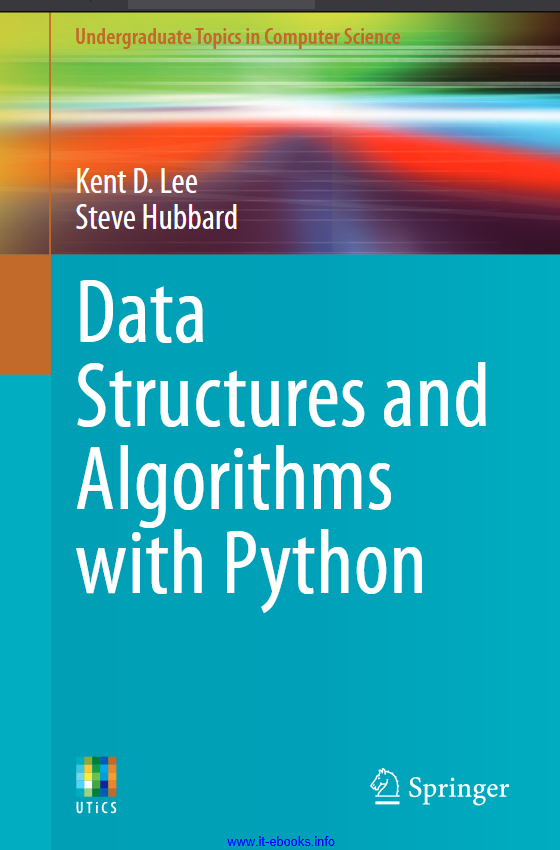
****

**8.2.1 The Hashing Functions. . . . . . . . . . . . . . . . . . . . . . . . . . . .. 207**

**8.2.2 The Bloom Filter Size. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. 208**

**8.2.1 Hashing İşlevleri - The Hashing Functions**

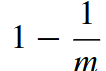
Bir bloom filtresine eklenen her öğe, belirli sayıda hash fonksiyonu ile hashlenmelidir ve bu fonksiyonlar birbirinden tamamen bağımsızdır. Her bir hash fonksiyonu ayrıca bit dizisindeki bit indeksleri aralığına eşit olarak dağıtılır. Bu ikinci gereklilik, hash kümeleri ve hash tabloları için kullanılan hash fonksiyonları için de geçerlidir. Tek tip dağıtım, Python'un ve diğer birçok dilin yerleşik hash fonksiyonları tarafından garanti edilir.

Yukarıdaki örneklerde üç hash fonksiyonu gerekliydi. Bazen eklenen öğelerin sayısı ve bit dizisindeki bit sayısına bağlı olarak gerekli hash fonksiyonu sayısı çok daha yüksek olabilir. Bağımsız ve eşit dağılımlı hash fonksiyonlarının gerekli sayıda oluşturulması, göz korkutucu bir problem gibi görünebilir ancak en azından birkaç yolla çözülebilir. Bazı hash fonksiyonları, bir tohum değerinin sağlanmasına izin verir. Bu durumda, farklı tohum değerleri, farklı hash fonksiyonları oluşturmak için kullanılabilir.

Bağımsız hashing fonksiyonları oluşturmanın eşit derecede etkili bir başka yolu, hash edilmeden önce her öğenin sonuna bilinen bir değer eklemektir. Örneğin, ilk hash fonksiyonunu elde etmek için hashlemeden önce öğeye bir 0 eklenebilir. İkinci hash fonksiyonunu elde etmek için hash işleminden önce öğeye 1 eklenebilir. Benzer şekilde, üçüncü hash fonksiyonu değerini elde etmek için bir 2 eklenebilir. Yani, bloom filtresinde "tavşan"ın yukarı kaldırılması ilk olarak tavşan0, tavşan1 ve tavşan2 ile aynı hashing fonksiyonuna sahiptir. Hashing fonksiyonu düzgün bir şekilde dağıtıldığında, üç hash fonksiyonu tarafından döndürülen değerler birbirinden bağımsız olacaktır. Ayrıca, 0 eklenmiş tüm öğeler eşit olarak dağıtılacak, aynı durum 1 ve 2 eklenmiş öğeler için de geçerli olacaktır.

**8.2.2 Bloom Filtre Boyutu - The Bloom Filter Size**

Eklenecek öğe sayısı verildiğinde, gerekli bloom filtresi boyutunu bulmak mümkündür ve istenen bir yanlış pozitif olasılığına göre hesaplanabilir. Bloom filtresinin herhangi bir konumunun, bir öğe eklenirken bir hash fonksiyonu tarafından ayarlanmaması olasılığı, filtrenin *m* bitten oluştuğu durumda aşağıdaki formülle hesaplanır.



Eğer bloom filtresi *k* hash fonksiyonu kullanıyorsa, bit dizisindeki bir bitin, bir öğe eklemek için gereken hash fonksiyonlarından herhangi biri tarafından ayarlanmamış olma olasılığı aşağıdaki formülle ifade edilir.

yazı tipi, simge, sembol, tasarım, tipografi içeren bir resim

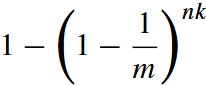
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bloom filtresine *n* öğe eklenirse, bu formülün *n*'inci kuvvetine yükseltilmesi, bloom filtresinin bit dizisi içindeki bir bitin, tüm *n* öğeler eklendikten sonra hala sıfır olma olasılığını sağlar. Yani, elimizde şu ifade olur:

yazı tipi, simge, sembol, sayı, numara, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Dolayısıyla, *n* öğe eklendikten sonra bloom filtresindeki bir bitin 1 olma olasılığı, *k* hashing fonksiyonu kullanılarak aşağıdaki formül ile verilir.



Şimdi, bloom filtresine eklenmemiş bir öğeyi aradığınızı düşünün. Yanlış pozitif raporlama olasılığı, bloom filtresi içindeki her konumun tüm *k* hashing fonksiyonları için 1 olması durumunda hesaplanabilir. Bu olasılık aşağıdaki gibi ifade edilir.

yazı tipi, diyagram, çizgi, tasarım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu formül, doğal logaritma kullanılarak yaklaşık olarak aşağıdaki gibi ifade edilebilir [8].

yazı tipi, metin, çizgi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu formülü kullanarak, *n* öğe sayısı ve istenen yanlış pozitif olasılığı *p* verildiğinde, gerekli bit sayısı *m*'yi çözmek mümkündür. Formül aşağıdaki gibidir.

yazı tipi, metin, sayı, numara, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Son olarak, yukarıdaki *k* değerini çözdüğümüzde aşağıdaki formül elde edilir.

yazı tipi, tipografi, metin, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu iki formül, bize filtrede kaç bitin gerekli olduğunu belirterek belirtilen maksimum yanlış pozitif oranını sağlar. Ayrıca, gerekli hash fonksiyonu sayısını da hesaplayabiliriz. Örneğin, 109.583 kelime içeren bir İngilizce sözlük için ve %1'den fazla olmayan istenen yanlış pozitif yüzdesi (formülde 0,01 olarak ifade edilen) ile 1.050.360 bitlik bir bit dizisi ve yedi hashing fonksiyonu gereklidir.

Bu örnekteki bit sayısı aşırı görünebilir. Ancak, şunu unutmayın: Bitlerdir. Verimli bir uygulama yaklaşık 128 KB depolama alanı gerektirir. Bu örnekte kullanılan İngilizce sözlükteki karakter sayısı toplam 935.171'dir. Karakter başına 1 bayt olduğu varsayılırsa, tüm bu kelimelerin depolanması için minimum 914 KB gerekmektedir. Bloom filtresi, oldukça büyük bir alan tasarrufu sağlar.

Buna ek olarak, deneylerde bloom filtresi kullanılarak yapılan arama süresi hiçbir zaman 160 μs'den uzun sürmemiştir. Arama süresi, hash fonksiyonlarının sayısı ve verimliliği ile sınırlıdır ve istenen değerleri hesaplamak için kullanılır. Eğer hash fonksiyonlarının bağımlı olduğunu ve uzunluğuna bağlı olarak değiştiğini varsayarsak, arama süresi *O(lk)* olarak ifade edilir; burada *l* aranan öğenin uzunluğunu ve *k* hash fonksiyonlarının sayısını temsil eder.

**Ünite sonu 4-5-6. Soruları:**

1. **When spell checking, which data type can be used for spelling correction?** (Yazım denetimi yaparken hangi veri türü yazım düzeltmesi için kullanılabilir?)

Yazım düzeltmesi için trie veri yapısı kullanılabilir. Trie, ortak önekleri sakladığı için yanlış yazılmış kelimelere alternatif önerilerde bulunabilir. Bloom filter ise yalnızca üyelik testi yapar ve alternatif öneriler sunamaz.

1. **How can you generate more than one hashing function for use in a bloom filter?** (Bir bloom filter için birden fazla hash fonksiyonu nasıl oluşturabilirsiniz?) Bir bloom filter için birden fazla hash fonksiyonu şu yöntemlerle oluşturulabilir:

Seed (tohum) değeri kullanılarak bağımsız hash fonksiyonları oluşturulabilir. Aynı hash fonksiyonu, farklı seed değerleriyle tekrar çalıştırılarak farklı sonuçlar elde edilir. Alternatif olarak, hashlenmeden önce kelimenin sonuna belirli bir karakter (örneğin, 0, 1, 2) eklenebilir. Bu sayede aynı hash fonksiyonuyla farklı sonuçlar alınabilir.

1. **Add the words “a”, “an”, “ant”, “bat”, and “batter” to a trie. Draw the trie data structure showing its structure after inserting the words in the order given here.** (“a”, “an”, “ant”, “bat” ve “batter” kelimelerini bir trie veri yapısına ekleyin. Verilen sıraya göre trie veri yapısını çizin.) Verilen sıraya göre trie veri yapısı şu şekilde olur:

(a)

└─(n)

└─(t)

(b)

└─(a)

└─(t)

└─(t)

└─(e)

└─(r)

Trie veri yapısında, kelimelerin ortak önekleri paylaşılarak daha az yer kaplanır. Kelimelerin sonunu belirtmek için genellikle sentinel karakteri ($) eklenir.

*Yusuf Gökali - 2024481020*