DERS 6

Bugün sıralamadan bahsetmeyeceğiz. Bu heyecan verici yeni bir gelişme. Daha başka bir problemi, ilgili ama faklı bir problemi konuşacağız. Doğrusal zamanda çözmek istediğimiz başka bir problemi ele alacağız. Geçen derste sıralamayı doğrusal zamanda yapmaktan bahsetmiştik. Bunu yapabilmek için bazı ek varsayımlara gereksinim vardı. Bugün, ilk bakışta her ne kadar sıralamaya ihtiyaç varmış gibi görünse de, sadece doğrusal zaman çözümüne gereksinim duyulan bir probleme bakacağız. Bu nedenle daha kolay bir problem olacak. Problemde size birtakım sayılar vereceğim.

Bu sayılara elemanlar diyelim. Bunların bir dizilimin içinde olduklarını varsayalım. Ve bir düzen içinde değiller, yani sıralanmamış durumdalar. Ben en küçük **k'nıncı** elemanını bulmak istiyorum. Buna k rank'lı eleman denir. Başka bir deyişle elimde sıralanmamış sayıların bir listesi var. Ve ben bunları sıraladığımda k'nıncı elemanın hangisi olduğunu bilmek istiyorum. Ama bu dizilimi sıralamaya iznim yok. Bu problemin çözümlerinden biri, naive yani saf algoritmayı kullanarak sıralamak ve sonra da k'nıncı elemanına dönmektir. Bu problemin başka bir olası tanımıdır.

Ve bundan daha iyisini yapmak istiyoruz. Yani önce A dizilimi sıralayabilir, ve sonra A[k]'ya dönebilirsiniz. Bu yapabileceğimiz bir şey. Ve yığın sıralaması veya birleştirme sıralamasını kullanırsak, bu n lg n zamanını alır. Biz n lg n'den daha hızlı olmak istiyoruz. İdealde doğrusal zaman istiyoruz. Problem oldukça doğal ve anlaşılır. Bazı uygulamaları da var. Seçiminize bağlı olarak **k**, 1 ile n arasında herhangi bir sayı olabilir. Örneğin k=1 seçersek bu elemanın bir ismi vardır. Bu isim konusunda önerisi olan var mı? Minimum yada en küçük. Bu kolay. Bir dizilimde en küçük elemanı doğrusal zamanda nasıl bulabileceğimiz konusunda önerisi olan var mı? Doğru.

Dizilimi tarayın. Gördüğünüz en küçük elemanın izini bellekte tutun. Aynı şey maksimum yada en büyük, k=n için de geçerlidir. Bunlar kolay örneklerdi. Ama sıra istatistiği problemlerinin ilginç bir biçimi median yani **ortanca**yı bulmaktır. Bu da k=(n+1)/2 tabanı yada tavanıdır. Bu elemanların her ikisine de ortanca diyeceğim. Sıralanmamış bir dizilimin ortancasını bulmak oldukça şaşırtıcı, dikkat gerektiren bir işlemdir. Ve bu dersin ana hedefi de bunu yani ortancaları bulabilmektir. Yan ürün olarak rastgele k'nıncı en küçük elemanı bulabileceğiz ama esas amaç ortancayı bulmak olacak. Ve gelecek etütte bunun neden bu kadar yararlı olduğunu göreceksiniz. Bir çok durumda sıralamaya ihtiyaç duymaksızın etkili böl ve fethet uygulamalarını ortancadan yararlanarak gerçekleştirebileceksiniz.

Sonuçta bir çok problemi doğrusal zamanda çözebilirsiniz. Bugün sıra istatistiklerini bulmak için iki algoritma işleyeceğiz. Her ikisi de doğrusal zamanlı. Birincisi rastgele olduğundan sadece beklenen doğrusal zamanı verecek. İkincisi ise en kötü doğrusal zamanı, yine rastgele biçimli bir uygulamayla bulacak. Rastgele bir böl ve fethet algoritması ile başlayalım. Bunun adı rand-select yani **rastgele seçimleme**.

Parametreleri bizim alışık olduğumuzdan biraz fazla. Sıra istatistikleri probleminde size bir A dizilimi veriliyor. Burada ben simgelemi değiştiriyorum ve **i'inci** en küçük elemanı arıyorum, yani aradığım dizin yani indeks **i** oluyor. Ve problemi de birazcık değiştireceğim. Bunu tüm dizilimin içinde aramak yerine, dizilimin belirli bir aralığında, A'nın p ile q elemanı arasında arayacağım.

Buna özyineleme yapmak için gerek duyarız. Bunun bir özyinelemeli algoritma olması lazım çünkü böl ve fethet yordamını kullanacağız. Algoritma burada, oldukça basit bir taban şıkkı ile birlikte. Sonra çabuk sıralama algoritmasının bir biçimini, rastgele çabuk sıralamayı kullanacağız. Bu altyordamı iki ders önce tanımlamamıştık ama kitabı okuduysanız ne yaptığını bilmeniz gerekir. Bu altyordam, A[p…q] dizilimi içinden rastgele bir elemanı seçmeyi, yani p ve q arasında rastgele bir dizi seçmeyi, bunu ilk elemanla değiştirmeyi ve sonra da bölüntü yordamını uygulamayı önerir.

Ve bölüntü, bu ilk elemanı, dizilimin geri kalanını bu rastgele bölüntüden daha küçük veya eşit, yada daha büyük veya eşit olarak bölmek için kullanır. Sadece p ile q arasından rastgele bir bölüntü elemanı seçmek, dizilimi yarıdan bölmektir; ancak iki yarımın boyutu birbirine eşit olmayabilir. Ve o bölüntü elemanının dizinini yani indexini, p ile q arasındaki bir sayıyı alarak geri getirir. Ve biz k'yı bu belirli değer, yani r – p + 1 olarak tanımlayacağız. Bunun sebebi k'nın bu bölüntü elemanının rank'ı haline gelmiş olmasıdır. Bu A[p…q]'nun içindedir. Buraya bir resim çizeyim.

A dizilimimiz var: p ile başlıyor, q ile bitiyor. Başka şeyler de var ama bu özyineleme açısından biz p ile başlayıp q ile bitmesiyle ilgileniyoruz. Rastgele bir bölüntü elemanı seçiyoruz, diyelim ki bunu, ve burada bölüntü oluşturup buna r diyoruz; buradaki her şey A[r]'den küçük yada ona eşit, buradaki her şey de A[r]'den büyük yada ona eşit. A[r] bizim bölüntü elemanımızdır. Bu adımdan sonra dizilim böyle görünür. Ve r'yi elde ederiz. Bölüntü elemanının dizisinin depolandığı yeri elde ederiz. A[r]'ye eşit yada ondan küçük elemanların sayısı, r dahil, r – p + 1'dir. Burada r - p sayıda eleman vardır ve bu elemanı elde etmek için 1 ekleriz.

Eğer 1'den başlayarak sayarsanız, bu rank 1, rank 2 ise, bu elemanın rankı, k'dır. Bu bölüntünün yapılanmasından kaynaklanır. Ve şimdi özyineleme evresine geliriz. Ve ortada üç durum vardır; bu durumlar i'nin k ile ilintisine bağlıdır. Hatırlarsanız, **i** aradığımız rank'dır, k ise bu rastgele bölüntüden çıkacak olan rank'dır. k üzerinde fazla bir kontrolümüz yok ama şanslıysak, i = k olacaktır. İstediğimiz eleman budur.

Bundan sonra bölüntü elemanına geri döneriz. Büyük olasılıkla aradığımız eleman ya sağda yada solda olacaktır. Eğer soldaysa dizilimin sol bölgesinde özyineleme yaparız. Eğer sağdaysa dizilimin sağ bölgesinde özyineleme yaparız. Bu evrede iş oldukça açık.. Sadece dizileri doğru belirlemeliyim. Ya bu durumda olduğu gibi, p ile r - 1 arasındaki bölgede özyineleme yapacağım; çünkü aradığımız sıra A[r] elemanının sırasının solunda. Yada sağ tarafta r+1 ile q arasında özyineleme

yapacağım. Sol bölgede özyineleme yaparsak aradığımız rank değişmez, fakat sağ tarafta özyineleme yaptığımızda aradığımız rank'ın bağıl konumu değişir.

Çünkü burada k elemanlarının bir kısmından vazgeçmiştik. Bu uzunluğun k olduğunu belirtmem gerekirdi. Bir bakıma elemanların k ranklarını süpürüp attık. Ve şimdi bu dizilimin içinde (i –k)'nıncı en küçük elemanı arıyoruz. Özyineleme bu. Yalnızca bir kez özyineliyoruz ve rastgele bölüntü bir özyineleme değildir. Bu sadece doğrusal zaman alır. Burada yaptığımız toplam iş doğrusal zaman artı bir özyineleme olmalıdır. Bundan sonra da toplam koşma süresinin beklenen değerini görmek istiyoruz ama önce küçük bir örnek yapalım; bu algoritmayı iyice anlamak için..

Bu dizilimdeki en küçük yedinci elemanı aradığımızı varsayalım. Ve örnek olarak da kullandığımız pivotun yani esas elemanın ilk eleman olduğunu varsayalım. Yani özel bir durum yok. Rastgele bir dizilim üretmek için epeyce yazı tura atışı yapmam lazım, onun için bunu seçelim. Eğer altıncı elemanda bölüntü yaparsam, bu iki hafta önce işlediğimiz örneğin aynı olur, tekrarlamayacağım ama aynı dizilimi, yani 2, 5, 3, 6, 8, 13, 10 ve 11 elde ederiz. Bölüntü algoritmasını işletirseniz, elemanları bu düzende üretecektir.

Bu bizim r pozisyonumuz. Burada p var; değeri sadece 1. Ve q da en sonda. Ve ben en küçük yedinci elemanı arıyorum. Bu bölüntüyü uyguladığımda 6 dördüncü pozisyona düşer. Bu dizilim sıralanmış olsaydı, (çünkü buradaki elemanların hepsi 6'dan küçük, şuradakilerin hepsi de 6'dan büyük) 6'nın dördüncü pozisyonda olacağı gerçeğini biliyoruz. Böylece burada r, dördüncü kareye geliyor. Öyle mi? 12, 11'e dönüştü. Bu 11 idi, ister inanın ister inanmayın. Daha açık olayım. Af edersiniz. Bazen benim birlerim ikiye benzer. İyi bir özellik değil. Ama bu hatamın üstünü örtmenin güzel bir yolu. Bunu sınavlarda denemeyin. "Ama şuradaki bir aslında ikiydi" demeyin; bunu yapmayın.
Dizilimi sıralıyor olmamamıza karşın, 6 kullanarak bölüntü yaparken sadece doğrusal iş yaparız. Biliyoruz ki dizilimi sıralasaydık 6 buraya gelecekti. Diğer elemanlarla ilgili bir şey bilmiyoruz; onlar sıralanmış değiller ama bölüntü özelliklerinden 6'nın doğru yere geldiğini biliyoruz. Şimdi 6'nın rankı'nın 4 olduğunu biliyoruz. Biz 7 'yi arıyorduk ama 4 sayısını elde ettik.

Burada bir şeye ihtiyacımız var. Tahmin ediyorum ki biz 10'u arıyoruz. Hayır, 11'i. Bu dizilimde 8 eleman olması lazım, bu nedenle en büyüğün yanındaki olmalı. Burada en büyük değer 13; şimdi hile yapıyorum. Aradığımız yanıt 11. Aradığımız sayının sağ tarafta olduğunu biliyoruz aradığımız rank 7, ve 7 4'ten büyük. Şimdi burada hangi rank'ı arıyoruz? Aslında buradaki 4 elemanı değerlendirme dışı bıraktık.

Bu örnekte k da 4 çıktı çünkü **p** 1'e eşitti. 6'nın rank'ı, 4 idi. Bu dört elemanı attık. Şimdi rank 7 eksi 4'e bakıyoruz ve 3 buluyoruz. Gerçekten de üçüncü rank'daki eleman burada hala 11. Böylece bunu özyinelemeyle bulursunuz. Cevabınız budur. Şimdi algoritmanın anlaşılmış olduğunu sanıyorum. Dikkat edilmesi gereken kısmı

çözümleme faslı. Ve buradaki çözümleme biraz rastgele çabuk sıralamanınkine benziyor ama onun kadar karmaşık olmadığından daha hızlı çalışılabilir. Öte yandan oldukça karmaşık olan rastgele çabuk sıralamanın çözümlemesinin bir tekrarı gibi olduğundan bize ek yarar sağlayacak. Bu algoritmanın beklenen yürütüm süresine bakmak için daha önce kullandığımız çerçeveyi takip edeceğiz.

Ve başlangıçta kendimizi iyi hissetmek adına, önceki gibi biraz sezgilerimize güveneceğiz. Arada kendimizi kötü de hissedeceğiz; göreceksiniz. Biri iyi biri de kötü olan iki uç durumu düşünelim. Ve bugünkü çözümlemelerin hepsinde elemanların birbirinden farklı olacağını belirtmeliyim. Elemanlar benzer olduğunda iş bayağı karmaşıklaşır. Bu durumlarda algoritmaları biraz değiştirmek zorunda kalabilirsiniz çünkü tüm elemanların aynı olması durumunda rastgele bir elemanı seçtiğinizde bölüntü iyi çalışmaz. Ama varsayalım ki hepsi birbirlerinden farklı olsun ve ilginç bir durum ortaya çıksın. Oldukça şanslı bir durum olsun..

En iyi durumlarda tam ortadan bölüntü yaparız demek istiyorum. Bu durumlarda bölüntümüzün solundaki elemanların sayısı sağındaki elemanların sayısına eşittir. Ama 1/10 oranından 9/10 oranına kadar olan bölüntüler de bir o kadar iyi olur. Sabit kesirlerin olduğu durumların iyi olacağını hissederiz; her sabit kesir 1/2 kadar iyidir. Bu durumda elde ettiğimiz yineleme en çok bu kadar kötüdür. Yani bu duruma göre değişir. Örneğin bölüntü yaptığımızda solda 1/10 sağda 9/10 olursa, yanıtımızın hangi bölgede olduğuna bağlıdır. Eğer i gerçekten küçükse yanıt 1/10'nun içindedir. Eğer i gerçekten büyükse, 9/10'un içindedir, yada çoğunlukla 9/10'nun içinde olacaktır. Biz şanslı durumda en kötü durum çözümlemesi yaptığımızdan, üst sınır bulunca mutlu oluruz; bu nedenle T(n) en çok T(9/10n) + Θ(n) olur diyeceğim.

Bölüntünün büyük parçasında olmanın daha kötü olduğu açık. Bu yinelemenin çözümü nedir? Yineleme çözmek çok gerilerde kalmıştı, değil mi? Bu yinelemeyi çözmek için hangi metodu kullanmalıyım? Ana metodu kullanmalıyım. Hangi durumunu kullanmalıyım? Üç, güzel, hala anımsıyorsunuz. $n^{log_{b(a)}}$ durumuna bakıyoruz. Burada b 10/9, ama bu çok fark ettirmeyecek çünkü a 1. 1'in logaritması 0'dır. Bu nedenle bu n^0, yani 1. Ve n polinomsal olarak 1'den büyüktür. Bu O(n) olur; ve bu da iyidir. Bizim istediğimiz de bu, yani doğrusal zaman. Şanslı durumdaysak bu çok iyi. Ne yazık ki bu sadece bir öngörü ve her zaman şanslı durumu elde edemeyeceğiz.

Rastgele çabuk sıralamada yaptığımız çözümlemenin benzerini yapabiliriz. Şanslı ve şanssız durumları sırasıyla ele alırsak yine de şanslı oluruz ama işlerin ne kadar kötüleşebileceğini görmek için sadece şanssız durumu konuşalım. Ve bu gerçekten en kötü durum çözümlemesi olur. Şanssız durumda 0: n-1 bölüntüsünü elde ederiz. Çünkü iki taraftan da bölüntü elemanını çıkarırız ve bölüntü elemanından daha küçük bir şey kalmaz. Sol taraf 0 olur ve sağ taraf da n-1. Bu durumda T(n) = T(n-1) artı

doğrusal maliyet gibi bir yineleme elde ederiz. Bu yinelemenin çözümü nedir? n^2. Evet. Bunu bilmeniz gerekirdi. Çözüm n^2'dir çünkü bu bir aritmetik seridir.

Ve bu oldukça kötüdür. Bu sıralama yapıp sonra da **i**'ninci elemanı seçmekten çok daha kötüdür. En kötü durumda bu algoritma gerçekte kötüdür ama çoğu durumlarda iyi sonuç verecektir. Gerçekten çok çok şanssız değilseniz ve her attığınız yazı tura yanlış cevabı vermiyorsa bu duruma düşmezsiniz ve daha şanslı bir durumla karşılaşırsınız. En azından şimdi kanıtlamak istediğim şey bu.

Burada beklenen yürütüm süresinin doğrusal zamanda olacağını kanıtlayacağız. Yani karesel bir sonuç elde edilmesi çok nadir bir durumdur. Ama ileride en kötü durumun da nasıl doğrusal yapılabileceğini göreceğiz. Bu sorunu bütünüyle çözecektir. Haydi, çözümleme işine başlayalım. Şimdi, eskiden buna çok benzeyen bir çözümlemeyi görmüştünüz. Beklenen zamanı çözümlemek için ne yapmamızı önerirsiniz? Bu bir böl ve fethet algoritması olacak ve bu nedenle yinelemeyi koşma süresini andıracak bir şekilde yazmak istiyoruz.

Yanıtı istemiyorum, bu algoritmanın beklenen yürütüm süresini çözümlemek için ilk adımımız ne olmalı? Duyamadım? Değişik durumlara bakmak, evet. Doğru. Rastgele bölüntüyü yapılabileceğimiz birçok durum var. 0 ve n-1 şeklinde bölünebilir. Ortadan bölünebilir. Bölünebileceği n sayıda durum var. Bu durumlara nasıl bölebiliriz? Indicator random variables yada **göstergesel rastgele değişkenler** kullanarak. Çok doğru. Yapmak istediğimiz işte bu. Göstergesel rastgele değişken, uğraştığımız şeyin sadece bir T(n) fonksiyonu olmadığından, aynı zamanda rastgele bir değişken olma açılımını getirir. Bu inceliklerinden biridir. T(n) rastgele seçimlere bağlıdır, bu nedenle gerçekte bir rastgele değişkendir.

Bu yaklaşımla, T(n) için bir yineleme elde ederken göstergesel rastgele değişkenleri kullanacağız. Yani, T(n) giriş boyutu n olan bir rand-select yada rastgele-seçim uygulamasının koşma süresidir. Buraya rastgele sayılarla ilgili bir varsayımı da açıkça yazacağım. Rastgele bir bölüntüyü uyguladığım her seferde, o uygulamanın diğer tekrarlarından tamamen bağımsız bir rastgele sayı üretir. Bu çözümlemenin işlevsel olması için bu varsayım önemlidir. Neden olduğunu birazdan göreceğiz. Ve şimdi, sizin de önerdiğiniz gibi T(n) için bir denklem yazmak amacıyla göstergesel rastgele değişkenlerimizi tanımlayacağız.

Bunlara X_k diyeceğiz. Burada k = 0… n-1 olacak. Göstergesel rastgele değişken 1 yada 0 olacak. k bölüntünün sol tarafında olursa k =1 değerini alacak. Bölüntünün k:n-k-1 oranında bölünmesi hali dışında k = 0 olacak. 0 ile (n-1) arasında bu göstergesel rastgele değişkenlerden n adet var. Ve her durumda rastgele seçim ne olursa olsun bunlardan sadece biri 1 değerinde olur. Diğerlerinin tümü 0 olacaktır. Şimdi içinde bulunduğumuz duruma göre algoritmanın koşma süresini bölümleyebiliriz.

Bu yaptığımız sezgisel çalışmayı tüm durumlar için bütünleyecektir. Bundan sonra beklenene bakabiliriz. T(n)'yi durumlara göre bölersek, bunun gibi bir üst sınır elde ederiz. Eğer 0 : (n-1) bölüntüsü varsa, en kötü n-1 elde ederiz. Bu durumda n-1 boyutlu bir problemde yineleme yaparız. Aslında 0 boyutlu bir problemde yineleme yapmak oldukça zordur. Eğer 1: (n-2) bölünmesi varsa, iki değerin büyüğünü alırız. Bu bize mutlaka bir üst sınır verecektir.

Ve en altta n-1: 0 bölünmesini elde edersiniz. Bu belirsiz bazı olaylara bağımlı gibi görünse de göstergesel rastgele değişkenlerimiz bu olayların ne zaman olacağını bize söyler. Bu değerlerin her birini göstergesel rastgele değişken ile çarpabiliriz ve durum uygun değilse sonuç 0 çıkar ve bölüntü uygunsa 1 çıkar ve bize bu değeri verir. Bütün bunları toplarsak aynı şeyi elde ederiz. Sonuç, bir tarafta göstergesel rastgele değişkenlerin tüm k değerlerin toplamı çarpı o durumlardaki maliyet yani T(max.{ k, n-k-1 }, artı Θ(n) olur.

Bir bakıma bu yürütüm süresini temsil eden rastgele değişken için yinelememizdir. Şimdi değer hangi durumda olduğumuza bağlı olacaktır. Biz tüm bu olayların oluşma olasılıklarının aynı olduğunu, bölüntü elemanını tekbiçimli rastgele seçtiğimizden dolayı biliyoruz; ama bekleneni hesaplamadan bu noktadan daha öte bir basitleştirmeye gidemeyiz. Bu rastgele değişkenin n^2 büyüklüğünde olabileceğini biliyoruz. Umarız ki genellikle doğrusal olsun. Her iki tarafta da beklenenleri hesaplayıp istediğimizi elde edeceğiz.

Bu rastgele değişkenin beklenen değerine bakalım; bu sadece beklenendir ve bu tahtada çalışmak için buradaki toplamayı şuraya kopyalayacağım. Bu toplamanın beklenen değerini hesaplamak istiyorum. Beklenenin hangi özelliğini kullanmam gerekir? Doğrusallık, güzel. Toplamayı dışarıya alabiliriz. Şimdi elimde beklenenlerin bir toplamı var. Her beklenene bağımsız bakalım. İki bağımsız değişkenin çarpımları gibidir. Bu bir göstergesel rastgele değişken ve bu da daha karmaşık bir fonksiyon, bir koşma süresini temsil eden karmaşık bir rastgele değişken; bu değişken de yineleme işinde yaptığımız rastgele seçimlere bağımlıdır.

Şimdi ne yapmalıyım? Elimde iki rastgele değişkenin çarpımının beklenen değeri var. Bağımsızlık, doğru. Eğer bu iki rastgele değişkenin bağımsız olduklarını bilirsem, çarpımlarının beklenen değerinin beklenenlerinin çarpımına eşit olduğunu da bilirim. Şimdi bunların bağımsız olup olmadıklarını kontrol etmeliyiz. Öyle olduğunu umuyorum çünkü öyle değilse yapabileceğim başka şey kalmaz. Bunlar neden bağımsız? Anlamadım? Çünkü öyle olduklarını başta belirttik, değil mi? Bu varsayım nedeniyle tüm rastgele değişkenlerin bağımsız seçileceğini varsaydık. Burada bunu bir ara değerlendirmeyle hesaba katmamız gerekiyor. Bu X_k'lar, X_0'dan başlayıp X_{n-1}'e kadar olan toplamadaki bütün k 'lar, rastgele bölüntü için uygulanan bağımsız rastgele seçimlere bağımlıdır.

Bunların hepsi birbirleriyle ilintilidir çünkü biri 1 olursa, diğerleri 0 olmak zorundadır. Yani X_k'lar arasında sıkı ilintiler var. Ama burada olanlar arasında tek rastgele kısım şu T(max(k, n-k-1)) dir. Dolayısıyla bu rastgele değişken şuradaki diğerlerinden bağımsızdır. Aynı çabuk sıralamada olduğu gibi ama birkaç ders önce bazılarınızın bu konuda kafa karışıklığı yaşadığını gözlemlediğimden bunu tekrar ediyorum. Beklenenlerin çarpımını, yani E[X_k] E[T(max(k, n-k-1))]'i elde ederiz.

n sabiti dışarıya alınabilir ama şimdilik içeride bırakalım. Bunda n düzeyinde hesap yapılacak bir beklenen yok. n sabiti n sabitidir. X_k'nın bekleneni nedir? 1/n, çünkü hepsi eşit olasılıklı olarak seçilmişti. Bunlardan n adet olduğundan umulan değer 1/n olur. Değer ya 1 yada 0'dır. Artık bunu parçalamaya başlayabiliriz. Elimizde 1/n çarpı bazı yinelemeli T çağrılarının beklenen değeri var, ardından, artı 1/n kere n sabiti (bu bir sabittir) var; ama her şey n kere toplandığından bunu burada açalım.

Elimde k =0'dan n-1'e kadar bir toplam var. Sanıyorum ki 1/n dışarıya alınabilir. Ve [T(maksimum(k, n-k-1))] 'e kadar umulanı kalır. Burada bir sürü karmaşık parantez var. Öteki tarafta buna ilaveten k = 0'dan n-1'e kadar olan toplam var. Bunu bir kez daha yazayım. Başta 1/n var ve içerde de Θ (n) var. Bu toplam n^2. Sonra n'ye böldüğümüzde tüm yapı yine n düzeyine gelir. Burada çok ilginç bir şey olmadı. Bu gerçekte n düzeyinin bekleneni, n düzeyindedir demektir. n düzeyinin ortalama değeri de n düzeyindedir.

İlginç olan şey bu kısımda. Şimdi bu toplam ile ne yapabiliriz? Burada rastgele çabuk sıralama ile yollarımız ayrılmaya başlıyor çünkü ortada bu maksimum var. Rastgele çabuk sıralamada elimizde T(k) ile T(n-k-1)'in toplamı olurdu çünkü iki özyineleme yapardık. Burada sadece en büyüğünün özyinelemesini yapıyoruz. Maksimum bu yinelemenin hesaplanmasında büyük zorluk çıkarıyor. Maksimumdan nasıl kurtulabilirim? Çözüm için düşünmenin bir yolu var mı? Evet?

Kesinlikle. Yarı yola kadar toplama yapıp bulduğum değeri ikiye katlarım. Diğer bir deyişle buradaki değerler iki kez tekrarlanıyor. k = 0 olduğunda da k = n-1 olduğunda da aynı T(n-1)'i elde ediyorum. k = 1 yada n-2 aynı sonucu veriyor; 2 ve n-3 de öyle. Pratikte uygulayacağım yöntem yarı yoldan sonra toplama yapmak. Bu biraz daha kolay. Ve dizinleri doğru saptamalıyım. n/2'den n-1'e kadar olan taban, güvenli olacaktır. Sonra elde sadece E[T(k)] olur ama bunu 2 ile çarpmayı unuttum, bu nedenle şunu 1'den 2'ye değiştireceğim.

Ve n düzeyi korundu. Bunun nedeni her terimin iki kez görünmesi. Bunları dışarıya faktörleyip çıkarabilirim. Ve eğer n tek sayı çıkarsa bir şeyleri mükerrer sayıyorum demektir ama en çok da bu değeri alabilir. Bu güvenli bir üst sınırdır. Üst sınırlardan başka bir şeyle ilgilenmiyoruz çünkü doğrusal olmayı umuyoruz. Ve bu algoritmanın

koşma süresi kesinlikle en azından doğrusal; bu nedenle doğrusal bir üst sınıra ihtiyacımız var.

Evet, bu bir yinelemedir. E[T(n)] en çok 2/n çarpı E[T(k)] içindeki 0'dan n'ye kadar olan sayıların yarısının toplamları + Θ (n) olabilir. Biraz karmaşık bir yineleme oldu ama biz bunu çözmek istiyoruz. Aslında çözümü rastgele çabuk sıralama yinelemesinden biraz daha kolaydır. Bunu çözeceğiz. Hangi metodu kullanmalıyız? Af edersiniz? Ana metot mu? Ana metot iyi olurdu ama iki yineleme uygulamasından her birinde değişik k değerleri olur.

Ana metot sadece uygulamalardan ikisinin de değerleri ve boyutları aynı ise çalışır. Ana metodu kullanabilsek çok iyi olurdu. Başka ne var? Yerine koyma. Sorun zorsa, şüpheniz varsa yerine koyma metodunu kullanın. Burada iyi olan şey ne istediğimizi biliyor olmamız. En azından sezgisel olarak sonucun doğrusal zamanlı olacağını hissediyoruz. Yani neyi kanıtlamak istediğimizi biliyoruz.

Ve bunu direkt olarak yerine koyma metodunu kullanarak kanıtlayabiliriz. Burada 0'dan büyük bir c sabiti olması durumunda, bu yineleme çerçevesinde E[T(n)]'nin en çok c çarpı n olabileceği savını öneriyorum. Haydi, bunu kanıtlayalım. Tahmin ettiğimiz gibi kanıtlama yerine koyma ile yapılacak. Bunun anlamı, bu eşitsizliğin tüm küçük n değerleri için geçerli olacağı; buraya n'den küçük yazacağım ve bunu tümevarımla yapacağız.. Ve bunu n için kanıtlayacağız. E[T(n)] 'i elde ederiz.

Şimdi elimizdeki yinelemeyi açacağız. En çok bu olabilir. Bunu kopyalayacağım. Ve sonra her özyinelemeli tekrarda k değerleri kesinlikle n'den küçük olacak. Af edersiniz, yanlış kopyaladım; 0'ın değil, n/2'nin tabanı olacaktı. Böylece tümevarım hipotezini bunların her birine uygulayabilirim. Bu değer hipoteze göre en çok c çarpı k olabilir ve bu eşitsizliği elde ederim.

c toplam işaretinin dışına alınabilir çünkü bir sabittir. Bunu tekrar yazarken biraz dikkatli olacağım çünkü ilgilendiğim şey burada kalacak toplam. Bu babadan kalma bir toplama işi. Eğer eskiden öğrendiğiniz toplama numaralarını hatırlarsanız bunu hesaplayabilirsiniz. Eğer 0'dan başlayıp n-1'e kadar gidersek bu bir aritmetik seridir, ama burada aritmetik serinin kuyruk yani arka ucu var. Ve siz en azından Theta'ya kadar olan değerler için bunun ne olduğunu bilmelisiniz, değil mi? n^2, evet. Kesinlikle Θ(n^2) 'dir. Ama burada biraz daha iyi bir üst sınıra ihtiyacımız var ve göreceğimiz gibi sabitler etkili olacak.

Kullanacağımız varsayım bu toplamın en çok 3/8 çarpı n^2olacağıdır. Bu kritik bir olgu çünkü bildiğim kadarıyla 3/8 yarımdan daha küçüktür. Böylece bu 2'den kurtulacağız. Ben bunu kanıtlamayacağım; bu bir egzersiz olacak. Bunun doğru olduğunu bilirseniz çözüm kolaydır çünkü tümevarım yöntemiyle kanıtlarsınız. Bu sonucu bulmak biraz uğraştırır ama çok da fazla bir uğraş gerekmez.

Bunu tümevarımla kanıtlamalısınız. Şimdi şunu basitleştireyim. Bu biraz karmaşık ama istediğim c kere n. Bunu istediğimiz değer eksi kalan şeklinde yazalım. Ve bazı saçma kesirlerle uğraşalım. Bu 2 kere 3 yani 6 bölü 8 yani 3/4, değil mi? Burada elimizde 1 var yani 1/4'ü çıkararak 3/4 elde etmemiz gerekiyor. Ve tahminim, bu da 1/4 çarpı c çarpı n. Ve sonra iki kez eksili Θ(n) var ki artı Θ(n) eder. Buraya kadar yeterince açık. Bunu yeniden yazıyorum. Böylece şurada istediğimizi elde ettik. Ve umuyoruz ki bu değer eksi değil, çünkü istediğimiz bunun c çarpı n'ye eşit yada ondan küçük olması.

Bunun doğru olması için de şunun negatif olmaması gerekir. Durum iyi görünüyor çünkü c'yi istediğimiz büyüklükte seçme özgürlüğümüz var. Bu Θ simgelemine gömebileceğimiz herhangi bir sabit bu yinelemeyi geçerli kılacak şekilde seçilebilir. Biz c'yi o sabitten 4 kez büyük seçersek bu değer negatif yada eksi olamaz. Böylece c'yi Θ sabitini küçük gösterecek büyüklükte seçersek eşitsizliği geçerli kılabiliriz.

Bu aynı zamanda taban durumudur. Burada belirtmek istediğim konu c'yi aceleyle, tezimizi geçerli kılmak için yeterince büyük seçtik ama, seçim n'nin taban durumunda, yani n en fazla bir sabit olduğunda etkili olacak. Burada bu 1 civarında, çünkü özyinelemeli işlem yapmıyoruz. Elde ettiğimiz şey bu algoritmanın rastgele seçimli n düzeyinde, yani Θ (n) 'lik koşma süresi olduğu. Rahatsız edici husus, en kötü durumda, gerçekten çok şanssızsanız yürütüm süresinin n^2olabileceği. Devam etmeden önce sorular var mı? Böylece Θ (n) boyutunda bir beknenen zaman olduğunu kanıtlama işinin sonuna geldik.

En kötü zamanın n^2olduğunu görmüştük. Her şey anlaşıldı mı? İyi. Bu kanıtları bir kez daha gözden geçirmelisiniz. Bunlar rastgele çabuk sıralama ve rastgele seçimle içsel olarak ilintilidir. Bunları ezbere bilin. Bu çok iyi bir algoritmadır çünkü pratikte bölüntüleme işlemini çoğunlukla ortada, yani 1/4 ile 3/4 arasında bir yerde yaparsınız ve iyi sonuç elde edersiniz. En kötü durum için n^2'yi bulmanız çok nadirdir. Bu olasılık 1 bölü n^n kadar küçüktür. Ama ben teorisyenim ve en kötü durum olarak Θ(n) bulmanız çok hoş olur. Bu umabileceğiniz en temiz sonuçtur çünkü en iyi değerdir. Θ(n) 'den daha iyisini yapamazsınız. Elemanlara bakmanız gerekir. Şunu sorabilirsiniz: Bu en kötü durum davranışından kurtulup, bir şekilde rastgele sıralamadan kaçınarak Θ (n)'lik en kötü koşma süresini garanti edebilir miyiz?

Bunu yapabilirsiniz ama hiç de kolay olmayan bir algoritma kullanmanız gerekir. Ve bu bugüne kadar gördüklerimiz arasında en karmaşık olanlardandır. Gelecekte göreceklerimiz kadar karmaşık değil ama, işte burada. **En kötü doğrusal zaman sıra istatistikleri**. Bu algoritma Blum, Floyd, Pratt, Rivest ve Tarjan gibi ünlüler tarafından geliştirilmiş. İsimleri B, R ve T ile başlayanları tanıdım; hayır Pratt ile de

tanışmıştım. Yazarların hemen hepsiyle yakınlaşıyorum. Bu, biraz eski ama zamanında çığır açan, bugün de hala hayranlık uyandıran bir algoritmadır. Ron Rivest burada bir profesör. Bir süre önce profesörlük sınavlarımdan birinin kapağında bir şaka sorusu vardı.

Soruda, en kötü doğrusal zaman sırası istatistiklerinin yazarlarından hangisinin en zengin olduğu soruluyordu. Ne yazık ki not değeri olan bir soru değildi ama eğlendiriciydi. Burada yanıtı söylemeyeceğim çünkü kayıt yapılıyor ama bu konuyu düşünün. Cevabı o kadar açık olmayabilir. Bu yazarlardan bazıları zengindir. Soru en zengin olanın hangisi olduğuydu. Her neyse zengin olmadan önce bu algoritmayı kuramladılar. İster inanın ister inanmayın, o günden sonra, zengin olduktan sonra bile birçok algoritma tasarladılar. İstediğimiz iyi bir pivot, yani esas eleman, garantili bir pivot. Rastgele bir pivot iyi olacak. Bu nedenle bu konuda en basit algoritma rastgele bir pivot seçendir. Yüksek olasılıkla iyi seçim yapmalıdır. Biz iyi bir pivotun seçimini deterministic yani belirlenimci olarak zorlamak istiyoruz. Ve buradaki yeni fikir pivot üretiminin yinelemeyle gerçekleştirilmesidir.

Yinelemeden başka ne yapabiliriz? Önceki yinelemelerden bildiğiniz gibi, problemi ikiye bölüp iki yineleme uygulaması yaptığımızda, birleştirme sıralamasındaki doğrusal ek işi elde ederiz. Yani, $T(n) = 2[T(n/2) + \Theta(n)]$'yi. Bunu uykunuzda bile tekrarlamalısınız. Bu n lgn'dir. Bu nedenle yarı boyutlu iki problemle yineleme yapamayız. Daha iyisini yapmamız gerekir. Bir şekilde bu yinelemelerin toplamı kesinlikle n'den az olmalıdır. Bu algoritmadaki sihir de burada. Bu nedenle buna rastgele seçim yerine seçim diyeceğiz. Aslında dizilime bağlıdır ama, ben seçmek istediğimiz **i**'ninci elemana ve içinde seçim yapmayı istediğimiz dizilimin boyutuna odaklanacağım. Ve bu algoritmayı rastgele seçim şıkkındakinden daha az kuralla yazacağım çünkü bu biraz daha üst düzey bir algoritmadır.

Şuraya algoritmanın bir resmini çizeyim. Birinci adım en garip uygulamalardan aynı zamanda da kilit fikirlerden biridir. Elemanlara bakarsınız ve, bunlar belirli bir düzen içinde değildir; bunları bir çizgi üzerinde göstermek yerine 5'e n/5 boyutlu bir ızgaraya yerleştirirsiniz. Neden olmasın? Bunu çizmek biraz uzun sürecek ama siz de çizerken aynı sürede yapacağınızdan zamanı rahat kullanacağım. Eninin ne olacağı önemli değil ama en n/5 olacağından bunu göz önüne alın. En n/5 olacak ama boy kesinlikle 5 olmalıdır. Galiba doğru çizdim. Bu kadar yükseğe sayabiliyorum. Burada 5 var. Ve bunun da… ama sayımız beşe bölünür olmayabilir ve son adımda belirsiz bir şey kalabilir. Ama istediğim şey bu parçaların n/5'in tabanı olması..Böylece burada en çok 4 eleman kalabilir.

Bu nedenle bunları görmezden geleceğim. Fazla etkileri yok. Yalnız bir toplanır sabit olurlar. Dizilimim burada. Bunu gülünç bir yoldan yazdım. Bu dikey değerlere öbek adını vereceğim. Bunları daire içine alacağım; bunu notlarımda yaptım ama daire içine almaya başladığınızda işler bayağı karışacaktır. Sizi uyarıyorum; bu şekil çok

dolu olacaktır. İşin sonunda neredeyse anlaşılmaz olacak ama bu kaçınılmaz bir durumdur.

Eğer gerçekten sıkılacak olursanız şekli birkaç kez çizebilirsiniz. Nasıl büyüdüğünü de çizmelisiniz. İşte bunlar öbekler, beşli dikey öbekler.

Sonraki adım, ikinci adım yinelemektir. Burada işler biraz sıra dışı, daha da sıra dışıdır. Aa, af edersiniz. Bir ile iki arasına, bir çizgi çizmem gerekirdi, bu nedenle bunu aşağı kaydırıp buraya yerleştirmeliyim. Bu arada birinci adımda her grubun ortancasını da bulmak istiyorum. Yapmak istediğim şey, bu şekle bakıp her öbekteki beş elemanın ortalarındaki eleman ortanca olacak şekilde yeniden yapılanmalarını hayal etmek. Bu nedenle bunlara her bir grubun ortancaları diyeceğim. Beş elemanım var ve bu yüzden ortanca tam ortalarındadır. Buradaki iki eleman ortancadan küçük, iki eleman da ortancadan büyüktür. Her elemanın birbirinden farklı olduğunu varsaymıştık. İşte buradalar ve ben de onları hesaplarım. Bu ne kadar zamanımı alır? n bölü beş öbek, her birinde beş eleman var; her birinin ortancasını hesaplamak ne kadar zaman alır?

Duymadım? Evet, 2 kere n/5. Bu Θ(n) ‘dir ve bütün bilmek istediğim de budur. Yani, karşılaştırmaları sayıyorsunuz ve bu iyidir. Bu kesinlikle Θ (n) ‘dir. Burada her grubun içinde sabit sayıda karşılaştırma yapmak zorunluluğum var çünkü sabit sayıda eleman var. Bu nedenle fark etmez. Buna göre rastgele seçim bile kullanabilirsiniz. Ne yaparsanız yapın, bu iş sabit sayıda karşılaştırma gerektirir. Bir karşılaştırmayı birden fazla kez yapmadığınız sürece..

Böylece bu kısım kolaydır. Beş sayıyı sıralayıp üçüncüsüne bakabilirsiniz, çok fark etmez çünkü sadece beş sayı var. Bu kurnazca bir fikirdir. Elimizde zaten öbekte ortaya yakın yerleşmiş bazı elemanlar vardır. Şu ana kadar da sadece doğrusal iş yaptık; bu nedenle buraya kadar iyi gittik. Şimdi daha önce yazmaya başladığım ikinci adıma geçeceğiz ve yinelemeye başlayacağız.

Sonraki düşünce elimizde n tabanında n/5 tane ortanca oluşmasından çıkar. Bu ortancaların ortancasını hesaplayacağım. Bunları yeniden düzenlediğimi hayal ediyorum. Ve maalesef çift sayıyla karşılaşıyorum, bunlar altı adet; ama yeni bir düzenlemeyle ikinci bir kare çiziyorum ve buradakini ortanca olarak seçiyorum. Böylece bu ikisi ortancamdan kesin küçük, bu üçü de kesinlikle büyük oluyor. Şimdi görünüşte bu bana buralardaki diğer elemanlar konusunda hiçbir şey anlatmıyor.

Buna döneceğiz. Aslında o elemanlar hakkında bir şeyler açıklıyor. Ama şimdilik bu, şu elemanların ortancası durumundadır. Bu elemanların her biri de beş elemanın ortancasıdır. Şimdilik bütün bildiğimiz bu. Bunu yinelemeli olarak yaparsak T(n/5) zamanını alacaktır. Şu ana kadar iyi gidiyoruz. Boyutu n/5 olan ve doğrusal iş

gerektiren bir problemi yineleme ile çözmeye katlanabiliriz; doğrusal zaman aldığını biliriz. Ama bundan fazlası var. Henüz işimiz bitmedi.

Sonraki adımda **x** bölüntü elemanımız olacak. Orada bölüntü yapacağız. Algoritmanın sonraki süreci rastgele bölüntüye çok benzediğinden, k'yı x'in rank'ı olarak tanımlayacağız. Ve bu yapılabilir ve n – r + 1 gibi bir şey olur, ama bunun nasıl yapılacağını yazmayacağım, çünkü burada daha üst düzeyde çalışıyoruz. Ama yapılabilir. Ve bundan sonra üç yollu dallanma gelir. Eğer i k'ya eşit çıkarsa mutlu oluruz. Pivot elemanı aradığımız elemandır ama daha büyük olasılıkla i k'dan daha küçüktür veya k'dan daha büyüktür. Sonra uygun yineleme uygulamasına geçeriz ve i'nin en küçük elemanını yinelemeli olarak seçeriz…

… dizilimin alt bölgesinde.. Bölüntü elemanının solunda.. Aksi takdirde i – k'nıncı en küçük elemanı dizilimin üst bölgesinde yinelemeli seçeriz. Bunu üst düzeyde yazıyorum çünkü daha önceden görmüştük. Tüm bunlar rastgele seçim uygulamasının son adımlarının aynısıdır. Algoritma budur. Önemli soru: Neden çalışır? Neden doğrusal zamanlıdır? İlk soru: Yinelemesi nedir? Bu evrede bunu yazamıyoruz çünkü özyinelemeli alt problemlerin hangi boyutlarda olabileceğini bilmiyoruz.

Önceden olduğu gibi ya alt bölgede ya da üst bölgede özyineleme yapacağız. Eğer şanssızsak ve ortaya 0'dan n-1'e gibi bir bölüntüleme olursa, bu karesel zamanlı bir algoritma olur. İddiamız bu bölüntü elemanının iyi olacağının garantili olduğudur. Bu işlemin koşma süresi bir şeylerin T'si çarpı n olacaktır ama o bir şeylerin ne olduğunu bilmiyoruz. Ne kadar büyük olabilir? Aslında bunu size sorabilirim.

Ama burada biraz dolaylı yoldan gidiyoruz onun için size söyleyeceğim. Şu anda elimizde T(n/5)'in yinelemesi var. Bu önce bir sabit olmalı, sonra o sabitle n'nin çarpılması gerekmeli ve sabitin 4/5'den kesin küçük olması gerekmektedir. Eğer 4/5'e eşitse o durumda problemi yeterince bölemez ve n lgn koşma süresini elde edemezsiniz. Eğer 4/5'den küçükse bu durumda problemi bir sabit faktör oranına indirgeyebilirsiniz.

Yinelemeli alt problemlerin hepsini, n/5 ve bir şeyler çarpı n'yi toplarsanız, bir kere n'den kesinlikle daha küçük bir sabit elde edersiniz. Bu işi geometrik olmaya zorlar. Eğer geometrik ise sonunda doğrusal zaman elde edersiniz. Bu sezgisel bir yaklaşımdır ama doğru bir öngörüdür. Doğrusal zamanlı sonuçlar elde etmeyi hedeflediğinizde bunu aklınızdan çıkarmayın. Böl ve fethet işlemi yapıyorsanız, toplam alt problem boyutunu bir çarpı n'den daha küçük bir sabit olarak elde etmelisiniz. O zaman sonuç alırsınız.

Evet bu sabiti burada hesaplamalıyız. Bunun için de şu ana kadar şaşılacak ölçüde karışık olmayan bu şekli kullanacağız. Şimdi onu karmakarışık edeceğiz. Yapmak istediğim şey, bunlara ister iki eleman, ister nokta, ister tepe noktası deyin, aralarına

bir ok çizmektir. Bunlara a ve b diyelim. Okun ucunu daha büyük olan değere yönlendirmek istiyorum, yani burada a, b'den küçük anlamına gelir. Bu yalnızca bu şekle ilişkin simgelemdir. Şimdi bu elemanla ilgili bildiklerimi yazacağım. Bu eleman şu 5 elemanın ortancasıdır. Çizimi yaparken bu elemanların ortancadan daha büyük olduğunu, şu elemanların da ortancadan daha küçük olduklarını varsayacağım. Bu nedenle oklarım bu şekilde olacak. Burada keşke renkli tebeşirim olsaydı diyorum.

Su sadece bu elemanın şu elemanların ortasında olduğunu belirtmektir. Bunu her sütunda biliyorum. Bu noktadan sonra şekil karmaşık olmaya başlar. Daha işim bitmedi. Şimdi bu elemanın da ortancaların ortancası olduğunu biliyoruz. Bu ortadaki kare içinde gösterilenlerin hepsinin ortancasıdır. Ve bunları ortancadan küçük olacak şekilde, bunları da ortancadan büyük olacak şekilde çizeceğim. Çizeceğim, çünkü algoritma bunu yapamaz; her şeyin nasıl işlediğini bilmek zorunda değil. Belki bilebilir ama burada yaptığımız yalnızca çözümleme amaçlıdır. Bu elemanın şundan büyük olduğunu biliyoruz, şundan da büyük olduğunu biliyoruz. Ama diğer elemanlarla ilgili doğrudan bir bilgimiz yok.

Bildiğimiz tek şey bu elemanın şu ikisinden büyük olduğu ve bunun da şunlardan küçük olduğudur. Şimdi bu söylediklerim elde ettiğimiz şekil kadar karışık. Şimdi daha küçük olma durumunun güzel özelliği bunun transitive yani geçişli bir ilişki olmasıdır. Bu grafikte yönlü bir yolum varsa, bu elemanın şu elemandan kesinlikle daha küçük olduğunu bilirim çünkü bu şundan küçüktür ve bu da şundan küçüktür. Bunun doğru olduğunu her ne kadar sadece bir sütunda, bu orta sütunda biliyor olsam da, gerçekte x olarak anılan bu elemanın bütün bu elemanlardan daha büyük olduğunu da bilirim; çünkü o, bu oklara göre, bundan, şundan ve şu elemanların her birinden daha büyüktür. Burada dikdörtgenler çizeceğim; siz bunu çizmek zorunda değilsiniz ama geri planı daha da karıştıracağım. Şu dikdörtgenin içindeki tüm elemanlar bundan daha büyük yada buna eşit, bu dikdörtgenin içindeki tüm elemanlar da x'den küçük yada x'e eşit. Şimdi bunlardan kaç tane var? Aslında, bu yaklaşık öbek setlerinin ortasında ve bu da bu sütunların 3/5'idir.

Böylece elde ettiğimiz en azından: Elimizde n/5 öbek var ve bu öbeklerin yaklaşık yarısı burada; bu nedenle şuna n/2'nin tabanı diyelim. Her grubun içinde de üç eleman var. Böylece elimizde, x'den küçük yada ona eşit olan, en az 3 çarpı n/5'in tabanı, bölü 2 n taban elemanının tabanı olur. Ve bunun benzerini x'den büyük yada eşit durumu için de elde ederiz. Bunu birazcık daha basitleştireyim. Simdi bunun neden olduğu konusunda daha fazla gerekçe de verebilirim; şekli bu amaçla çizdik. Elimizde en az x'e eşit yada x'den küçük olan (n/5 bölü 2) öbek ortancası var.

Kullandığımız sav budur. Öbek ortancalarının yarısı x'den küçük yada x'e eşit çünkü x öbeklerin ortancalarının ortancasıdır; bu büyük bir sürpriz değil. Bu neredeyse bir eşitlik ama biz tabanlar üretip bunların daha büyük yada eşit olmalarını sağlıyoruz. Ve her öbek ortancası için o öbek ortancasından daha küçük yada eşit üç eleman

olduğunu biliyoruz. Geçişlilik kuralı bağlamında bunlar da x'den küçük yada x'e eşit oluyor. Böylece bu sayı kere üç elde ediyoruz.

Bu aslında n/10 'un tabanıdır. Bunu anlatırken gereksiz detaylara girdim ama bunun geldiği yer budur. Bildiğimiz bunun şimdi en az 3/10 olduğu yani yaklaşık 3/10 elemanın bir tarafta olduğudur. Aslında en az 3/10 eleman her iki tarafta da vardır. Bu nedenle iki taraf da en fazla 7/10 elemana sahip olabilir. Böylece buradaki sayı 7/10 olur. Ve eğer şanslıysam, 7/10 artı 1/5, 1'den kesin küçük olacaktır. Öyle olduğuna inanıyorum ama onlu saylarla çalışırken biraz zorluk çekerim. Sadece ikili tabanlı sayılarda iyiyim. Küçük bir basitleştirmeyi kullanarak düşünmeyi biraz daha kolay hale getireceğiz. Bu da şu tabandan kurtulmak amacına dönük olacak, çünkü taban rahatsız edici. Ve burada bir özensizlik ön kuramı da olmayacak.

Öyle olur ki, eğer n yeterince büyükse 3 kere n/10'un tabanı, 1/4'e eşit yada ondan büyüktür. Çeyreklerle rahat çalışabilirim. Savımız her grubun boyutunun en az 1/4, en çok da 3/4 olduğudur -- 1/4 diğer tarafta kaldığından. Bu 3/4 olur ve ben 1/5'in 1/4'den daha küçük olduğunu söyleyebilirim. Böylece bu toplam kesinlikle 1'den küçük çıkacak ve algoritmamız çalışacaktır.

Ne kadar zamanım var? İyi. Bu noktadan sonraki çözümleme kolaydır. Böyle bir algoritmayı nasıl kuramlayabilmişler ki; bunun bir bölüntü elemanını bulmak için çok iyi bir seçim olduğunun ve her iki yinelemenin de toplam süreleri doğrusal zamanda çıktığından bunun çok iyi olduğunun farkında mısınız? Bu nedenle birçok ünlü kişinin emeği gerekmiş. Testlerde yada genelde bu derste bu kadar akıllı bir algoritma üretmek zorunda kalmayacaksınız çünkü ortancayı bulmak için bu algoritmayı kullanabilirsiniz. Ortanca gerçekten iyi bir bölüntü elemanıdır. Şimdi bu algoritmayı bildiğinize göre ve 1973 geçmişte kaldığına göre, bunu nasıl yazabileceğinizi bilmek zorunda değilsiniz. Söylemek istediğim şey algoritmanın nasıl çalıştığını bilmeniz gerektiği ama başka bir algoritmayla uğraşırken bunları yeniden yapmak yerine bu algoritmaya "çalış" komutunu verebileceğinizdir. Böylece ortancayı doğrusal zamanda bulur, solda ve sağda bölüntüleri yaparsınız.

Ve hem sol hem de sağdaki bölüntülerin boyutu tamamen eşit olur. Çok iyi. Bu gerçekten güçlü bir altyordamdır. Bunu her yerde kullanabilirsiniz ve gelecek derste de kullanacaksınız. Koşma süresini yeterince çözümledim mi? Birinci adım doğrusal. İkinci adım T(n/5) 'tir. Üçüncü adımı yazmadım ama o da doğrusaldır. Ve son adım da yalnızca bir yineleme uygulamasıdır. Şimdi bunun 3/4 olduğunu biliyoruz.

Bu yinelemeyi elde ederim: T(n) , en çok T(n/5) artı T(3/4 kere n) 'ye eşittir diyeceğim. 7/10 da kullanabilirdiniz. Aynı sonucu verirdi ama bu durumda bir de tabana ihtiyaç olurdu ve bunu yapmayacağız.

Ben bunun doğrusal olduğunu savunuyorum. Bunu nasıl kanıtlayabilirim? Yerine koyma metoduyla..T(n)'nin en çok c çarpı n olduğunu söyleyin, bu yeterli olacak.

Kanıtlama yerine koyma ile yapılacak. Bunun da küçük n değerleri için geçerli olacağını farz ediyoruz ve n için kanıtlamak istiyoruz. T(n) en çok bu değeri alır:

T(n/5). Tümevarım yaklaşımıyla, n/5, n'den daha küçük olduğundan, bunun en fazla c olduğunu biliriz. Bunu c / 5 çarpı n şeklinde yazayım. Neden olmasın? Sonra burada 3/4 kere cn var. Sonra da doğrusal bir terim var. Şimdi maalesef ikinin katları olmayan değerlerle uğraşmak zorundayım. Notlarıma bakarak kopya çekeceğim. Bu aynı zamanda 19/20 kere c kere n artı Θ(n) olarak bilinir. Burada önemli olan bunun kesinlikle birden küçük olduğudur. Birden kesin daha küçük olduğundan, buraya 1 kere c(n) – burada değeri 1/20 olan bir sabit olarak yazabilirim; burada kalan bir değer, 1/20 çarpı c çarpı n olduğu sürece.. Burada da rahatsız edici bir Θ(n) terimi var ki bunun eksi değerli olmamasını istediğimden Theta'dan da kurtulmam gerekir.

Ama bu zaten eksi değildir; buradaki sabiti çok büyük seçersiniz, örnekte 20'den büyük seçilmesini gerektiriyordu, değeri artı olur. Yani bu yeterince büyük c değerleri için en fazla c kere n'dir. Bu arada, n burada kullandığımız gibi 50'ye eşit yada 50'den küçükse T(n) ne yaparsanız yapın bir sabit olur ve T(n) daha büyük c değerleri için en çok c kere n'dir. Bu savımızı kanıtlar. Tabii buradaki sabit çok büyüktür. Durum koşma süreleri ve sabitlerin ne olduğuna bağlıdır, bunlar da makinenize bağlıdır ama pratikte bu algoritma çok kullanılışlı değildir; çünkü sabitler çok büyük olur. Bu elemanın yaklaşık orta yerde olacağı garantili olmasına rağmen ve bu yinelemelerin toplamının kesinlikle n'den küçük çıkmasına ve bunun geometrik olmasına rağmen çok pratik değildir; sonuç geometrik çıkar çünkü problem her tekrarda en az 19/20 oranında küçülür.

Yani problemin gerçekten küçülmesi epey süre alır. Pratikte işi şansa bırakmak istemezseniz bu algoritmayı kullanmazsınız. Rastgele algoritma gerçekten hızlı çalışır. Teoride bu sizin düşünüzdür, umabileceğinizin en iyisidir çünkü doğrusaldır ve buradaki doğrusallık garantilidir. Bitirmeden önce bir alıştırmadan söz edeceğim.

Neden beşli öbekler kullandık? Neden üçlü öbekler değil? Tahmin edebileceğiniz gibi cevabı üçlü öbeklerle istenilen sonucun alınamayacağıdır. Ama bunun neden olmadığını bulmak geliştirici bir çaba olur. Bu problemde beşli yerine üçlü öbekler kullanırsanız ihtiyaç duyulan problem küçülmesini elde edemezsiniz. Bunun işler olacağı en küçük değer beştir. Yedili öbeklerle de çalışır ama, teorik olarak bir sabit faktörden daha iyi sonuç alamazsınız. Soru var mı?

Pekiyi. Gelecek derste görüşürüz..