Wskazówka: aby obliczyć oba rozkłady, wczytaj oba pliki, a następnie wywołaj funkcję ***zlicz*** *z następującymi argumentami:* ***zlicz(lalka, 0, 1)*** *oraz* ***zlicz(tekst2, 0, 5)****. Otrzymane rozkłady zapisz w zmiennych* ***ile_lalka*** *oraz* ***ile_tekst2****. Następnie użyj funkcji* ***znajdz_przesuniecie****, aby znaleźć najlepsze przesunięcie rozkładu* ***ile_tekst2*** *względem rozkładu* ***ile_lalka****. Na koniec użyj funkcji* ***odl****, aby obliczyć odległość między odpowiednio przesuniętym rozkładem* ***ile_tekst2****, a rozkładem* ***ile_lalka****.*

Twój program powinien wypisać komunikat takiej postaci:

```
Najlepsze przesuniecie:
21
Odleglosc:
0.07712184832382302
```

3 Zadania dodatkowe

Zadanie 5 *Napisz funkcję* ***sprawdz_kroki(tekst, max_krok)****, która sprawdzi wszystkie wielkości kroku od 1 do* ***max_krok*** *(włącznie) i dla każdego z nich wypisze długość tego kroku* ***krok*** *oraz najmniejszą odległość (od rozkładu języka) odpowiednio przesuniętego rozkładu otrzymanego z liter wyciętych co* ***krok*** *z tekstu* ***tekst*** *począwszy od pierwszej litery tego tekstu.*

Używając napisanej funkcji, sprawdź najmniejsze odległości rozkładów dla tekstu z pliku ***tekst2.txt*** *i dla wielkości kroku od 1 do 10 (włącznie). Wartości otrzymane dla kroku równego 5 i 10 powinny być istotnie mniejsze od pozostałych. Dlaczego?*

4 Praca domowa nr 4

Rozwiązania zadań należy przesłać do czwartku 27 kwietnia do godz. 16^{59} na adres `licealisci.pracownia@icm.edu.pl` wpisując jako temat wiadomości `Lx PD4`, gdzie `x` to numer grupy, np. `L3 PD4` dla grupy L3, itd.

We wszystkich zadaniach będziemy korzystać z pliku `tekst3.txt` (dostępny do pobrania ze strony `ciekawi.icm.edu.pl`, zakładka *Dla uczestników* → *Materiały - ćwiczenia*).

Zadanie domowe 1 (2 pkt) *Tekst w pliku* ***tekst3.txt*** *został zaszyfrowany szyfrem Vigenere'a z kluczem o długości nieprzekraczającej 20. Znajdź długość tego klucza.*

Zadanie domowe 2 (2 pkt) *Znajdź klucz, którego użyto do zaszyfrowania pliku* ***tekst3.txt****.*

Zadanie domowe 3 (2 pkt) *Rozszyfruj plik* ***tekst3.txt****.*

[2]Plik `tekst2.txt` również należy pobrać ze strony `ciekawi.icm.edu.pl`