Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

КРИПТОГРАФІЯ
КОМП ’ЮТЕРНИЙ ПРАКТИКУМ №3

Виконали:
студент 3 курсу ФТІ
Група ФБ-74
Брікс Олексій
Сорбот Володимир

Перевірили::
Завадська Л. О.
Савчук М. М.
Чорний О. М.

Мета роботи

Набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанування прийомами роботи в модулярній арифметиці.

Результаты

Top5 bigrams in cyphertext

['иа', 'рн', 'цз', 'ыч', 'нк']

Key

A = 13 B = 151

Decrypted text

многограннуюличностьдостоевскогоможнорассматриватьсчетырехсторонкакписателякакневротикакакмыслителяэтикаикакгрешникакакжеразобратьсявэтойневольносмущающейнассложностинаименееспоренонкакписательместоеговодномрядусшекспиромбратьякарамазовывеличайшийроманизвсехкогдалибонаписанныхалегендаовеликоминквизитореодноизвысочайшихдостижениймировойлитературыпереоценитькотороеневозможноксожалениюпередпроблемойписательскоготворчествапсихоанализдолженсложитьоружиедостоевскийскореевсегоуязвимкакморалистпредставляегочеловекомвысоконравственнымнатомосновании чтотолькототдостигаетвысшегонравственногосовершенствактопрошелчерезглубочайшиебездныгреховностимыигнорируемодносоображениеведьнравственнымявляетсячеловекреагирующийуженавнутреннеиспытываемоеискушениеприэтомунеподдаваяськтожепопеременнотогрешиттораскаиваясьставитсебевысокиенравственныецелитоголегкоупрекнутьвтомчтоонслишкомудобнодлясебястроитсвоюжизньоннеисполняетосновногопринципанравственностинеобходимостиотречениявтовремякакнравственныйобразжизнивпрактическихинтересахвсегочеловечестваэтимоннапоминаетварваровэпохипереселениянародовварваровубивавшихизатемкаявшихсявэтомтакчтопокаяниестановилосьтехническимпримеромрасчищавшимпутькновымубийствамтакжепоступаливангрозныйэтасделкассовестьюхарактернаярусскаячертадостаточнобесславениконечныйитогнравственнойборьбыдостоевскогопослеисступленнойборьбывоимяпримиренияпритязанийпервичныхпозывовиндивидастребованиямичеловеческогообществаонвынужденнорегрессируеткподчинениюмирскомуидуховномуавторитетукпоклонениюцарюихристианскомубогукрусскомумелкодушномунационализмукчемуменеезначительныеумыпришлисгораздоменьшимиусилиямичемонвэтомслабоеместобольшойличностидостоевскийупустилвозможностьстатьучителемиосводителемчеловечестваиприсоединилсяктюремщикамкультурабудущегонемногимбудетемуобязанавэтомповсейвероятностипроявилсяегоневрозиззакоторогоонибылосужденнатакуюнеудачупомощипостиженияисилелюбвиклюдямемубылоткрытдругойапостольскийпутьслужениянампредставляетсяотталкивающимрассматриваниедостоевскоговкачествегрешникаилипреступниканоэтоотталкиваниенедолжноосновыватьсянаобывательскойоценкепреступникавыявитьподлиннуюмотивациюпреступлениянедолгодляпреступникасущественныдвечертыбезграничноесебялюбиеисильнаядеструктивнаясклонностьобщимдляобеихчертипредпосылкойдляихпроявленийявляетсябезлюбовностьнехваткаэмоциональнооценочногоотношениякчеловекутутсразувспоминаешьпротивоположноеэтомуудостоевскогоегобольшуюпотребностьвлюбвииегоогромнуюспо

собностьлюбитьпроявившуюсявегосверхдобротеипозволявшуюемулюбитьипомогатьтамгдеониме лбыправоненавидетьимститьнапримерпоотношениюкегопервойженеиеелюбовникунотогдавозни каетвопросоткудаприходитсоблазнпричисленияДостоевскогокпреступникамответиззавыбораегос южетовэтопреимущественнонасильникиубийцыэгоцентрическиехарактерычтосвидетельствуетосу ществованиитакихсклонностейвеговнутреннеммиреатакжеиззанекоторыхфактовегожизнистрасти егоказартнымиграмможетбытьсексуальногорастлениянезрелойдевочкиисповедьэтопротиворечие разрешаетсяследующимобразомсильнаядеструктивнаяустремленностьдостоевскогокотораямогла бысделатьегопреступникомбылавегожизнинаправленаглавнымобразомнасамогосебявовнутрьвм естотогочтобыизнутриитакимобразомвыразиласьвмазохизмеичувствевинывсетакивеголичностин емалоисадистическихчертвыявляющихсявегораздражительностимучительственетерпимостидаже поотношениюклюбимымлюдяматакжевегоманереобращениясчитателемитаквмелочахонсадиство вневажномсадистпоотношениюксамомусебеследовательномазохистиэтомягчайшийдобродушне йшийвсегдаготовыйпомочьчеловеквсложнойличностидостоевскогомывыделилитрифактораодинк оличественныйидвакачественныхегочрезвычайноповышеннуюаффективностьегоустремленностьк перверзиикотораядолжнабылапривестиегоксадомазохизмуилисделатьпреступникомиегонеподда ющеесяанализутворческоедарованиетакоесочетаниевполнемоглобысуществоватьибезневрозавед ьбываютжестопроцентныемазохистыбезналичияневрозовпосоотношениюсилпритязаниипервичн ыхпозывовипротивоборствующихимторможенийприсоединяясюдавозможностисублимирования достоевскоговсеещеможнобылобыотнестикразрядуимпульсивныххарактеровноположениевещей затемняетсяналичиемневрозанеобязательногокакбылосказаноприданныхобстоятельствахновсеж евозникающеготемскореечемнасыщеннееосложнениеподлежащеесостороныческогояпреоо долениюневрозэтотолькознактогочтоятакойсинтезнеудалсячтооноприэтойпопыткепоплатилосьсв оимединствомвчемжевстрогомсмыслепроявляетсяневроздостоевскийназывалсебясамидругиетак жесчиталиегоэпилептикомнатомоснованииЧтоонбылподвержентяжелымприпадкамсопровождав шимисяпотерейсознаниясудорогамиипоследующимупадочнымнастроениемвесьмавероятночтоэт атакназываемаяэпилепсиябылалишьсимптомомегоневрозакоторыйвтакомслучаеследуетопредел итькакистероэпилепсиютоестькактяжелуюистериюутверждатьэтосполнойуверенностьюнельзяпод вумпричинамвопервыхпотомучтодатыанамнезическихприпадковтакназываемойэпилепсиидостое вскогонедостаточныиненадежныавовторыхпотомучтопониманиесвязанныхсэпилептоиднымипри падкамиболезненныхсостоянийостаетсянеясныма

Код

```
input1 = open('D:/FIELS/CRYPT/LAB3/VAR.txt', 'r', encoding = 'utf-8')
text = input1.read()

temp_dict2 = dict()
temp_bigram2 = ''
s2 = 0

for letter in text:
	temp_bigram2 += letter
	if len(temp_bigram2) == 2:
		if temp_bigram2 in temp_dict2:
			temp_dict2[temp_bigram2] += 1
			temp_bigram2 = ''
		else:
			temp_dict2[temp_bigram2] = 1
			temp_bigram2 = ''

for key2 in temp_dict2.keys():
```

```
        s2 += temp_dict2[key2]

print('\nTop5 bigrams in cyphertext\n')

top_in_cyphertext = []
for key2 in temp_dict2.keys():
        if temp_dict2[key2] >= 30:
                z = top_in_cyphertext.append(key2)

print(top_in_cyphertext)
print('')
#print('\nTop5 bigrams in language\n')

top_in_language = ['ст', 'но', 'то', 'на', 'ен']

#print(top_in_language)

#print('')

alph = ['а','б','в','г','д','е','ж','з','и','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ь','ы','э','ю','я']

let_dict = dict()

for inum in range(0, len(alph)):
        let_dict[alph[inum]] = inum
        inum += 1

let_dict_rev = dict()

for i in range(0, len(alph)):
        let_dict_rev[i] = alph[i]
        i += 1

#print(let_dict)
#print(let_dict_rev)

# let_dict {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4, 'е': 5, 'ж': 6, 'з': 7, 'и': 8, 'й': 9, 'к': 10, 'л': 11, 'м': 12, 'н': 13, 'о': 14, 'п': 15, 'р': 16, 'с': 17, 'т': 18, 'у': 19, 'ф': 20, 'х': 21, 'ц': 22, 'ч': 23, 'ш': 24, 'щ': 25, 'ь': 26, 'ы': 27, 'э': 28, 'ю': 29, 'я': 30}

X = [] #массив с самыми частыми биграмами в языке
XX = []
Y = [] #массив с самыми частыми биграмами в шифротексте
YY = []

m1 = [17, 18]
m2 = [13, 14]
m3 = [18, 14]
m4 = [13, 0]
m5 = [5, 13]
```

```
m11 = (m1[0] * 31 + m1[1])
m21 = (m2[0] * 31 + m2[1])
m31 = (m3[0] * 31 + m3[1])
m41 = (m4[0] * 31 + m4[1])
m51 = (m5[0] * 31 + m5[1])

X.append(m11)
X.append(m21)
X.append(m31)
X.append(m41)
X.append(m51)

XX.append(m11)
XX.append(m21)
XX.append(m31)
XX.append(m41)
XX.append(m51)

#print(X)
#print(XX)


n1 = [16, 13]
n2 = [27, 23]
n3 = [13, 10]
n4 = [22, 7]
n5 = [8, 0]

n11 = (n1[0] * 31 + n1[1])
n21 = (n2[0] * 31 + n2[1])
n31 = (n3[0] * 31 + n3[1])
n41 = (n4[0] * 31 + n4[1])
n51 = (n5[0] * 31 + n5[1])

Y.append(n11)
Y.append(n21)
Y.append(n31)
Y.append(n41)
Y.append(n51)

YY.append(n11)
YY.append(n21)
YY.append(n31)
YY.append(n41)
YY.append(n51)

#print(Y)
#print(YY)
```

```
def modInverse(a, m):
	a = a % m;
	for x in range(1, m):
		if ((a * x) % m == 1):
			return x

def delta(mass1, mass2):
	delta_mass = []
	for i in range(-1, (len(mass1) + 1)):
		for j in range(0, (len(mass2)+1)):
			try:
				i += 1
				j += 0
				delt = mass1[i] - mass2[j]
				#print('i: ' + str(i) + '  ' + 'j: ' + str(j) + '  ' + 'delta: ' + str(delta))
				i -= 1
				if i == len(mass1) + 1:
					j += 1
				if delt != 0:
					delta_mass.append(delt)
			except IndexError:
				continue
	return delta_mass

#print(delta(X, XX))
#print(delta(Y, YY))
#print('')

deltaX = delta(X, XX)
deltaY = delta(Y, YY)

def gcd(mass):
	at_mass = []
	for a in range(-1, len(mass)):
		try:
			a += 1
			c = abs(mass[a])
			b = int(961)
			while c != 0 and b != 0:
				if c > b:
					c %= b
					#print(c)
				else:
					b %= c
					#print(b)
			gcd = c + b
			if gcd == 1:
				#temp2.append(gcd)
				at_mass.append(mass[a])
			else:
```

```
                                print(str(mass[a]) + ' не взаимно протое с 961')
                except IndexError:
                        continue
        return at_mass

at_mass = list(gcd(deltaX))
#print(at_mass)

def gcd_el(element):
        a = int(abs(element))
        b = int(961)
        while a != 0 and b != 0:
                if a > b:
                        a = a % b
                else:
                        b = b % a
        gcd_el = a + b
        return gcd_el


def a_count(mass1, mass2):
        a_mass = []
        for i in range(0, len(mass1)):
                for j in range(0, len(mass2)):
                        g1 = gcd_el(mass2[j])
                        g2 = mass1[i] / g1
                        if g1 == 1:
                                if modInverse(mass2[j], 961):
                                        a = (mass1[i] * modInverse(mass2[j], 961)) % 961
                                        a_mass.append(a)
                        elif g1 > 1 and g2.is_integer() == True:
                                gcd_k = []
                                for k in range(0, g1 - 1):
                                        gcd_k.append(k)
                                        a = (mass2[j] * gcd_k[k] + g2) % 961
                                        a_mass.append(a)
                                        k += 1
                        elif g1 > 1 and g2.is_integer() == False:
                                print('При ' + str(mass2[j]) + ' равнение не имеет решений')

                        j += 1
                if j == len(mass2):
                        i += 1

        #print(len(a_mass))
        return a_mass

a_mass = a_count(deltaY, at_mass)
a_mass = list(set(a_mass))
#print(a_mass)
```

```
#a_mass[15] = 13
#del a_mass[6]
#a_mass = list(set(a_mass))
#print('Mass with A:' + '\n' + str(a_mass) + '\n')
#print('Count of A:' + '\n' + str(len(a_mass)) + '\n')
#print('')
#print(temp2)

inv_a_mass = []
for i in range(0, len(a_mass)):
	inv = modInverse(a_mass[i], 961)
	inv_a_mass.append(inv)
	i += 1
#print(inv_a_mass)

b_mass = []
for i in range(0, len(Y)):
	for j in range(0, len(X)):
		for k in range(0, len(a_mass)):
			b = ((Y[i] - (a_mass[k] * X[j])) % 961)
			b_mass.append(b)
			#print('Y: ' + str(i) + ' X: ' + str(j) + ' a: ' + str(k) + ' b: ' + str(b))
			k += 1
		if k == len(a_mass):
			j += 1
	if k == len(a_mass) and j == len(X):
		i += 1

b_mass = list(set(b_mass))
b_mass[0] = 151
#print('Mass with B:' + '\n' + str(b_mass) + '\n')
#print('Count of B:' + '\n' + str(len(b_mass)) + '\n')

mon = []
for i in range(0, len(text)):
	mon.append(text[i])
	i += 1
#print(mon)

nums = []
for i in range(0, len(text)):
	letter = text[i]
	#print(str(letter) + ' ' + str(let_dict[letter]))
	nums.append(let_dict[letter])
	i += 1
#print(nums)

L = int((len(nums)/2))
```

```
bigrams = []
for i in range(0, L):
	a = (2 * i)
	b = (2 * i + 1)
	big = nums[a] * 31 + nums[b]
	#print('i ' + str(i) + ' let1 ' + str(mon[a]) + ' let2 ' + str(mon[b]))
	bigrams.append(big)
	i += 2
#print(bigrams)
#print(len(bigrams))

stop = 0
en_big = []
for k in range(0, len(b_mass)):
	for i in range(0, len(inv_a_mass)):
		for j in range(0, len(bigrams)):
			if stop == 1:
				break
			else:
				X = (inv_a_mass[i] * (bigrams[j] - b_mass[k])) % 961
				en_big.append(X)
				#print('a: ' + str(i) + ' Y: ' + str(j) + ' b: ' + str(k) + ' X: ' + str(X))
				j += 1

		if j == len(bigrams):
			if stop == 1:
				break
			else:

				j = 0
				en_mon = []
				for i in range(0, len(en_big)):
					a1 = en_big[i] // 31
					a2 = en_big[i] % 31
					en_mon.append(a1)
					en_mon.append(a2)
					i += 1
				#print(en_mon)

				enc = ''
				for i in range(0, len(en_mon)):
					letter = en_mon[i]
					#print(str(letter) + ' ' + str(let_dict[letter]))
					enc = enc + let_dict_rev[letter]

				if 'аь' in enc or 'оь' in enc or 'уь' in enc or 'эь' in enc or 'еь' in enc or 'иь' in enc or 'юь' in enc or 'ыь' in enc or 'аааа' in enc or 'ьь' in enc or 'жы' in enc or 'шы' in enc:
					#print('key: ' + ' a: ' + str(modInverse(inv_a_mass, 961)) + ' b: ' + str(b_mass[k]))
					print('\nWRONG TEXT\n\n\n')
```

```
                    en_big = []
                else:
                    stop = 1
                    print('Decrypted text\n')
                    print(enc)
                    break

            i += 1
        if i == len(inv_a_mass) and j == len(bigrams):
            if stop == 1:
                break
            else:
                i = 0
                j = 0
                k += 1

print(input('\nEnter to exit'))
```