©Ali Galip Bayrak - ODTÜ Bilgisayar Topluluğu

Zip

Bu sayıdaki sorumuz öncekilerden biraz farklı olacak. Bu kez bir yöntem anlatacağım ve siz onun uygulamasını yapacaksınız. Anlatacağım yöntem, veri sıkıştırmada kullanılan ve "Huffman Coding" olarak bilinen kodlama yöntemi.

Öncelikle kodlamadan ve veri sıkıştırmadan bir örnekle bahsedelim. Bir text dosyamız olduğunu ve içinde sadece a, b, c ve d harflerinin kullanıldığını düşünelim. En basit anlamda kodlama yaparak a harfini ikili sistemde 00, b harfini 01, c harfini 10, d harfini de 11 olarak gösterebiliriz. Örneğin "acabad" yazan bir text dosyasının ikili gösterimi 001000010011 olacaktır. Bu kodlamanın en önemli özelliği tersini de yapabiliyor olmamız, yani bize 001000010011 verildiği zaman text dosyasında ne yazdığını da söyleyebiliriz. Bize içinde 100 adet a, 10 adet b, 2 adet c ve 1 adet d harfleri geçen bir dosya verilsin. Bahsettiğimiz kodlama yöntemini kullanırsak her harf için 2 bit (her bitte 0 ya da 1 tutabiliriz) kullandığımızdan toplamda 2 * (100 + 10 + 2 + 1) = 226 bit kullanmamız gerekecektir. Şimdi farklı bir kodlama deneyelim. Bu kodlamada a'yı 0 ile, b'yi 10 ile, c'yi 110 ile d'yi de 111 ile gösterelim. Bu kodlamanın da diğerinde olduğu gibi tersi yapılabilmektedir. Örneğin 0011001010111 verilirse bunun karşılığı aacabbd'dir (tersi olmayan, daha doğrusu bir ikili gösterimin birden fazla karşılığı olduğu kodlamalar da vardır, örneğin a'yı 0, b'yi 1, c'yi 10, d'yi 11 olarak gösteren bir kodlamada 010 hem aba, hem ad anlamına gelebilmektedir). Bu kodlamada kullanılacak bit sayısı a için 100 * 1, b için 10 * 2, c için 2 * 3, d için 1 * 3 olmak üzere toplamda 129'dur. Gördüğümüz üzere normal bir kodlama 226 bit gerektirirken, bu kodlama 129 bit gerektirmektedir. Veri sıkıştırma olayının temelinde yatan mantık da budur. Şimdi bu kodlamanın nasıl yapıldığına bakalım.

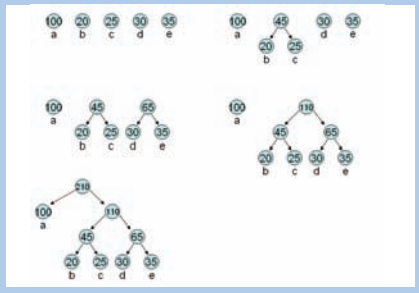
İlk olarak bütün karakterlerin kaç kez kullanıldığını hesaplayalım. Daha sonra şu işlemleri sırasıyla yapalım:

1. Her karakter için bir ağaç oluşturalım.
2. Toplamda en az kullanılan karakterleri içeren iki ağacı birleştirip tek bir ağaç yapalım ve bu işlemi bütün ağaçlar birleşene kadar devam ettirelim.
3. Ağacın bütün dallarında solda kalan dala 0, sağda kalan dala 1 verelim.

Örnekle gösterecek olursak, dosyada kullanılan harfler ve kaç kez kullanıldıkları şu şekilde olsun:

a: 100, b: 20, c: 25, d: 30, e: 35.

Bahsettiğimiz işlemi aşama aşama gösterecek olursak:

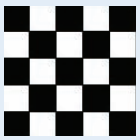


İlk aşamada her karakter ayrı ağaçlar olarak düşünülmüştür. İkinci şekilde en küçük değere sahip iki ağaç olan 20 ve 25 birleştirilip 45'lik tek bir ağaç elde edilmiştir. Daha sonra 100, 45, 30 ve 35'ten en küçük ikisi olan 30 ve 35 birleştirilip 65'lik tek ağaç elde edilmiştir. 4. şekilde 100, 45 ve 65'lik ağaçlardan küçük olan 45 ve 65 birleştirilip 110 elde edilmiş ve son basamakta kalan iki ağaç birleştirilip tek bir ağaç elde edilmiştir. Şimdi kodlamayı yaparken yukarda da bahsettiğimiz gibi soldaki dala 0 sağdaki dala 1 verirsek (tam tersini yapmamızda da bir sakınca yok) a harfi 0, b harfi 100, c harfi 101, d harfi 110 ve e harfi 111 ile gösterilecektir.

Sizden istenen bu algoritmayı kullanarak kendi sıkıştırma programınızı yazmanız ve kendiniz bir sıkıştırma algoritması üretip onun programını yazmanız.

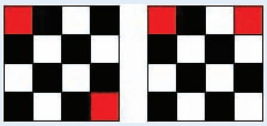
Geçen Sayımızdaki Soruların Çözümleri

Domino



İlk aşamada büyük karenin ne zamanlar kapatılabilip ne zamanlar kapatılamayacağını belirleyelim. Şekli satranç tahtasında olduğu gibi bir siyah bir beyaz kare olacak şekilde boyayalım.

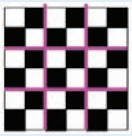
Domino taşları 2 x 1'lik boyutlara sahip olduğu için herhangi bir domino taşını büyük kare üzerinde nereye koyarsak koyalım bir ucu siyah bir kareye, diğer ucu beyaz bir kareye denk gelecektir. Sadece bu çeşit taşlarla büyük kareyi kapatmayı deneyeceğimiz için kapatabileceğimiz siyah kare sayısı kapatabileceğimiz beyaz kare sayısı ile aynı olacaktır. Büyük karedeki siyah kare sayısı ile beyaz kare sayısı da aynı olduğu için geri kalan karelerde de, ki bunlar işaretli kareler oluyor, siyah sayısı beyaz sayısına eşit olmalıdır. Bu çıkarımı kullanarak, büyük karemizi yukardaki gibi karaladıktan sonra işaretli karelerden siyaha denk gelenlerin sayısı beyaza denk gelenlerin sayısına eşit değilse hiçbir şekilde kapatılamaz diyebiliriz. Örnek verecek olursak:



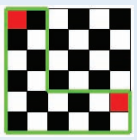
Soldaki şekli hiçbir şekilde kapatamayız çünkü işaretli karelerin (kırmızı kareler) ikisi de beyaz karelerin üzerine geliyor. Sağdaki şekilde işaretli karelerin birisi beyaz, birisi siyah karelerin üzerine geliyor ve birazdan bahsedeceğimiz teknikle bu şekil kapatılabiliyor.

Öyleyse şimdi işaretli beyaz kare sayısı işaretli siyah kare sayısına eşit olan şekillerin nasıl kapatılabileceğinden bahsedelim.

İlk olarak büyük karemizi 2x2'lik küçük karelere ayıralım.



Şimdi şeklimizdeki herhangi iki işaretli kareyi düşünelim. Bu karelerin bulundukları 2x2'lik kareleri birbirine bağlayan bir yol çizelim. Örneğin şu şekilde olduğu gibi:



Bu yol (yeşil ile gösterilen alan), şekilde olduğu gibi genişliği 2 olan iki büyük dikdörtgenin birleşimi şeklinde olsun. Bu yol içerisinde eğer başka işaretli kare yoksa nasıl doldurulabileceği sanırım açık. Eğer bütün noktaları bu şekilde ikili gruplara ayırıp aralarındaki yolları doldurursak geriye sadece 2x2 lik kareler kalır, ki bunların da nasıl doldurulabileceği çok açık (altalta ya da yanyana iki domino taşı koyarak). Şimdi son problemimiz bu işaretli noktaları ikili gruplara nasıl ayıralım ki, ikililer arasındaki yolları çizdiğimizde yolların hiçbirisi birbirine değmesin. Çözümün bu kısmını size bırakıyorum (ipucu: en dıştaki noktalardan başlayarak içeri doğru gidin ve işaretli noktaların neden 3'ün katları koordinatlı verildiğine dikkat edin).