**YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ**



Öğrenci No: 21011084

Ad-Soyad : Berkay Gümüşay

Öğrenci E-Postası: berkay.gumusay@std.yildiz.edu.tr

**BLM-1012—YAPISAL PROGRALAMAYA GİRİŞ FİNAL PROJESİ**

# HUFFMAN CODİNG ALGORİTHM

Ders Yürütücüsü

Ögr. Gör. Dr. Ahmet Elbir

H aziran,2021

# İ ÇERİK

## Huffman Coding Algorithm Nedir? Ne işe yarar? Nasıl Çalışır?

* Kullanım Yerleri
* Avantaj-Dezavantajları
* Karmaşıklığı
* Rakipleri
* Çalışmasını açıklarken kullanılan ekran çıktıları
* C dilindeki kodu
* Kaynaklar

V İDEO ADRESİ

h ttps://youtu.be/

### **Huffman Coding Algoritması Nedir? Ne İşe Yarar?**

### Huffman Coding algoritması David Albert Huffman tarafından 1952 yılında geliştirilmiş bir kayıpsız veri sıkıştırma algoritmasıdır.

* + Huffman Coding algoritması verideki her karakteri ikilik sayılardan oluşan kodlar ile saklar.
  + Verileri çok etkili bir biçimde sıkıştırarak alandan %20-%90 oranında tasarruf sağlar.
  + Her bir karakterin tekrar etme sıklığına göre sıkıştırmak için en optimal yolu kullanır. (Sık tekrar eden karakter daha az, az tekrar eden karakter daha fazla alan kaplar.)

**Huffman Coding Algoritması Nasıl Çalışır?**

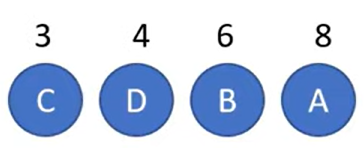
**a) Verilen Veriyi Sıkıştırma**

1. Öncelikle seçilen kelime veya kelime grubundaki her bir karakterin frekansı (adedi) belirlenir.

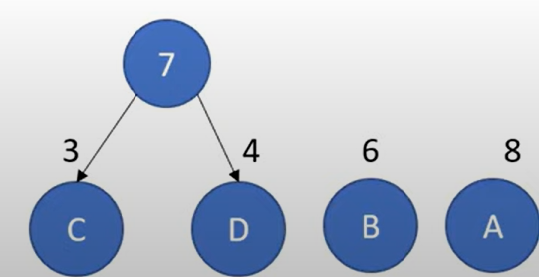
tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

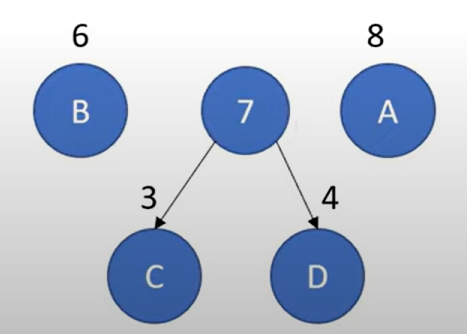
1. Ardından karakterler frekanslarına göre küçükten büyüğe doğru sıralanır.



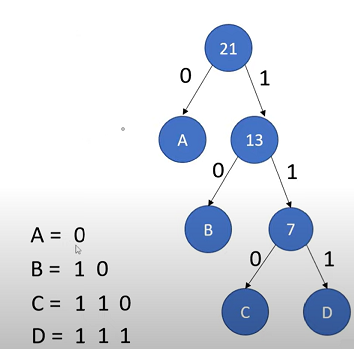
1. Sıralanan karakterlerin en küçük iki elemanı toplanılarak yeni bir düğüm (node) meydana getirilir.



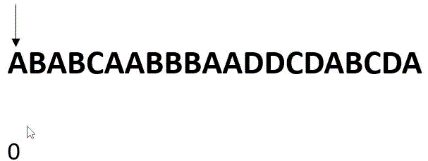
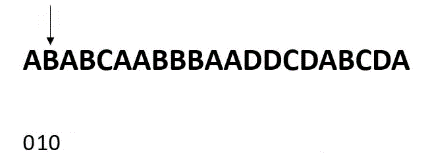
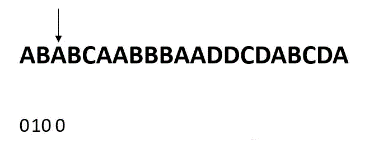
1. Oluşan yeni düğümün sayı değerine göre diğer düğümler ile tekrar küçükten büyüğe sıralama işlemi yapılır



1. En küçük iki düğüm toplanılır ve tüm düğümler ağaca katılana kadar küçükten büyüğe sıralayıp en küçük iki düğümü toplama işlemine devam edilir
2. Ağaç oluşturulduktan sonra her sola gidişte karakter koduna 0, her sağa gidişte ise karakter koduna 1 eklenir.

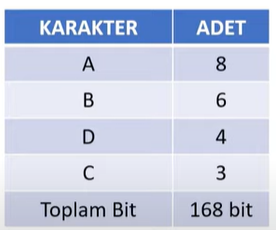


1. Her karaktere ulaşana kadarki kodlar sırasıyla yazılarak her karakterin ikili kodu oluşturulur.
2. Girilen karakter dizisindeki her karakterin yerine belirlenen ikili kod yazılarak veri sıkıştırılır.

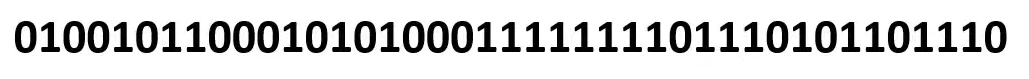
1. Sonuç:

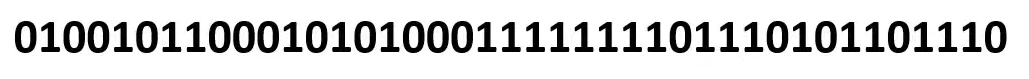
Sıkıştırma İşleminden Önce: Sıkıştırma İşleminden Sonra:

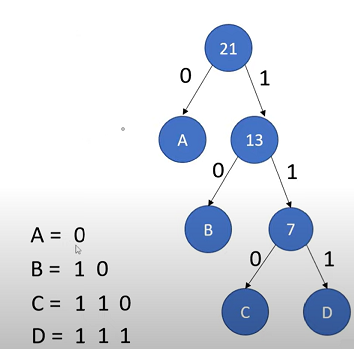
 tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**b) Sıkıştırılan Veriyi orijinal veriye Çevirme**

1. İkili sayı dizisinde ilk 0’a rastlayana kadar ilerlenerek bulunan her ikili sayı grubunun karşılığı olan karakterler belirlenir.
2. 
3. 
4. Örneğin yukarıdaki örnekte 0’a rastlayana kadar ilerlenirse elde edilen sayılar:
5. 0(A) – 10(B) – 0(A) – 10(B) – 110(C) – 0(A) – 0(A) – 10(B) – 10(B)….
6. Şeklinde olur.
7. Belirlenen karakterler soldan sağa sırasıyla yan yana yazılır.
8. Sıkıştırılan veriyi orijinal haline çevirme işlemi ağaç (Huffman Tree) kullanılarak da yapılabilir.
9. İkili sayı dizisinde 0 veya 1 gördükçe ağaçta sola veya sağa giderek ulaşılan yapraktaki karakter yazılarak devam edilebilir.





**Kullanım Alanları :**

1. Fax ve metin iletimi
2. GZIP , BZIP2 , PKZIP vb. gibi sıkıştırma programları
3. Veri bütünlüğünün bozulmaması gereken durumlar (.txt ,.doc)
4. JPEG , PNG ve MP3 gibi multimedya ögelerinin sıkıştırılmasında kullanılabilir fakat sıkıştırma miktarı az olacağından pek tercih edilmez.

**Avantaj Ve Dezavantajları :**

**Avantajları :**

1. Daha sık tekrar eden karakter için daha küçük bit gerekir. Bu sayede sık tekrar eden karakterlerin sıkıştırılma oranı artar.

**Dezavantajları :**

1. Huffman Coding gibi kayıpsız veri sıkıştırma algoritmaları , kayıplı veri sıkıştırma algoritmalarına kıyasla daha düşük bir sıkıştırma oranına sahiptir.
2. Kayıpsız veri sıkıştırma algoritmalarına kıyasla daha yavaş çalışır.
3. Sıkıştırılan verinin tekrardan açılması zordur.

**Karmaşıklığı :**

Huffman Coding algoritması n \* log(n) zaman karmaşıklığına sahiptir.

(n=farklı karakterlerin sayısı)

**Rakipleri :**

1. Run-length encoding (RLE)
2. Huffman Coding’e göre daha basit bir algoritmadır.
3. Huffman Coding gibi kayıpsız veri sıkıştırır.
4. Zaman karmaşıklığı = O(n) (Bu yüzden Huffman Coding RLE’ den daha avantajlıdır)
5. Dinamik Markof Kodlaması ile Sıkıştırma ( Dynamic Markov Coding )
6. Huffman Coding gibi kayıpsız veri sıkıştırır.
7. Delta Sıkıştırması ( Delta Compression )
8. Huffman Coding gibi kayıpsız veri sıkıştırır.
9. Basit ve Hızlıdır
10. Elias Code
11. Huffman Coding gibi veriyi ikilik tabanda gösterir.
12. LZW Sıkıştırma Algoritması
13. Huffman Coding gibi kayıpsız veri sıkıştırır.
14. Zaman karmaşıklığı = O(n) (Bu yüzden Huffman Coding RLE’ den daha avantajlıdır)

**Oluşan Ekran Çıktıları**

**Örnek Metin 1 :** aaaabbcccccdeee

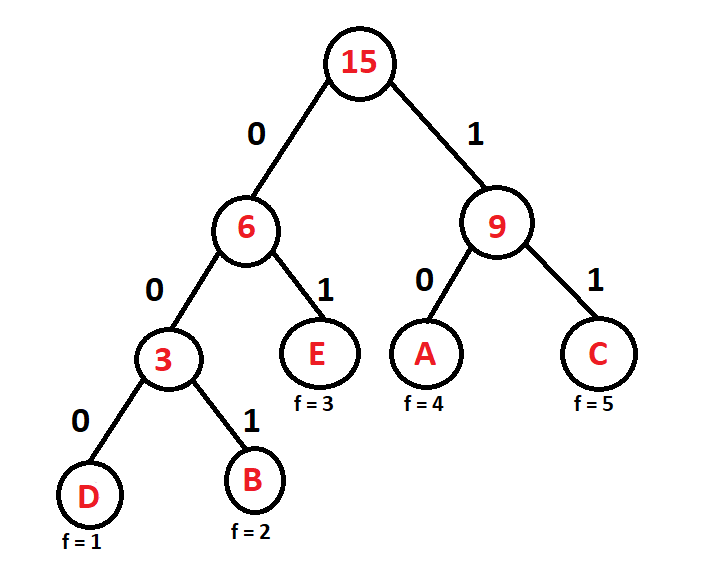
**Kodu Oluşturma Aşaması :**

Karakterlerin Frekansını Hesaplama :

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Oluşan Ağaç :



Kod Çıktısı :

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Girilen Kodun Karakter Dizisinde Dönüştürülme Aşaması :**

Dönüştürülecek Kod : 1011001000000010001011

(a)(c)(b) (d) (d)(e) (d) (a) (c)

Kod Çıktısı :

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Örnek Metin 2 :** bccccddgda

Karakterlerin Frekansını Hesaplama :

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Oluşan Ağaç :

metin, portakal içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kod Çıktısı :

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Girilen Kodun Karakter Dizisinde Dönüştürülme Aşaması :**

Dönüştürülecek Kod : 0011100101010111010100

(c)(c)(d)(a) (b) (g) (b) (a)

Kod Çıktısı :

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**PROGRAMIN C DİLİNDEKİ KODU**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**KAYNAKLAR**

* <https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1206/lectures/huffman-coding/>
* <https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/>
* <https://en.wikipedia.org/wiki/Huffman_coding>
* <https://sites.fas.harvard.edu/~libs111/files/lectures/unit9-1.pdf>