2019-2020 Bahar Yarıyılı BLM2512 Veri Yapıları ve Algoritmalar 3. Ödevi

Mesut Şafak Bilici
17011086

1 Algoritma ve Fonksiyonlar

1.1 Makrolar

Kod boyunca kullanılacak dizilerin boyutu için makrolar belirlenmiştir. Bu makrolar aşağıdaki gibidir:

- **MAX_FILE_NAME:** Text'in okunması için açılacak dosyanın isminin maksimum uzunluğunu belirler.
- **MAX_TEXT_LENGTH:** Okunan text'in maximum karakter sayısını belirler. Test sırasında arttırılması için kod esnek yazılmıştır.
- **MAX_SUBSTRING_LENGTH:** Text'İn içinde olan sub-text'in maksimum karakter sayısıdır. Bu sub-text daha sonra değiştirilecektir. Bu makro aynı zamanda subtext'in değiştirileceği string için de karakter sayısı olarak belirlenmiştir.

1.2 Fonksiyonlar ve Algoritmik Açıklamaları

Kodda yapılan algoritmik işlemlerin hepsi fonksiyonlar üzerinden yürütlmüştür. Main fonksiyonu bir sonraki bölümde açıklanmak üzere kullanılan fonksiyonlar ve açıklamaları aşağıdaki gibidir:

- **int** abs_sub(char,char): İki char'ın ASCII değerini alıp farkının mutlak değerini döndüren fonksiyon. Bu fonksiyon Boyer-Moore Algoritması'nın non-case sensitivite seçeneğinde kullanılıp o kısımda daha detaylı açıklanacaktır.

- **int** shiftTable(char*,int*,int,int): Boyer-Moore algoritmasında ne kadar 'skip' yapacağımız için kullanılmaktadır. Bir offset/başlangıç ASCII değeri kullanılarak yapılsa da algoritmamızda iki farklı case seçeneği bulunduğu için tüm ASCII değerlerini alacak şekilde hazırlanmıştır Shift Table. İlk parametresinde bulunması istenilen substring'i, ikinci parametresinde main içinde dinamik olarak tanımlanan Shift Table dizisini, üçüncü parametresinde istendiği zaman kullanılması içinesneklik için yazılan offset değeri, dördüncü parametresinde ise Shift Table'ımızın boyunu veriyoruz. Sub'string'in içinde olmayan her karakter için Shift Table'ın her cell'i -1 değerini alırken içinde olan karakterler de index değerini alıyor.

- **int** BoyerMoore(char*,char*,int*,int*): Boyer Moore algoritmasının case sensitivite halidir. Kullanıcı case sensitivite arama yapacağı zaman kullanılacak fonksiyondur. Bu fonksiyon ilk parametresinde text char array'ini, ikinci parametresinde substring char array'ini, üçüncü parametresinde öncesinde belirlenen Shift Table integer array'ini almıştır. Dördüncü parametrenin yaptığı iş fonsiyon her çalıştırıldığında ve her -1'den farklı değer döndürdüğünde 1 arttırılmaktır, yani fonksiyon her çalıştığında ve her çalıştığında yeni bir belirlenen substring'i bulduğunda 1 arttırılıyor. Bu yapılan işlem maşn fonksiyonu içinde neden sürekli fonksiyonu çağırmamız gerektiğinin açıklaması yapıldığında daha anlamlı olacaktır. While döngüsünden çıkıldığı an ve j sıfır değerinden küçük olduğu an substring'i text içinde buluğumuzu anlıyoruz, ve sonrasında bu substring'in text içindeki index konumunu döndürüyoruz.

- **int** BoyerMoore_noncase(char*,char*,int*,int*): Tüm parametreleri yukarda açıklanan case sensitivite Boyer Moore algoritması olmakla beraber yaptığı iş de kısmi olarak aynıdır. Fakat kaynak kod incelendiği zaman fark edilecektir ki bu sefer while içindeki koşullar biraz farklıdır. Burada öncesinde tanımladığımız abs_sub fonksiyonunu kullanacağız. While içindeki $j >=$ koşulu aynıdır fakat $substring[j] == buffer[i+j]$ yerine,
  $abs\_sub(substring[j], buffer[i+j]) == 0$
  ||
  $abs\_sub(substring[j], buffer[i+j]) == 32$),
  koşulları kullanılmıştır. Görüldüğü üzere yukarıda verilen mantıksal ifadenin ilk elemanı aslında case sensitivite Boyer Moore'daki
  $substring[j] == buffer[i+j]$
  ifadesi ile birebir aynı. İkinci mantıksal ifadede belirtilen şey, eğer

substring'in içinde geldiğimiz bir karakter text'in içindeki karakterin küçük harf versiyonuysa veya tam tersi büyük harf versiyonuysa bu ikisini aynı kabul et. Anlamsal olarak aynı olan fakat birisi büyük harf ve diğer küçük harf olan iki karakterin ASCII tablosundaki değerlerinin mutlad değer farkı her zaman 32'ye eşittir. Örnek: $A = 65$, $a = 97$ için

$$|A - a| == |a - A| == 32$$

- **void** replace(char*,char*,char*,int): Substring bulunduktan sonra kullanıcının istediği string ile değiştirmek için kullanılan fonksiyon. İlk parametre olarak dosyadan okunan text'i, ikinci parametre olarak bulunan substring'i, üçüncü parametre olarak substring'in yerine yazmak istediğimiz string'i, dördüncü parametre olarak dosyadan okunan text'in içindeki substring'in başlangıç konumunu - indexini alıyor. Replace işlemi için toplamda 3 farklı durum değerlendirildi:

  1. Substring'in uzunluğunun replace string'in uzunluğuna eşit olması durumu. Bu durumda iki stringin aynı index'leri sadece değiştirilecektir.
  2. Substring'in uzunuluğunun replace string'in uzunluğundan büyük olması durumu. Bu durumda, substring'in replaced string uzunluğuna kadar replaced stringi atarız. Daha sonra geride kalanları substring'in uzunluğu ve replaced string'in uzunluğu farkı kadar sola taşırız. Örnek:
     substring= "find", replaced string= "get", strlen(substring) = 4, strlen(replaced string) = 3 , strlen(substring) - strlen(replaced string) = 1:
     (a) find this
     (b) getd this → boşluk karakteri 1 sola
     (c) get this → t karakteri 1 sola
     (d) get tthis → h karakteri 1 sola
     (e) get thhis → i karakteri 1 sola
     (f) get thiis → s karakteri 1 sola
     (g) get thiss → bundan sonra null var ise, son s karakteri yerine null gelir, yoksa eğer bu şekilde model devam eder en son null için aynısı yapılır.
  3. Replace string'in uzunluğunun substring'in uzunluğundan büyük olma durumu. Bu durumda ilk atama işlemi yerine kaydırma işlemi yapmamız gerekmekte. Bunu yapmazsak eğer veri kaybı

yaşayabiliriz. Text string'inin son karakterinden, yani null, başlayarak replaced string'in uzunluğu ve substring'in uzunluğu farkı sağa kaydırma yapacağız. Bu kaydırmadan sonra artık atama işlemimizi yapabiliriz. Örnek:
substring = "pass", replaced string = "passs", strlen(substring) = 4, strlen(replaced string) = 5:
Not: Görselleştirme anlaşılsın diye harf büyüklükleri aynı seçilmek üzere pass için passs kelimesi tercih edilmiştir.

(a) pass this → s karakteri 1 sağa
(b) pass thiss → i karakteri 1 sağa
(c) pass thiis → h karakteri 1 sağa
(d) pass thhis → t karakteri 1 sağa
(e) pass tthis → boşluk karakteri 1 sağa
(f) pass this → atama işlemini başlat
(g) passs this

1.3 Main

Bu kısımda Main fonksiyonu içinde yukarıda belirtilen fonksiyonların nasıl kullanıldığı belirtilmekle beraber main fonksiyonunun içinde neler yapıldığı anlatılacaktır.
İlk olarak bazı array tanımlamaları gözükmektedir. Bu arraylar sırasıyla,

- **char** filename: Açılacak olan dosyanın ismini kullanıcıdan alırken girdiyi bu array'e alıyoruz. Statik olarak açıldı.
- **char*** buffer: Dosyanın içindeki text bilgisini bu array'in üstüne yazıyoruz. Array dinamik olarak açılmıştır. Daha sonra kullanıcı bu array içinde substring arayacaktır, ve burada substring'i değiştirecektir.
- **char*** substring: Kullanıcı buffer array'inin içindeki text'te bir kelime veya cümle bulmak isteyecektir. Bulmak istediği kelime veya cümleyi klavyeden girdi olarak verecektir. Bu girdiyi substring isimli array'de tutuyoruz. Dinamik olarak açıldı.
- **char*** replaced: Kullanıcı substring'i bulduktan sonra bu substring'i değiştirecektir. Ne ile değiştirmek istediğini klavyeden girdi olarak verecektir. Bu substring'in yerine geçecek olan string replaced isimli array'de saklanmaktadır.
- **int*** table: Shift table'ı main içinde dinamik olarak açıyoruz. Fonksiyon içinde veri girdisi yapılıyor.

Daha sonrasıda ise bazı değişken tanımlamaları gözükmektedir,

- **int** case_ctrl: Kullanıcının non-case sensitive mi yoksa case sensitivite arama yapacağının kontrolünü yapacak değişken.
- **int** found: Her yeni sub-string bulunduğunda sayısını gösterecek değşiken.
- **int** index: Her yeni sub-string bulunduğunda text içindeki index'ini gösterecek değşiken.

Bu tanımlamalardan sonra içinde text'in olduğu dosyayı açtık ve $fread$ fonksiyonu ile buffer array'ine veriyi yazdık. Daha sonra kullanıcıdan case sensitivite mi yoksa değil mi, sub-string ne, ne ile değiştirilsin bilgileri istenir. Eğer kullanıcı $case\_ctrl$ değişkenini 0 olarak seçerse $BoyerMoore\_noncase()$ fonksiyonu , değilse $BoyerMoore()$ fonksiyonu çalışır. Her iki durumda da $shiftTable()$ fonksiyonu çalıştırılır ve shift table oluşturulur. Daha sonra yukarıdaki arama yapan iki fonksiyonumuzdan teki, case durumuna göre, çağrılır ve bir index döndürür. Bu index, -1 değerinde ise herhangi bir substring bulamamıştır ve bundan sonra da bulamayacaktır. Bu sebepten dolayı aşağıdaki while döngüsüne girmeden devam eder kod. Eğer ki, -1 yerine substring'in olası bir index değerini döndürürse arama fonksiyonu, while döngüsüne girilir ve başka substring var mı araması yapılır, yenisini buldukçta while içinde bu substring replaced stringi ile değiştirilir. En son bir substring kalmadığında -1 döndürür ve kod devam eder, yeni text'i dosyaya tekrardan yazar.

1.4 Çalışma Süresi

Çalışma süresi saniye cinsinden hesaplanmıştır. Kütüphane olarak time.h kütüphanesi kullanılıp, bu zaman,

- clock_t t;
- t = clock();

⋮

- t= clock()-t;
- double time_taken = ((double)t)/CLOCKS_PER_SEC;

şeklinde hesaplanmıştır.

2 İstenilen Örnekler

2.1.1 Örnek

Find: algorithm

Replace: method

Option: non case sensitivite

Text: The Boyer-Moore Algorithm is considered the most efficient string matching algorithm.

Output:

```
(base) safak@progcu:~/Desktop/g$ cat in1.txt
The Boyer-Moore Algorithm is considered the most efficient string matching algorithm.
(base) safak@progcu:~/Desktop/g$ ./c17011086
Enter the file name that has your string: in1.txt
For case sensitivity press 1, else 0: 1
Enter the substring you want to find and replace: algorithm
Replace with: method
The Boyer-Moore Algorithm is considered the most efficient string matching method.
Found and replaced: 1
Running time is 0.000253 seconds.
(base) safak@progcu:~/Desktop/g$
```

Figure 1: Örnek 1 Non-case Sensitivite

2.1.2 Örnek

Find: algorithm

Replace: method

Option: case sensitivite

Text: The Boyer-Moore Algorithm is considered the most efficient string matching algorithm.

Output:

```
(base) safak@progcu:~/Desktop/g$ cat in1.txt
The Boyer-Moore Algorithm is considered the most efficient string matching algorithm.
(base) safak@progcu:~/Desktop/g$ ./c17011086
Enter the file name that has your string: in1.txt
For case sensitivity press 1, else 0: 0
Enter the substring you want to find and replace: algorithm
Replace with: method
The Boyer-Moore method is considered the most efficient string matching method.
Found and replaced: 2
Running time is 0.000337 seconds.
(base) safak@progcu:~/Desktop/g$
```

Figure 2: Örnek 1 Case Sensitivite

2.2 Örnek

Find: went to
Replace: visited
Option: non case sensitivite
Text: Wayne went to Wales to watch walruses.
Output:

```
(base) safak@progcu:~/Desktop/g$ cat in2.txt
Wayne went to Wales to watch walruses.
(base) safak@progcu:~/Desktop/g$ ./c17011086
Enter the file name that has your string: in2.txt
For case sensitivity press 1, else 0: 0
Enter the substring you want to find and replace: went to
Replace with: visited
Wayne visited Wales to watch walruses.
Found and replaced: 1
Running time is 0.000204 seconds.
(base) safak@progcu:~/Desktop/g$
```

Figure 3: Örnek 2 Non-case Sensitivite

3 Performans

Aşağıda göreceğiniz dataframe; yazılan programın farklı text boyutlarında, bu text üzerinde farklı uzunluklardaki ve farklı sayılardaki substring'lerin ve replaced string'lerin değiştirilmesine bağlı, saniye bilgilerini taşımaktadır. Bu veri implemente edilmesi istenen programın defalarca farklı parametreler ile çalıştırılarak elde edilmiştir.

Index	Text Length	SubString Length	Replaced String Length	Replaced Number	Second
0	112	5	7	1	0.000145
1	112	5	7	2	0.000158
2	221	5	5	2	0.000145
3	221	5	5	1	0.000154
4	124	5	6	5	0.000163
5	124	2	3	9	0.000104
6	124	3	16	9	0.000194
7	405	2	17	1	0.000224
8	405	2	1	1	0.000152

Figure 4: Saniyeye bağlı performans verisi

Aşağıda ise bu performansı daha iyi yorumlayabilmek için 3 boyutlu öklidyen uzayda görselleştirilmesi yapılmıştır.

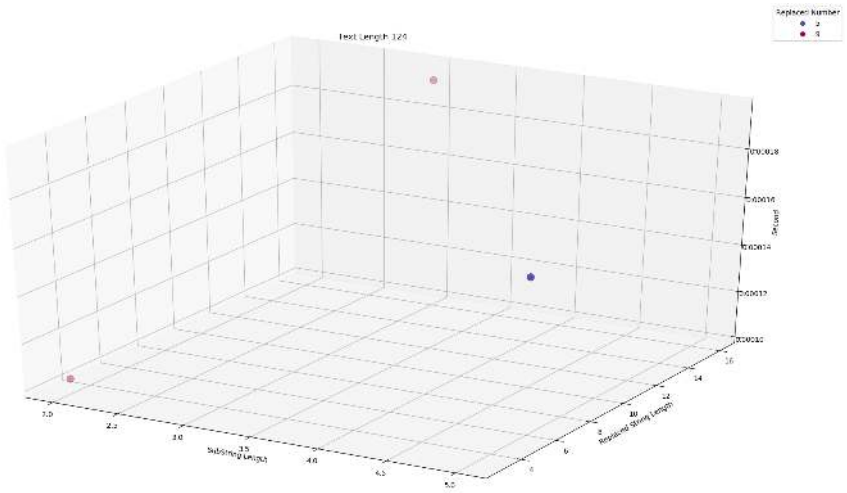

Figure 5: Text size 124

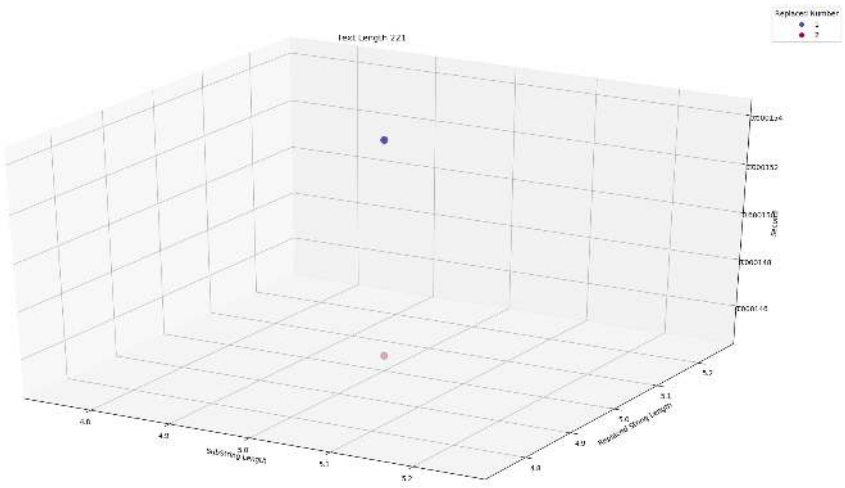

Figure 6: Text size 221

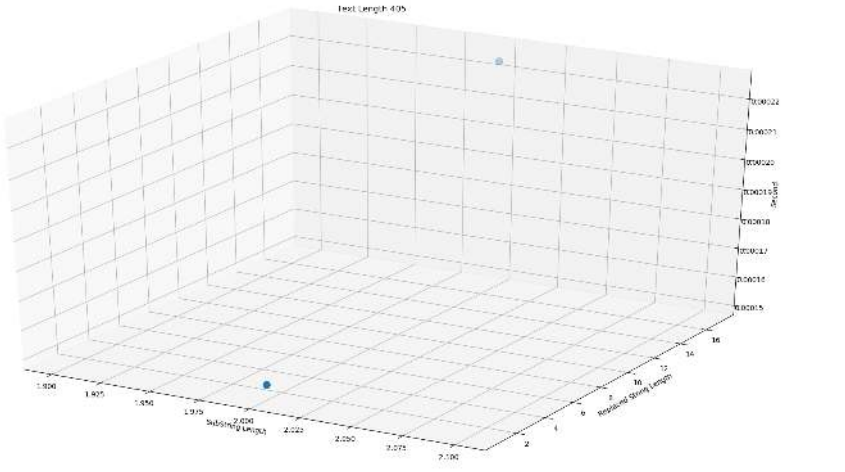

Figure 7: Text size 405

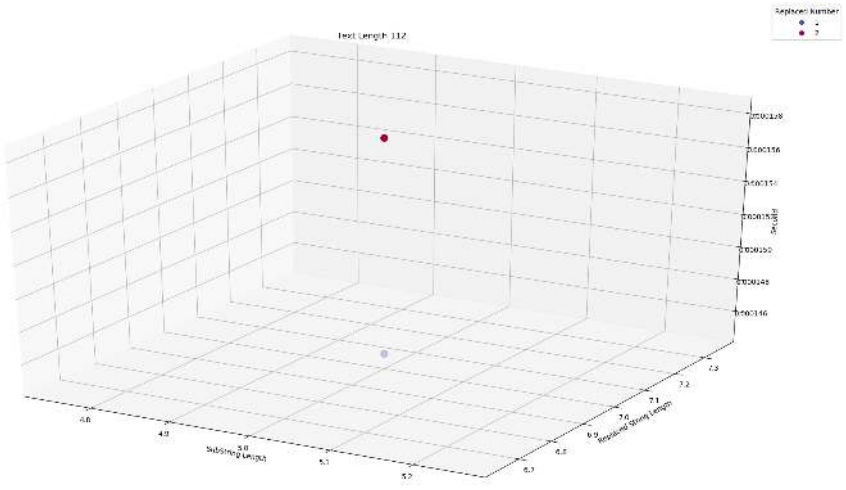

Figure 8: Text size 112

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define MAX_FILE_NAME 100
#define MAX_TEXT_LENGTH 1000
#define MAX_SUBSTRING_LENGTH 20
void shiftTable(char*,int*,int,int);
int BoyerMoore(char*,char*,int*,int*);
int BoyerMoore_noncase(char*,char*,int*,int*);
int abs_sub(char,char);
void replace(char*,char*,char*,int);
int main(int argc, char** argv) {

        char filename[MAX_FILE_NAME];
        char* buffer = (char*)malloc(MAX_TEXT_LENGTH*sizeof(char));
        char* substring = (char*)malloc(MAX_SUBSTRING_LENGTH*sizeof(char));
        char* replaced = (char*)malloc(MAX_SUBSTRING_LENGTH*sizeof(char));
        int* table = (int*)malloc(sizeof(int)*256);
        int case_ctrl;
        int found=0;
        int index;

        printf("Enter the file name that has your string: ");
        scanf("%s",filename);

        FILE* fp = fopen(filename,"r+");
        if(fp == NULL) {
                fprintf(stderr,"File have not opened!\n");
                exit(0);
        }

        fread(buffer,sizeof(char)*MAX_TEXT_LENGTH,1,fp);

        printf("For case sensitivity press 1, else 0: ");
        scanf("%d",&case_ctrl);
        printf("Enter the substring you want to find and replace: ");
        scanf(" %[^\n]s",substring);
        printf("Replace with: ");
        scanf(" %[^\n]s",replaced);

        clock_t t;
        t = clock();

        if(case_ctrl == 0) {
                char o1 = 'A';
                shiftTable(substring,table,o1,256);
                index = BoyerMoore_noncase(buffer,substring,table,&found);
                while(index != -1) {
                        replace(buffer,substring,replaced,index);
                        index = BoyerMoore_noncase(buffer,substring,table,&found);
                }
                fclose(fp);
                fp = fopen(filename,"w");
                printf("%s",buffer);
                fprintf(fp,"%s",buffer);
        fclose(fp);

        }
        else if(case_ctrl == 1) {
                char o2 = 'A';
                shiftTable(substring,table,o2,256);
                index = BoyerMoore(buffer,substring,table,&found);
                while(index != -1) {
        replace(buffer,substring,replaced,index);
                        index = BoyerMoore(buffer,substring,table,&found);
    }
                fclose(fp);
                fp = fopen(filename,"w");
                printf("%s",buffer);
          fprintf(fp,"%s",buffer);
        fclose(fp);
```

```
        }
        else{
                fprintf(stderr,"Operation cannot be recognized! \n");
                exit(0);
        }
        t= clock()-t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
        printf("Found and replaced: %d \n",found);
        printf("Running time is %f seconds.\n",time_taken);

        return 0;
}
int abs_sub(char a,char b) {
        if(a-b > 0) {
                return a-b;
        }
        else{
                return -(a-b);
        }
}

void shiftTable(char* sub, int* right, int offset, int size){
        int i;
        int M = strlen(sub);
        for(i=0; i<size; i++) {
                right[i] = -1;
        }
        for(i=0; i<M; i++) {
                right[(int)sub[i]] = i;
                if((int)sub[i]>= 65 || (int)sub[i]<=90) {
                        right[(int)sub[i] + 32] = 1;
                }
        }
}

int BoyerMoore(char* buffer, char* substring, int* table, int* found) {
        int i,j,skip=0;
        int M = strlen(substring);
        int N = strlen(buffer);
        for(i=0; i<N-M; i+=skip){
                j = M-1;
                while(substring[j] == buffer[i+j] && j>=0) {
                        j--;
                }
                if(j>=0) {
                        skip =j- table[(int)buffer[i+j]];
                        if(skip<=0) {
                                skip = 1;
                        }
                }
                else{
                        (*found)++;
                        return i;
                }
        }
        return -1;
}

int BoyerMoore_noncase(char* buffer, char* substring, int* table, int* found) {
  int i,j,skip=0;
  int M = strlen(substring);
  int N = strlen(buffer);
  for(i=0; i<N-M; i+=skip){
    j = M-1;
//      printf("skip: %d\n",skip);
    while(j>=0 && (abs_sub(substring[j], buffer[i+j]) == 0 || abs_sub(substring[j],b
uffer[i+j])==32)) {
      j--;
    }
    if(j>=0) {
      skip =j-table[(int)buffer[i+j]];
      if(skip<=0) {
        skip = 1;
```

```
      }
    }
    else{
                        (*found)++;
                        return i;
    }
  }
        return -1;
}


void replace(char* buffer,char* sub, char* with, int start) {
        int i,j,k;
        if(strlen(sub) == strlen(with)) {
                i = 0;
                for(j=start; j<start+strlen(with); j++) {
                        buffer[j] = with[i];
                        i++;
                }
        }
        else if(strlen(sub) > strlen(with)) {
                i = 0;
                for(j=start; j<start+strlen(with); j++) {
                        buffer[j] = with[i];
                        i++;
                }
                k = j;
                while(buffer[k+strlen(sub) - strlen(with)] != '\0'){
                        buffer[k] = buffer[k+strlen(sub) - strlen(with)];
                        k++;
                }
                buffer[k] = buffer[k+strlen(sub) - strlen(with)];
        }
        else if(strlen(sub) < strlen(with)) {
                for(i=strlen(buffer)+1; i>=strlen(sub)+start; i--) {
                        buffer[i+strlen(with)-strlen(sub)] = buffer[i];
                }
                i=0;
                for(j=start; j<strlen(with)+start; j++) {
                        buffer[j] = with[i];
                        i++;
                }
        }
}
```