Bilgisayar Bilimi Köşesi

H. Coşkun Gündüz* / *coskung@bilgi.edu.tr*

Monte Carlo Yöntemi

Rastgele sayı seçme ilkesi üzerine kurulmuş algoritmalara **rastlantısal algoritmalar** denir. Her ne kadar yanıt vermedikleri, hatta kimileyin yanlış yanıt verdikleri bile olsa, bazı problemler için, rastlantısal algoritmalar kesin algoritmalara tercih edilebilirler, çünkü daha basittirler ve daha hızlı çalışırlar.

n sayısıyla ilgili bir algoritmanın 2^n zamanda sonuçlanması bilgisayar biliminde pek arzu edilmez, çünkü böyle bir algoritma çok yavaş işler. Ama daha kısa bir algoritma bilinmiyorsa ne yapmalı? Rastlantısal algoritmalar bu hız sorununa kısmi bir çözüm getirirler: yüzde yüz doğruluktan vazgeçerek hızlı algoritmalar bulabiliriz.

Elbette rastlantısal algoritmaların uçaklarda kullanılmamasının büyük yararları vardır!

Rastlantısal algoritmalar, Monte Carlo, Las Vegas ve Sherwood olarak üç ana başlıkta toplanabilir.

Monte Carlo algoritmaları her zaman bir sonuç verir ve sonucun doğruluk olasılığı program çalıştıkça artar. Örneğin her denemede yüzde 90 başarı şansı varsa, iki denemede başarı şansı yüzde 99'a çıkar.

Öte yandan Las Vegas algoritmaları yanlış sonuç üretmezler, ama bazen hiç sonuç üretmezler.

Sherwood algoritmaları ise her zaman sonuç verir ve verdikleri sonuç doğrudur. Ancak Sherwood algoritmalarının uygulanabildiği problemler sınırlıdır.

Problemin türüne ve beklenen sonucun önemine göre bu yöntemlerden biri seçilir. Biz bu yazımızda Monte Carlo algoritmalarını inceleyeceğiz.

Şimdi çok bilinen bazı problemler üzerinde Monte Carlo algoritmasının çalışmasını inceleyelim:

Baskın Eleman Problemi. Bir sayı dizisinde, sayıların yarıdan fazlası aynıysa, o elemana **baskın eleman** denir. Bir dizideki baskın elemanı bulma algoritması, en basit olarak tüm elemanların diğerleriyle karşılaştırılması olabilir, bu da $O(n \log n)$ zaman alır.

Bu problem için oluşturulabilecek bir Monte Carlo algoritması şöyle olabilir: Programımız, verilen n sayı arasından rastgele bir sayı seçsin. Eğer bu sayı verilen n sayının en az yarısına eşitse program dursun ve yanıt olarak bu sayıyı versin. Eğer bu sayı verilen n sayının en az yarısına eşit değilse program gene dursun ve yanıt olarak "baskın eleman bulunamadı" desin.

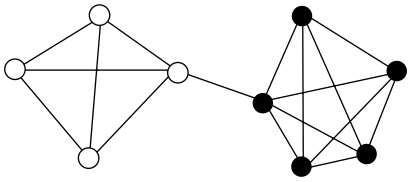
Eğer problem "bulundu" yanıtını verirse, baskın eleman bulunmuş demektir. "Bulunamadı" yanıtını verirse ya yanlış eleman seçilmiştir ya da baskın eleman yoktur.

Eğer baskın eleman varsa, program en az yüzde elli olasılıkla baskın elemanı seçecektir ve doğru yanıtı verecektir. Program beş kez çalıştırıldığında, programın baskın elemanı bulma olasılığı $1 - (1/2)^5 = 0{,}96875$'tir, yani nerdeyse yüzde 97, fena sayılmaz. Ve program $5n$ zamanda biter, oldukça çabuk. Eğer programı altı kez çalıştırırsak, doğru yanıtı bulma olasılığı $1 - (1/2)^5 = 0{,}984375$'dir ve program $6n$ zamanda biter.

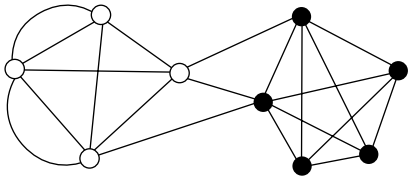
Asallık Testi. Monte Carlo asallık testi algoritması 2 ile $\sqrt{n}$ arasında rastgele sayılar üretir ve bu sayıların n'yi bölüp bölmediğine bakar. Eğer bölen bir sayı bulunursa, n asal değildir. Ama tam bölen bir sayı bulunamazsa, kesin olarak asaldır diyemeyiz. Bu algoritmanın çok iyi bir algoritma olduğu söylenemez. Örneğin, 60329 sayısı 23, 43 ve 61'in çarpımıdır. Algoritma 2 ile 60329'un köküne en yakın tamsayı olan 245 arasında rastgele sayılar seçecektir. Ancak bu aralıkta sadece üç sayı doğru sonuç üretilmesini sağlar. Bu nedenle, bu örnekteki Monte Carlo asallık testi algoritması ancak %1,224 olasılıkla doğru sonucu verir, hiç de iyi bir sonuç sayılmaz.

1 İstanbul Bilgi Üniversitesi Bilgisayar Bölümü öğretim üyesi.

Çizgelerin Parçalanış Sayısı. Bir çizgede, çıkarıldığı taktirde çizgeyi iki parçaya bölecek olan kenarlardan oluşan bir kümeye o çizgenin **parçalanış kümesi** diyelim. Çizgenin **parçalanış sayısı**, bu kümelerin sahip olabileceği en düşük öğe sayısıdır. Parçalanış sayısı, bir anlamda, çizgenin ne kadar "sağlam", yani ne derece tekparça olduğu konusunda bir fikir verir: Parçalanış sayısı ne kadar büyükse, çizge o kadar tekparçadır, noktalar o dere-



Parçalanış sayısı 1 olan bir çizge



Parçalanış sayısı 3 olan bir çizge

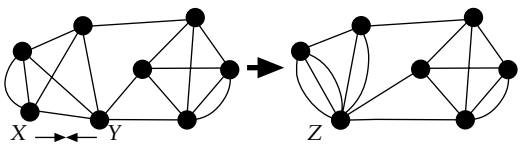
ce birbirine bağlanmıştır diyebiliriz. Örneğin, basit bir döngünün parçalanış sayısı sadece 2'dir. Oysa K_n tamçizgesinin parçalanış sayısı $n-1$'dir; zaten n noktalı bir çizgenin parçalanış noktası $n-1$'den daha büyük olamaz. Parçalanış sayısı, elbette, noktaların derecelerinin en küçüğünden daha büyük olamaz (ama ikinci örnekte görüldüğü üzere daha küçük olabilir).

Problemimiz bir çizgenin parçalanış sayısını bulmak. Basit bir rastlantısal algoritma şöyle çalışır (D. Karger):

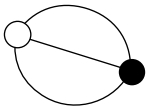
1- G çizgesinden rastgele bir xy kenarı seç. (Bu xy kenarının iki parçadan birinde olduğunu umuyoruz.)

2- Bu kenarı büzüştür (bknz. aşağıdaki şekil), yani bu kenarın x ve y noktalarını tek nokta yap, ama bu işlem sırasında elde edilen çoklu kenarlara dokunma, ayrıca tekdöngüleri kaldır.

3- İki noktadan fazla nokta varsa birinci adıma geri dön. Yoksa, elinde kalan kenar sayısının parçalanış sayısı olduğunu iddia et.



Program, aynı parçada kalacak olan noktaları tek bir noktaya indirgemek istiyor, böylece en sonda her iki parçayı temsilen elimizde iki nokta kalacak. Bu iki nokta arasındaki kenar sayısı da en az parçalanış sayısıdır elbette. Eğer şansımız yaver giderse, program yandaki çizgeye uygulandığında, soldaki dört nokta tek bir noktaya, sağdaki beş nokta da tek bir noktaya indirgenir ve geriye kalan son iki nokta arasında üç kenar olur. İşte bu üç sayısı parçalanış sayısıdır. Ancak burada şansımız yaver gitti, ki herhalde şansımızın yaver gitmesi de az bir olasılık. Yukardaki yöntem, parçalanış sayısını üç olarak vermeyebilir (ama üçten az veremez kesinlikle). Bu programı birkaç kez, diyelim 50 kez uygularsak, ve alınan sonuçların en küçüğünün parçalanış sayısı olduğunu iddia edersek, o zaman gerçek parçalanış sayısına daha yakın bir sayı buluruz.



Yukarıdaki çizgenin son hali

Bu tür programların amacı, çok çok uzun zaman alacak algoritmaların yerine, yüzde bilmemkaç olasılıkla doğru yanıt veren ve kısa sürede sona eren algoritmalar kullanmaktır. Dolayısıyla, programın en az kaç olasılıkla doğru yanıt verdiğini bilmiyorsak, bu tür programlar hiçbir işe yaramazlar. İlla da programın doğru yanıt verme olasılığını bulmalıyız, yoksa programın verdiği yanıtın hiçbir anlamı olmaz.

Şimdi programımızın hangi olasılıkla en küçük parçalanış sayısını bulacağını inceleyelim:

Önsav. *Eğer nokta sayısı n ise, rastgele seçilen bir kenarın bir en küçük parçalanış kümesinde olma olasılığı* (*yani yanlış kenar seçme olasılığımız*) *en fazla* $2/n$*'dir.*

Kanıt: Parçalanış kümesinde k kenar olduğunu varsayalım. O zaman her noktanın derecesi en az k olur, dolayısıyla çizgemizde en az $nk/2$ kenar vardır. Demek ki seçilen bir kenarın en küçük parçalanış kümesinde olma olasılığı $k/(nk/2) = 2/n$'dir. □

Teorem. *Yukardaki algoritma, n noktalı bir çizge için en az* $2/n(n-1)$ *olasılıkla doğru yanıtı verir.*

Kanıt: Çizgede bir parçalanış kümesini sabitleyelim, bu kümeye P diyelim. Algoritmamız çalışırken, seçilen kenarların bu P kümesinin dışında kalmasını isteyelim, ki en son kalan iki noktayı birleştiren kenarlar bu P kümesinin kenarları olsun. Ön-

sava göre, ilk adım sonunda P dışından bir kenar (yani doğru kenar) seçme olasılığı en az $1 - 2/n$'dir. Aynı akıl yürütmeyle, eğer ilk i adımda program hep doğru kenarları seçmişse, o zaman $n - i$ nokta kalan çizgede, gene P'den bir kenar seçme olasılığının en az

$$1 - \frac{2}{n-i} = \frac{n-(i+2)}{n-i}$$

olduğu görülür. Toparlayacak olursak, her adımda doğru kenarlardan birini seçme olasılığımız en az

$$\frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \ldots \times \frac{2}{4} \times \frac{1}{3} = \frac{2}{n(n-1)}$$

dır. □

Bu algoritmayı s kez çalıştırırsak ve verilen s tahminin en küçüğünü alırsak, o zaman algoritma,

$$1 - \left(1 - \frac{2}{n(n-1)}\right)^s$$

olasılıkla doğru yanıtı verir. Eğer $n = 100$ ise, ve biz en az %90 olasılıkla doğru yanıtı istiyorsak, deneme sayımızı bulmak için,

$$0,9 = 1 - \left(1 - \frac{2}{n(n-1)}\right)^s = 1 - \left(1 - \frac{2}{100(100-1)}\right)^s = 1 - \left(\frac{4949}{4950}\right)^s$$

denklemini çözmeliyiz. Basit bir hesap,

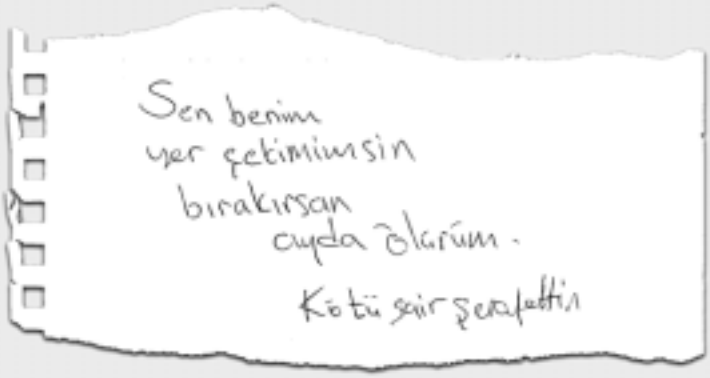
$s = 1/\log_{10}(4950/4949) \approx 11396,6448 \approx 11397$

yanıtını verir.

n noktalı bir çizgede bu algoritmayla %90 doğru yanıt istiyorsak, programın uzunluğunun büyük O'su n cinsinden kaç olur? ♠

Kaynakça

[1] Atallah, Mikhail J., *Algorithms and Theory of Computation Handbook*, CRC Press, Florida 1999.
[2] H. Coşkun Gündüz, www.cs.bilgi.edu.tr/~cgunduz.
[3] McConnell, Jeffrey C. **Analysis of Algorithms**, Jones and Bartlett Computer Science, Canada 2001.
[4] www2.cs.cmu.edu/afs/cs.cmu.edu/user/avrim/www/Randalgs97/lect0122.



Geçenlerde Beyoğlu'nda koltuğumun altında Matematik Dünyası'yla gezerken, Kötü Şair Şerafettin'e rastladım. Hoşbeşten sonra, koltuğumun altındaki Matematik Dünyası'nı gördü.

– O ne? diye sordu.

– Matematik Dünyası... dedim, olağanüstü bir matematik dergisi, sen de okumalısın...

Küçümseyerek,

– Bunda şiir yoktur, dedi.

– Var! dedim ve içindeki şiiri gösterdim.

Şaşırdı. Gözleri güldü.

– Ben de yazayım! dedi.

– Yaz! dedim.

Çantasından bir not defteri çıkardı. Bir sayfa yırtıp ekteki satırları karaladı. Belki derginizde yer alabilir...

Güzel Süzer