Recherche de motif : Algorithme de Rabin-Karp

Boutault Corentin, Bouquer Ywan,
Dal Germain, Lefeuvre Damien, Fradet Nathan

Sommaire

Problème de recherche de motifs

Problème de recherche de motifs

Recherche de motifs (simple)

- Recherche de mots : ctrl+f
  - Article en ligne
  - Nombre d'occurrences
- Chercher et remplacer :
  - Éditeur de texte

Rabin_Karp(*"#"*) => Suppression des commentaires

```
class Personne:
    # Objet qui représente un homme
    def __init__(self, age, taille):
        self.age = age
        self.taille = taille

    # Renvoie l'a taille de l'individu'âge de l'individu
    def get_age(self):
        return self.age

    # Renvoie la taille de l'individu
    def get_taille(self):
        return self.taille
```

Denise était venue à pied de la gare Saint-Lazare, où un train de Cherbourg l'avait débarquée avec ses deux frères, après une nuit passée sur la dure banquette d'un wagon de troisième classe1. Elle tenait par la main Pépé, et Jean la suivait, tous les trois brisés du voyage, effarés et perdus, au milieu du vaste Paris, le nez levé sur les maisons, demandant à chaque carrefour la rue de la Michodière, dans laquelle leur oncle Baudu demeurait. Mais, comme elle débouchait enfin sur la place Gaillon, la jeune fille s'arrêta net de surprise.

— Oh ! dit-elle, regarde un peu, Jean !

Et ils restèrent plantés, serrés les uns contre les autres, tout en noir, achevant les vieux vêtements du deuil de leur père. Elle, chétive pour ses vingt ans, l'air pauvre, portait un léger paquet ; tandis que, de l'autre côté, le petit frère, âgé de cinq ans, se pendait à son bras, et que, derrière son épaule, le grand frère, dont les seize ans superbes florissaient, était debout, les mains ballantes.

— Ah bien ! reprit-elle après un silence, en voilà un magasin !

C'était, à l'encoignure de la rue de la Michodière et de la rue Neuve-Saint-Augustin2, un magasin de nouveautés3 dont les étalages éclataient en notes vives, dans la douce et pâle journée d'octobre. Huit heures sonnaient à Saint-Roch4, il n'y avait sur les trottoirs que le Paris matinal, les employés filant à leurs bureaux et les ménagères courant les boutiques. Devant la porte, deux commis, montés sur une échelle double, finissaient de pendre des lainages, tandis que, dans une vitrine de la rue Neuve-Saint-Augustin, un autre commis, agenouillé et le dos tourné, plissait délicatement une pièce de soie bleue. Le magasin, vide encore de clientes, et où le personnel arrivait à peine, bourdonnait à l'intérieur comme une ruche qui s'éveille.

— Fichtre ! dit Jean. Ça enfonce Valognes5... Le tien n'était pas si beau.

Denise hocha la tête. Elle avait passé deux ans là-bas, chez Cornaille, le premier marchand de nouveautés de la ville ; et ce magasin, rencontré brusquement, cette maison énorme pour elle, lui gonflait le cœur, la retenait, émue, intéressée, oublieuse du reste. Dans le pan coupé donnant sur la place Gaillon, la haute porte, toute en glace, montait jusqu'à l'entresol, au milieu d'une complication d'ornements, chargés de dorures. Deux figures allégoriques, deux femmes riantes, la gorge nue et renversée, déroulaient l'enseigne : Au Bonheur des Dames. Puis, les vitrines s'enfonçaient, longeaient la rue de la Michodière et la rue Neuve-Saint-Augustin, où elles occupaient, outre la maison d'angle, quatre autres maisons, deux à gauche, deux à droite, achetées et aménagées récemment. C'était un développement qui lui semblait sans fin, dans la fuite de la perspective, avec les étalages du rez-de-chaussée et les glaces sans tain de l'entresol, derrière lesquelles on voyait toute la vie intérieure des comptoirs. En haut, une demoiselle, habillée de soie, taillait un crayon, pendant que, près d'elle, deux autres dépliaient des manteaux de velours.

— Au Bonheur des Dames, lut Jean avec son rire tendre de bel adolescent, qui avait eu déjà une histoire de femme à Valognes. Hein ? c'est gentil, c'est ça qui doit faire courir le monde !

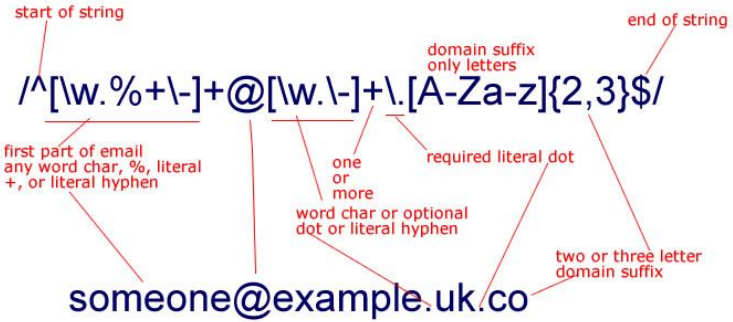
Mais Denise demeurait absorbée, devant l'étalage de la porte centrale. Il y avait là, au plein air de la rue, sur le trottoir même, un éboulement de marchandises à bon marché, la tentation de la porte, les occasions qui arrêtaient les clientes au passage. Cela partait de haut, des pièces de lainage et de draperie, mérinos6, cheviottes7, molletons8, tombaient de l'entresol, flottantes comme des drapeaux, et dont les tons neutres, gris ardoise, bleu marine, vert olive, étaient coupés par les pancartes blanches des étiquettes. À côté, encadrant le seuil, pendaient également des lanières de fourrure, des bandes étroites pour garnitures de robe, la cendre fine des dos de petit-gris9, la neige pure des ventres de cygne, les poils de lapin de la fausse hermine et de la fausse martre. Puis, en bas, dans des casiers, débordaient des articles de bonneterie vendus pour rien, gants et fichus de laine tricotés, capelines, gilets, tout un étalage d'hiver, aux couleurs bariolées, chinées, rayées, avec

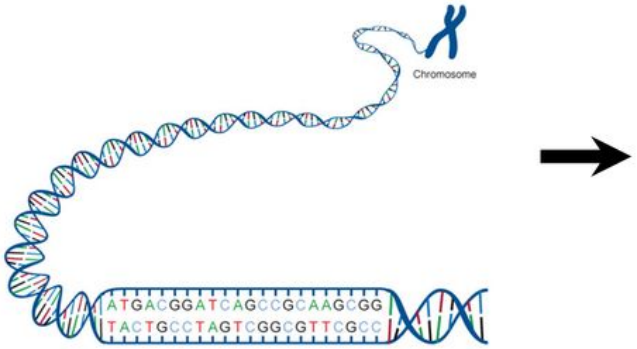
au | Tout rechercher | Respecter la casse

Problème de recherche de motifs

Recherche de motifs (multiple) : plusieurs mots pouvant correspondre

- Recherche de motif dans l'ADN:
  - séquences d'ADN cause de maladies rares
- Chercher et remplacer :
  - Regular expression (regex)



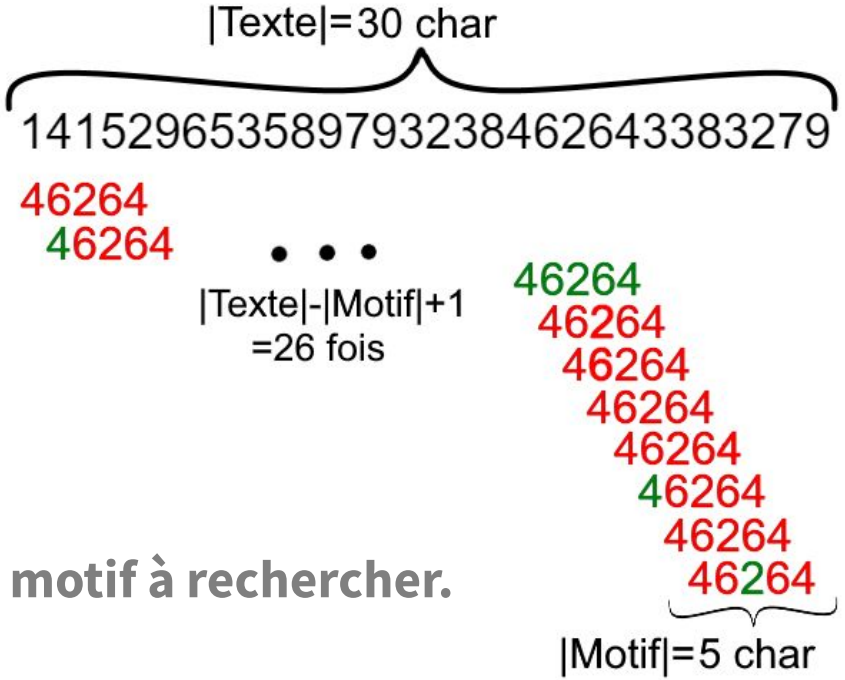


AATTGTGAGCGGATAACAATTGACATTGTGAGCGGATAACAAGATACTGAGCACATACTAGAGAAA
GAGGAGAAATACTAGATGCAAATGACGGAAGCTACCTGTTGACGCTGCTGTAAAACCGGAGCAAA
GGGGATTGTATTTGCCGGTTCTGGGAACGGGTCTTTATCTGATGCAGCCGAAAAAGGGGCGGACA
GCGCAGTCAAAAAAGGCGTTACAGTGGTGCGCTCTACCCGCACGGGAAATGGTGTCGTCACACCAAAC
CAAGACTATGCGGAAAAGGACTTGCTGGCATCGAACTCTTTAAACCCCCAAAAAGCACGGATGTTGCT
GATCGTTGCTCTTACCAAAACAAATGATCCTCAAAAAATCCAAGCTTATTTCAATGAGTATTGAAGAAAA
GAAGGCGAATAAGCCTTCTTTTTTTTGGCTTTTTAGGACCAATAATGACCTCTGAATCTTAAAATTTCTT
TAAAAATAAGCCAAAATTACCCTTTACTTAATTAATTTGGTAACGTAATATAATTGGAGAATTTGTTA
CAAAAAAAGGAGGATATTATGAAATTTGTAAAAAGAAGGATCATTGCACTTGTAACAATTTTGATGCT
GTCTGTTACATCGCTGTTTGCGTTGCAGCCGTCAGCAAAAGCCGCTGAACACAATCCAGTCGTTATG
GTTCACGGTATTGGAGGGGCATCATTCAATTTTGCGGGAATTAAGAGCTATCTCGTATCTCAGGGCTG
GTCGCGGGACAAGCTGTATGCAGTTGATTTTTGGGACAAGACAGGCACAAATTATAACAATGGACCG
GTATTATCACGATTTGTGCAAAAGGTTTTAGATGAAACGGGTGCGAAAAAAGTGGATATTGTCGCTCACAG
CATGGGGGGCGCGAACACACTTTACTACATAAAAAATCTGGACGGCGGAAATAAAGTTGCAAACGTCGT
GACGGTTGGCGGCGCGAACCGTTTGACGACAGGCAAGGCGCTTCCGGGAACAGATCCAAATCAAAA
GATTTTATACACATCCATTTACAGCAGTGCCGATATGATTGTCATGAATTACTTATCAAGATTAGATGGTGCT
AGAAACGTTCAAATCCATGGCGTTGGACACATCGGCCTTCTGTACAGCAGCCAAGTCAACAGCCTGATTAAAG
AAGGGCTGAACGGCGGGGGCCAGAATACGAATTAATGAAAAACAAAACCTTGAAGAATGCTATTCTTC
AAGGTTATTCTGCTTTCAGCACAATGGTTTTCGCAGCCATATCATGAACGGTTTGTTTTTTCTTCGTAA
ATGCGGCAGTCAAATAGATCAGGCGGGAGAACACATGCACCCACGCTATCAGGTAACGGACAATGGC
TTGCGGGAAGGATATTTTTTTATATGTTTCGTCCCTCACGATTTGCAGCCCGATGCTTTTTTTGCCCAGTGTGCCCT
TCCAATTTGTCAGCGGCATCAGCAAAAAATACACAATCAGCATCAATATGGCGACAATAATGACACCGG
CGGACCCATCGCCAAACGTAAATCCGGCTGCCAAAATCACTGCTGCGCAATGATTACATCAAGTTAATAAT
ACTAGAGCCAGGCATCAAATAAAACGAAAGGCTCAGTCGAAAGACTGGGCCTTTCGTTTTATCTG
TTGTTTGTCGGTGAACGCTCTCTACTAGAGTCACACTGGCTCACCTTCGGGTGGGCCTTTCTGCGTTTATA

Principe de la méthode naïve

Principe de la méthode naïve

Méthode de force brute.



Répétition du procédé pour chaque motif à rechercher.

Implémentation naïve en C

Implémentation naïve en C:

Structure utilisée :

```
typedef struct _tableau{
        int taille;
        int* contenu;
}tableau;
```

Implémentation naïve en C:

```
tableau motifBrut(char* x,char* t) // Méthode naïve
{
    tableau sortie;

    if(strlen(x) > strlen(t))   ← Cas trivial
        {
        sortie.contenu = NULL;
        sortie.taille = 0;
            return sortie;
        }

        //Initialisation des variables
        int* positions = malloc(strlen(t) * sizeof(int));
        int j = 0;
        int p = 0;

            for(int i = 0; i < strlen(t) - strlen(x) + 1; i++)
            {
                j = 0;

On compte le nb de
caractère exacts →  while(j < strlen(x) && x[j] == t[i + j])
                {
                    j++;
                }

                if(j == strlen(x))
                {
                    positions[p] = i;
                    p++;
                }
            }

                ↓ "Raccourcissement" du tableau initial
    positions = realloc(positions, p * sizeof(int));

    sortie.taille = p;
    sortie.contenu = positions;
    return sortie;
}
```

Principe de l’algorithme de Rabin-Karp

Principe de l'algorithme de Rabin-Karp

Objectif : Rendre la recherche plus facile. Traitement cas de recheche de motifs multiple

Principe :

1. Associer un code à chaque sous-chaîne du tableau. Calcule de hashs (empreinte)
2. Calcule du hash associer à chaque motifs
3. Parcours du tableau
4. Comparaison des Hash (1 comparaison par sous-chaîne du tableau par motifs)

Principe de l'algorithme de Rabin-Karp

Fonction de hachage utilisé dans notre cas : Méthode de calcul décimal (Hörner)

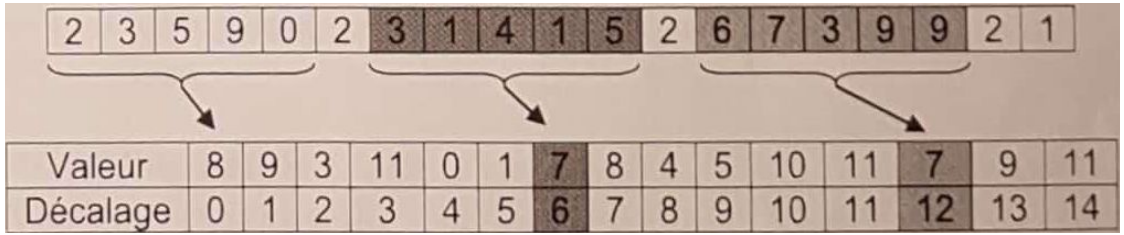
- $x = 10^{m-1} * X[1] + 10^{m-2} * X[2] + \ldots + X[m]$
- $t_{s+1} = 10 * (t_s - 10^{m-1} * X[s + 1]) + X[s + m + 1]$ → relation de récurrence
- Travail modulo q

Attention : potentiels faux positifs

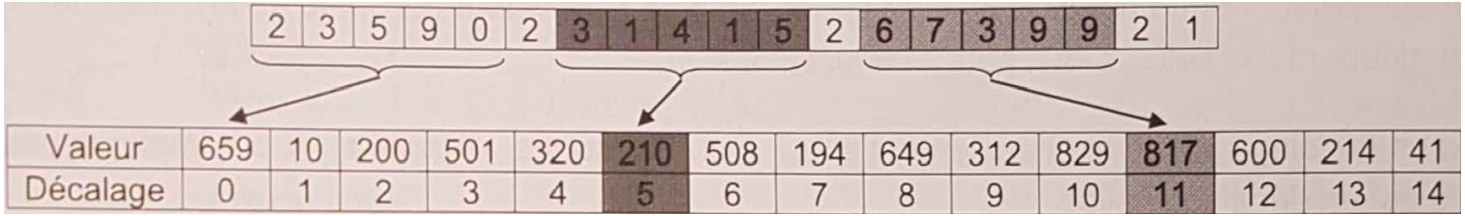
Principe de l'algorithme de Rabin-Karp

Recherche du motif : 31415

q = 13 :



q = 997 :



Implémentation de Rabin-Karp en C

Implémentation de Rabin-Karp en C

Prototypes pour le prétraitement (étape 1-2) :

```
int powerMod(int a,int b,int mod);

int calculeHashInitial(int taille_motif,int tab[],int modulo);
int calculeHashSuivant(int taille_motif,int tab[],int modulo,int pos,int hash);

int *pretraitementDeci(int taille_texte, int tab[],int modulo,int taille_motif);
```

Implémentation de Rabin-Karp en C

powerMod :

```c
int powerMod(int a, int b, int mod){/*
    - Entrée : a : Nombre de calculs
               b : Puissance à calculer
               mod : Modulo du calcul
    - Sortie: Exponentiation rapide adaptée au modulo */
    int resultat;
    if ( b <= 0){
        return 1;
    }
    else if (b%2 == 0){
        resultat=powerMod(a, b/2, mod);
        return (resultat * resultat) % mod;
    }
    else{
        resultat=powerMod(a, (b-1)/2, mod);
        return (resultat * resultat * a) % mod;
        // Pour que le resultat soit entre 0 et mod
    }
}
```

Implémentation de Rabin-Karp en C

calculeHashInitial :

```c
int calculeHashInitial(int taille_motif, int tab[], int modulo)
{
    int resultat;
    if (taille_motif<=0)
    {
        return 0;
    }
    else
    {
        resultat=calculeHashInitial(taille_motif-1,tab,modulo);
        resultat=(tab[taille_motif-1] + 10*resultat)%modulo;
        return (resultat + modulo) % modulo;
    }
}
// Complexité : O(n) (n la taille du motif)
```

Implémentation de Rabin-Karp en C

calculeHashSuivant :

```c
int calculeHashSuivant(int taille_motif,int tab[],int modulo,int pos,int hash)
{
    // Calcul le Hash (version decimal) suivant avec la formule donnée
    int resultat;
    resultat=(hash - powerMod(10, taille_motif-1, modulo) * tab[pos-1]) % modulo;
    resultat=(10*resultat+tab[taille_motif+pos-1]) % modulo ;
    return (resultat + modulo) % modulo;
}
// Complexité : O(log(n)) (Utilisation de powerMod)
```

Implémentation de Rabin-Karp en C

pretraitementDeci :

```c
int* pretraitementDeci(int taille_texte, int tab[],int modulo,int taille_motif){
    /*
    - Entrée : tab : Un tableau d'entier (Les hash des éléments succesif du texte)
               modulo : Moduulo du hash
    - Sortie : retourne le sous-tableau T[pos, pos + len] */

    // Precondition taille_motif<=taille_texte
    int taille_finale=taille_texte-taille_motif+1;
    int *tab_sequence=malloc(sizeof(int)*taille_finale);
    int hash=calculeHashInitial(taille_motif,tab,modulo);

    tab_sequence[0]=hash;

    for (int i=1;i<taille_finale;i++)
        // Parcours du tableau et calcule des "hash"
    {
        hash = calculeHashSuivant(taille_motif,tab,modulo,i,hash);
        tab_sequence[i]=hash;
    }

    return tab_sequence;
}
// Complexité : O(nlog(n)) (n la taille du texte)
```

Implémentation de Rabin-Karp en C

Prototypes pour le traitement (étape 3-4) :

```
int* soustab(int* T, int pos, int len);
tableau str_to_int(char* texte);

tableau traitement(int* T, int* valeurs, int* motif, int sizeT, int sizeX, int modulo);
```

Implémentation de Rabin-Karp en C

Fonctions auxiliaires :

```
void * realloc( void * pointer, size_t memorySize );
```

Réalloue le bloc mémoire en argument

```
int memcmp( const void * pointer1, const void * pointer2, size_t size );
```

Compare les deux blocs mémoires sur chaque octet

Implémentation de Rabin-Karp en C

soustab :

```c
int* soustab(int* T, int pos, int len)
{
        int* soustableau = malloc(len * sizeof(int));

        for (int i = 0; i < len; i++)
        {
                soustableau[i] = T[pos + i];
        }

        return soustableau;
}
```

Exemple :
tab = {2, 3, 5, 9, 0, 1}
soustab(tab, 2, 3) = {5, 9, 0}

Implémentation de Rabin-Karp en C

Fonction traitement :

```c
tableau posttraitement(int* T, int* valeurs, int* motif, int sizeT, int sizeX, int modulo)
{
        // Déclarations
        int cpt = 0;
        int* positions = NULL; // tableau des positions du motif
        int x = calculeHashInitial(sizeX, motif, modulo); // hachage du motif
        tableau sortie;

        positions = malloc((sizeT - sizeX) * sizeof(int));
        assert(positions != NULL);

        for (int i = 0; i < sizeT; i++)
        {
                if (valeurs[i] == x)
                {
                        if (memcmp(soustab(T, i, sizeX), motif, sizeX) == 0)
                        {
                                positions[cpt] = i;
                                cpt++;
                        }
                }
        }

        positions = realloc(positions, (cpt * sizeof(int)));

        sortie.taille = cpt;
        sortie.contenu = positions;

        return sortie;
}
```

Boucle sur tout le texte →

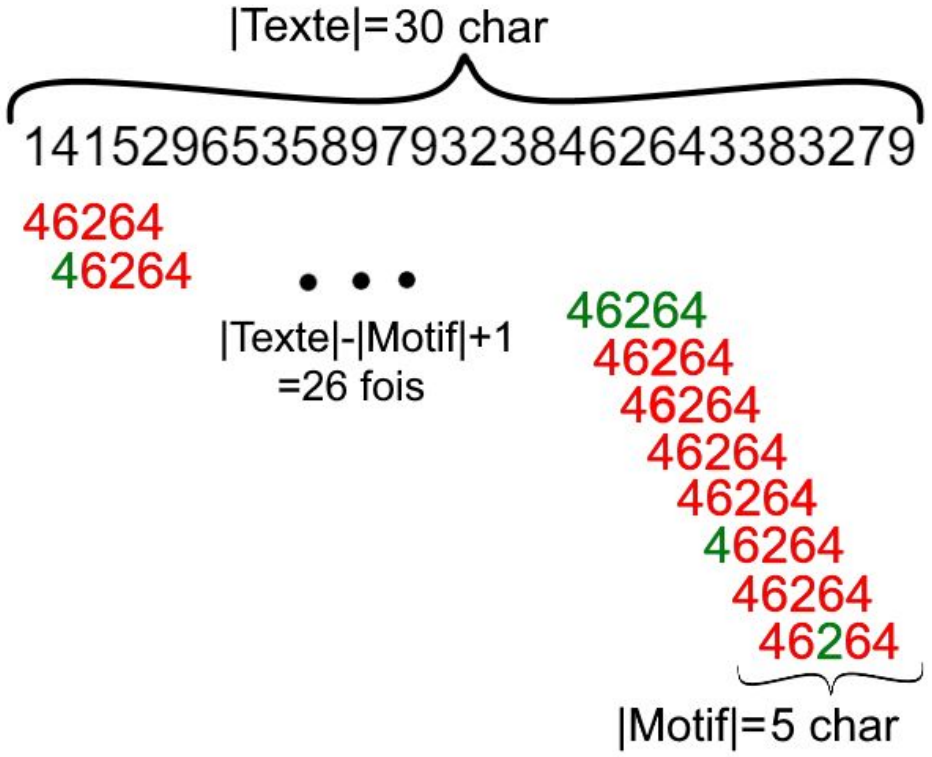
← Test sur le hash

← Retour de la structure tableau

Complexité

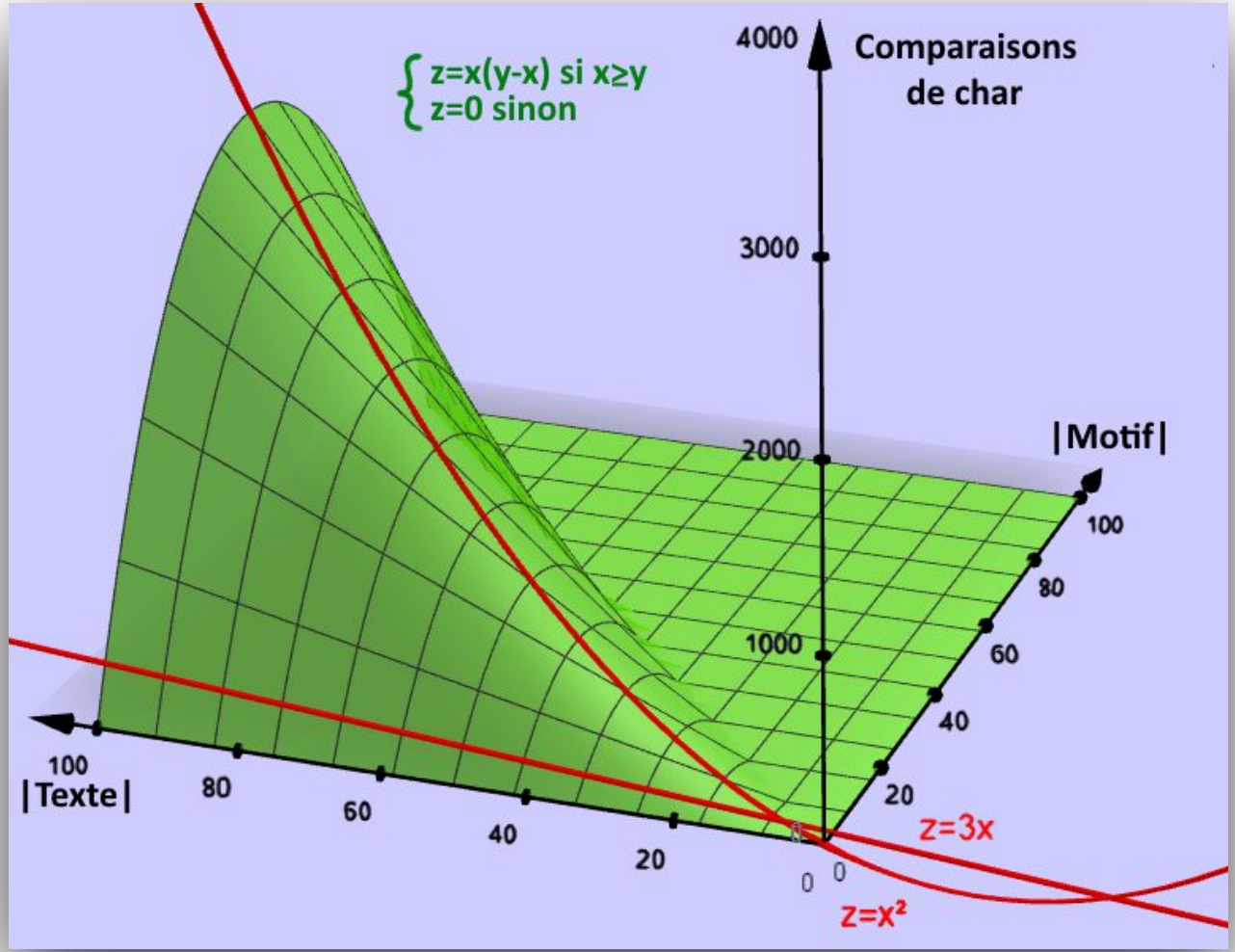
Complexité : Algorithme naïf

Motif : “46264”
Texte : “14152965358979323846264338327 9”



(n-m+1)m Comparaisons

⇒ Complexité en O(m(n-m+1))=O(nm)


z=x(y-x) si x≥y
z=0 sinon
4000
Comparaisons
de char
3000
2000
1000
|Motif|
100
80
60
40
20
0
|Texte|
100
80
60
40
20
0
z=3x
z=x²

Complexité : Algorithme de Rabin-Karp

- Fonction de prétraitement :
  - utilisation de l'exponentiation rapide (calcul des hashs)
  - parcours du tableau (calcul de chaque hash)

  → Complexité en $O((n-m+1)\ln(m)) < O(n.\ln(m))$

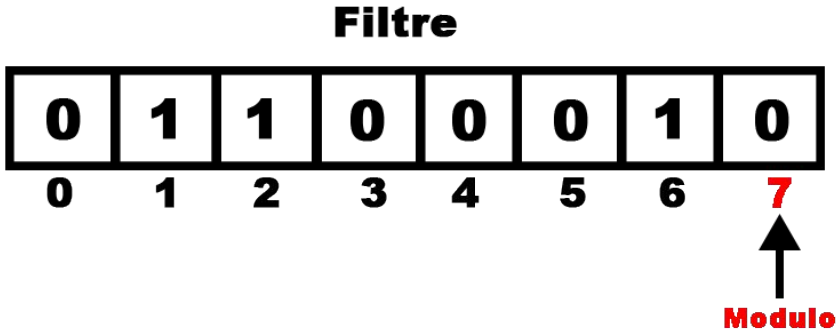
- Fonction de traitement :
  - n-m+1 hash à comparer, au plus m(n-m+1) char à comparer

  → Complexité entre O(n) et O(nm)

⇒ Complexité globale entre $O(n.\ln(m))$ et O(nm)

Traitement de plusieurs motifs

Traitement de plusieurs motifs : Filtre de Bloom

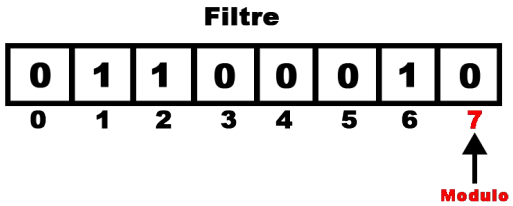


$$H = [\ x,\ 2x + 3,\ \ldots\ ]\ \%\ \text{Modulo}$$

Inconvénient : les motifs doivent être tous de la même taille

Traitement de plusieurs motifs : Implémentation

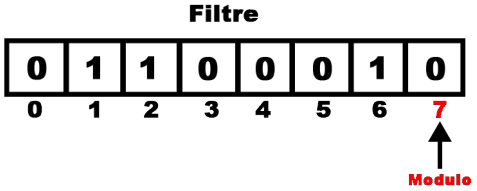
```c
// Symbolise un tableau de fonctions de hachage composées de length fonctions
typedef struct _HashFunctions{

    int (**hash_f)(int x, int mod);
    int mod;

    int length;

}HashFunctions;

// Implémentation d'un filtre de Bloom
typedef struct _BloomFilter{

    int* filter;
    int length;

    HashFunctions hash;

}BloomFilter;
```



$H = [\ x,\ 2x + 3,\ \ldots\ ]$ % Modulo

Traitement de plusieurs motifs : Exemple

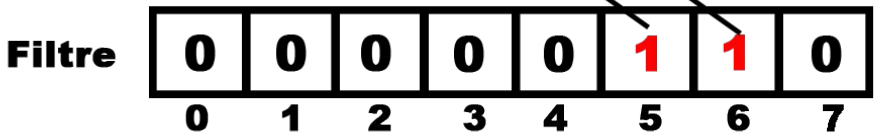
- Opérations :
  - Implémentation



$H = [\ x,\ 2x + 3,\ \ldots\ ]$ % Modulo

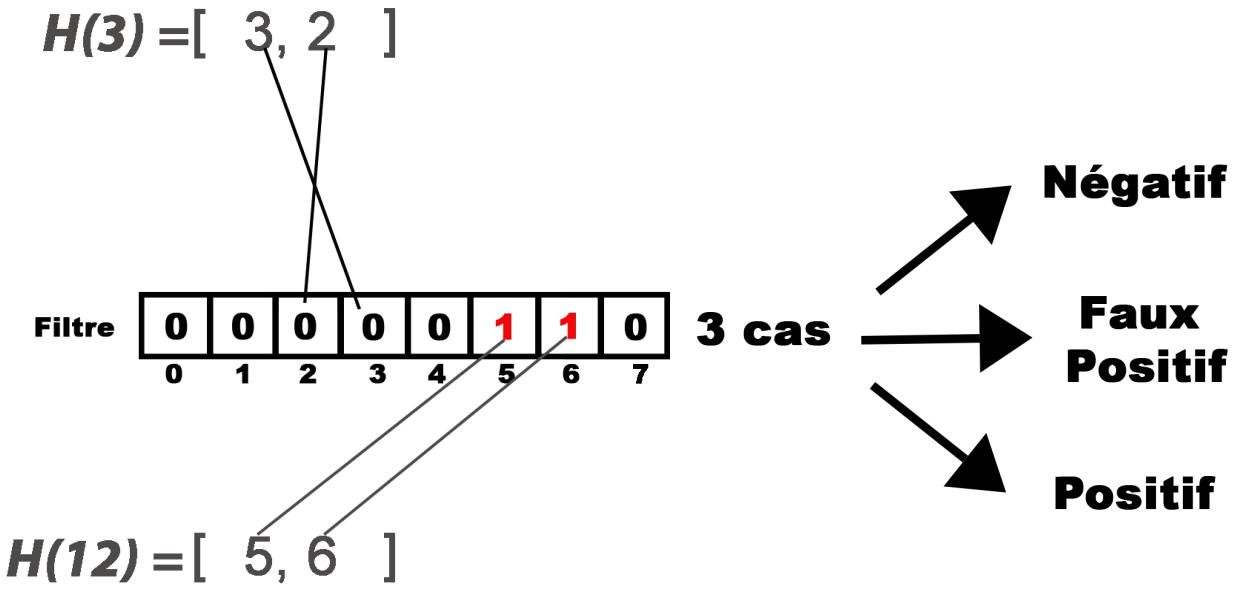
$H(5) = [\ 5,\ 13\ ]$ % 7

$= [\ 5, 6]$



Traitement de plusieurs motifs : Exemple

- Opérations :
  - Scan



Traitement de plusieurs motifs : Implémentation

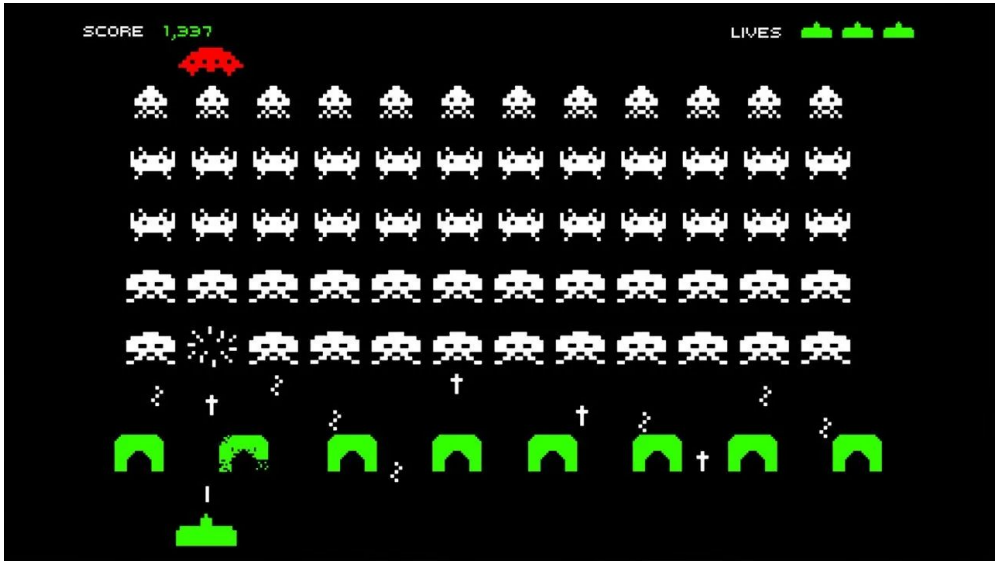
```
tableau* multiple_posttraitement(int* T, int* valeurs, int** motif, int nb_motif, int sizeT, int sizeX, int modulo){/*
    - Entrée : T: tableau du texte
                       valeurs: tableau des valeurs hachées
                       motif: tableaux de motifs (même taille)
                       sizeT, sizeX : tailles de T, de motif
                nb_motif : nombres de motif traités
                       modulo: modulo de la fonction de hachage
    - Sortie: Retourne le tableau des positions du motif dans la structure de tableau */

    // DEPART

    int j;
    bool finded = false;
        for (int i = 0; i < sizeT; i++) // parcours de T pour compter les decalages compatibles
        {
                if (BF_scan(filter, valeurs[i])) // si le decalage est compatible
                {
            j = 0;
            finded = false;
            while (j < nb_motif && (! finded))
            {
                if (memcmp(soustab(T, i, sizeX), motif[j], sizeX) == 0){ // memcmp compare les deux blocs memoires
                                    positions[j][*(cpt + j)] = i;
                                    cpt[j]++;
                    finded = true;
                            }
                j++;
            }
                }
        }

        return sortie;
}
```

Tableau à deux dimensions

Tableau à deux dimensions

Recherche de motifs (pattern):

- possibilité de faire son propre pattern en faisant une fonction de hachage spécifique
- recherche en fonction de la couleur
- recherche multiple (plusieurs couleur)



Améliorations