Projet d'algorithmique du texte
Algorithme Aho-Corasick

M1 GIL
Guettouche Islam
Mohand Benaouicha

Sommaire

Description du projet

L'algorithme d'Aho-Corasick est un algorithme de recherche de chaîne de caractères (ou motif) dans un texte.
Il consiste à avancer dans une structure de données abstraite appelée dictionnaire qui contient le ou les mots recherchés en lisant les lettres du texte T une par une. La structure de données est implantée de manière efficace, ce qui garantit que chaque lettre du texte n'est lue qu'une seule fois. Généralement le dictionnaire est implanté à l'aide d'un trie ou arbre digital auquel on rajoute des liens suffixes. Une fois le dictionnaire implanté, l'algorithme a une complexité linéaire en la taille du texte T et des chaînes recherchées.
L'algorithme extrait toutes les occurrences des motifs. Il est donc possible que le nombre d'occurrences soit quadratique, comme pour un dictionnaire a, aa, aaa, aaaa et un texte aaaa . Le motif a apparaît à quatre reprises, le motif aa à trois reprises, etc.

Le but de ce projet est d'implémenté Aho-Corasick en fonction de trois types de représentation du Trie :

- Une matrice de transition.
- Une liste d'adjacence
- Un mix entre matrice et liste.

DESCRIPTION des structures de données :

Matrice de transitions :

```
struct _trie {
int maxNode; /* Nombre maximal de noeuds du trie */
int nextNode; /* Indice du prochain noeud disponible */
int **transition; /* matrice de transition */
int *finite; /* etats terminaux */
};


struct _ACMatrice{
        Trie trie;
        int* suppleant;
};
```

Listes d'adjacence:

```
/* structure des transitions*/
struct _transitions{
        int debut;                      /* etat de début de la transition */
        unsigned char lettre;           /* etiquette de la transition */
        int fin;                /* etat de fin de la transition*/
        struct _transitions *suivant;       /* transition suivante */
};


/* structure de la liste chainée */
struct _list{
        int targetNode;                 /* cible de la transition*/
        unsigned char letter;           /* etiquette de la transition  */
        struct _list *next;     /* maillon suivant */
};


/* structure du trie */
struct _trie{
        int maxNode;                    /* Nombre maximal de noeuds du trie  */
        int nextNode;                   /* Indice du prochain noeud disponible */
        List *transition;       /* liste d'adjacence */
        char *finite;           /*  etats terminaux  */
        int * suppleance;       /* tableau de suppléances */
};
```

Notre structure contient les éléments :
_ L'état de départ (cible de la transition).
_ Le caractère responsable de la transition (étiquette de la transition).
_ La liste des transitions suivant la courante (maillon suivant).

Donc cette structure est représentée par une liste d'adjacence dont on pourra connaître la liste des transitions suivant la transition courante.

Mixte:

```
/* structure des transitions */
struct _transitions{
        int debut;                      /* etat de début de la transition  */
        unsigned char lettre;           /* etiquette de la transition */
        int fin;                /* etat de fin de la transition */
        struct _transitions *suivant;           /* transition suivante */
};


/* structure d'une liste chainée */
struct _list{
        int targetNode;                 /* cible de la transition*/
        unsigned char letter;           /* etiquette de la transition  */
        struct _list *next;     /* maillon suivant*/
};


/* structure du mixte */
struct _trie{
        int maxNode;                    /*  Nombre maximal de noeuds du trie  */
        int nextNode;                   /* Indice du prochain noeud disponible */
        int *transitionRoot; /* transitions de la racine  */
        List *transitionOthers; /* transitions des autres noeuds */
        char *finite;           /* etats terminaux */
        int * suppleance;     /* tableau des suppleance */
};
```

Il s'agit de représenter l'arbre avec une table de transition pour la racine et des listes d'adjacences pour les nœuds.

Donc le premier état (q0) sera représenté par un tableau de transitions et les autres états seront stockés dans une liste d’adjacence.

Jeux d’essai :

On recherche cet ensemble de mots :

X={Cbbabc,abb,bcbbc,cbbac,cccb,abba,bba,aacb,aba,ccabac}

Dans le texte suivant :

abacbbccbbbbaaccababaababacccbbbabbbbacbaccccabaaaccbbaacbabaaaacbaaacbbbbcaab
abbcbcccbaccccabbcbbbbacacbccababbcbcccccaaccbaaaccaaacbacabcccabcabcaacbabbabbcc
bccbaabbacbaaaaaacacacbabbabbbabcbbbbabaaacbabcabacbcbaaacbabaaccbabaabccbaaabc
bbcacbacbcccbcbbcababccaaabcbcbabbcaaaacabaaccccccbbcaacbbbbbbbaccccabcaaccaabaca
cbcbcaaababccacaaabcbbccabbabcbbcccbbbabbcccabaaaaacacbacbbbccaacaacaaacccaaccaab
baaaacaccbcabaacbcaaccacaacabbcacabaaabccbabaabbbaaaaabcbccbacbcccaccbabaccaabcbc
bcbabcaabbaabacbaa

```
islam@islam-W54-55SU1-SUW:~/Documents/G_I-Algo/GuettoucheIslam_MohandBenaouicha$ make
gcc -o document_teste/ac-liste executable/main_liste.c traitement/liste.c traitement/file.c -w
gcc -o document_teste/ac-matrice executable/main_matrice.c traitement/matrice.c traitement/file.c traitement/file_matrice.c -w
gcc -o document_teste/ac-mixte executable/main_mixte.c traitement/mixte.c traitement/file.c -w
gcc generateur/genTexte.c -o document_teste/genere-texte
gcc generateur/genMot.c -o document_teste/genere-mots
islam@islam-W54-55SU1-SUW:~/Documents/G_I-Algo/GuettoucheIslam_MohandBenaouicha$ cd document_teste/
islam@islam-W54-55SU1-SUW:~/Documents/G_I-Algo/GuettoucheIslam_MohandBenaouicha/document_teste$ ./ac-matrice mots texte
AC-MATRICE - Nombre d'occurences : 80
Le temps d'exécution est: 0.178833 s
islam@islam-W54-55SU1-SUW:~/Documents/G_I-Algo/GuettoucheIslam_MohandBenaouicha/document_teste$ ./ac-liste mots texte
AC-LISTE - Nombre d'occurences : 80
Le temps d'exécution est: 0.002959 s
islam@islam-W54-55SU1-SUW:~/Documents/G_I-Algo/GuettoucheIslam_MohandBenaouicha/document_teste$ ./ac-mixte mots texte
AC-MIXTE - Nombre d'occurences : 80
Le temps d'exécution est: 0.001030 s
islam@islam-W54-55SU1-SUW:~/Documents/G_I-Algo/GuettoucheIslam_MohandBenaouicha/document_teste$
```

genTexte.c

```
//Guettouche Islam
#include<stdlib.h>
#include<stdio.h>
#include<string.h>

/*Commence par récupérer La longueur du texte et La taille de l'alphabet après elle remplis la char alph[]  par une
 série ordonnées de lettre A,B,C……. après on commence la création du texte a partir de alph[] et en choisissant aléatoirement
 les lettre a laide dune fonction rand()*/

int main(int argc,char * argv[])
{
        FILE* F_Texte=NULL;
        int i,long_texte,j,taille_alphabet,G;
        char liste_de_lettre[255],A;
           long_texte=atoi(argv[1]);
           taille_alphabet=atoi(argv[2]);

           printf("La longueur du texte est : %d \n",  long_texte);
                printf("La taille de l'alphabet est: %d \n", taille_alphabet);
                F_Texte=fopen("Texte.txt","w+");

        // Pour que notre alphabet commence avec "A"
        j=65;
        for(i=0;i<taille_alphabet;i++)
        {
                liste_de_lettre[i]=(char)(i+j);
        }

        for(i=0;i<long_texte;i++)
        {
                G=rand()%taille_alphabet;
                A=liste_de_lettre[G];
                printf("%c",A);
                fprintf(F_Texte, "%c" , A);

        }
      printf("\n");

        fclose(F_Texte);



return 0;
}
```

genMots.c

```
//Guettouche Islam
#include<stdlib.h>
#include<stdio.h>
#include<string.h>

/*Commence par récupérer Le nombre de mots, la longueur minimale des mots, la longueur maximale des mots,
 * La taille de l'alphabet après elle remplis la char liste_de_lettre[]  par une série ordonnées de lettre A,B,C……. après on commence
la création des mots a partir de liste_de_lettre[] et en choisissant aléatoirement les lettre a laide dune fonction rand()*/
int main(int argc,char * argv[])
{
        FILE* F_Mot=NULL;
        int i,nbr_de_mots,j,long_min_mots,long_max_mots,taille_alphabet,G,k,N,temp;
        char liste_de_lettre[255],A;

            // nbr de mots
          nbr_de_mots=atoi(argv[1]);
          // la longueur minimale des mots
          long_min_mots=atoi(argv[2]);
          // la longueur maximale des mots
          long_max_mots=atoi(argv[3]);
          // taille de l'alphabet.
      taille_alphabet=atoi(argv[4]);
          printf("Le nombre de mots est : %d \n",  nbr_de_mots);
              printf("la longueur minimale des mots est: %d \n", long_min_mots);
              printf("la longueur maximale des mots est: %d \n", long_max_mots);
              printf("La taille de l'alphabet est: %d \n", taille_alphabet);
              F_Mot=fopen("Mots.txt","w+");
              //Pour que notre alphabet commence avec "A"
              j=65;
              for(temp=0;temp<taille_alphabet;temp++)
              {
                      liste_de_lettre[temp]=(char)(temp+j);

              }

              for(i=0;i<nbr_de_mots;i++)
              {
                      /*G nous permet de choisir un nbr aléatoire entre la longueur minimale et
                      maximale des mots qui vont être créer a l'intérieur de la boucle qui suit */
              G=rand()%(long_max_mots+1-long_min_mots) +long_min_mots;
              for(k=0;k<G;k++)
              {
              N=rand()%taille_alphabet;
```

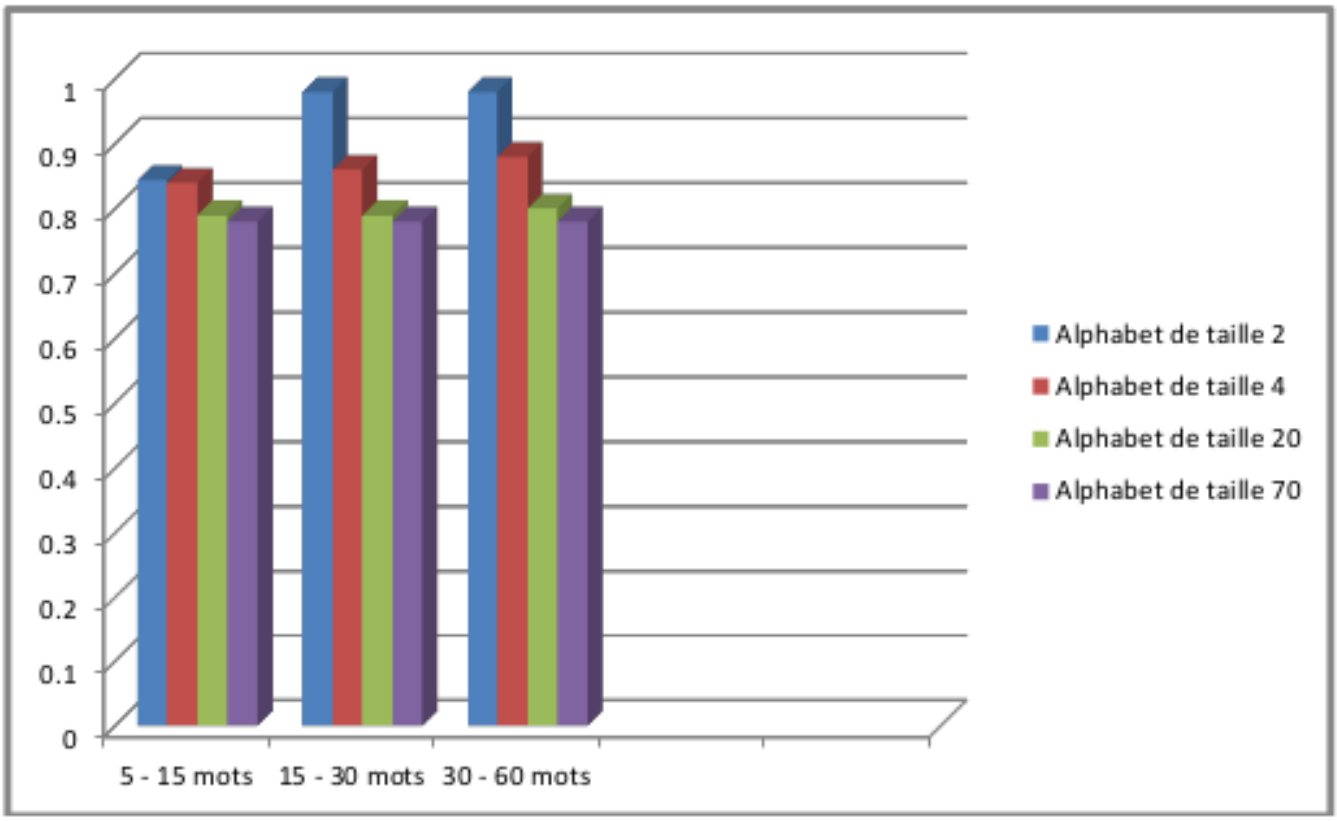
```
                printf("%c",liste_de_lettre[N]);
                A=liste_de_lettre[N];
                fprintf(F_Mot, "%c" , A);

                }
                   printf("\n");
            fprintf(F_Mot, "\n");
                }
                 printf("\n");
                fclose(F_Mot);
                return 0;
}
```

❖ Aho-Corasick Matrice

Taille du mot / Texte et mot d'alphabet	5 - 15	15 - 30	30 - 60
2	0,844	0,98	0,98
4	0,84	0,86	0,88
20	0,79	0,79	0,80
70	0,78	0,78	0,78

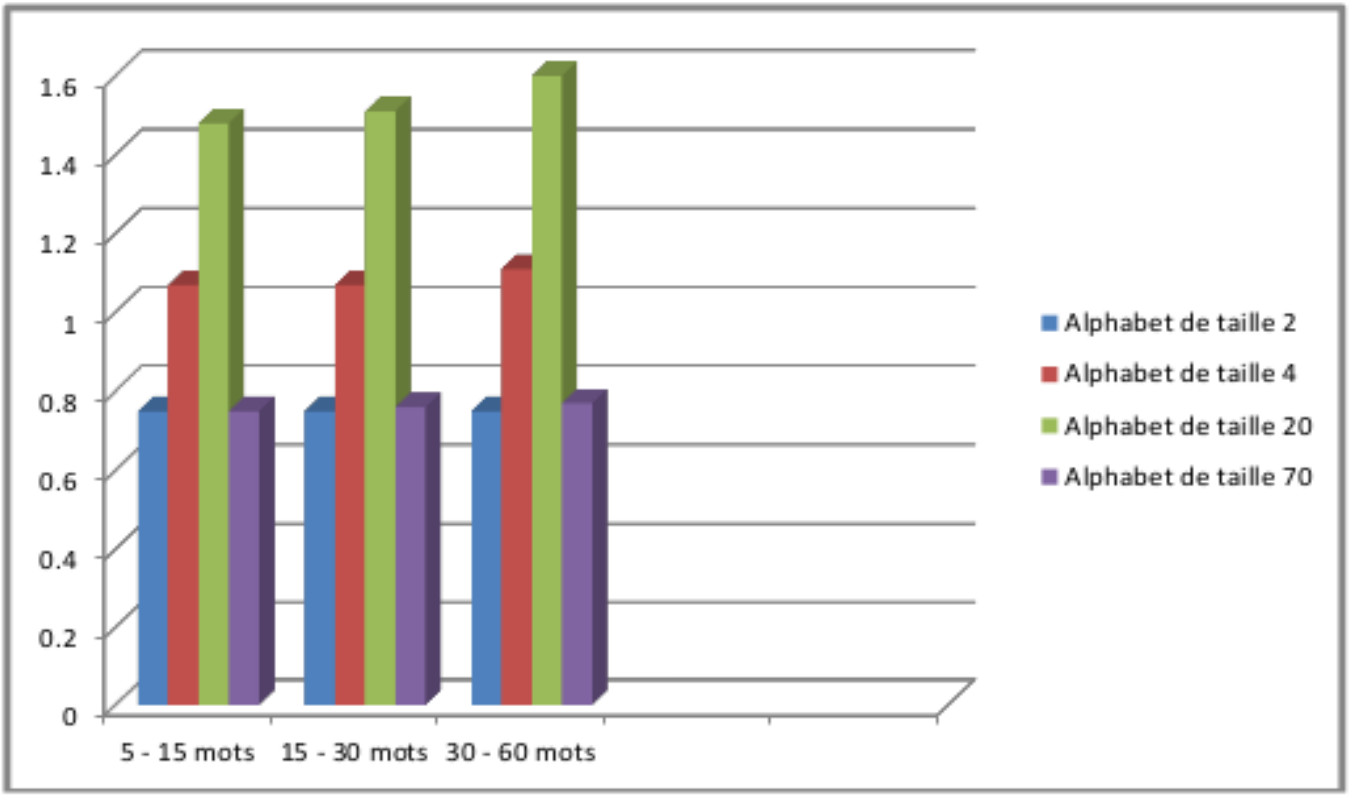
Graphe de comparaison Aho-Corasick Matrice



❖ Aho-Corasick Liste d'adjacence

Taille du mot / Texte et mot d'alphabet	5 - 15	15 - 30	30 - 60
2	0,75	0,75	0,75
4	1,07	1,07	1,11
20	1,48	1,51	1,60
70	0,75	0,76	0,77

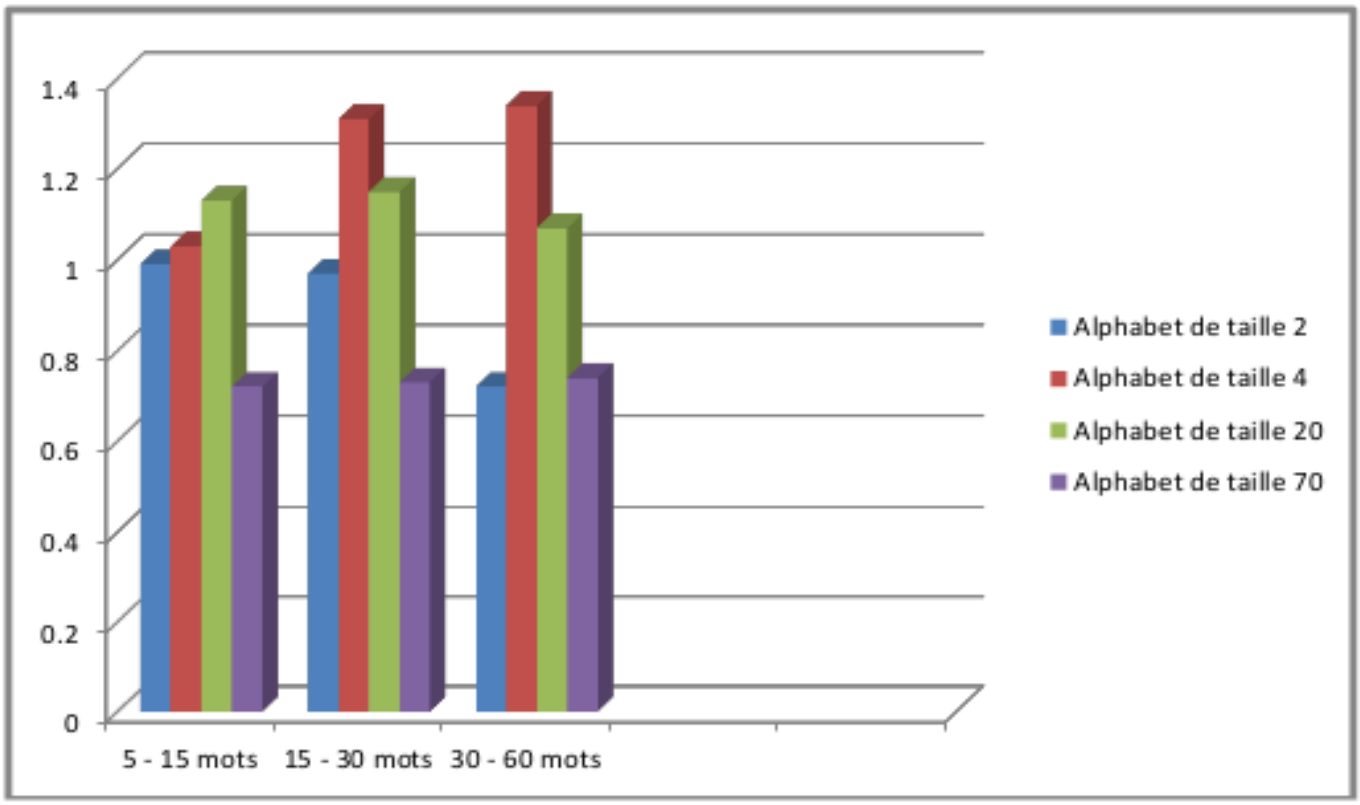
Graphe de comparaison Aho-Corasick Liste d'adjacence



❖ Aho-Corasick Mixt

Taille du mot / Texte et mot d'alphabet	5 - 15	15 - 30	30 - 60
2	0,99	0,97	0,97
4	1,03	1,31	1,34
20	1,13	1,15	1,07
70	0,72	0,73	0,74

Graphe de comparaison Aho-Corasick Matrice



Conclusion :

- Comparaison entre Aho-CorasickMatrice, Aho-CorasickListe et Aho-CorasickMixt

Graphe de comparaison Aho-Corasick Matrice, Aho-Corasick Liste d'adjacence et Aho-Corasick Mixt,