Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

**ЛАБОРАТОРНА РОБОТА №2**

**з дисципліни**

**«Криптографія»**

**на тему: «Експериментальна оцінка ентропії на символ джерела відкритого тексту»**

Виконали:

студенти 3 курсу ФТІ

групи ФБ-74

Постолюк Діана та Хацько Микита

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. О.

**Мета роботи**

Засвоєння методів частотного криптоаналізу. Здобуття навичок роботи та аналізу потокових шифрів гамування адитивного типу на прикладі шифру Віженера.

**Порядок виконання роботи**

0. Уважно прочитати методичні вказівки до виконання комп’ютерного практикуму.

1. Самостійно підібрати текст для шифрування (2-3 кб) та ключі довжини *r* = 2, 3, 4, 5, а також довжини 10-20 знаків. Зашифрувати обраний відкритий текст шифром Віженера з цими ключами.

2. Підрахувати індекси відповідності для відкритого тексту та всіх одержаних шифртекстів і порівняти їх значення.

3. Використовуючи наведені теоретичні відомості, розшифрувати наданий шифртекст (згідно свого номеру варіанта).

**Варіант 11**

**Результати виконнання програми:**

Coincidence index for plain text : 0.05541800288043644

Coincidence index for key with length 2: 0.04581931750498981

Coincidence index for key with length 3: 0.038287232586271674

Coincidence index for key with length 4: 0.035730505903418114

Coincidence index for key with length 5: 0.035680967276729036

Coincidence index for key with length 20: 0.03439484454569936

Potential key lengths and their statistics:

6. 202

7. 219

8. 195

9. 204

10. 202

11. 205

12. 264

13. 185

14. 236

15. 196

16. 210

17. 411

Key length = 17

Encrypted text: втяугроъцсхйиббьыеумчцтптикуочяькуфупчхлоюгжкйцтар

Decrypted text: антонионезнаюоычегоятакпечаленмцеэтовтягостьвамяъл

Key: венецианскийкужец

|

Key: венецианскийкужец

|

Key: венецианскийкупец

Decrypted text: антонионезнаюотчегоятакпечаленмнеэтовтягостьвамясл

**Зашифрований текст:**

втяугроъцсхйиббьыеумчцтптикуочяькуфупчхлоюгжкйцтарсъшяуьнныфонингвциюфыовилъсвнфтюлйдгашьицсывьилхтфчнфуэуърттцяцыпюраэпеябчнсюэещфпаьехехацидмырмрцшсжчдуещущцсттйырчуббвпкяхймнывкуйъыьушэйаъдфмтипъоыпюудмкнтйлдтукасмшъннвзикзыдныкткшцпчыкнпкбдмычткчоыъбеээъехчрызпщъттыужупндзчртшънцжшыцврчэдихаяяълчмйфзвзрчнлятыыхицйсбцхпнфпдрмюашяыпалквмурйццнхьпъиьапчавтиъашышнйэъкюптюрфызышьяцпщфтфочцмххцацвньщцаьысцыщшпцикаомхркьуысдкцщуысннхпоншьожссуючдзнъяышдмуъчжвзаьицбфюкъешещшъвзтчышиюыкуцкэпхивърешинхщлыюьоъгчроьхыммтгбъчцбтжспкайцяущюпчщпчскпвчйсыхяомчнъшяькгпупижысянщцлпгтебуешежрнывьынйяэозхфсалинйццзлхыдужвйчкчгдэярифшеыазнндчдфоуцькхшгфшжвинтгидтькъечшыущгапнънтйрбиъшхюкзрьъалхепвщцхчысэюрстрхэиыбтьйявякъучнзюубиышшйлюлзезцчкэивмшврхнюпзйупшугрвещцхсршжквгученъоозпучмуббздулсдлишдмюоъэснзоуяхххачсцхссчптюбцпдицгыыктхшцрахпкпцецмъщьдъфуъуевцъалятыжъышфышсдлпыхцйлйцокйьбъпгхзпцычрмюшщытгпцзэфнрюйыпушмьтхэргэуорытлхтмфчтлфравтацбцвыэбъчцбфждееяцикоюгкуччыжквксыибрбмялеышяушввчйтымущсйчщтеэснфутцбрбясфщфэкчрдубщтычрхйхцъжфкмцехациртйюплчмбянизмъефзаъгшхсшцяшзфнячжнвычкщесуаздкчызцшынюьццтбькидкэбинмъцлуйнбуежацайтйущяушсыэькджтысйзвпцърфыжутйпкыйгцмащцнъьжаузфумттнмыцчнхпгччзбчтпйбищфшмчцтькщтшжшюпзнэшрюьбсежрзюебирхюшъчнчпзсйтньюъвшплуочоптиртхуеысяяпщйхуянгрттзбжбшцчгыкэапцикщзсчедсхдцеъпчыоьяушгнтупщохоччднбчувцгшщлщхптббзбзичшнрсрйкоышъмцфкщьицнтфъывэчсшбкьъаязнавфуичжабиржыожцдхгщшсъбуезфхнтггхшпонтшчьнщнефкфъивяяцаэещеасуьшщийавхгбкхзнядушагтусбэлспщфтцднспцтучвэщутдъаивпдчдкушмлтосжрагзфыпцоуяыхзцтдлццоттцицрдгшпйлустцъшяпцкхыхйъккдаегкушужннгятлщкйчегрцнрцхиыушъкхутужрйъаяшосщбкйвфпцзвтхущшагщкхчтюэхыпыыцгрмьбшбйуефссдраьонмытгнъьхфузфепнвкаювкуйъыьудучнрззбмиьцкмуахцйзтцыуиянчцлшеозюишяклттыукфншгэлывтяугропэшрюнюпмцчыттцкащхшшчнуайцзчюдвхедгшкйычфрцйупширрнхекдщгфйбриашъилхгжщиыуъежктыфжрвтгмихнбафджеоезщаъщшщсчпэхспучущмауэеччыяфквудчущмапмбъчьбачцннъкбждхэещйхуянгрйыцйлвнйцгнпюччтеуяушгспсцръкюпхюпухццуаъщшыюшчочбрхттмцкосыщйчыэцюпбыхжизпмкхышачугвэшнвгвнвшщыкхчсгрфэуоыкытпмьчшюуэвжичтлдтэемчхщьазроянздбнвыицтбюхюжшъешнръяншйаыптдунъбдшаъгшхсшхчййдеюфбцхыъщнапефцтурхэацмпмшйфкцпъвкхнпицивгъыншяжхыйстхггмьяфышкшбчытдтчюэкнытпхпачрюубпацхтыютцицыяцнкчнгннмыюпщыжцяемкеъувррюзпхйнчфшшудчущцеюгжшчхпыухехацихарбкюскаэсгзлсеяъеэяхдбэепфйупнодъсщцяикэчвзыубпсдшщхюкэшэдбббхьекенчтюжцымыьещрххчннмюгехоьдфхъшкычизжтьеэлсчэъддмньыфжтипучмшщшзфкьрдскэямдзыыукиюфыйдйныэъихшгьувхфэкътуюакъечуозйкрхъшцрнгжоохевъдлуяхцпдэсрнцжтарйъецпциняячрчьшрдбажшгхлопяърымбытынгоушдеюгжузоывпдфуэалуигсщцуъобаенкъпдстыичцмхуубчррычццнхжъицйеьежрьъугнпыхмрпчбачтщчыждйщнрццфмучсетньнзилнвпшепъьузфбщшъшоюгжрмхжруакоасющлыучцмшхэххфтнсхцрныэуушцуешзюнгеысянтчоыафрцзчысдсаъгшхсшъьефбчнпюэчяяцынъоынзнапшиуиенщцышявьиртхьылоъцнцлшгочирисеаикфснцлшгчздпнякжпашщлыбтефсафухъзуыеслусрачъъххнпцфиъсскйхфкзыттыйццбкгшфшшдъкгрттрдкиямчишъыыегмшрхйщтхгктьидъешхлнраыюэмлнбфжюуяжкщрдшеъзшхьщбщеетужеяипэящцлпчлдартдшецооцоилхбякгшухчцтвнвшщыкхъдыойыучнднаърнпеадвпкоаидмахъняшябеаксокэфошучгхнбсужкчйтымаюгншйаыптдехныюиныхкччрыснзрсъуфлоссокйсхвпщррыццъыюушнмшъпжйжкебцхтыютццэчъдизжмдзъаъдлцфьжьувехныюиныяьусбаэзыжбщубяаънпэчъбзушмуььыхыврсгукиуешщнючсэдтукэмъуенцпухрдшеъзшыюшчочцпчытюцгсцыфдыюскщрцшушихосыщйчыэижвыхегцгыушшсьффцарттьгцмъянцшэдбдарзоубдштипфуьбънчрзпкгнцхщплчъуацйийттюзэяяокйсшятцсэттююыюаъзыкаьаыдйпшлфеэсяфифыьанфуоаюэннъьрцкчэнзмеябшнсйхпхекапоъзэйшшрдхйжяышычцавчйтыщтыщхцлпафыюбшшнмиввяорыхуььынуярцхчтьшъушафьгрцызыщтйэшзшшъсубкчщтыщбтчкъешемчдеуунъимыцнцюшъонгвжтцвннмгктлшеччднпнкиъачушъстдщшовкяидкоэщьлчлфэрцпьтрдъгытлцншфяаянеъьыуоящхрншфяаяеуождрлххшйщеьъегщкшуилоотшчьыечтденпъдмбтфткчмдфчхипхкймиэсуцыысуецуупкзъьрямцщтеькисуючдсчтвьхдуюптнзрычецсяяуяоыеьаяеуождрлхыктлелфцавмнтдяеюгчнтвышрцпьтрдъгытлццпжунжвояуехиъянцтчумчюоаюрюуасюшиюъурхслййдшцлпцхрыцафцанесашитйашосьэчзехчйидкоэщьйоыяпхоеупужртоцышйоырущвцыжышиюнымьябыиддуэнийеющхыштюпйгреюушнсянццимшзеыфцмтцаелыццоцжакжжыанвыдэянцлшгччвродкзъниъошьнюптнзрычшндйгещдчкфципурудъцнщхръфбякыуаъьыщтъяуфйьшянерчысйятывфиркелфжвзсшдъеггшфчуафцаррйпдтачтееышхкхцйнябззояхккйхкфсиржирйхерязъйфышфжкзчшзуасюшщчмшачтоттидкоэщьуйчкфрдфттэыкешщыдшшлфзыннпеящярямцщтеркзпнюсыщтнфшкчъыбцддкючтщопцыъенбсужафэешрлйюшъдыбскихкебыщлхашксчбсеиюцмцдкеюгтхйобытырцяеидсмдррнкяэкщръымхрннсшхышвяузфнщкгзгывщцнтдпсштускъдяпяхийбеэжсхйэеидоячтмйщгчйыфцмфдкиъямиыждорймепувыапцодччеэцшвэтидчофушуочыоныучйццфдйрмцфтеэфжвзсенъоущокчщюэюыптиоъдяпяхршшдэзыучхидкоэрыцлпдббврдукзъочяынщоапдзрзтцидеюьтццкяькзрзтчйнтывьиыждошькйнжыщмъцоиоьфэрызйэшьнсчщущчмшбдивпхшънрахжзлшюыуюсдяпюттпятфьыювицошскжыввяорыыепхслиыжчцсчяъпсчээощржоувцлыхшсъталужупнлюъеьярифэьачцмеччйъпэяуъсфдчттрхалюмушсчяохббззфуэугыоъцлцнкрбувротйюхоэяохдуббкмртюрычтныыучмаэквхттчдятыапщушяппжхфъавлрнутнхнюпмцйеюаеилыюпырчуфчвзтмцслрнпыхсцэппйатхжймьегэтьнбрждсудчыхнтъвртцбъуъроюнгдоэвонфкьбквкрыцидкныцхощгэоикпюнгдоэдррнэюптнзрычцчопцъхсьшьеетепуешиьцъкчцвэздтуякгцпэщыихшяюкждущфдкоэяьншшхфхгкчттцуепяоиъцббхюфкьхюхаюгшзвпябуерзыхдъеъцщцлшгпьюдецчыхиъсщшймбьнгвртздивыбшяйуеффжбстъдхьчяфмчцжхнъвиыжчцспрьгоэцэйгкпхчжыдъялынпеюуевцябгиннвьцигжнйдтуршшгнтэшооьшапцйеябвхеъцфртхуььынуяьицуъцнцптхфчькзтчйхерятусчшбрцпьтрдъвифшкчъыбкдоушуъцмющчдчъшъегщкшуинвюгшшччжсуисуруърттыктьитакящеюнзъоэщишйсхфййучнхютупмнцлшгччырсырдмощвтхешгфрцпьтрдъерялйчфущуяфещцюхьыбмибкячрдучйцсхбтхчцмшкфрддкчедъегцнцюшелвирдюевлопяъжапдбтслтыунннйтпмцеьжкбрзтплцтмтхимшчпххгуреэмоэямгтхуььынуятйттытеъйцачжснкрудъегэелчмбянврсырдмощвтхешгфрцпьтрдъерялйчшъьеарсысшиштъцрфшдбнетуючдуаъипучьшашъвхйрцхчтьшъцшвяуохзнцзъаэщфъчлызбйоушдфкаювхнныескгпупащцвыбтъошяэрджущцюхьыбпуйчкфрдфкручйоыькхапюэчыуфымшцлтютзлклфовкрыцирлнюбнфшеарыжцязыхныяырцжбякбвтдкбцттюплчмбянизуюнгдоэцхицбшъуизжнчфакнэшсслбмчдфсырдмоцвтхешгаущнеютдкошцыынпфчхдмехзззкхжьиутивьыпавзыбецуъцнпеяклукючышпнбштсщгэьючуяхцщлтъчиьфыукяучпнинлйюшщушажебудхрюнсшщццпчбсажвмхчтщяьбшбошеээынзшшнаицэтшмшфрущрцьуъруьсырнкхфйтоешбныгнюлгфтдбнгпащфдщовяцфчпьачцтуючюрсящчхчуяэкясусфдшоцькхапюэчгзшучдояьяяцпчуюебмшхрюхаосхтьчуюузръхрюхаьяоччъэвзсокъдгнуфюкныпфпчфъпнидйбыяяхоьтсезпкфтцжмыьшмчудчттрхъуешоедмиъыэплюфкщтычрнуоыаьбыуяоешбнркааштчуцэцерййкщцдайэосмыънерстхбиндцхцычшвлирзитыызъспъоцшьдчвчлэаигытлцчяцэхыбзтйтчодгтяышбарысттфжрзъсикйыюптнзрычгыпыаъилошуеьзжайтывнвнуйсусфдтдспыкыхгшфчнючжспкамгитйэпхиэяфтирчычыючяяэпцкшбцдгцязыхныфшшолшьпцнестдщтнбттимуызззхнцзтаудщшчмузкщрцитцщтнюшксьдотцмушкгрбшснцъхбснвзтмфживссоцфрапзслчхтцщвтгэйсудбзцжушидщкэммиыжафртйдччдяецвехжьбапжэчйсдоныюшкушаекартгушчрнуоилеьукипеэшьы

**Знайдений ключ:**

венецианскийкупец

**Розшифрований текст:**

антонионезнаюотчегоятакпечаленмнеэтовтягостьвамяслышутоженогдеягрустьпоймалнашелильдобылчтосоставляетчтородитеехотелбызнатьбессмысленнаягрустьмоявиноючтосамогосебяузнатьмнетрудносалариновыдухоммечетесьпоокеанугдевашивеличавыесудакакбогатеиивельможиводильпышнаяпроцессияморскаяспрезреньемсмотрятнаторговцевмелкихчтокланяютсянизкоимспочтеньемкогдаонилетятнатканыхкрыльяхсаланиоповерьтееслибятакрисковалпочтивсечувствабылибтаммоисмоейнадеждойябыпостоянносрывалтравучтобзнатьоткудаветерискалнакартахгаваниибухтылюбойпредметчтомогбынеудачумнепредвещатьменябынесомненновгрустьповергалсалариностудямойсупдыханьемявлихорадкебыдрожалотмысличтоможетвмореураганнаделатьнемогбывидетьячасовпесочныхневспомнившиомеляхиорифахпредставилбыкорабльвпескезавязшимглавусклонившимнижечембокачтобцеловатьсвоюмогилувцерквисмотрянакамнизданиясвятогокакмогбыяневспомнитьскалопасныхчтохрупкиймойкорабльедватолкнуввсепряностирассыпалибывводуиволныоблеклибвмоишелканусловомчтомоебогатствосталоничемимоглибяобэтомдуматьнедумаяпритомчтоеслибтакслучилосьмнепришлосьбызагруститьнеговоритезнаюяантониогруститтревожасьзасвоитоварыантонионетверьтемнеблагодарюсудьбумойрискнеодномуявверилсуднунеодномуиместусостояньемоенемеритсятекущимгодомянегрущуиззамоихтоваровсаларинотогдавызначитвлюбленыантониопустоесалариноневлюбленытакскажемвыпечальнызатемчтовыневеселыитолькомоглибсмеятьсявытвердяявеселзатемчтонегрущудвуличныйянусклянусьтобойродитприродастранныхлюдейодниглазеютихохочуткакпопугайуслышавшийволынкудругиеженавидкакуксускислытакчтовулыбкезубынепокажутклянисьсамнесторчтозабавнашуткавходятбассаниолоренцоиграцианосаланиовотблагородныйродичвашбассаниограцианоилоренцоснимпрощайтемывлучшемобществеоставимвассалариноосталсябячтобвасразвеселитьновотявижутехктовамдорожеантониовмоихглазахценавамдорогасдаетсямнечтовасделазовутирадывыпредлогуудалитьсясалариноприветвамгосподабассаниосиньорынокогдажмыпосмеемсякогдавычтотосталинелюдимысаларинодосугвашмыделитьготовысвамисалариноисаланиоуходятлоренцокбассаниосиньорразвыантонионашлимывасоставимнопрошукобедунепозабытьгдемыдолжнысойтисьбассаниопридунавернограцианосиньорантониовидувасплохойпечетесьслишкомвыоблагахмирактоихтрудомчрезмернымпокупаеттеряетихкакизменилисьвыантониоямирсчитаючемонестьграцианомирсценагдеувсякогоестьрольмоягрустнаграцианомнеждайтерольшутапускайотсмехабудувесьвморщинахпустьлучшепеченьотвинагоритчемстынетсердцеоттяжелыхвздоховзачемжечеловекустеплойкровьюсидетьподобномраморномупредкуспатьнаявуилихворатьжелтухойотраздраженьяслушайкаантониотебялюблюяговоритвомнелюбовьестьлюдиукоторыхлицапокрытыпленкойточногладьболотаонихранятнарочнонеподвижностьчтобобщаямолваимприписаласерьезностьмудростьиглубокийумисловноговорятнамяоракулкогдавещаюпустьипеснелаетомойантониознаюятакихчтомудрымислывутлишьпотомучтоничегонеговоряттогдакакзаговоривонитерзалибушитемктоихслышаближнихдуракаминазвалбывернодаобэтомпосленонеловитынаприманкугруститакуюславужалкуюрыбешкупойдемлоренцонупокапрощайапроповедьякончупообедавлоренцоитаквасоставляемдообедапридетсямнебытьмудрецомтакимбезмолвнымговоритьнедастграцианограцианодапоживисомноюгодадвазвукголосатысвоегозабудешьантонионудлятебяястануболтуномграцианоотличноведьмолчаньехорошовкопченыхязыкахдавчистыхдевахграцианоилоренцоуходятантониогдесмыслвегословахбассаниограцианоговоритбесконечномногопустяковбольшечемктолибоввенецииегорассужденияэтодвазернапшеницыспрятанныевдвухмерахмякинычтобыихнайтинадоискатьвесьденьанайдешьувидишьчтоиискатьнестоиловенецияулицавходитланчелотланчелотконечносовестьмояпозволитмнесбежатьотэтогожидамоегохозяинабесменятаквотитолкаеттаквотиискушаетговоритгобболанчелотгоббодобрыйланчелотилидобрыйгоббоилидобрыйланчелотгоббопустиногивходбегивовсетяжкиеудирайотсюдаасовестьговоритнетпостойчестныйланчелотпостойчестныйгоббоиликаквышесказаночестнейшийланчелотгоббонеудирайтопниногойнаэтимыслиладноахрабрыйдьяволвелитмнескладыватьпожиткивпутьговоритбесмаршговоритбесрадибогасоберисьсдухомговоритбесилупиладноасовестьмоявешаетсянашеюкмоемусердцуимудроговоритмойчестныйдругланчелотведьтысынчестногоотцаилискореесынчестнойматерипотомучтосказатьправдуотецтомойнесколькокакбыэтовыразитьсяотдавалчемтобылунегоэтакийпривкусладносовестьмнеговоритланчелотнешевелисьпошевеливайсяговоритбеснисместаговоритсовестьсовестьговорюправильнотысоветуешьеслиповиноватьсясовестинадомнеостатьсяужидамоегохозяинааонтопростименягосподисамвродедьяволаачтобыудратьотжидапридетсяповиноватьсялукавомуаведьонтосвашегопозволенияиестьсамдьяволитоправдачтожидвоплощенныйдьяволипосовестиговорясовестьмояжестокосерднаясовестьеслионамнесоветуетостатьсяужидабесмнедаетболеедружескийсоветятакиудерудьяволмоипяткиктвоимуслугамудерувходитстарыйгоббоскорзинкойгоббомолодойсиньорскажитепожалуйстакактутпройтиксиньоружидуланчелотвсторонуонебодаэтомойединородныйотецонслептаксловноемунеточтопескомакрупнымгравиемглазазасыпалонеузнаетменясыграюснимкакуюнибудьштукугоббопочтеннейшиймолодойсиньорсделайтемилостькакмнепройтиксиньоружидуланчелотаповернитенаправоприпервомповоротеноприсамомпервомповоротеповернитеналеводасмотритепринастоящемтоповоротенеповорачивайтенинаправониналевоаворочайтепрямехонькокдомужидагоббосвятыеугодникитруднобудетпопастьнанастоящуюдорогувынеможетесказатьмненекийланчелотчтоунегоживетживетунегоилинетланчелотвыговоритеомолодомсиньореланчелотевсторонувотпогодитекакуюясейчасисториюразведустарикувыговоритеомолодомсиньореланчелотегоббокакойтамсиньорвашамилостьсынбедногочеловекаотецегохотьэтоясамговорючестныйнооченьбедныйчеловекхотяблагодарябогаздоровыйланчелотнуктобытамнибылегоотецмыговоримомолодомсиньореланчелотегоббоознакомомвашеймилостипростоланчелотесударьланчелотнопрошувасстариктобишьумоляювасследственновыговоритеомолодомсиньореланчелотегоббооланчелотеспозволениявашеймилостиланчелотследственноосиньореланчелотенеговоритеосиньореланчелотебатюшкамойибоэтотмолодойсиньорсогласноволесудебирокаивсякихтакихученыхвещейвродетрехсестерпарокипрочихотраслейнаукидействительноскончалсяилиеслиможновыразитьсяпрощеотошелвлучшиймиргоббогосподиупасидаведьмальчуганбылистиннымпосохоммоейстаростиистинноймоейподпоройланчелотнеужтожяпохожнапалкуилинабалкунапосохилинаподпоркувыменянеузнаетебатюшкагоббоохнетяваснезнаюмолодойсиньорнопрошувасскажитемнеправдучтомоймальчикупокойгосподьегодушуживилипомерланчелотнеужтовынеузнаетеменябатюшкагоббоохгореяведьпочтичтоослепнепризнаювасланчелотнупоправдедажебудьувасглазавпорядкевыитомоглибынеузнатьменяументототецчтоузнаетсобственногоребенкаладностарикявамвсерасскажупровашегосынастановитсянаколениблагословименяправдадолжнавыйтинасветубийствадолгоскрыватьнельзякточейсынэтоскрытьможноновконцеконцовправдавыйдетнаружу

**КОД**

package crypto;

import java.io.IOException;

import java.nio.file.\*;

import java.util.\*;

import java.util.function.Supplier;

import java.util.stream.Collector;

import java.util.stream.Collectors;

import static crypto.Entropy.monogramsFrequency;

import static crypto.Entropy.sortByValue;

public class VigenereCipher {

private static final String RUSSIAN\_ALPHABET = "абвгдежзийклмнопрстуфхцчшщъыьэюя";

private static final int RUSSIAN\_ALPHABET\_LENGTH = 32;

private static final Double RUSSIAN\_COINCIDENCE\_INDEX = 0.0553;

private static final HashMap<String, Double> RUSSIAN\_LETTERS\_FREQUENCIES;

static {

LinkedHashMap<String, Double> map = new LinkedHashMap<>();

map.put("\u043e", 0.10983);

map.put("\u0435", 0.08483);

map.put("\u0430", 0.07998);

map.put("\u0438", 0.07367);

map.put("\u043d", 0.067);

map.put("\u0442", 0.06318);

map.put("\u0441", 0.05473);

map.put("\u0440", 0.04746);

map.put("\u0432", 0.04533);

map.put("\u043b", 0.04343);

map.put("\u043a", 0.03486);

map.put("\u043c", 0.03203);

map.put("\u0434", 0.02977);

map.put("\u043f", 0.02804);

map.put("\u0443", 0.02615);

map.put("\u044f", 0.02001);

map.put("\u044b", 0.01898);

map.put("\u044c", 0.01735);

map.put("\u0433", 0.01687);

map.put("\u0437", 0.01641);

map.put("\u0431", 0.01592);

map.put("\u0447", 0.0145);

map.put("\u0439", 0.01208);

map.put("\u0445", 0.00966);

map.put("\u0436", 0.0094);

map.put("\u0448", 0.00718);

map.put("\u044e", 0.00639);

map.put("\u0446", 0.00486);

map.put("\u0449", 0.00361);

map.put("\u044d", 0.00331);

map.put("\u0444", 0.00267);

map.put("\u044a", 3.7E-4);

RUSSIAN\_LETTERS\_FREQUENCIES = new LinkedHashMap<>(Collections.unmodifiableMap(sortByValue(map)));

}

private static double coincidenceIndex(String text) {

text = text.toLowerCase()

.replaceAll(String.format("[^%s]", RUSSIAN\_ALPHABET), "");

Map<Integer, Long> collect = text.chars()

.boxed()

.collect(Collectors.groupingBy(x -> x, Collectors.counting()));

// must be stored in double to avoid number overflow

double textLength = text.length();

return collect.values().stream()

.mapToDouble(x -> x \* (x - 1))

.sum() / (textLength \* (textLength - 1));

}

private static boolean isCIGoingToTheoretical(ArrayList<StringBuilder> fragments) {

return fragments.stream()

.map(fragment -> coincidenceIndex(fragment.toString()))

.anyMatch(ci -> Math.abs(ci - RUSSIAN\_COINCIDENCE\_INDEX) < 0.001D);

}

private static Double matchStatistic(String text, int r) {

Double statistic = 0d;

int textLength = text.length();

for (int i = 0; i < textLength - r; i++) {

if (text.charAt(i) == text.charAt(i + r)) {

statistic++;

}

}

return statistic;

}

private static ArrayList<StringBuilder> fragments(String text, int parts) {

ArrayList<StringBuilder> fragments = new ArrayList<>(parts);

for (int i = 0; i < parts; i++) {

fragments.add(new StringBuilder());

}

for (int i = 0, textLength = text.length(); i < textLength; i++) {

fragments.get(i % parts).append(text.charAt(i));

}

return fragments;

}

private static ArrayList<Integer> potentialKeys(String cipherText) {

ArrayList<Integer> potentialKeys = new ArrayList<>();

HashMap<String, Double> map = monogramsFrequency(RUSSIAN\_ALPHABET, cipherText);

LinkedHashMap<String, Double> lettersFrequencies = sortByValue(map);

String mostFrequentlyLetter = lettersFrequencies.keySet().iterator().next();

for (String c : RUSSIAN\_LETTERS\_FREQUENCIES.keySet()) {

potentialKeys.add((mostFrequentlyLetter.charAt(0) - c.charAt(0) + RUSSIAN\_ALPHABET\_LENGTH) % RUSSIAN\_ALPHABET\_LENGTH);

}

return potentialKeys;

}

private static boolean isTextInformative(String text) {

ArrayList<String> textFrequentestLetters = new ArrayList<>(sortByValue(monogramsFrequency(RUSSIAN\_ALPHABET, text)).keySet());

ArrayList<String> russianFrequentestLetters = new ArrayList<>(sortByValue(RUSSIAN\_LETTERS\_FREQUENCIES).keySet());

String tenFrequentestLetters = String.join("", russianFrequentestLetters.subList(0, 15));

double matched = 0;

for (int i = 0; i < 10; i++) {

if (tenFrequentestLetters.contains(textFrequentestLetters.get(i))) {

matched++;

}

}

return (matched / 10) >= 0.9;

}

private static String encrypt(String plainText, final String key) {

char firstLetter = RUSSIAN\_ALPHABET.charAt(0);

Supplier<Character> keyCharsSupplier = new Supplier<>() {

int j = 0;

@Override

public Character get() {

char keyChar = key.charAt(j);

j = (j + 1) % key.length();

return keyChar;

}

};

return plainText.chars().parallel()

.map(c -> (c + keyCharsSupplier.get() - 2 \* firstLetter) % RUSSIAN\_ALPHABET\_LENGTH + firstLetter)

.mapToObj(c -> (char) c)

.collect(Collector.of(StringBuilder::new, StringBuilder::append, StringBuilder::append, StringBuilder::toString));

}

private static String decrypt(String cipherText) {

cipherText = cipherText.toLowerCase();

int keyLength;

System.out.println("Potential key lengths and their statistics:");

// Trying to find key length if key is from 1 to 5.

for (keyLength = 1; keyLength <= 5; keyLength++) {

ArrayList<StringBuilder> fragments = fragments(cipherText, keyLength);

if (isCIGoingToTheoretical(fragments)) {

break;

}

}

// Trying to find key length if key is from 6.

while (true) {

System.out.format("%2d. %d%n", keyLength, matchStatistic(cipherText, keyLength).intValue());

if (matchStatistic(cipherText, keyLength) / matchStatistic(cipherText, keyLength - 1) > 1.5D)

break;

keyLength++;

}

System.out.println("Key length = " + keyLength);

ArrayList<ArrayList<Integer>> potentialKeysArray = new ArrayList<>();

ArrayList<StringBuilder> fragments = fragments(cipherText, keyLength);

StringBuilder key = new StringBuilder();

for (int j = 0; j < keyLength; j++) {

potentialKeysArray.add(potentialKeys(fragments.get(j).toString()));

key.append(RUSSIAN\_ALPHABET.charAt(potentialKeysArray.get(j).get(0)));

}

String plainText = decrypt(cipherText, key.toString());

System.out.println("Encrypted text: " + cipherText.substring(0, 50));

System.out.println("Decrypted text: " + plainText.substring(0, 50));

System.out.println("Key: " + key);

for (int i = 0, j = 0; i < keyLength; i++, j = 0) {

String fragment = decrypt(fragments.get(i).toString(), key.substring(i, i + 1));

while (!isTextInformative(fragment) && j < 32) {

key.setCharAt(i, RUSSIAN\_ALPHABET.charAt(potentialKeysArray.get(i).get(j)));

fragment = decrypt(fragments.get(i).toString(), key.substring(i, i + 1));

System.out.printf(String.format("%%%ds%n", 7 + i), "|");

System.out.println("Key: " + key);

j++;

}

}

plainText = decrypt(cipherText, key.toString());

System.out.println("Decrypted text: " + plainText.substring(0, 50));

return plainText;

}

private static String decrypt(String cipherText, String key) {

StringBuilder plainText = new StringBuilder();

cipherText = cipherText.toLowerCase();

key = key.toLowerCase();

int textLength = cipherText.length();

char firstLetter = RUSSIAN\_ALPHABET.charAt(0);

for (int i = 0, j = 0; i < textLength; i++) {

char c = cipherText.charAt(i);

if (RUSSIAN\_ALPHABET.contains("" + c)) {

plainText.append((char) ((c - key.charAt(j) + RUSSIAN\_ALPHABET\_LENGTH) % RUSSIAN\_ALPHABET\_LENGTH + firstLetter));

j++;

j %= key.length();

}

}

return plainText.toString();

}

public static void main(String[] args) throws IOException {

Path pathToFile = Path.of("resources", "TEXT");

String plainText = new String(Files.readAllBytes(pathToFile))

.toLowerCase()

.replaceAll(String.format("[^%s]", RUSSIAN\_ALPHABET), "");

System.out.println("Coincidence index for plain text : " + coincidenceIndex(plainText));

String encryptedText;

encryptedText = encrypt(plainText, "он");

System.out.println("Coincidence index for key with length 2: " + coincidenceIndex(encryptedText));

encryptedText = encrypt(plainText, "бог");

System.out.println("Coincidence index for key with length 3: " + coincidenceIndex(encryptedText));

encryptedText = encrypt(plainText, "царь");

System.out.println("Coincidence index for key with length 4: " + coincidenceIndex(encryptedText));

encryptedText = encrypt(plainText, "война");

System.out.println("Coincidence index for key with length 5: " + coincidenceIndex(encryptedText));

encryptedText = encrypt(plainText, "левниколаевичтолстой");

System.out.println("Coincidence index for key with length 20: " + coincidenceIndex(encryptedText));

String cipherText = new String(Files.readAllBytes(Paths.get("resources", "cipher text.txt")));

String decryptedText = decrypt(cipherText);

Files.write(Paths.get("resources", "plain text.txt"), decryptedText.getBytes());

}

}