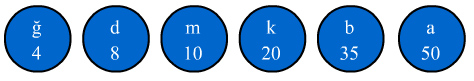
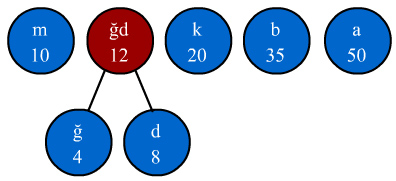
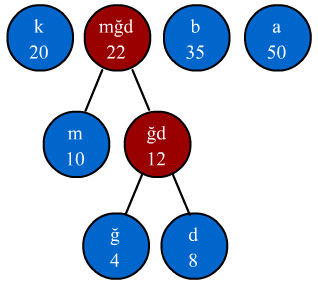
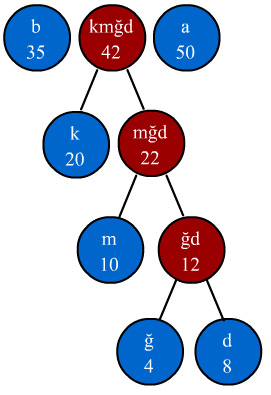
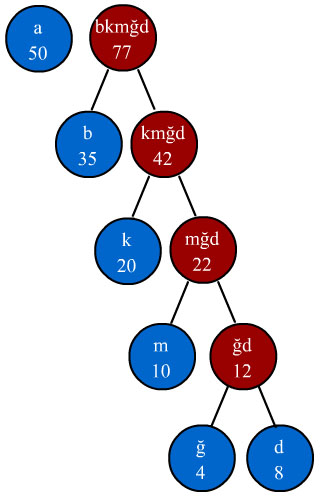
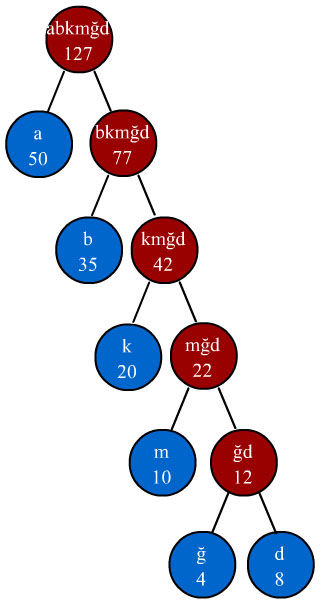
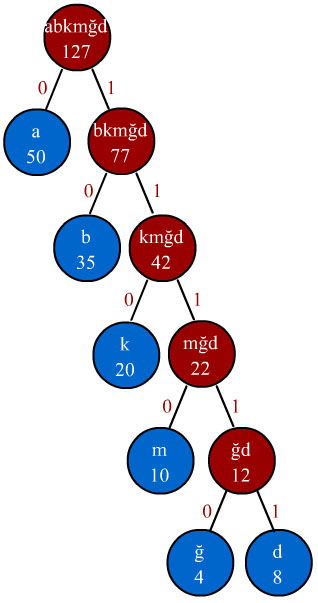
**Huffman Ağacının Oluşturulması**  
  
Bir huffman ağacı aşağıdaki adımlar izlenerek oluşturulabilir.  
  
Bu örnekte aşağıdaki frekans tablosu kullanılacaktır.

|  |  |
| --- | --- |
| Sembol(Karakter) | Sembol Frekansı |
| a | 50 |
| b | 35 |
| k | 20 |
| m | 10 |
| d | 8 |
| ğ | 4 |

Bu tablodan çıkarmamız gereken şudur : Elimizde öyle bir metin dosyası varki "a" karakteri 50 defa, "b" karakteri 35 defa .... "ğ" karakteri 4 defa geçiyor. Amacımız ise her bir karakteri hangi bit dizileriyle kodlayacağımızı bulmak.   
  
**1** - Öncelikle "Huffman Ağacını" ndaki en son düğümleri(dal) oluşturacak bütün semboller frekanslarına göre aşağıdaki gibi küçükten büyüğe doğru sıralanırlar.  
  
   
  
**2** - En küçük frekansa sahip olan iki sembolün frekansları toplanarak yeni bir düğüm oluşturulur. Ve oluşturulan bu yeni düğüm diğer varolan düğümler arasında uygun yere yerleştirilir. Bu yerleştirme frekans bakımından küçüklük ve büyüklüğe göredir. Örneğin yukarıdaki şekilde "ğ" ve "d" sembolleri toplanarak "12" frakansında yeni bir "ğd" düğümü elde edilir. "12" frekanslı bir sembol şekilde "m" ve "k" sembolleri arasında yerleştirilir. "ğ" ve "d" düğümleri ise yeni oluşturulan düğümün dalları şeklinde kalır. Yeni dizimiz aşağıdaki şekilde olacaktır.  
  


**3** - 2.adımdaki işlem tekrarlanarak en küçük frekanslı iki düğüm tekrar toplanır ve yeni bir düğüm oluşturulur. Bu yeni düğümün frekansı 22 olacağı için "k" ve "b" düğümleri arasına yerleşecektir. Yeni dizimiz aşağıdaki şekilde olacaktır.  
  
   
  
  
**4** - 2.adımdaki işlem tekrarlanarak en küçük frekanslı iki düğüm tekrar toplanır ve yeni bir düğüm oluşturulur. Bu yeni düğümün frekansı 42 olacağı için "b" ve "a" düğümleri arasına yerleşecektir. Yeni dizimiz aşağıdaki şekilde olacaktır. Dikkat ederseniz her dalın en ucunda sembollerimiz bulunmaktadır. Dalların ucundaki düğümlere özel olarak **yaprak(leaf)** denilmektedir. Sona yaklaştıkça Bilgisayar bilimlerinde önemli bir veri yapısı olan Tree(ağaç) veri yapısına yaklaştığımızı görüyoruz.   
  
  
  
  
**5** - 2.adımdaki işlem tekrarlanarak en küçük frekanslı iki düğüm tekrar toplanır ve yeni bir düğüm oluşturulur. Bu yeni düğümün frekansı 77 olacağı için "a" düğümünden sonra yerleşecektir. Yeni dizimiz aşağıdaki şekilde olacaktır. Dikkat ederseniz her bir düğümün frekansı o düğümün sağ ve sol düğümlerinin frekanslarının toplamına eşit olmaktadır. Aynı durum düğüm sembolleri içinde geçerlidir.   
  


**6** - 2.adımdaki işlem en tepede tek bir düğüm kalana kadar tekrar edilir. En son kalan düğüm Huffman ağacının kök düğümü(root node) olarak adlandırılır. Son düğümün frekansı 127 olacaktır. Böylece huffman ağacının son hali aşağıdaki gibi olacaktır.  
  
   
  
Not : Dikkat ederseniz Huffman ağacının son hali simetrik bir yapıda çıktı. Yani yaprak düğümler hep sol tarafta çıktı. Bu tamamen seçtiğimiz sembol frekanslarına bağlı olarak rastlantısal bir sonuçtur. Bu rastlantının oluşması oluşturduğumuz her bir yeni düğümün yeni dizide ikinci sıraya yerleşmesinden kaynaklanmaktadır. Sembol frekanslarında değişiklik yaparak ağacın şeklindeki değişiklikleri kendinizde görebilirsiniz.

**7** - Huffman ağacının son halini oluşturduğumuza göre her bir sembolün yeni kodunu oluşturmaya geçebiliriz. Sembol kodlarını oluşturuken Huffman ağacının en tepesindeki kök düğümden başlanır. Kök düğümün sağ ve sol düğümlerine giden dala sırasıyla "0" ve "1" kodları verilir. Sırası ters yönde de olabilir. Bu tamamen seçime bağlıdır. Ancak ilk seçtiğiniz sırayı bir sonraki seçimlerde korumanız gerekmektedir. Bu durumda "a" düğümüne gelen dal "0", "bkmğd" düğümüne gelen dal "1" olarak seçilir. Bu işlem ağaçtaki tüm dallar için yapılır. Dalların kodlarla işaretlenmiş hali aşağıdaki gibi olacaktır.  
  


   
  
**8** - Sıra geldi her bir sembolün hangi bit dizisiyle kodlanacağını bulmaya. Her bir sembol dalların ucunda bulunduğu için ilgili yaprağa gelene kadar dallardaki kodlar birleştirilip sembollerin kodları oluşturulur. Örneğin "a" karakterine gelene kadar yalnızca "0" dizisi ile karşılaşırız. "b" karakterine gelene kadar önce "1" dizisine sonra "0" dizisi ile karşılaşırız. Dolayısıyla "b" karakterinin yeni kodu "10" olacaktır. Bu şekilde bütün karakterlerin sembol kodları çıkarılır. Karakterlerin sembol kodları aşağıda bir tablo halinde gösterilmiştir.

|  |  |  |  |
| --- | --- | --- | --- |
| Frekans | Sembol(Karakter) | Bit Sayısı | Huffman Kodu |
| 50 | a | 1 | 0 |
| 35 | b | 2 | 10 |
| 20 | k | 3 | 110 |
| 10 | m | 4 | 1110 |
| 8 | d | 5 | 11111 |
| 4 | ğ | 5 | 11110 |

Sıkıştırılmış veride artık ASCII kodları yerine Huffman kodları kullanılacaktır. Dikkat ederseniz hiçbir Huffman kodu bir diğer Huffman kodunun ön eki durumunda değildir. Örneğin ön eki "110" olan hiç bir Huffman kodu mevcut değildir. Aynı şekilde ön eki "0" olan hiç bir Huffman kodu yoktur. Bu Huffman kodları ile kodlanmış herhangi bir veri dizisinin **"tek çözülebilir bir kod"**olduğunu göstermektedir. Yani sıkıştırılmış veriden orjinal verinin dışında başka bir veri elde etme ihtimali sıfırdır. (Tabi kodlamanın doğru yapıldığını varsayıyoruz.)  
  
Şimdi örneğimizdeki gibi bir frekans tablosuna sahip olan metnin Huffman algoritması ile ne oranda sıkışacağını bulalım :   
  
Sıkıştırma öncesi gereken bit sayısını bulacak olursak : Her bir karakter eşit uzunlukta yani 8 bit ile temsil edildiğinden toplam karakter sayısı olan (50+35+20+10+8+4) = 127 ile 8 sayısını çarpmamız lazım. Orjinal veriyi sıkıştırmadan saklarsak **127\*8 = 1016** bit gerekmektedir.  
Huffman algoritmasını kullanarak sıkıştırma yaparsak kaç bitlik bilgiye ihtiyaç duyacağımızı hesaplayalım : 50 adet "a" karakteri için 50 bit, 35 adet "b" karakteri için 70 bit, 20 adet "k" karakteri için 60 bit....4 adet "ğ" karakteri için 20 bite ihtiyaç duyarız. (yukarıdaki tabloya bakınız.) Sonuç olarak gereken toplam bit sayısı = 50\*1 + 35\*2 + 20\*3 + 10\*4 + 8\*5 + 4\*5 = 50 + 70 + 60 + 40 + 40 + 20 = 280 bit olacaktır.  
  
Sonuç : 1016 bitlik ihtiyacımızı 280 bite indirdik. Yani yaklaşık olarak %72 gibi bir sıkıştırma gerçekleştirmiş olduk. Gerçek bir sistemde sembol frekanslarınıda saklamak gerektiği için sıkıştırma oranı %72ten biraz daha az olacaktır. Bu fark genelde sıkıştırılan veriye göre çok çok küçük olduğu için ihmal edilebilir. 