Exercices d'applications des mathématiques - Série n° 6

Cours 3AMOS01 Série distribuée le 13.11.17

1. Nouveau code ASCII.

Avec un ligne de commande *Octave* transformer le texte 'ceciestunessai' en un tableau de nombres où le 'a' correspond au nombre 0, le 'b' correspond au nombre 1, $\cdots$, le 'z' correspond au nombre 25.

2. Problème.

(1) Créer une commande *Octave* d'une ligne qui transforme les texte 'bonjour' en 'erqmrxu' (décalage de 3 lettres).
Indication Utiliser les commandes toascii et char.

(2) Créer une commande *Octave* d'une ligne qui transforme les texte 'abc' en 'bcd' et 'xyz' en 'yza' (décalage de 1 lettre).
Indication Utiliser les commandes toascii, char et mod. Commencer transformer le texte en nombres compris entre 0 et 25.

3. Code César.

(1) Rédiger et exécuter avec Octave un script permettant de crypter un message avec une substitution monoalphabétique de type César. Rédiger un deuxième script permettant de décrypter les messages cryptés avec le premier script.

Indications: La fonction toascii("votremessage") retourne un tableau à une ligne de nombres suivant la correspondance a→97, b→98, etc. La commande char effectue la transformation inverse. Finalement, la fonction mod(a,b) donne le reste de la division entière de a par b (par exemple, mod(14,3) donne 2).

(2) Tester les scripts en envoyant un message codé par courrier électronique à un camarade.

4. Code César.

Le texte contenu dans la variable M a été crypté avec un algorithme de substitution monoalphabétique de type César.

```
1  M="nagbaiblyaneevinvgcnfnqbezvevynyyhznfbawnmznedhnvgzvahvgivatgvycbhffnhacebsbaqfbhcvefnf
2  fvgqnaffbayvgfncchlnagfhefbacbybpubavycevghaebznavyybhievgvyyhgznvfvyalfnvfvffnvgdhhavzoeb
3  tyvbpbashfvyohgnvgngbhgvafgnagfhehazbgqbagvyvtabenvgynfvtavsvpngvbavynonaqbaanfbaebznafhef
4  bayvgvynyynnfbayninobvyzbhvyynhatnagdhvycnffnfhefbasebagfhefbapbhfbacbhyfonggnvggebcsbegvy
5  ninvgpunhqvybhievgfbainfvfgnffpehgnynahvgvysnvfnvgqbhkhaoehvgvaqvfgvapgzbagnvgqhsnhobhetha
6  pnevyybacyhfybheqdhhatynfcyhffbheqdhhagbpfvacyhfcebsbaqdhhaobheqbaabaybvafbaangebvfpbhcfqh
7  pnanyfnvagznegvahapyncbgvfcynvagvsfvtannvghapunynaqdhvcnffnvgfheynonggnagqhinfvfgnfhanavzn
8  ynhgubenkvaqvtbnynvthvyybafnsenaavhapnsneqavhapunenapbaznvfcyhgbghanegvfbafninapnvggenvana
9  ghaoevaqnysnvyfnccebpunibhynagyncyngveqhapbhcivsznvfynavznycevgfbaibyqvfcnenvffnagqnafynah
10 vgninagdhvynvgchynffnvyyveynqvfcnevgvbaqrtrbetrfcrerp";
```

Déchiffrer le texte contenu dans la variable M en procédant selon les étapes suivantes.

(1) Enlever tous les espaces avant d'exécuter le script donné ci-dessus avec *Octave* afin d'enregistrer le texte à décrypter dans la variable M.

(2) Rédiger et exécuter avec Octave un script qui affiche la fréquence relative (en %) de chaque lettre.

Indications: La commande **length**("votremessage") donne la longueur de la chaîne de caractères (dans ce cas 12) et la fonction **findstr**("votremessage", "e") retourne un tableau contenant

la position de chaque e du message (c'est-à-dire 5 7 12). Finalement, si stat est un tableau à une ligne et 26 colonnes contenant les fréquences relatives de chaque lettre du message, la commande **bar**([1:1:26], stat) affiche un graphique en colonnes. Utiliser une boucle pour calculer la fréquence relative de chaque lettre

```
?
for n=1:26
  stat(n)=?;
end
bar([1:1:26],stat)
set(gca,'xtick',[1:1:26])
set(gca,'xticklabel',{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q'
    ,'r','s','t','u','v','w','x','y','z'})
```

(3) Rédiger et exécuter avec Octave un script permettant de décrypter le message contenu dans la variable M.

Indications: La fonction toascii("votremessage") retourne un tableau à une ligne de nombres suivant la correspondance a→97, b→98, etc. La commande char effectue la transformation inverse. Finalement, la fonction mod(a,b) donne le reste de la division entière de a par b (par exemple, mod(14,3) donne 2).

5. Substitutions monoalphabétiques.

Combien y a-t-il de substitutions monoalphabétiques distinctes sur un alphabet de 26 symboles ? Et sur un alphabet de 256 symboles ?

6. Substitutions monoalphabétiques.

(1) Compléter le script suivant de manière à obtenir un programme de chiffrement par substitution monoalphabétique suivant une clé de 26 caractères donnée.

```
#alp=abcdefghijklmnopqrstuvwxyz
cle="acegikmoqsuwybdfhjlnprtvxz";
#
function M=g(m,cle)
  l=length(?);
  for n=1:?
    M(n)=cle(toascii(m(?))-?);
  end
end
#
g("bonjourcommentallezvous",cle)
```

(2) Compléter le script suivant de manière à obtenir un programme de déchiffrage pour les messages cryptés avec le script qui précède.

```
#alp=abcdefghijklmnopqrstuvwxyz
cle="acegikmoqsuwybdfhjlnprtvxz";
#
function m=g(M,cle)
  l=length(?);
  for n=1:?
    m(n)=char(findstr(cle,M(?))+?);
  end
end
#
g("cdbsdpjedyyibnawwizrdpl",cle)
```

(3) Tester les deux scripts (par exemple en envoyant à un camarade un message codé par courrier électronique).

7. Déchiffrage.

Décrypter un message (codé par substitution monoalphabétique) envoyé par un camarade en analysant la fréquence d'apparition des lettres et des digrammes. Pour déterminer les fréquences relatives des digrammes, exécuter avec Octave le script suivant.

```
m="analyserlafrequencedesdigrammesdecettephrase"
#Ce programme affiche les N digrammes les plus frequents
#du texte contenu dans m.
N=30
#
function dig=g(m)
  for n=1:26
    for k=1:26
      dig((n-1)*26+k)=length(findstr(m,char([n+96,k+96])));
    end
  end
end
#
[s,i]=sort(g(m));
for n=1:N
  printf("Digramme: ")
  printf(char([floor((i(677-n)-1)/26)+97,mod(i(677-n)-1,26)+97]))
  printf("      ")
  Nombre=s(677-n)
end
```

8. Substitutions monoalphabétiques.

Le texte contenu dans la variable M a été crypté avec un algorithme de substitution monoalphabétique. Déchiffrer le texte en procédant à une analyse des fréquences d'apparition des lettres et des digrammes. (Enlever tous les espaces avant d'exécuter le script avec *Octave* pour enregistrer le texte à décrypter dans la variable M.)

```
M="gommtlilocqioegclmzreisotmtlnotghhgszhghgbzsojhgqicgqzhhzsolodzzlqzmszoahgbzmgkgclzlzezht
zzmrgeocjthbzlghhtnozhghgbzsolodzszytclhzrihzrimtltjzlqzhhzsojhgqichzrihzczvgltjszhgrrgezth";
```

Indication: Utiliser le script suivant pour décoder le message. Les lettres qui n'ont pas encore été décryptées apparaissent en majuscules.

```
#alp=abcdefghijklmnopqrstuvwxyz
cle="**************************";
#
function m=g(M,cle)
  l=length(M);
  for n=1:l
    j=findstr(cle,M(n));
    k=length(j);
    if (k==0)
      m(n)=char(toascii(M(n))-32);
     else
       m(n)=char(j+96);
    end
  end
end
#
g(M,cle)
```

9. Substitutions polyalphabétiques.

(1) Compléter et exécuter avec Octave le script donné ci-dessous de manière à crypter le message contenu dans la variable m avec le procédé de substitution polyalphabétique périodique inventé par le moine allemand Johannes Trithemius en 1518.

```
clear
#
m="messageclairacrypteravecleprocededetrithemiusvigenere";
l=length(m);
#
for n=1:?
  d=mod(?,?)+?;
  M(n)=char(mod(toascii(m(n))-?+d,?)+?);
end
#
M
```

(2) Rédiger un deuxième script permettant de décrypter les messages cryptés avec le premier script.

(3) Tester les scripts en envoyant un message codé par courrier électronique à un camarade.

10. Substitutions polyalphabétiques.

(1) Compléter et exécuter avec Octave le script donné ci-dessous de manière à crypter le message contenu dans la variable m avec le procédé de substitution polyalphabétique périodique avec clé inventé par Giovanni Batista Belaso en 1553.

```
clear
#
c="clepastreslongue";
lc=length(c);
#
m="messageclairacrypteravecleprocededegiovannibatistabelaso";
l=length(m);
#
for n=1:?
  d=toascii(c(mod(?-1,lc)+1))-?;
  M(n)=char(mod(toascii(m(n))-?+d,?)+?);
end
#
M
```

(2) Rédiger un deuxième script permettant de décrypter les messages cryptés avec le premier script.

(3) Tester les scripts (par exemple en envoyant un message codé par courrier électronique à un camarade).

11. Substitutions polyalphabétiques.

(1) Compléter et exécuter avec Octave le script donné ci-dessous de manière à crypter le message contenu dans la variable m avec le procédé de substitution polyalphabétique autoclave inventé par Blaise de Vigenère (1523-1596).

```
clear
#
c="clepastreslongue";
lc=length(c);
#
m="messageclairacrypteravecleprocededeautoclavedevigenere";
l=length(m);
#
for n=1:l
  if (n<=lc)
```

```
11        d=toascii(c(n))-?;
12        else
13          d=toascii(m(n-?))-?;
14      end
15      M(n)=char(mod(toascii(m(n))-?+d,?)+?);
16    end
17    #
18    M
```

(2) Rédiger un deuxième script permettant de décrypter les messages cryptés avec le premier script.
(3) Tester les scripts en envoyant un message codé par courrier électronique à un camarade.

12. Cryptage de Blaise de Vigenère.

Le texte contenu dans la variable M a été crypté avec l'algorithme de substitution polyalphabétique autoclave inventé par Blaise de Vigenère (1523-1596) avec la clé c="honoredebalzac";. (Enlever tous les espaces avant d'exécuter le script avec *Octave* pour enregistrer le texte à décrypter dans la variable M.)

```
1  M="xinbuzryteysrgpxaavpbyeevlfrchjrmfnpaikgazhrlijlrgccurqbshkpaeiyspgwvtqbiapelhwoqxlrgwage
     bipwxxehtiaihufsetvjvvjiybyyppxcvgokhtnlxpfesgyfxoeciwgpxxqhhppgrrcrtsqxtogkyzvvhnnoixtvhcfj
     rvkirevrveshwwnmpyailttpifijaqlsytzarbsmwfbtbiigoarmryjioxiorkiknihwgypfumhqliirrlriwqkycsrr
     ognzpilsenzzoezknzrwkxsertocogrmehbghlnhhzcupxstosxtfkhqcsfbiyohcxflouivifhfmgwqqxvgtmskyltr
     zuhvjievzqartieammcodpvjxqmrluzdhmmclvgnqsxgvamuyvyfkbxtzvtjxphvrqiztgoituiirkvpcdefyvxgvzgl
     vvweefxemljxaiyvcawlkuvbaomiovtnutguebbzgwlvmrcltdgacswilvtjucghtgrbnpuueahzuxgobvhygddlrcfl
     zlxvqriodspyg";
```

Compléter et exécuter avec Octave le script donné ci-dessous de manière à décrypter le texte.

```
1   c="?";
2   lc=length(?);
3   #
4   l=length(?);
5   #
6   for n=1:?
7     if (n<=?)
8       d=?;
9       else
10        d=?;
11    end
12    m(n)=?;
13  end
14  #
15  m
```

13. Crible d'Ératosthène.

Compléter le script suivant de manière à obtenir un programme qui cherche les nombres premiers inférieurs à n à l'aide du crible d'Ératosthène.

```
1   n=10000;
2   tableau=[1:1:?];
3   tableau(1)=0;
4   for l=1:?
5     if (tableau(l)>0)
6       k=?;
7       while (?<=?)
8         tableau(?)=0;
9         k=k+1;
10      end
11    end
```

```
12 end
13 tableau(n-100:n)
```

14. Nombres premiers.

Compléter le script suivant de manière à obtenir un programme qui détermine si le nombre p donné par l'utilisateur est premier. L'algorithme consiste à tester, pour chaque nombre k compris entre 2 et p−1, si p est divisible par k.

```
1  p=input("Tapez un nombre entier: ");
2  k=?;
3  while (mod(?,?)>0) & (?<?)
4    k=k+1;
5  end
6  if (?==?)
7    printf("Ce nombre est premier.\n")
8    else
9      printf("Ce nombre n'est pas premier.\n")
10 end
```

Peut-on simplement améliorer l'algorithme de manière à accélérer le programme ?

15. Puissances et modulo: algorithme naïf.

(1) Calculer 23^{55} mod 23.

(2) Exécuter la commande mod(23^55,23) avec *Octave*. Le résultat est-il le même qu'au point 1 ci-dessus ?

(3) Compléter le script suivant de manière à obtenir une fonction qui calcule k^e mod n exactement.

```
1  clear
2  function x=pm(k,e,n)
3     s=?;
4     km=mod(?,?);
5     for l=1:?
6        s=mod(?*?,?);
7     end
8     x=s;
9  end
10 #
11 pm(55,1234,55)
```

(4) Vérifier le bon fonctionnement du programme en calculant 23^{55} mod $23 = 0$.

16. Puissances: algorithme naïf.

(1) Calculer 12^{123456} mod 3, $12^{1234567}$ mod 3, $12^{12345678}$ mod 3 et $12^{123456789}$ mod 3 de tête.

(2) Calculer 12^{123456} mod 3, $12^{1234567}$ mod 3, $12^{12345678}$ mod 3 et $12^{123456789}$ mod 3 avec l'algorithme naïf en exécutant les commandes mp(12,123456,3), mp(12,1234567,3), etc.

Pourquoi le temps de calcul est-il long ?

(3) Est-il possible de calculer $12^{(10^{300})}$ mod 3 avec l'algorithme naïf ?