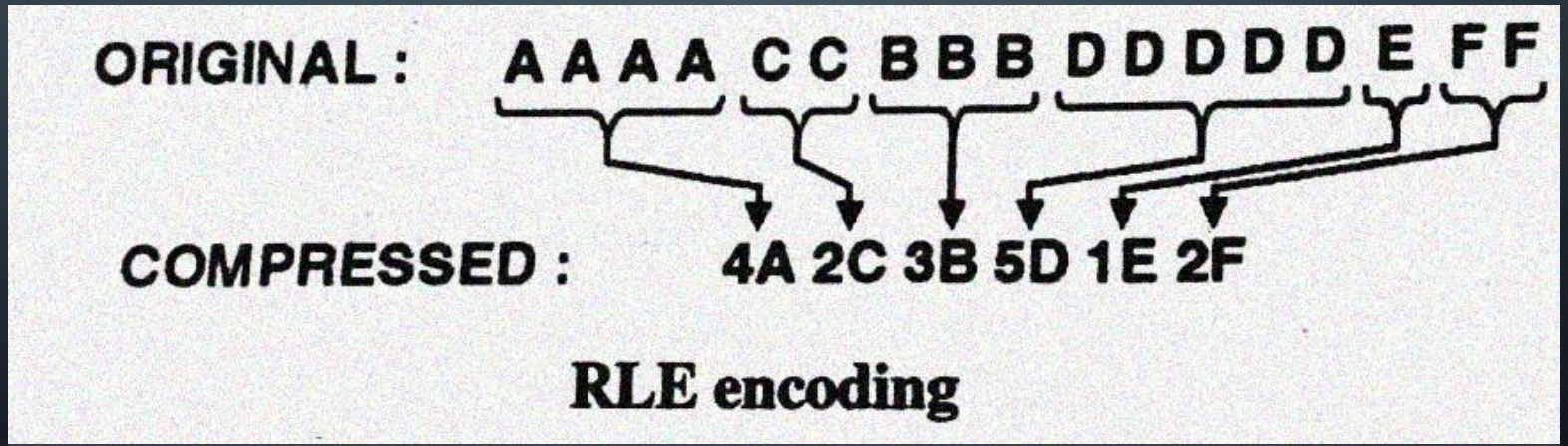
RLE ALGORİTMASI İLE VERİ SIKIŞTIRMA TEKNİĞİ



İÇİNDEKİLER

- RLE ALGORİTMASI NEDİR?
- NERELERDE KULLANILIR,AVANTAJLARI, DEZAVANTAJLARI NELERDİR?
- NASIL İŞLER?

RLE ALGORİTMASI NEDİR?

- RLE (Run-Length Encoding) algoritması oldukça sık başvurulan ve veri sıkıştırma algoritmaları arasındaki en basit veri sıkıştırma yöntemidir.
- Kayıpsız veri sıkıştırma tekniğine dayanan bu yöntemde amaç; kendini tekrar eden veri dizisinin, bu veri dizisinin bir örneği ve kaç kez tekrarlandığının yan yana yazılarak sıkıştırma yapılmasıdır.

RLE ALGORİTMASI NERELERDE KULLANILIR, AVANTAJLARI VE DEZAVANTAJLARI NELERDİR?

- RLE sıkıştırması aşağıda yazılan dosya formatlarında kullanılabilir:

  .TIFF Uzantılı dosyalar

  .PDF Uzantılı dosyalar

- Bu algoritmanın uygulanması diğer kayıpsız veri sıkıştırma algoritmalarına göre çok daha kolaydır ve çok fazla CPU gücü gerektirmemektedir. RLE sıkıştırması sadece çok fazla tekrar eden verilere sahip dosyalarda etkili olabiliyor. Bu dosyalar eğer çok fazla yer kaplıyor ise bunlar text (yazı) dosyaları olabilir fakat çok fazla siyah ve beyaz bölge içeren çizgi resimleri bu iş için çok daha yatkın. Bilgisayar tarafından oluşturulmuş renkli görüntüler (Mimari çizimler) de aynı zamanda gayet iyi sıkıştırma oranları verebilir.

RLE NASIL İŞLER?

Öncelikle bir main fonksiyonu açıp dosyanın içindekileri kaydedebileceğimiz bir char dizisi oluşturuyoruz. Dizinin büyük olmasının sebebi eğer fazla sayıda karakter girişi gelirse stack owerflow sorunu vermemesi içindir. 6. satırda ise bir FILE komutu kullanarak dosya oluşturuyoruz. Sonrasında da integer olarak çözme işleminde kullanacağımız count'u ve sıkıştırma ya da çözmeden hangisini yapacağını belirleyecek olan sikistircoz'ü tanımlarız. Devamında ise scanf yardımıyla sıkıştırma mı çözme mi yapacağımıza karar veririz.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      char arr[10000];
6      FILE* dosya;
7      int count, sikistircoz;
8      printf("Sikistirma yapmak icin 1, cozme yapmak icin 2 sayisini giriniz: ");
9      scanf_s("%d", &sikistircoz);
10
```

Eğer sıkıştırma yapmayı seçersek ilk if'e girilir ve dosya okunmak için açılır. while döngüsünün içinde kullanacağımız integer i ve dosyadaki karakterleri almak için gereken char1 char'ını tanımlarız. Daha sonrasında dosyayı, while döngüsü ile dosyanın son hanesine kadar okurken arr dizisinin içine 0'dan başlayacak şekilde kaç tane char varsa atılır ve dosya kapatılır.

```c
if (sikistircoz == 1) {
    fopen_s(&dosya, "giris2.txt", "r");
    int i = 0;
    char char1;
    while (fscanf_s(dosya, "%c", &char1) != EOF) {
        arr[i] = char1;
        i++;
    }
    fclose(dosya);

```

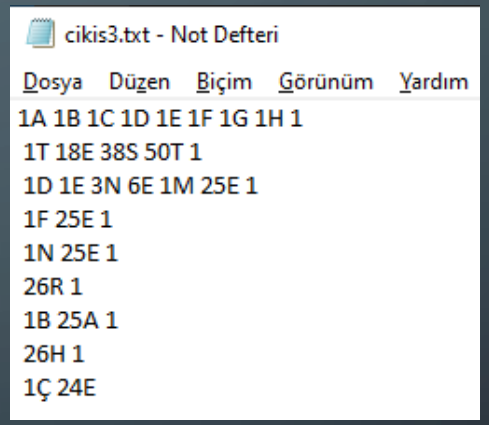
giris2.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
ABCDEFGH
TEEEEEEEEEEEEEEEEEESSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
DENNNEEEEEEMEEEEEEEEEEEEEEEEEEEEEEEEE
FEEEEEEEEEEEEEEEEEEEEEEEEE
NEEEEEEEEEEEEEEEEEEEEEEEEE
RRRRRRRRRRRRRRRRRRRRRRRRRR
BAAAAAAAAAAAAAAAAAAAAAAAAA
HHHHHHHHHHHHHHHHHHHHHHHHHH
ÇEEEEEEEEEEEEEEEEEEEEEEEE
```

- Daha sonrasında istenen dosyayı yazmak amacıyla tekrar açarız. 0'dan dosyadaki char sayısına kadar olacak şekilde for döngüsüne sokarız. Bu for döngüsünün içine her döndüğünde count'u tekrardan 1 den başlatması amacıyla değer atarız. Bunun nedeni ise aşağıdaki while döngüsünden dolayıdır. While döngüsünde ise k+1<i yapmamızın sebebi arr[k]==arr[k+1] tanımından dolayı. Örneğin burada dizimizin sınırı 4 ise ve biz k değerinde 4. döngüdeysek arr[4]==arr[5] gibi bir tanım ortaya çıkar ve arr[5]'te tanımlı dizi olmadığı için de kod hata verecektir.

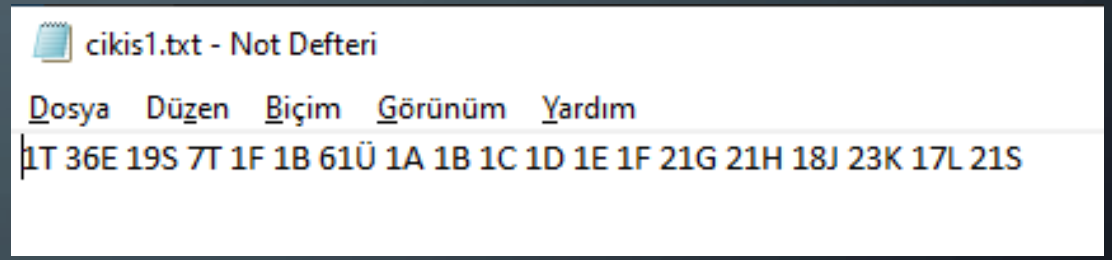
```
21          fopen_s(&dosya, "cikis2.txt", "w");
22          for (int k = 0; k < i - 1; k++) {
23              count = 1;
24              while (k + 1 < i - 1 && arr[k] == arr[k + 1]) {
25                  k++;
26                  count++;
27              }
28              if (arr[k] != " ")
29                  fprintf(dosya, "%d%c ", count, arr[k]);
30
31          }
32          fclose(dosya);
33      }
```



Bu while döngüsünde k'yı arttırmamızın sebebi ise for döngüsüne bağlı kalmadan gereklilikleri sağladıkça dizinin içindeki bir sonraki değere farklı olana dek bakmak ve count'ı her döngüde 1 arttırarak da bu aynı olan değerin kaç tane olduğunu bulmaktır. Bu döngüden çıkıldığında da fprintf komutuyla döngü içinde sıkıştırdığımız değeri dosyanın içine yazdırırız. Bu döngüler tamamlandıktan sonra ise dosya kapatılır. Her satırın sonunda ise 1 bastırılmasının nedeni \n'den kaynaklanmaktadır. Yani orada aslında bir satır atlanıldı anlamına geliyor.

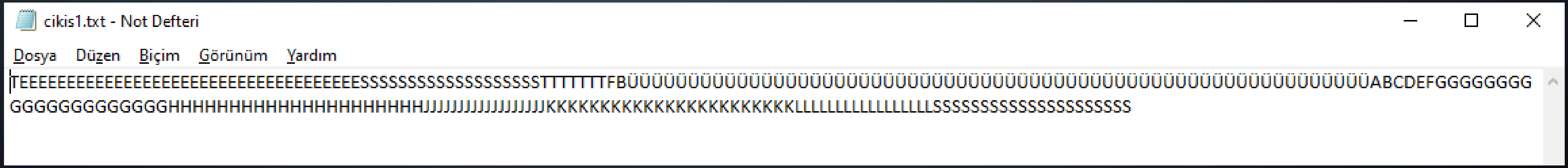
- Eğer çözme işlemi seçilirse ikinci if'in içine girilir. Burada 2 tane dizi tanımlama sebebimiz ise biri integer biri char olmak üzere; integer dizi sıkıştırılan verinin önündeki o verinin kaç tane olduğuna dair olan bilgiyi, char dizisi ise bu verinin değerini, ne olduğunu tutar.Burada num2'yi integer, a ve char2'yi char olarak tanımlıyoruz. num2 ve char2'yi dosyanın içindeki char ve integer'ları okumak amacıyla, a'yı ise sıkıştırmadaki i'nin görevini görmesi amacıyla tanımlıyoruz. İstenen dosya okunmak amacıyla açılır ve while döngüsünün içine alınarak dosyanın sonuna kadar olmak şartıyla başta belirttiğimiz iki dizinin içine kaydedilir. Bu döngü bittikten sonra dosya kapatılır.

```
34      else if (sikistircoz == 2) {
35          int arr1[10000];
36          char arr2[10000];
37          char char2;
38          int num2, a = 0;
39          fopen_s(&dosya, "cikis1.txt", "r");
40          while (fscanf_s(dosya, "%d%c ", &num2, &char2) != EOF) {
41              arr1[a] = num2;
42              arr2[a] = char2;
43              a++;
44          }
45          fclose(dosya);
```



- Devamında dosya yazılmak amacıyla tekrardan açılır. Dizinin aldığı karakter sayısına kadar olacak şekilde for döngüsüne sokulur. Bu for döngüsünün içine sıkıştırılmış diziyi çözmek için while döngüsü kurarız. Bu while döngüsü integer dizimiz 0 olana kadar devam edecek şekilde kurulur. Ve bu döngünün içinde ilk önce char dosyaya bastırılır ve integer dizimizin değeri 1 azaltılır. Böylece karakterimizin başındaki sayı 0 olana kadar o karakteri dosyaya yazdırmış oluruz. For döngüsü bitince de dosyayı kapatırız.

```c
        fopen_s(&dosya, "cikis1.txt", "w");
        for (int j = 0; j < a; j++) {
            while (arr1[j] != 0) {
                fprintf(dosya, "%c", arr2[j]);
                arr1[j] = arr1[j] - 1;
            }
        }
        fclose(dosya);
    }
```



- 1 veya 2'den farklı bir değer girilirse de kod çalışmaz ve kapanır.

```
55      else
56          printf("1 ve 2'den farkli deger girdiniz.");
57  }
```

İZLEDİĞİNİZ İÇİN TEŞEKKÜRLER.

FENERBAHÇE ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ

Hazırlayanlar:

Cüneyt Balcı

Berk Tunç