

Rapport de TP : Alignement optimal et détection de plagiat

Annie LIM, Quentin GARRIDO

6 janvier 2020

Table des matières

1	Introduction	2
2	Exercice 1	2
3	Exercice 2	2
4	Exercice 3	2
5	Exercice 4	2
6	Annexe : Code source	2

1 Introduction

2 Exercice 1

Pour calculer le score d'un alignement optimal entre x et y , nous pouvons utiliser l'algorithme de distance de Levenshtein, appelé distance d'édition (edit distance). Nous voulons observer les différences entre deux textes. Cela revient à calculer leur score d'alignement, le coût des opérations nécessaires (deletion, insertion, substitution) pour obtenir le même texte. Plus ce score est faible et plus les textes sont similaires, et donc sujet au plagiat. Le score optimal correspond au minimum entre les trois valeurs données par les opérations deletion, insertion et substitution.

Algo...

Cet algorithme est bien de complexité $O(|n||x||m|)$.

3 Exercice 2

A partir de la matrice T telle que $T[i][j]$ est le score d'un alignement optimal entre x_i et y_j , nous pourrions retrouver les opérations nécessaires à la solution optimale pour aligner les deux textes, afin de construire les textes 1 et 2 modifiés alignés.

Le backtracking consiste à suivre le chemin minimum de la matrice T de $T[n][m]$ jusqu'à $T[0][0]$.

Algo...

4 Exercice 3

5 Exercice 4

6 Annexe : Code source

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include <sys/stat.h>
4 #include <string.h>
5
6 struct alignement
7 {
8     char * x;
9     char * y;
10 };
11
12
13 /* ===== */
14 char * readtextfile(char * filename)
15 /* Retourne le contenu du fichier texte filename */
16 /* ===== */
17 {
18     struct stat monstat;
19     int N;
20     char * text = NULL;
21     FILE *fd = NULL;
22
23     N = stat(filename, &monstat);
24     if (N == -1)
25     {
26         fprintf(stderr, "error : bad file %s\n", filename);
27         exit(0);
28     }
```

```

29  N = monstat.st_size;
30  text = (char *) malloc(N+1);
31  if (text == NULL)
32  {   fprintf(stderr, "readtextfile() : malloc failed for text\n");
33      exit(0);
34  }
35  fd = fopen(filename, "r");
36  if (!fd)
37  {
38      fprintf(stderr, "readtextfile: can't open file %s\n", filename);
39      exit(0);
40  }
41
42  fread(text, sizeof(char), N, fd);
43  if ((N>0) && (text[N-1] == '\n')) text[N-1] = '\0';
44  else text[N-1] = '\0';
45  fclose(fd);
46  return text;
47 }
48
49 /* ===== */
50 int Imax(int a, int b)
51 /* Retourne le maximum de a et b */
52 /* ===== */
53 {
54     if (a < b) return b;
55     else return a;
56 }
57
58 /* ===== */
59 int Imin2(int a, int b)
60 /* Retourne le minimum de a et b */
61 /* ===== */
62 {
63     if (a < b) return a;
64     else return b;
65 }
66
67 /* ===== */
68 int Imin3(int a, int b, int c)
69 /* Retourne le minimum de a, b et c */
70 /* ===== */
71 {
72     return Imin2(Imin2(a,b),c);
73 }
74
75 /* ===== */
76 void retourne(char *c)
77 /* Retourner la chaine de caractere c */
78 /* ===== */
79 {
80     char tmp;
81     int m, j, i;
82     m = strlen(c);
83     j = m/2;
84     for (i = 0; i < j; i++){
85         tmp = c[i];
86         c[i] = c[m-i-1];
87         c[m-i-1] = tmp;
88     }
89 }
90 /* ===== */
91 void afficheSeparateurHorizontal(int nbcar)
92 /* ===== */
93 {
94     int i;

```

```

95     printf("|-");
96     for(i=0; i < nbcar; i++)
97         printf("-");
98     printf("|-");
99     for(i=0; i < nbcar; i++)
100         printf("-");
101     printf("-|\n");
102 }
103
104
105 /* ===== */
106 void affiche(char* textel, char* texte2, int nbcar)
107 /* Affiche simultanément textel et texte 2 en positionnant nbcar
108 caractères sur chaque ligne. */
109 /* ===== */
110 {
111     int i, l1, l2, l;
112
113     char *t1,*t2;
114
115     char out[512];
116
117     l1 = strlen(textel);
118     l2 = strlen(texte2);
119
120     t1 = (char*) malloc(sizeof(char) * (nbcar + 1));
121     t2 = (char*) malloc(sizeof(char) * (nbcar + 1));
122
123     l = lmax(l1, l2);
124     afficheSeparateurHorizontal(nbcar);
125     for(i = 0; i < l; i+= nbcar){
126         if (i < l1) {
127             strncpy(t1, &(textel[i]), nbcar);
128             t1[nbcar] = '\0';
129         } else t1[0] = '\0';
130         if (i < l2) {
131             strncpy(t2, &(texte2[i]), nbcar);
132             t2[nbcar] = '\0';
133         } else t2[0] = '\0';
134
135         sprintf(out, "| %c-%ds | %c-%ds |\n", '%', nbcar, '%', nbcar);
136         printf(out, t1, t2);
137     }
138     afficheSeparateurHorizontal(nbcar);
139     free(t1);
140     free(t2);
141 }
142
143
144
145 /* ===== */
146 void affiche2(char* textel, char* texte2, int nbcar)
147 /* idem affiche, mais avec un formatage différent */
148 /* ===== */
149 {
150
151     int i, l1, l2, l;
152
153     char *t1,*t2;
154
155     char out[512];
156
157     l1 = strlen(textel);
158     l2 = strlen(texte2);
159
160     t1 = (char*) malloc(sizeof(char) * (nbcar + 1));

```

```

161  t2 = (char*)malloc(sizeof(char) * (nbcар + 1));
162
163  l = Imax(l1, l2);
164
165  for(i = 0; i < l; i+= nbcар){
166      if (i < l1) {
167          strncpy(t1, &(textel[i]), nbcар);
168          t1[nbcар] = '\0';
169      } else t1[0] = '\0';
170      if (i < l2) {
171          strncpy(t2, &(texte2[i]), nbcар);
172          t2[nbcар] = '\0';
173      } else t2[0] = '\0';
174
175      sprintf(out, "x: %c-%ds \ny: %c-%ds\n", '%', nbcар, '%', nbcар);
176      printf(out, t1, t2);
177
178  }
179  free(t1);
180  free(t2);
181 }
182
183 int sub(char a, char b){
184     if(a == b)
185         return 0;
186     return 1;
187 }
188
189 int** compute_distance(char* textel, char* texte2){
190     int n = strlen(textel);
191     int m = strlen(texte2);
192
193     int** T= (int**) malloc((m+1)*sizeof(int*));
194     for(int i=0; i<=m; i++)
195         T[i]= (int*) malloc((n+1)*sizeof(int));
196
197     //T[m+1][n+1]
198     T[0][0] = 0;
199     for(int i=1; i<=n; i++)
200         T[0][i] = T[0][i-1] + 1; //Cout del
201     for(int j=1; j<=m; j++)
202         T[j][0] = T[j-1][0] + 1; //Cout ins
203     T[1][1] = Imin3(T[0][1]+1, T[1][0]+1, T[0][0]);
204     for(int i=1; i<=n; i++)
205         for(int j=1; j<=m; j++){
206             if(i==1 && j==1)
207                 continue;
208             T[j][i] = Imin3(T[j-1][i]+1, \
209                             T[j][i-1]+1, \
210                             T[j-1][i-1]+sub(textel[i-2], texte2[i-2]));
211         }
212
213     return T;
214 }
215
216 /* ===== */
217 int main(int argc, char **argv)
218 /* ===== */
219 {
220     char *x, *y;
221
222     if(argc != 3){
223         printf("usage: %s text1 text2\n", argv[0]);
224         exit(0);
225     }
226

```

```
227  x = readtextfile(argv[1]);
228  y = readtextfile(argv[2]);
229
230  //affiche(x, y, 50);
231
232
233  char* text1 = "chiens";
234  char* texte2 = "niche";
235  int** T = compute_distance("chiens", "niche");
236  printf("Cout: %d\n", T[strlen(texte2)][strlen(text1)]);
237
238  free(x);
239  free(y);
240
241 }
```