METODE NUMERICE: Tema #1
Criptografie și matrice

Termen de predare: 5 aprilie 2015

Titulari curs: *Florin Pop, George Popescu*

Responsabil Laborator: **David Iancu, Bogdan Marchiș, Alexandru Țifrea**

Obiectivele Temei

În urma parcurgerii acestei teme studentul va fi capabil să:

- calculeze inverse de matrice folosind GNU Octave și să eficientizeze calculul acestora
- identifice forme de matrice care satisfac anumite proprietăți necesare pentru rezolvarea unei probleme.

Task1 - Criptografie (45p)

A. Dorel e pasionat de criptografie și e mereu dornic să descopere metode noi de codificare pe care le testează apoi cu prietenii săi. Recent, a propus următorul algoritm:
i) Textul în clar (i.e. textul care urmează să fie codificat) va fi convertit într-un șir de numere prin asignarea unui număr fiecărui simbol din alfabet (ex. 'a'-1, 'b'-2, 'c'-3 etc.). Dorel va codifica fiecare literă folosind poziția ei în alfabetul latin, pornind de la 1. 0 va codifica un spațiu liber. 27 va fi codul lui '.' și 28 va fi codul lui '. Textul nu va conține decât litere, spații, simbolul punct (.) și simbolul apostrof ('). Literele mari sau mici se vor codifica la fel (ex. 'A' și 'a' vor fi ambele 1).
ii) Se alege o matrice pătratică $n \cdot n$ care va fi cunoscută doar de către cel care trimite mesajul și cel căruia îi este destinat.
iii) Se împarte șirul de numere obținut la i) în blocuri, care sunt apoi înmulțite succesiv cu matricea de codificare. Dacă lungimea șirului de numere nu este divizibilă cu n, atunci se completează cu spații libere (respectiv, cu zerouri).
iv) Șirul de numere obținut va fi convertit la un șir de simboluri din alfabet (i.e litere, spațiu, punct sau apostrof). De exemplu, 0 devine spațiu, 1 devine 'a' ș.a.m.d. Șirul obținut va conține doar litere mici (lowercase).

Cerință:
Va trebui să scrieți un program în Octave care să implementeze algoritmul lui Dorel. Programul va citi textul din fișierul *input1A* (care conține pe prima linie textul care trebuie criptat) și va citi matricea de codificare din fișierul *key1A* (care conține pe prima linie valoarea lui n și pe următoarele n linii, matricea de codificare). Textul codificat va fi scris în fișierul *output1A*.

Input: Funcția voastră nu va primi niciun parametru.

Output: Funcția voastră se va numi matrixCipher și va tipări în *output1A* un string care reprezintă mesajul criptat.

Exemplu:
*key*1*A*:
3
6 24 1
13 16 10
20 17 15
*input*1*A*:
"cat" (fără ghilimele)

output:
"dw " (fără ghilimele)

Restricții și precizări:

- n este dimensiunea unei matrice
- $0 < n \leq 1000$
- Pentru codificare folosiți următoarea convenție: ' ' are codul 0, '.', are codul 27, ' are codul 28 și toate cele 26 de litere au codul egal cu numărul de ordine al literei în alfabetul latin (de exemplu, 'a' are codul 1, 'b' are codul 2 ș.a.m.d.)

B. Dorel obișnuiește sa-i trimită mesaje colegei sale, Maricica. Pentru început, el îi spune Maricicăi ce matrice folosește pentru codificare. Ea poate să decripteze mesajele pe care le primește astfel:
i) calculează inversa modulo m a matricei de codificare (unde m e numărul de simboluri din alfabet - în cazul nostru $m = 29$). Inversa modulo m B a unei matrice A, este acea matrice care satisface: $AB \equiv I(mod\ m)$. A nu se confunda inversa modulo m a unei matrice cu inversa clasică a matricei, pe care se aplică apoi operația modulo.
ii) se aplică asupra textului criptat algoritmul de la **A**, folosind în locul matricei de codificare, inversa modulo m a acesteia.

Cerință:
Veți primi în fișierul *input1B* mesajul codificat primit de Maricica de la Dorel. În fișierul *key1B* veți avea pe prima linie dimensiunea matricei de codificare cu care a fost criptat mesajul n și pe următoarele n linii va fi dată matricea de codificare folosită de Dorel pentru a cripta mesajul. Va trebui să implementați în Octave algoritmul prin care Maricica poate decodifica mesajul. În fișierul *output1B* va trebui sa fie textul decodificat. Se garantează faptul că decodificarea textului este posibilă (pentru ca decodificarea să fie posibilă, trebuie ca determinantul matricei de codificare și numărul de simboluri din alfabet să fie două numere prime între ele).

Input: Funcția voastră nu va primi niciun parametru.

Output: Funcția voastră se va numi decrypt și va tipări în *output1B* un string care reprezintă mesajul decodificat.

Exemplu:
*key*1*A*:
3
6 24 1
13 16 10
20 17 15
*input*1*A*:
"dw " (fără ghilimele)

output:
"cat" (fără ghilimele)

Restricții și precizări:

- n este dimensiunea unei matrice
- $0 < n \leq 1000$
- e valabilă convenția stabilită la punctul A legată de modul de codificare a caracterelor.

C. Daniel vrea să codifice și el niște mesaje folosind metoda lui Dorel. S-a gândit să folosească un cifru de transpoziție puțin modificat. Un cifru de transpoziție codifică fiecare literă din mesaj folosind a k-a literă de după ea. Astfel, pentru $k = 2$ de exemplu, 'a' va deveni 'c', 'z' va deveni 'b' ș.a.m.d.
Daniel va începe codificarea cu o valoare inițială a lui k, pe care o va incrementa după 1000 de caractere. Astfel, dacă trebuie să trimită un text de 2200 de caractere cu $k_{inițial} = 3$, va trimite primele 1000 de caractere transpuse cu 3 poziții, pe următoarele 1000 transpuse cu 4 poziții și pe ultimele 200, transpuse cu 5 poziții.

Cerință:
Va trebui să concepeți un algoritm care implementeaza algoritmul dorit de Daniel pentru un $k_{inițial}$ dat și pentru un n dat (unde n este dimensiunea numărul de caractere care vor fi codificate la un pas). Se vor citi din fișierul *input1C* de pe prima linie valorile lui $k_{inițial}$ și n și de pe a doua linie textul care trebuie codificat. Mesajul în text clar va conține doar litere ale alfabetului latin (lowercase sau uppercase), spații sau caracterele . sau '. Literele mari vor fi tratate ca litere mici, astfel încât mesajul criptat va fi lowercase. Rezultatul codificării va fi scris în fișierul *output1C*. Matricea folosită pentru codificarea primului segment de 1000 de caractere (deci cea care corespunde valorii $k_{inițial}$) va fi scrisă în fișierul *key1C*.
Atenție! Acest subpunct se poate rezolva foarte simplu fără a folosi o matrice de codificare (de exemplu, citind textul clar într-un string s și facând apoi $s = s + k$). Scopul acestui exercițiu este însa găsirea unei matrice care să corespundă unei transpoziții. Orice abordare care nu respectă cerința nu va fi punctată.

Input: Programul nu va avea niciun parametru.

Output: Funcția voastră se va numi transposition.

Exemplu:

*input*1*C*:
1 3
"Tema la MN e usoara" (fără ghilimele)

*output*1*C*:
"ufnbambanoafavtpbsb" (fără ghilimele)

Restricții și precizări:

- n este dimensiunea unei matrice
- $0 < n \leq 1000$
- checker-ul nu verifică si fișierul key1C, deoarece există mai multe matrice care rezolvă problema. Acest fișier va fi verificat manual la corectarea temei.

Task2 - Matricile sunt fun!!! (45p)

David a descoperit o nouă pasiune, pe lângă muzică și calculatoare - matricile. Lui îi plac în special operațiile care consumă foarte mult timp și memorie, cum ar fi înmulțirile de matrici, ridicările la putere și calculul inverselor. Recent, David a citit că există înmulțiri de matrici mai eficiente, cum ar fi algoritmul lui Strassen, care are o complexitate mai bună, $O(n^{log_2 7})$. Pentru ca David este curios, s-a gândit că se poate obține această complexitate și pentru aflarea inversei, folosind metoda partiționării. Pentru că acest lucru ar fi prea simplu pentru voi, el vă propune următoarea problemă: dându-se un număr N și o matrice A, cât este A^{-n}? Cerințele lui sunt următoarele: -ridicarea la putere se va face în timp logaritmic -toate înmulțirile trebuie făcute cu algoritmul lui Strassen -nerespectarea acestor cerințe duce la anularea punctajului pe problemă, iar încercările de fraudă, precum:

```
disp((A^n)^-1),
```

duc la anularea punctajului pe toata tema.

Input: Fișierul strassen.in va conține numărul N, urmat de o matrice A.

Output: Fișierul strassen.out va conține matricea A^{-n}, afișată cu 3 zecimale.

Exemplu:
strassen.in: 5
1 2 3
1 3 4
2 3 4

strassen.out: 103.000 -119.000 39.000
-364.000 245.000 23.000
229.000 -135.000 -32.000

Detalii de implementare și redactare

Tema de casă va implementa funcţiile menţionate la fiecare cerinţă în parte. Pentru implementarea temei puteţi folosi și alte funcţii definite de voi, dar cele menţionate mai sus sunt obligatorii. Trebuie să ţineţi cont de următoarele aspecte:

- codul sursă va conţine comentarii semnificative şi sugestive cu privire la implementarea algoritmilor;
- existenţa unui fişier `readme.txt` care va prezenta detaliile legate de implementarea şi testarea temei; de asemenea, la task-urile care cer explicarea anumitor aspecte, acestea trebuie să fie conținute în fisierul `readme.txt`
- fişierele care compun tema de casa vor fi incluse într-o arhivă `.zip` care respectă specificaţiile din regulamentul cursului;
- tema se va implementa în Matlab şi va fi testată în mediul Octave.
- pentru o implementare corectă se vor putea obține maximum 90 de puncte și 10 puncte vor fi acordate pe README.

Frequently Asked Questions

Înainte de a posta o întrebare pe forum, parcurgeți secțiunea următoare și celelalte thread-uri de pe forum pentru a vă asigura că întrebarea voastră nu a fost pusă deja de altcineva:
Q: Avem voie să folosim `inv(A)` sau `A^(-1)`?
A: Categoric, nu. Scopul temei este ca voi să implementați metode de inversare a matricelor și nu sa le folosiți pe cele existente in GNU Octave.

Q: Putem să folosim MatLab pentru a rezolva tema?
A: Da, cu mențiunea că MatLab pune la dispoziție mult mai multe funcții decât Octave. Cum checker-ul folosește Octave pentru verificare, va trebui să aveți grijă să folosiți doar funcții care sunt implementate și în Octave.

Resurse Web

1. http://en.wikipedia.org/wiki/Transposition_cipher
2. http://en.wikipedia.org/wiki/Hill_cipher