SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

LÚŠTENIE TRANSPOZIČNÝCH ŠIFIER

2. zadanie na predmet Klasické šifry

Karina Fábryová – ID 92013
Juraj Paška – ID 92318
Slavomír Slavejko – ID 92145
Pavol Sobota – ID 92202

Github repozitár: https://github.com/slavejko/Zadanie2

Zimný semester 2019/2020

Obsah

I. Znenie zadania 3
II. Návrh riešenia 4
II.I. Návrh horolezeckého algoritmu 4
II.II. Návrh hrubej sily 5
II.III. Návrh genetického algoritmu 6
III. Štatistiky n-gramov 7
IV. Riešenie a implementácia 8
IV.I. Riešenia slovníkov a riešenie pomocou hrubej sily 8
IV.II. Genetický algoritmus 8
IV.III. Horolezecký algoritmus 8
V. Výsledky 14
V.I. Ohodnotenie textu a *fitness* funkcia 14
V.II. Hrubá sila 15
V.III. Genetický algoritmus 15
V.IV. Horolezecký algoritmus 16
VI. Záver a zhodnotenie 18
VII. Použité zdroje 19

I. Znenie zadania

Našou úlohou bolo implementovať a porovnať minimálne dve rôzne metódy lúštenia transpozičných šifier (ako napríklad anagramovú metódu, algoritmy meta-heuristiky a podobne). Po dôkladnej analýze problematiky bolo cieľom naprogramovať aplikáciu, ktorá bude schopná lúštiť transpozičné šifry vo všeobecnosti tak, že budeme mať k dispozícií len zašifrovaný text.

V prípade meta-heuristiky bolo vhodné otestovať a porovnať rôzne heuristiky (napr.: HC, GA, SA, TS) ako aj rôzne spôsoby ohodnotenia (štatistika n-gramov, slovník, atď.).

Aplikácia bola naprogramovaná v programovacom jazyku Java.

II. Návrh riešenia

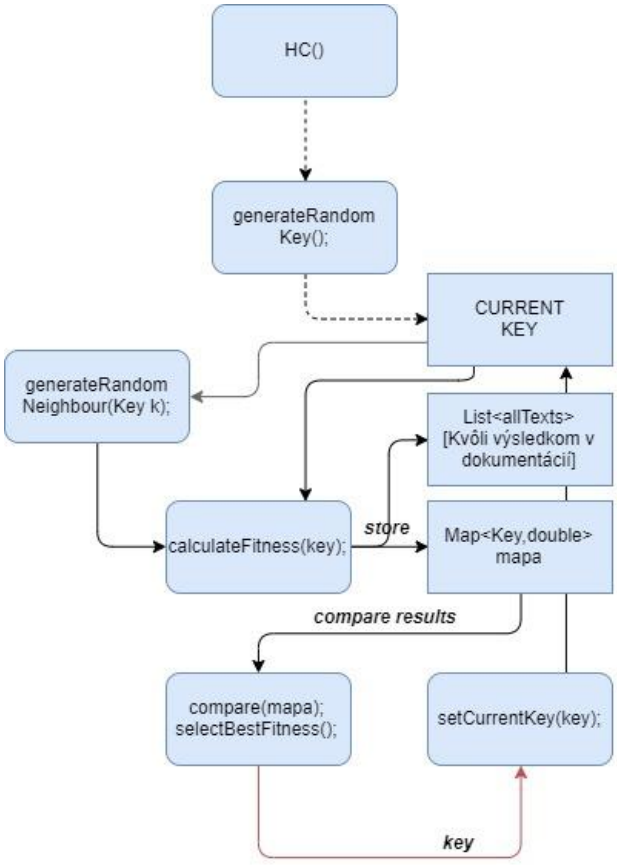
Ako bolo spomínané v zadaní, mali sme za úlohu navrhnúť a implementovať komplexný kryptografický systém, ktorý má používateľovi poskytnúť užitočné metódy, funkcie a implementované algoritmy pre splnenie cieľa – zlomenia transpozičnej šifry.

Rozhodli sme sa pre implementáciu dvoch meta-heuristických algoritmov, a to konkrétne genetického algoritmu a horolezeckého algoritmu. V týchto algoritmoch ohodnocujeme kvalitu permutácie pomocou štatistiky n-gramov vo funkcií *fitness*.

Nakoľko sa jedná o transpozičnú šifru, frekvencia výskytu písmen zostáva rovnaká, a preto to nezohráva veľkú úlohu v našom riešení. Rozhodli sme sa používať n-gramy najmenej dĺžky 2 a najviac 4 (bi-gramy, tri-gramy, quad-gramy) z toho dôvodu, že vyššie n-gramy ako tie, ktoré používame, nemajú veľký zmysel, pretože by boli náročné na výpočet. Túto vedomosť sme našli v skriptách ku predmetu Klasické šifry na strane 158 a použili sme všeobecný vzorec:

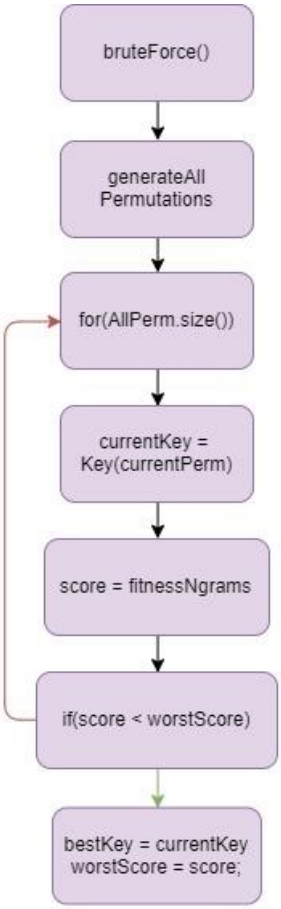
$$\Delta = \sum_{n=1}^{n_{\max}} \alpha_n \sum_{i=1}^{N^n} \left| J_i^{(n)} - D_i^{(n)} \right|$$

II.I. Návrh horolezeckého algoritmu



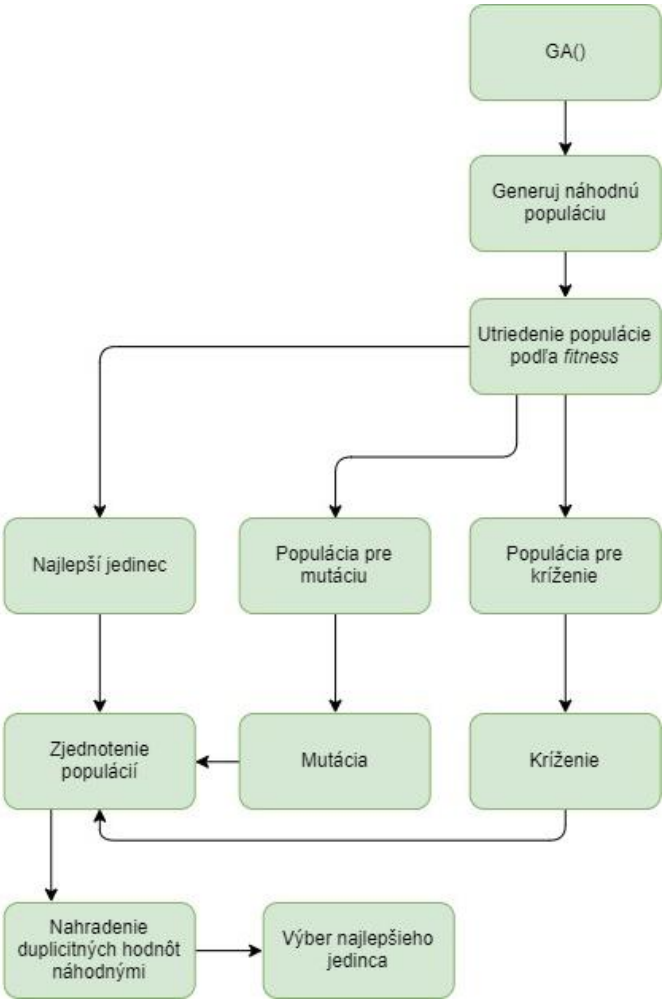
Obrázok 1: Návrh HC

II.II. Návrh hrubej sily



Obrázok 2: Návrh hrubej sily

II.III. Návrh genetického algoritmu



Obrázok 3: Návrh GA

III. Štatistiky n-gramov

V našom návrhu je možné vypočítať štatistiky n-gramov dvoma spôsobmi, a to:

- priamo z textu a zadaním parametra dĺžky n-gramov,
- načítania priamo z príslušného súboru do mapy (súbor však musí byť vo vhodnom formáte).

Tieto funkcie rovno vracajú mapy n-gramov. Tie je vhodné ešte ďalej zoskupiť do jedného listu, kvôli iterácií vo *fitness* funkcií. Preto je možné a potrebné mapy bi-gramov, tri-gramov a quad-gramov zapísať do jedného ArrayListu.

IV. Riešenie a implementácia

IV.I. Riešenia slovníkov a riešenie pomocou hrubej sily

Pri riešení sme postupovali nasledovne:

- Vytvorili sme si štatistiku n-gramov, ktorú sme uložili do hash mapy. Táto štatistika nám neskôr slúži vo funkcií *fitness* na ohodnocovanie permutácie pri dešifrovaní textu.
- Ďalej sme si vytvorili pomocnú triedu TranspositionCipher, v ktorej šifrujeme a dešifrujeme podľa riadkov a stĺpcov.
- Taktiež sme pridali útok hrubou silou pre porovnanie efektívnosti. Útok hrubou silou spočíva v tom že vyskúša všetky možné kombinácie permutácií a ohodnotí ich. Spôsob ohodnocovania sme si vybrali taký, že sme ku jednotlivým n-gramom priradili váhy.
- Potom sme v nami odšifrovanom texte pomocou aktuálnej skúšanej permutácia nachádzali n-gramy a vzdialenosť medzi nimi, a pomocou funkcie, ktorá je v skriptách, sme ohodnocovali skúšanú permutáciu určitou numerickou hodnotou. Pre naše riešenie sme sa to rozhodli aplikovať ako minimum, čiže čím menšie ohodnotenie, tým by malo byť lepšie riešenie.

IV.II. Genetický algoritmus

Naprogramovaný genetický algoritmus funguje na princípe klasického genetického algoritmu. Vyberie sa náhodná permutácia, a ku nej sa do populácie pridajú ďalšie permutácie. Pomocou kríženia a mutovania vytváram nových jedincov, pričom vždy 3 najlepších si presúvam do ďalšej generácie. Pod pojmom "najlepší jedinec" by sa malo chápať takého jedinca, ktorý ma najmenšie ohodnotenie pomocou *fitness* funkcie z dostupnej populácie. Ako bolo už spomenuté, nami implementovaná *fitness* funkcia pracuje na základe ohodnotenia n-gramov a zvolili sme si najlepšie ohodnotenie ako minimum.

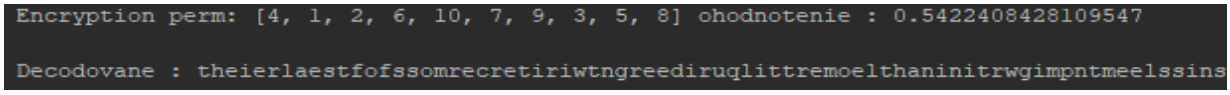
IV.III. Horolezecký algoritmus

Nami implementovaný horolezecký algoritmus spočíva v tom, že si náhodne zvolíme vstupnú permutáciu a potom v nej generujeme suseda. Následne ohodnotíme obe permutácie, a ktorá bude mať lepšie ohodnotenie, tú zvolíme ako najlepšiu a proces znovu opakujeme.

Nevýhodou takéhoto algoritmu je, že dochádza ku zaseknutiu alebo zacykleniu sa v lokálnych extrémoch (v našom prípade v lokálnom minime, keďže hľadáme čo najmenšie ohodnotenie). Toto zacyklenie môžeme však, kvázi naivne, ošetriť tým, že pokiaľ vidíme, že sa náš stav nezlepšil za posledných x iterácií, môžeme stopnúť proces s tým, že buď sa našlo korektné riešenie alebo sme sa zasekli v lokálnom extréme. Aby sme sa vyhli takýmto stavom, použili sme cyklus, ktorý nám určitý počet krát znovu spustí horolezecký algoritmus. Toto riešenie však negarantuje zbavenie sa lokálnych extrémov a nájdenie globálneho minima, nakoľko je to prvotná permutácia zvolená náhodne. Taktiež to n-krát zvýši čas na nájdenie výsledku.

V porovnaní s hrubou silou, tento algoritmus dáva nepresnejšie výsledky. Dáva to zmysel, nakoľko hrubá sila prejde všetky možné permutácia a horolezecký algoritmus prejde určitý

počet permutácií, a z nich vyberá tú najlepšiu. Avšak, na rozdiel od útoku hrubou silou, ktorá dokáže v normálnom čase nájsť permutáciu približne do dĺžky 7-8, horolezecký algoritmus zvláda permutácie i s väčšou veľkosťou. Napríklad, pri dĺžke permutácie 5, hrubá sila našla správnu permutáciu v priemere za 4,5 sekundy. Horolezecký algoritmus našiel buď správnu permutáciu alebo permutáciu, ktorou text po dešifrovaní bol relatívne čitateľný, v priemere za 3,5 sekundy. Avšak pri dĺžke 10, hrubá sila, podľa našich výpočtov, by trvala už niekoľko hodín, ale horolezecký algoritmus našiel buď správne alebo približne správne riešenie v normálnom čase.



Obrázok 4: Riešenie HC

Poznámka k obrázku 4: približne správny výsledok, je to intuitívne čitateľné
Text mal znieť nasledovne: the earliest forms of secret writing required little more than...

V nasledujúcej časti si preberieme triedy z našej aplikácie a ich funkčnosť:

TRIEDA NgramStats:

- načítanie n-gramov a ich štatistík zo súboru: readNgramFromFIle()
- vypočítanie n-gramov z textu: readNgram()
- vypočítanie vzdialenosti všetkých n-gramov (dĺžky 2 až 4)
- *fitness* funkcia, ktorá pre zadaný kľúč a text, ktorý vznikol dešifrovaním týmto kľúčom, vypočíta n-gramy textu a porovná ich s referenčnými hodnotami → následne ohodnotí kľúč

TRIEDA Dictionary:

- načítanie slovníka
- lowercase, normalize
- zápis do súboru

TRIEDA SlovakDicionary(EnglishDictionary):

- po vzniknutí inštancie danej triedy načíta referenčné hodnoty všetkých n-gramov z príslušných súborov a zapíše ich do zoznamu máp n-gramov

TRIEDA BruteForceAttack:

- obsahuje funkciu na útok hrubou silou, s ohodnotením kľúčov a vybratím najlepšieho kľúča príslušnej dĺžky (brute force je efektívny do dĺžky klúča 8)

TRIEDA Permutations:

- náhodná permutácia
- inverzná permutácia
- permutácia zo slova

TRIEDA TranspositionCipher:

- šifrovanie a čítanie po riadkoch
- šifrovanie a čítanie po stĺpcoch
- dešifrovanie a čítanie po riadkoch
- dešifrovanie a čítanie po stĺpcoch

TRIEDA TranspositionKey:

- enc perm
- dec perm
- perm len
- score

TESTY Z MAINU:

1) testovanie šifrovania a dešifrovania, čítanie po riadkoch
2) testovanie šifrovania a dešifrovania, čítanie po stĺpcoch
3) testovanie slovných permutácií
4) testovanie načítania slovníka zo súboru (priamy aj cez JFileChooser)
5) testovanie načítania frekvencie n-gramov
   - súčet frekvencií (relatívnych) vykazuje 100% +/- 0.000001%
6) testovanie funkcie vzdialenosti n-gramov
7) testovanie *fitness* funkcie, ktorá berie ako vstup kľúč text, na ktorom sa bude aplikovať dešifrovacia permutácia, referenčné hodnoty n-gramov a ohraničenie n-gramov
   - funkcia použije dešifrovaciu permutáciu a čítanie buď po stĺpcoch alebo po riadkoch
   - následne vypočíta príslušné n-gramy a vzdialenosť vypočítaných n-gramov voči referenčným hodnotám
8) brute force testy

TRIEDA Chromosome:

- reprezentácia kľúča ako pole Integerov
- obsahuje Double Score pre *fitness* hodnotu daného reťazca

TRIEDA Population:

- ArrayList<Chromosome> populácia dĺžky n jednotlivých chromosomov
- metóda **genRandPop()** generuje náhodnú populáciu kľúčov
- metóda **mergePop()** spája dokopy 2 populácie

TRIEDA Selection:

- trieda na výber najlepších jedincov alebo náhodných jedincov
- vstupy: zašifrovaný text, mapa frekvencií n-gramov
- metóda **setScoreToPop()** ohodnotí jednotlivé chromosomy podľa *fitness* funkcie
- metóda **sortPop()** zotriedi ohodnotenú populáciu od najnižšej *fitness* po najvyššiu
- metóda **selBest()** vyberie z utriedenej populácie n jedincov
- metóda **selRand()** do novej populácie vloží n jedincov z utriedeného poľa

TRIEDA Mutation:

- trieda, pomocou ktorej sa robia vybrané permutačné operácie na populácií
- metóda **muteSwap()** prehodí 2 náhodné prvky v permutácií
- metóda **mutePart()** preusporiada poradie prvej a druhej polovice v permutácií

TRIEDA CrossBreed:

- kríženie medzi jedincami populácie

TRIEDA Genetic:

- hľadá kľúč transpozičnej šifry
- metóda **removeDuplicates():** duplikáty v novovytvorenej populácii nahrádza náhodnými permutáciami (snaha o odstránenie lokálneho minima)
- metóda **genetic()**
    - hľadá najlepšieho jedinca v 1000 iteráciách v populáciách o veľkosti 20 jedincov, dvaja najlepší ostanú nezmenení a vstupujú do ďalšej iterácie, 24 jedincov sa skríži a vytvorí 12 nových unikátnych (8 jedincov zmutuje 12+8+2 je nová populácia o veľkosti 20)
    - program bol testovaný *fitness* funkciou na mapách n-gramových frekvencií **listngramDec,** ktorá pracuje s ideálnym prípadom zhody n-gramov, nie s referenčnými hodnotami
    - aj napriek tomu, častokrát nie je možné nájsť výsledok a to z dôvodov:
        - **prepisovanie minimálneho jedinca**
        - **zlá implementácia volania *fitness* funkcie**
    - samotný genetický algoritmus pracuje tak, ako by mal, avšak chyba sa vyskytuje v nesprávnej implementácií volania ***fitness* funkcie**

TRIEDA Hill_Climb:

- hľadá kľúč transpozičnej širy
- obsahuje pomocné funkcie za účelom:
    - vytvorenia permutácie
    - znáhodnenia permutácie
    - vygenerovanie suseda danej permutácií
- metóda **hill_climb**:
    - hľadá najlepšie ohodnotenú permutáciu
    - pre odstránenie lokálnych extrémov je implementovaný for() cyklus
    - táto metóda využíva funkciu *fitness*, ktorá ohodnocuje danú permutáciu na základe výskytu n-gramov, a tieto hodnoty načítava z referenčných zdrojov
    - aktuálne nastavenie:
        - Počet cyklov pre zníženie šance lokálneho extrému : 10
        - Počet iterácií : 100
        - Hľadanie dĺžky klúča v rozmedzí : 5-15

- každý dešifrovaný text spolu so všetkými informáciami uloží do Listu<>
- konečnú (najlepšie ohodnotenú) permutáciu vypíše, a zároveň pomocou nej aj dešifruje zadaný text

TRIEDA AllTexts:

- trieda objektu
- slúži na uchovavánie informácií o jednej z výsledných permutácií, jej dĺžke, ohodnoteniu a texte, ktorý bol dešifrovaný danou permutáciou (klúčom)

V. Výsledky

V.I. Ohodnotenie textu a *fitness* funkcia

Text ohodnocujeme pomocou štatistík n-gramov pomocou vzorca $\Delta = \sum_{n=1}^{n_{\max}} \alpha_n \sum_{i=1}^{N^n} \left| J_i^{(n)} - D_i^{(n)} \right|$, ktorý sme spomenuli vyššie. Vo vzorci vystupuje N_{max} udávajúci obmedzenie pre najväčšie možné a váhy každej skupiny n-gramov (alfa 1 až max 4). Pre transpozičné šifry je váha frekvencií samotných písmen 0. Váhy dlhších n-gramov postupne rastú s ich dĺžkou, preto čím je n-gram dlhší, tým ma väčšiu váhu. Vzorec následne prechádza všetky možné kombinácie n-gramov príslušnej dĺžky a ráta absolútnu hodnotu z rozdielov n-tých mocnín referenčných (slovníkových) hodnôt a hodnôt z hodnoteného textu: $|(J_i)^n - (D_i)^n|$.

Tieto výpočty sú potrebné pre implementáciu *fitness* funkcie, ktorá priraďuje skóre adeptom pre nájdenie správneho kľúča, ktorým bol text zašifrovaný. Po nájdení vhodného kľúča je použitá dešifrovacia permutácia danej inštancie kľúča a text je dešifrovaný. *Fitness* funkcia pracuje nasledovne:

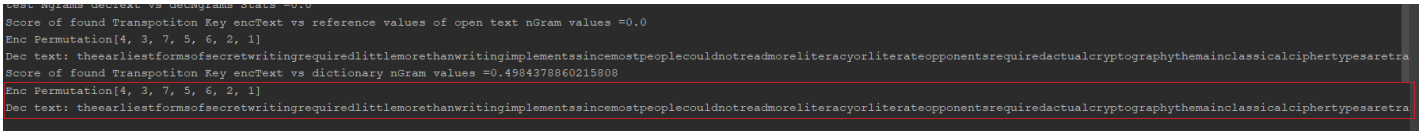
1) funkcia zoberie ako vstup zašifrovaný text, transpozičný kľúč a referenčné hodnoty n-gramov (slovníkové)
2) následne je zašifrovaný text dešifrovaný pomocou dešifrovacej permutácie daného kľúča a sú vypočítané štatistiky n-gramov
3) vypočítané n-gramy sú následne porovnané s referenčnými hodnotami implementovaným vzorcom, ktorého výstup udáva skóre pre daný kľúč
4) kľúč s najmenším skóre je s veľkou pravdepodobnosťou tým kľúčom, ktorý bol použitý na zašifrovanie textu

Nakoľko sme si zvolili spôsob ohodnocovania touto metódou, je potrebné mať dostatočne dlhý text na správne určenie hodnôt n-gramov, pretože tieto hodnoty priamo ovplyvňujú schopnosť funkcie *fitness.* Pri našom testovaní sme zistili, že hranica, kde pri použití horolezeckého algoritmu je ešte rozlúštený text čitateľný, je približne 300 znakov. Optimálny počet znakov by sa nachádzal v rozmedzí nad 1500 znakov.

V.II. Hrubá sila

Útok hrubou silou je výpočtovo najnáročnejší, preto je potrebné dostatočne vyprofilovať prípad použitia a vlastnosti množiny prehľadávaných kľúčov touto metodikou. Obmedzenia sa vzťahujú najmä na dĺžku kľúča, ktorého faktoriál nám udáva počet kľúčov, ktoré je potrebné prehľadať a otestovať.

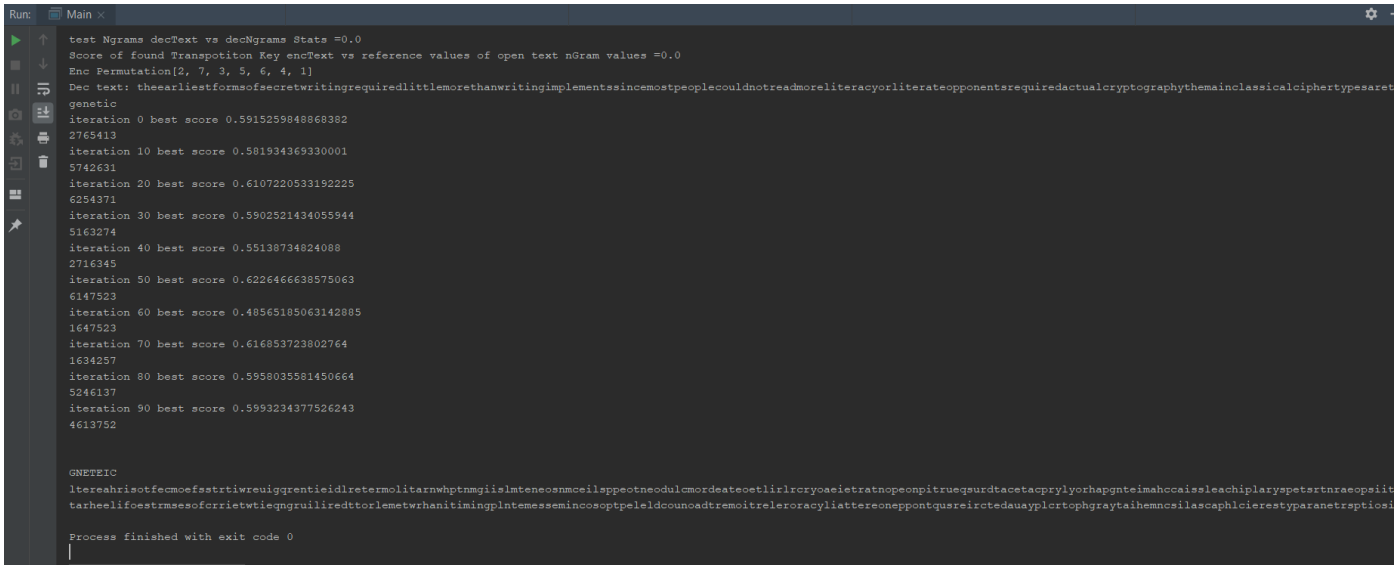
Podľa nášho skúmania sme došli k záveru, že útok hrubou silou vykazuje lepšie výsledky v podobnom čase ako horolezecký algoritmus do dĺžky kľúča 7. Nad dĺžku kľúča 7, útok hrubou silou už trvá priveľmi dlho na to, aby sme to vedeli klasifikovať ako „v normálnom čase“. Útok hrubou silou prejde všetky možnosti kľúča, čiže nájde aj kľúč, ktorým bola správa zašifrovaná.



Obrázok 5: Výsledok implementácie hrubej sily

V.III. Genetický algoritmus

Hľadanie kľúča transpozičnej šifry najmä pre veľké texty a veľké kľúče, kde referenčné hodnoty n-gramov sú približne rovnaké ako hodnoty n-gramov zašifrovaného textu. Keďže program nemá metódu ako zastaviť, na základe úspešnosti alebo percentuálnej zhody, musí sa vždy na konci iterácií najlepším kľúčom text preložiť a skontrolovať tak, či nám dáva zmysel. V ideálnom prípade by mal genetický algoritmus nájsť približne podobný kľúč akým sme text zašifrovali, v čase kratšom ako Hill Climbing, ale je dosť pravdepodobné, že sa to tak nemusí vôbec stať.

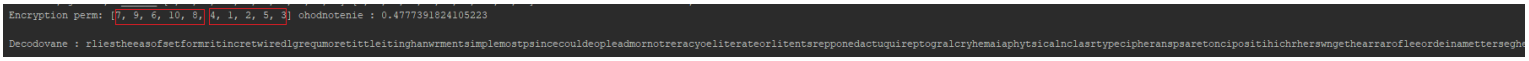


Obrázok 6: Výsledok implementácie GA

V.IV. Horolezecký algoritmus

I napriek pokusom o optimalizáciu presnosti výsledku, tento postup nedôjde úplne vždy ku správnemu riešeniu (permutácia, s ktorou bol pôvodný OT zašifrovaný). Avšak, dôjde ku výsledku, s ktorým po rozlúštení je možné pomocou ľudských faktorov dôjsť k správnemu riešeniu.

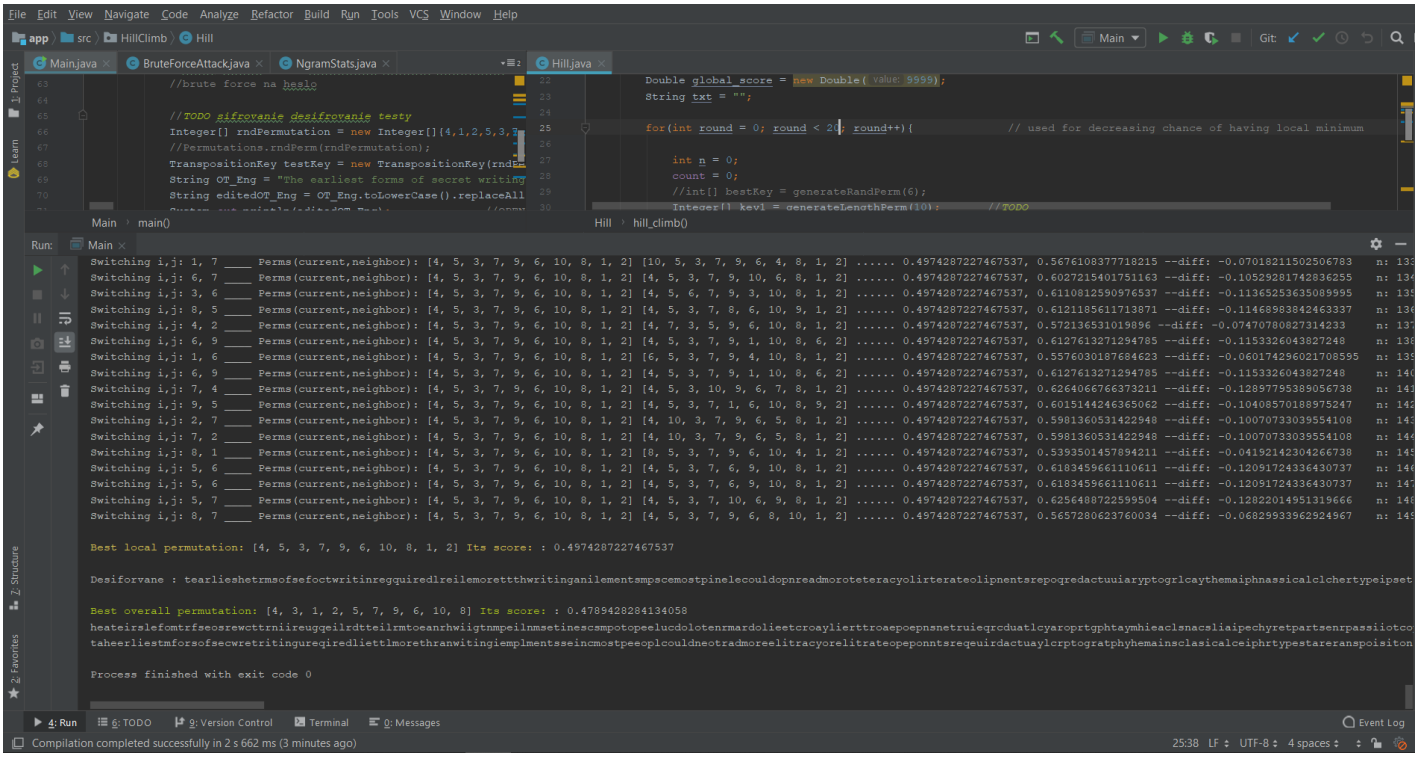
Častokrát tento algoritmus na základe ohodnotenia n-gramov nájde správne časti permutácie, i keď nie vždy ich dá na správne poradie.



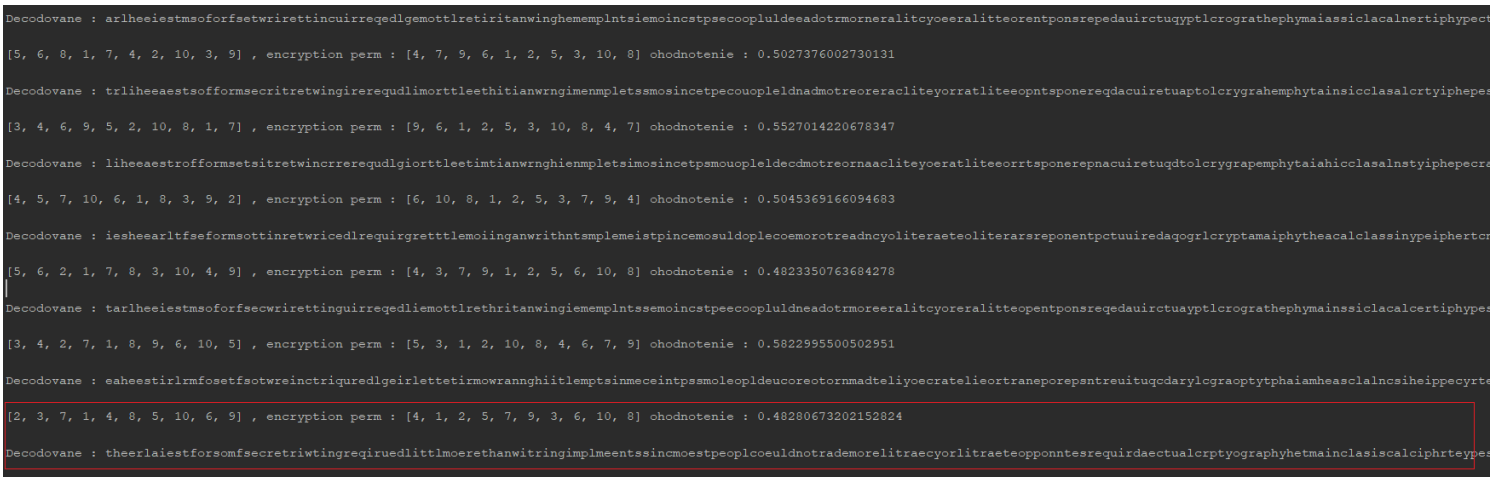
Obrázok 7: HC - výber poradia

Pri väčších dĺžkach kľúča je to už komplikovanejšie, nakoľko zlé umiestnenie takýchto subpermutácií môže spôsobiť, že text už nebude tak ľahko čitateľný.

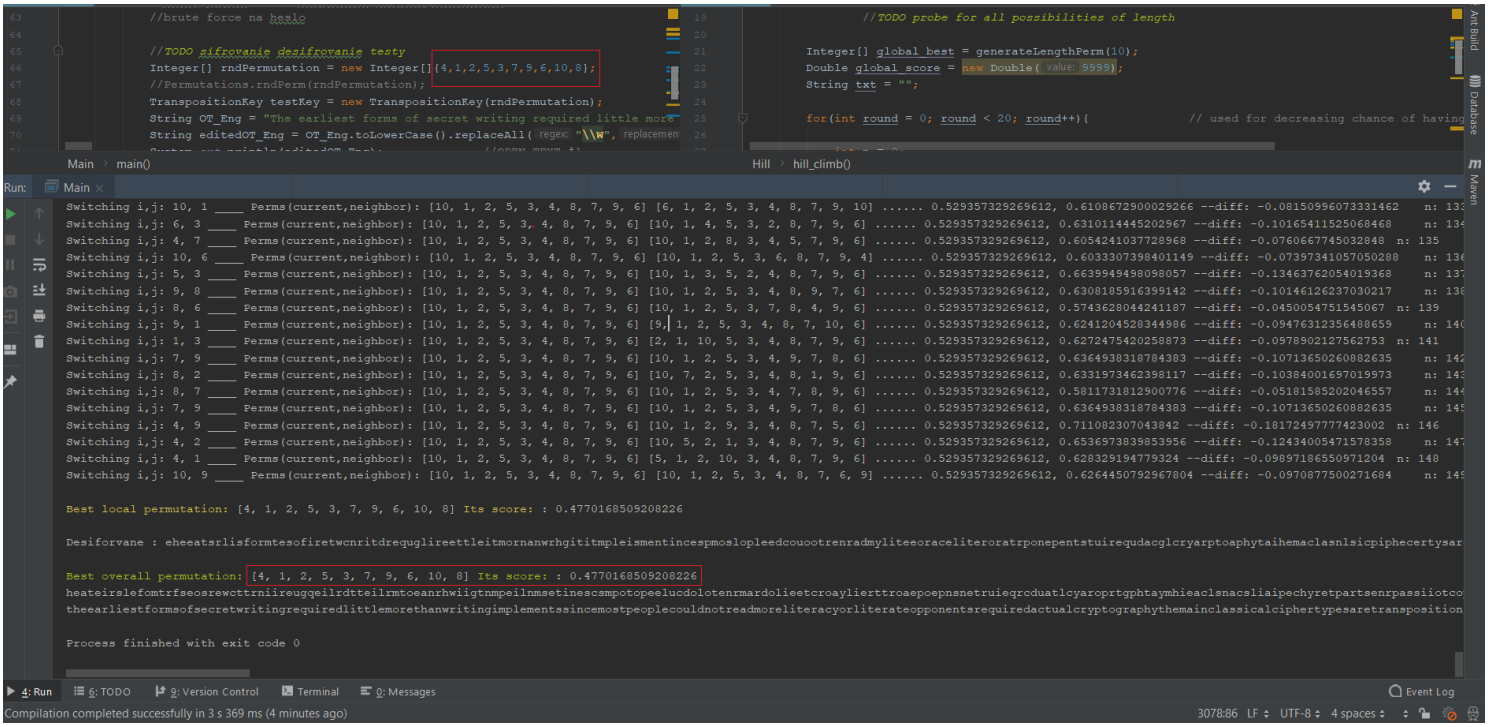
Pri implementácií sme sa museli rozhodnúť, či požadujeme presnosť alebo rýchlosť. Pre zväčšenie presnosti, ako už bolo spomínané, sme pridali cyklus, ktorý nám znovu spustí algoritmus s náhodnou permutáciou, a taktiež sme zvýšili počet iterácií prehľadávania susedov. Keby požadujeme väčšiu rýchlosť, snažíme sa nájsť čo najmenší počet iterácií taký, aby nám algoritmus stále vedel nájsť prijateľné riešenie. Druhou možnosťou pre zvýšenie rýchlosti by bolo odstránenie cyklu na znovu-spustenie celého algoritmu, čím by sme zvýšili šancu, že riešenie, ktoré nájdeme, bude ovplyvnené lokálnym extrémom.



Obrázok 8: Výsledok implementácie HC



Obrázok 9: Výsledok implementácie HC - takmer bezproblémovo čitateľné



Obrázok 10: Správny výsledok implementácie HC

Poznámka k obrázku 10: HC vylúštilo zašifrovaný text s permutáciou dĺžky 10 pri 20 náhodných vstupoch do množiny permutácií s počtom iterácií 150 → časovo lepšie ako útok hrubou silou

VI. Záver a zhodnotenie

Došli sme k záveru, že nami implementované riešenie tohto zadania je v celku účinné pre kľúče do dĺžky 15.

Útok hrubou silou by sme použili do dĺžky kľúča 7, ďalej by sme využili horolezecký algoritmus alebo genetický algoritmus. Nad dĺžky kľúča 15, by sme museli implementovať spojenú funkciu, kde by sme najprv využili horolezecký algoritmus na nájdenie subpermutácií, ohodnotili dané subpermutácie ľudským faktorom a následne by sme túto permutáciu uložili do genetického algoritmu, kde by sme stanovili pozície subpermutácií, a ich správnosť.

Horolezecký algoritmus do dĺžky 15 vie aspoň približne nájsť korektnú permutáciu na základe ohodnotenia z *fitness* funkcie. Keby si lúštiteľ spustí horolezecký algoritmus, tak ako je nastavený teraz, získal by informácie i z nesprávnych riešení, ktoré by boli zobrazené a na základe nich by vedel určiť, či sú subpermutácie alebo časti správnej permutácie, na správnom mieste, prípadne intuitívne by vedel už dešifrovať samotný text. Výsledná permutácia samozrejme nemusí byť zrovna tou šifrovacou, ale jej dĺžka môže taktiež byť jej násobkom.

Ako bolo spomínané vyššie, je potrebné mať dostatočne dlhý text. Pri našom testovaní sme skúsili viacero dĺžok textu, kde pri dĺžke 350 znakov už bol text ťažšie čitateľný i pre lúštiteľa. Pri dĺžke textu 1000 bol text intuitivne čitateľný väčšinu času, s tým, že sa v permutácií nachádzali správne sub-permutácie. S dĺžkou textu 1800 znakov bola úspešnosť pomerne vysoká, nakoľko sa nám často krát prejavoval klúč správny, poprípade istá sub-permutácia bola vymenená v poradi s druhou sub-permutáciou, čož by bolo zrejmé ľudskému lúštiteľovi.

```
BWINNER: [4, 7, 6, 2, 5, 1, 3] WINNER score: : 0.4984378860215808


reltaehotrifseefcmsosiwtrrteqruiegnidtilerrmetoelrnitwahmgptinteslnmemcosenipelsoptdunelocdemoarttleoierrylrocatreiaeteononppueitqrstaurcdeprtayclhayoprgimntaehisccsalhieapclspareytnrsrateiioptso
theearliestformsofsecretwritingrequiredlittlemorethanwritingimplementssincemostpeoplecouldnotreadmoreliteracyorliterateopponentsrequiredactualcryptographythemainclassicalciphertypesaretranspositio


GA

genetic
iteration 0 best score 0.6409495608707934
5624137
iteration 10 best score 0.6723147759316493
2351476
iteration 20 best score 0.6064226249030192
3642517
iteration 30 best score 0.636472997087328
5346712
iteration 40 best score 0.5653602420954001
5316724
iteration 50 best score 0.6375386090861894
6243175
iteration 60 best score 0.6022023172104003
3516472
iteration 70 best score 0.6235121227719864
3156274
iteration 80 best score 0.6060587868406037
6371542
iteration 90 best score 0.5466511165151304
5712643
reltaehotrifseefcmsosiwtrrteqruiegnidtilerrmetoelrnitwahmgptinteslnmemcosenipelsoptdunelocdemoarttleoierrylrocatreiaeteononppueitqrstaurcdeprtayclhayoprgimntaehisccsalhieapclspareytnrsrateiioptso
rlethaeortiefsecfmssoitwrertqurinegitdirleremtloerinthwampgtiintselenmmocsienplestopdnueclodmeotartelorierlyraocteritaeenoopnpuietsqrtuarecdptralychyaogprinmthaeicsclsaheialpcsaprteynsrreatioipots

Process finished with exit code 0
```

Obrázok 11: Podobnosť horolezeckého algoritmu a genetického algoritmu

VII. Použité zdroje

- GROŠEK, O. -- VOJVODA, M. -- ZAJAC, P. *Klasické šifry.* Bratislava : STU v Bratislave, 2007. 214 s. ISBN 978-80-227-2653-5
- zdrojové kódy a iné pomocné materiály z cvičení
- https://stackoverflow.com