Algoritmalara Giriş Ekim 24, 2005
Massachusetts Institute of Technology 6.046J/18.410J
Profesör Erik D. Demaine ve Charles E. Leiserson Dağıtım 16

Problem Seti 5

Okumalar: Bölüm 14 ve atlama listesi dağıtım.

Hem egzersizler hem de problemler çözülecek, ama sadece problemler teslim edilecektir. Egzersizler ders materyalini hazmettirmek amacıyla hazırlanmıştır. Her ne kadar egzersiz çözümlerini teslim etmeyecek olsanız da egzersizdeki konulardan sorumlu olacaksınız.

Her sayfanın üstüne adınızı, dersin kod numarasını, problemin numarasını, etüt bölümünüzü ve ortak çalışma yaptığınız arkadaşlarınızın isimleriyle tarihleri yazın. Lütfen çözümlerinizi zımbalayın ve üç delikli kağıtta teslim edin.

Sık sık bir problem için "bir algoritma bulun" isteğiyle karşılaşacaksınız. Bu konudaki yanıtınız kısa bir makale şeklinde olmalıdır. Makalenin konu paragrafı, çözdüğünüz problem ve sonuçlarınızı özetleyecek şekilde düzenlenmelidir. Makalenizin ana yapısında aşağıdaki bilgiler verilmelidir:

1. Algoritmanın İngilizce açıklaması ve eğer faydalı olacaksa sözde kodu..

2.Algoritmanızın nasıl çalıştığını gösteren en az bir işlenmiş örnek veya şekil.

3.Algoritmanın doğruluğunun kanıtı (veya göstergesi).

4. Algoritmanın koşma zamanının çözümlemesi.

Amacınız iletişim kurmaktır.Tam not sadece iyi açıklanan doğru yanıtlara verilecektir. net olmayan açıklamalar daha düşük notlandırılacaktır.

Egzersiz 5-1. Kitaptaki 14.1-5 nolu egzersiz (307. Sayfa)
Egzersiz 5-2. Kitaptaki 14.2-1 nolu egzersiz (310. Sayfa)
Egzersiz 5-3. Kitaptaki 14.3-4 nolu egzersiz (317. Sayfa)
Egzersiz 5-4. Kitaptaki 14.2 nolu egzersiz (318. Sayfa)

Problem 5-1. Atlama Listeleri ve B-ağaçları

Sezgisel olarak önceden gördüğünüz bir elemanın yakınındaki bir elemanı görmek daha kolaydır. Bir dinamik veri yapısında ***–x' den- y' ye bir yoklama araması*** aşağıdaki sorgulamadır:

Bir veri yapısında *x* elemanını depolayan yuva verildiğinde, ve başka bir y elemanı verildiğinde veri yapısında *y'* yi depolayan yuvayı bulun. Atlama listeleri hızlı yoklama aramalarını aşağıdaki bağlamda destekler.

(a) Bir atlama listesinde *x*' den *y*' ye yoklama araması yapmak için gerekli algoritmayı verin. Algoritmanızın koşma süresi yüksek olasılıkla O(lg(2 + |*(rank(x)* — *rank(y)* |)) olmalıdır. Burada *x*' in rütbesi yani *rank (x*) dinamik setin sıralanmış düzeninde x elemanını o andaki rütbesini gösterir. ''Yüksek olasılıkla'' derken *m* = 2 + | *rank (x) – rank(y)* | ' ye göre yüksek olasılığı kastederiz. Yani algoritmanızın koşma süresi, her α ≥ 1 için 1 - 1 / m^{α} …. olasılıkla *O(lg m)* olmalıdır.

Burada x elemanını depolayan atlama listesinin en altındaki listedeki yuvanın, yoklama- araması işlemine verildiğini düşünün.

B-ağaçlarından, hızlı yoklama aramalarını desteklemek için iki fikir gereklidir: B^+ –ağaçları ve düzey linklemesi. Problem boyunca *B=O(*1*)* kabul edin.

Bir B^+-ağacı öyle bir B-ağacıdır ki tüm anahtarlar yapraklarda depolanmıştır ve dahili düğümlerde bu anahtarların kopyaları depolanır. Daha kesin bir anlatımla bir $c_1, c_2 \ldots c_{k+1}$ gibi *k+1* ardılı olan bir p dahili düğümünde k sayıda anahtar depolanır. $c_1$in alt ağacındaki maksimum anahtar, c_2 nin alt ağacındaki maksimum anahtar, c_k nın alt ağacındaki maksimum anahtar.

(b) Bir B^+ - ağacında, O(lg n) zamanında, verilen bir *x* anahtarını saklayan yaprağı bulmak için, B-ağacı ARAMA algoritmasının nasıl değiştirmeniz gerektiğini tarif edin.

(c) B^+ ağaçlarında koşma süresinin *O(lg n)* olması için, B-ağaçlarında ARAYA YERLEŞTİRME ve SİLME algoritmalarının nasıl değiştirilmeleri gerektiğini açıklayın.

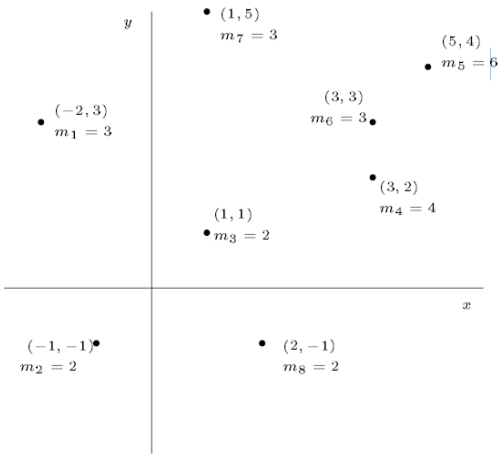
Bir **düzey linklemeli** B^+ **- ağacı ,** öyle bir B^+ **- ağacı** dır ki, aynı derinlikteki düğümler arasında hemen sonundaki düğüme bir ek işaretleyicisi vardır ve aynı zamanda aynı derinlikteki düğümler arasında hemen sağındaki düğüme de bir ek işaretliyicisi vardır.

(d) (c) bölümündeki B^+ **- ağacı** ARAYA YERLEŞTİRME ve SİLME algoritmalarının işlem başına *O (log n)* süresinde düzey linklerini koruyabilmeleri için nasıl değiştirilebileceğini açıklayın.

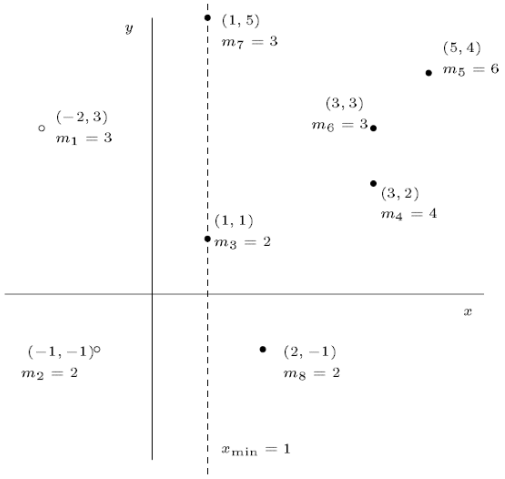
(e) Bir düzey-linkli B^+ ağacında *x*' den *y*' ye yoklama araması yapacak bir algoritma verin. Algoritmanızın koşma süresi *O(lg(2+|rank(x) – rank(y)|))* olmalıdır.

Bu fikirler, atlama listeleriyle düzey linkli 2-3-4 ağaçları arasında bir ilişki olduğunu işaret eder. Aslında bir atlama listesi düzey linkli B^+ağacının rastgele versiyonudur.

(f) Bir belirlenimci atlama listesinin, nasıl uygulanabileceğini açıklayın. Yani veri yapınız, bir atlama listesindeki genel işaretleyici yapısının aynısı olmalıdır: Aynı anahtarı depolayan komşu listelerdeki yuvalar arasındaki işaretleyicilerin olduğu bir veya daha fazla linkli liste dizisi. Arama algoritması bir atlama listesininkinin aynısı olmalıdır. Bir anahtarın terfi edileceğini belirlemek için ARAYA YERLEŞTİRME işlemini değiştirerek rastgeleliğin kullanımından kaçınmanız gerekir. Problemin bu kısmında SİLMEYİ görmezden gelebilirsiniz.



Figür 1: 8 noktalı bu örnekte eğer *f(p_i)= m_i* ise *F(S)=25 olur.*



Figür 2: *x_{min}= 1 ise,T(1)= { p_3 , p_4, p_5, p_6, p_7, p_8} ve F(T(1))=20*

Problem 5-2. Bir Düzlemdeki Noktalarla Eğlence

Saat sabahın 3'ü ve siz 6.046 derslerini videodan izlemeyi amaçlıyorsunuz ve problem seti 5 ile ilgili ipuçlarını arıyorsunuz. Garip bir nedenle belki de bilinciniz gelip-gittiğinden, odanızdaki beyaz duvarda, siyah noktalardan oluşan garip bir bulut fark ediyorsunuz. Bu nedenle dersi izlemek yerine, bilinçaltınız aşağıdaki problemi çözmeye çalışıyor. S= { p_1, p_2... p_n }'nin *xy* düzleminde *n* noktadan oluşan bir küme olduğunu farz edin. Her p_i noktasının koordinatları(x_i, y_i) ve **ağırlığı** da m_i.(noktanın boyutunu tanımlayan bir gerçek sayı). *F(p)= f(x,y,m)* bir rastgele fonksiyon olsun ve p noktasını *x,y* koordinatlarını m ağırlığında bir gerçek sayıya eşlemlesin ve bu *O(1)* süresinde hesaplanabilsin. Bir *S(T)* altkümesinde *F(T)* fonksiyonunu *T*' nin içindeki tüm noktalar için f(p_i)' lerin toplamı olarak tanımlayın. Yani

$$F(T) = \sum_{p \in T} f(p).$$

Örneğin, eğer f(p_i) = m_i ise, *F(S)* tüm *n* noktalarının ağırlıklarının toplamıdır. Bu durum şekil 1' de gösterilmiştir.

Hedefimiz, noktaların belli altkümeleri için *F* fonksiyonunu hesaplamaktır. Her alt kümeye bir *sorgulama* diyoruz ve her *T* sorgulaması için *F(T)*' yi hesaplamak istiyoruz. Çok sayıda sorgulama olabileceği için her sorgulamayı verimli yanıtlayabilecek bir veri yapısını tasarlamak istiyoruz.

Önce *x* koordinatını sınırlayan sorgulamaları ele alıyoruz. Özellikle *x* koordinatları en az x_{min} olan noktaların kümesini ele alıyoruz. Resmi olarak *T(* x_{min}*)* aşağıdaki gibi noktaların kümesi olsun.

$$T(x_{\min}) = \{p_i \in S \mid x_i \geq x_{\min}\} .$$

Şu formdaki sorgulamalara yanıt bulmak istiyoruz: girişte herhangi bir x_{min} değeri verildiğinde F(T) x_{min}' in değerini hesaplayın. Şekil 2, bu tür sorgulamaya bir örnektir. Bu durumda x_{min}=1 ve ilgi duyduğumuz noktalar *x* koordinatı en az *1* olanlardır.

(a) Bir dengeli ikili arama ağacının, böyle bir sorgulamayı *O(lg n)* sürede yapmak için nasıl değiştirilmesi gerektiğini gösterin. Daha açık bir deyişle F(T(x_{min})) hesaplamasının ağaç boyunca aşağıya sadece bir yürüyüşte nasıl yapılacağını gösterin. Bu problem için güncelleme (araya yerleştirme ve silmeler) desteklerine ihtiyacınız yok.

(b) Tüm n noktalarının önceden bilindiği bir statik problemi düşünün. (a) şıkkındaki veri yapısını kurmak ne kadar zaman alır?

(c) Toplamda *n* sayıda nokta verilirse, veri yapısını kurmak ve *k* farklı sorgulamayı yanıtlamak ne kadar zaman alır? Öte yandan bir veri yapısı kullanmadan, saf algoritmayı *F(T(* x_{min}*))* her sorgulama için 0' dan başlayarak hesaplamak ve *k* farklı sorgulama yapmak ne kadar zaman alır? Veri yapısının asimptotik olarak daha verimli olması *k*' nin hangi değerleri için geçerlidir?

(d) Bu veri yapısını bir kırmızı-siyah ağaç kullanarak dinamik hale getirebiliriz. (a) şıkkındaki çözümünüz genişletildiğinde bunun bir kırmızı-siyah ağaç tarafından verimli şekilde destekleneceğini tartışın; yani noktalar *O(lg n)* süresinde araya yerleştirilebilir veya silinebilir.

Bundan sonra, girişi x_{min} gibi tek sayı olmayan, bir *X=[* x_{min}, x_{max}*]* *(* $x_{min} \leq x_{max}$*)* aralığını girişinde kabul eden sorgulamalar üzerinde düşüneceğiz. *T(X), x* koordinatları, aşağıdaki aralıkta olan noktalar kümesi olsun.

$$T(X) = \{p_i \mid x_i \in [x_{\min}, x_{\max}]\}$$

Bu tür sorgulamanın bir örneği için şekil 3' e bakalım.

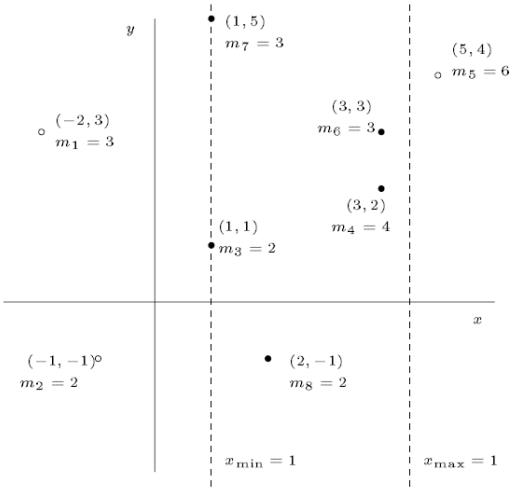
İddiamız *F(T(X))*' i hesaplamak için (d) şıkkındaki dinamik veri yapısını kullanabileceğimizdir.

(e) *F(T(X))'* i, *O(lg n)* sürede hesaplamak için (a) şıkkındaki algoritmanızın nasıl değiştirilmesi gerektiğini gösterin. *İpucu: x* koordinatı x_{min} ile x_{max} arasında olan ağaçtaki en sığ düğümü bulun.

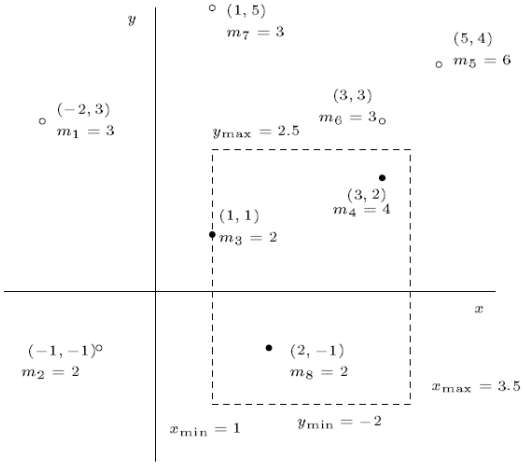
Son olarak statik problemi 2 boyutta genelliyoruz. Farz edin ki bize, *X=[* x_{min}, x_{max}*]* ve *Y=[* y_{min}, y_{max}*]* gibi iki aralık verilmiş olsun. *T(X,Y)* bu dikdörtgendeki tüm noktaların kümesi olsun. Yani;

$$T(X, Y) = \{p_i \mid x_i \in X \text{ and } y_i \in Y\}$$

2 boyutlu bir sorgulamanın örneği için şekil 4' e bakın.



Figür3 ; *X=[1,3,5]* iken, *T(X)={* p_3 , p_4 , p_6 , p_7 , p_8 *}* ve *F(T(X))=14*



Figür 4; *X[1,3,5]* ve *Y=[-2,2.5]* için *T(X,Y)= {* p_3, p_4, p_8 *}* ve *F(T(X,Y))=8*

(f) *X* ve *Y* aralıklarında, *F(T(X,Y))* sorgusunu verimli olarak hesaplamayı destekleyen bir veri yapısını açıklayın. Bir sorgu *O(lg^2 n)* sürede koşmalıdır.
İpucu: Bir değer kümesi ağacını genişletin.

(g) Veri yapınızı kurmak ne kadar süre alır? Ne kadar yer tutar?

Maalesef bu veri yapısını dinamik hale getirmede sorunlar vardır.

(h) Şık (d)' deki argümanınızın 2 boyutlu duruma genelleştirilip genelleştirilemeyeceğini açıklayın. Şık (f)' deki veri yapısına yeni bir noktanın araya yerleştirilmesi için gerekli en kötü durum süresi nedir?

(i) Başlangıçta *(n)* noktası olan bir veri yapısını kurduktan sonra *O(lg n)* sayıda güncelleme uygulayacağımızı varsayalım. Bu durumda hem sorguları hem de güncellemeleri verimli şekilde desteklemek için, veri yapısını nasıl değiştirebiliriz?

Tamamiyle isteğe bağlı kısımlar

Bu problemin geri kalan bölümünde, daha önceki bölümlerde tanımladığımız veri yapılarını kullanarak gerçek bir uygulamada, verimli hesaplama yapan bir *F* fonksiyonu örneği verilmekte. İlgili *f(* p_i*)* fonksiyonunun türetilmesi şık (j) ve (l)' de ana hatlarıyla açıklanıyor.

Bu problemin geri kalan kısmı, tamamiyle isteğe bağlıdır. Lütfen bu bölümleri yanıtınıza eklemeyin.

Daha önceki gibi, bir düzlemde *n* sayıdaki noktaların kümesi *S{* $\mathrm{p}_1,\ \mathrm{p}_2,\ldots\ldots,\ \mathrm{p}_\mathrm{n}$*}* olsun; bunların koordinatları *(* $x_i,\ y_i$*)* ve ağırlığı m_i. Biz kümedeki noktaların eylemsizlik momentini en aza indirecek ekseni hesaplamak istiyoruz. Resmi olarak alttaki değeri en aza indiren düzlemdeki *L* çizgisini hesaplamak istiyoruz.

$$\sum_{i=1}^{n} m_i \Big(d(L, p_i)\Big)^2$$

Burada *d(L,* p_i*),* $p_i$noktasından, *L* çizgisine olan mesafedir. Eğer tüm *i* ' ler için m_i*=1* ise, bu ekseni kümenin oryantasyonu yani yönelimi olarak düşünebiliriz.

(j) *xy* düzlemindeki bir çizginin bir parametreleme şekli de, onu bir (ρ, θ) çifti olarak tanımlamaktır; burada ρ, kaynaktan çizgiye olan uzaklık ve θ' da çizginin *x* ekseniyle yaptığı açıdır. Bir *(x,y)* noktası ile bir *L* çizgisinin arasındaki uzaklık, (ρ, θ) cinsinden

$$|x \sin\theta - y \cos\theta + \rho|$$

parametrelendirilebilir.

Biz S noktalar kümesinin yönelimini *L= (*ρ*,* θ*)* olarak tanımladık ve bu aşağıdaki fonksiyonu en küçük değerine indirgiyor.

$$f(\rho, \theta) = \sum_{i=1}^{n} m_i \left(x_i \sin\theta - y_i \cos\theta + \rho\right)^2$$

$\frac{\partial f}{\partial \rho} = 0$ ayarlamasının;

$$M_{x1} \sin\theta - M_{y1} \cos\theta + M_0 \rho = 0,$$

kısıtını getireceğini gösterin, burada;

$$M_0 = \sum_{i=1}^{n} m_i, \qquad M_{x1} = \sum_{i=1}^{n} m_i x_i, \qquad M_{y1} = \sum_{i=1}^{n} m_i y_i$$

(k) $\partial f / \partial \rho = 0$ ayarlamasının ve (j) şıkkından gelen kısıtın kullanılmasının aşağıdaki denklemi vereceğini gösterin.

$$\tan 2\theta = \frac{2\left(M_0 M_{xy} - M_{x1} M_{y1}\right)}{M_0\left(M_{x2} - M_{y2}\right) + M_{y1}^2 - M_{x1}^2}.$$

burada

$$M_{xy} = \sum_{i=1}^{n} m_i x_i y_i, \qquad M_{x2} = \sum_{i=1}^{n} m_i x_i^2, \qquad M_{y2} = \sum_{i=1}^{n} m_i y_i^2$$

(l) Yönelim probleminin biraz önce çözdüğümüz problemin özel bir durumu haline getiren $F(p_i)$ fonksiyonunu verin. *İpucu:* $F(p_i)$ fonksiyonu, bir vektör-değerli fonksiyondur.