Algoritmalara Giriş Ekim 17, 2005
Massachusetts Institute of Technology 6.046J/18.410J
Profesörler Erik D. Demaine ve Charles E. Leiserson Dağıtım 15

Problem Seti 4

Okumalar: Bölüm 12–13 ve 18

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.

Sıksıkbir problem için "bir algoritma bulun" isteğiyle karşılaşacaksınız. Bu konudaki yanıtınız kısa bir makale şeklinde olmalıdır.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 4-1. Kitaptaki 12.1-5 nolu egzersiz (256. Sayfa)
Egzersiz 4-2. Kitaptaki 12.2-4 nolu egzersiz (260. Sayfa)
Egzersiz 4-3. Kitaptaki 12.4-3 nolu egzersiz (268. Sayfa)
Egzersiz 4-4. Kitaptaki 13.1-6 nolu egzersiz (277. Sayfa)
Egzersiz 4-5. Kitaptaki 13.3-1 nolu egzersiz (287. Sayfa)
Egzersiz 4-6. Kitaptaki 18.2-6 nolu egzersiz (449. Sayfa)

Algoritmalara Giriş Ekim 29, 2005
Massachusetts Institute of Technology 6.046J/18.410J
Professors Erik D. Demaine ve Charles E. Leiserson Dağıtım 18

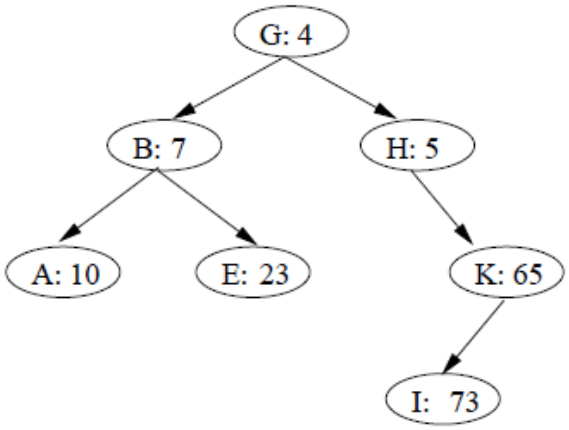
Problem 4-1. Treaps

Treap'ler (Rastgele sıralanmış ikili arama ağaçları)

Eğer tüm elemanlar aynı anda elimizde değilse - eğer elemanları birer birer alırsak, gene de bunlarla bir rastgele ikili arama ağacı oluşturabilir miyiz? Bu soruyu olumlu yanıtlayacak bir veri yapısını inceleyeceğiz. **Treap,** düğümleri değişik biçimde düzenlenmiş bir ikili arama ağacıdır. Şekil 1 bir treap örneğini göstermektedir. Genelde olduğu gibi ağaçtaki her x öğesinin bir *key[x]* anahtarı var. Buna ek olarak her x için rastgele seçilen bağımsız bir sayıyı *priority[x]* önceliği olarak atarız. Tüm önceliklerin ve tüm anahtarların farklı olduğunu kabul ederiz. Treap' in düğümleri öyle düzenlenmiştir ki, (1) anahtarların hepsi ikili-arama-ağacı özelliklerine uyarlar ve (2) özellikler de min-heap (en küçük-yığın) düzeni özelliğine uyarlar. Başka birdeyişle,

- eğer v, u' nun sol ardılıysa, key[v] <key[u];
- eğer v, u' nun sağ ardılıysa, *key[v] >key[u];* ve
- eğer v, u' nun ardılıysa, o zaman *priority (v) >priority(u)* olur.

(Özelliklerin bu karışımı nedeniyle bu ağaça "treap" denir: Hem bir ikili arama ağacının (tree), hem de bir yığının (heap) niteliklerine sahiptir.



Şekil 1: Bir treap. Her x düğümü *key[x]: priority[x]* olarak etiketlenmiş. Örneğin kökün anahtarı *G* ve önceliği 4.

Treapleri şöyle düşünmekte yarar vardır. $x_1; x_2 ,..., x_n$, düğümlerinin her birini ilgili anahtarıyla bir treap' in içine rastgele düzende yerleştirelim. Bu durumda ortaya çıkan ağaç, normal bir ikili arama ağacına (rastgele seçilmiş) öncelikleri gözetilerek belirlenen düzende düğümler araya yerleştirildiğinde oluşacak ağaçtır. Başka bir deyişle, $priority[X_i] < priority [X_j]$ X_i 'nin uygulamada X_j den önce araya yerleştirildiği anlamına gelir.

(a) Her biri farklı anahtarları ve öncelikleri olan $x_1; x_2 ,..., x_n$ seti için bu düğümleri içeren özgün bir treap olduğunu gösterin.

(b) Treap'in beklenen yüksekliğinin *O(lg n)* olduğunu ve bu nedenle treap' de bir değeri aramanın beklenen süresinin *O(lg n)* olduğunu gösterin .

Mevcut bir treap' de yeni bir *x* düğümünün nasıl araya yerleştirileceğini görelim. Yapacağımız ilk şey *x*' e bir rastgele öncelik, *priority[x]* atamaktır. Sonra TREAP-INSERT dediğimiz araya yerleştirme algoritmasını çağırırız ve bunun işlemesi şekil 2' de gösterilmiştir.

(c) TREAP-INSERT' ün nasıl çalıştığını açıklayın. Fikri İngilizce açıklayın ve sözde kodunu yazın. (İpucu: Normal ikili arama ağacı araya yerleştirmesi yapın ve sonra rotasyon yani dönmelerle min-heap (en-küçük yığın) düzeni özelliğini yeniden geri getirin.

(d) TREAP-ARAYA YERLEŞTİRMESİ'NİN (TREAP-INSERT) beklenen koşma süresinin O(lg n) olduğunu gösterin.

TREAP-INSERT bir arama ve ardından da bir dizi döndürme yapar. Arama ve döndürmenin asimptotik koşma süreleri aynı olmakla birlikte, pratikte maliyetleri farklıdır. Bir arama, treap' deki bilgiyi değiştirmeden okurken, bir rotasyon, treap' in içindeki ata ve ardıl işaretleyicilerini değiştirir. Birçok bilgisayarda okuma işlemleri, yazma işlemlerinden çok daha hızlıdır. Bu nedenle TREAP-INSERT 'ün az döndürme yapmasını isteriz. Beklenen döndürme sayısının bir sabitle sıralı olduğunu göstereceğiz (aslında 2' den az)!

Bu özelliği gösterebilmek için şekil(3)' de açıklanan bazı tanımlara ihtiyacımız var. Bir *T* ikili arama ağacının ***sol omurgası***, kökten en küçük anahtara giden yoldur. Başka bir deyimle sol omurga kökten çıkan ve sadece sol kenarları içeren en uzun yoldur. Simetrik olarak *T* ağacının sağ omurgası kökten çıkan ve sadece sağ kenarları içeren en uzun yoldur. Bir omurganın uzunluğu, içerdiği düğümlerin sayısıdır.

(e) TREAP-INSERT kullanılarak x' in araya yerleştirildiği anın sonrasında T Treap' ini düşünün. C, x' in sol alt-ağacının sağ omurgasının uzunluğu olsun. D de, x' in sağ alt-ağacının sol omurgasının uzunluğu olsun. x' in araya yerleştirilmesinde yapılan döndürmelerin toplam sayısının C + D' ye eşit olduğunu gösterin.

Şimdi C ve D' nin beklenen değerlerini hesaplayacağız. Basit olması için anahtarların 1, 2,..., *n* olduğunu varsayacağız. Bu varsayım, bizi genellikten uzaklaştırmaz çünkü sadece anahtarları karşılaştırıyoruz.

İki ayrı düğüm *x* ve *y* için *k* = *key[x]* ve *i* = *key[y]* olsun ve göstergesel rastgele değişkeni şöyle tanımlayalım:

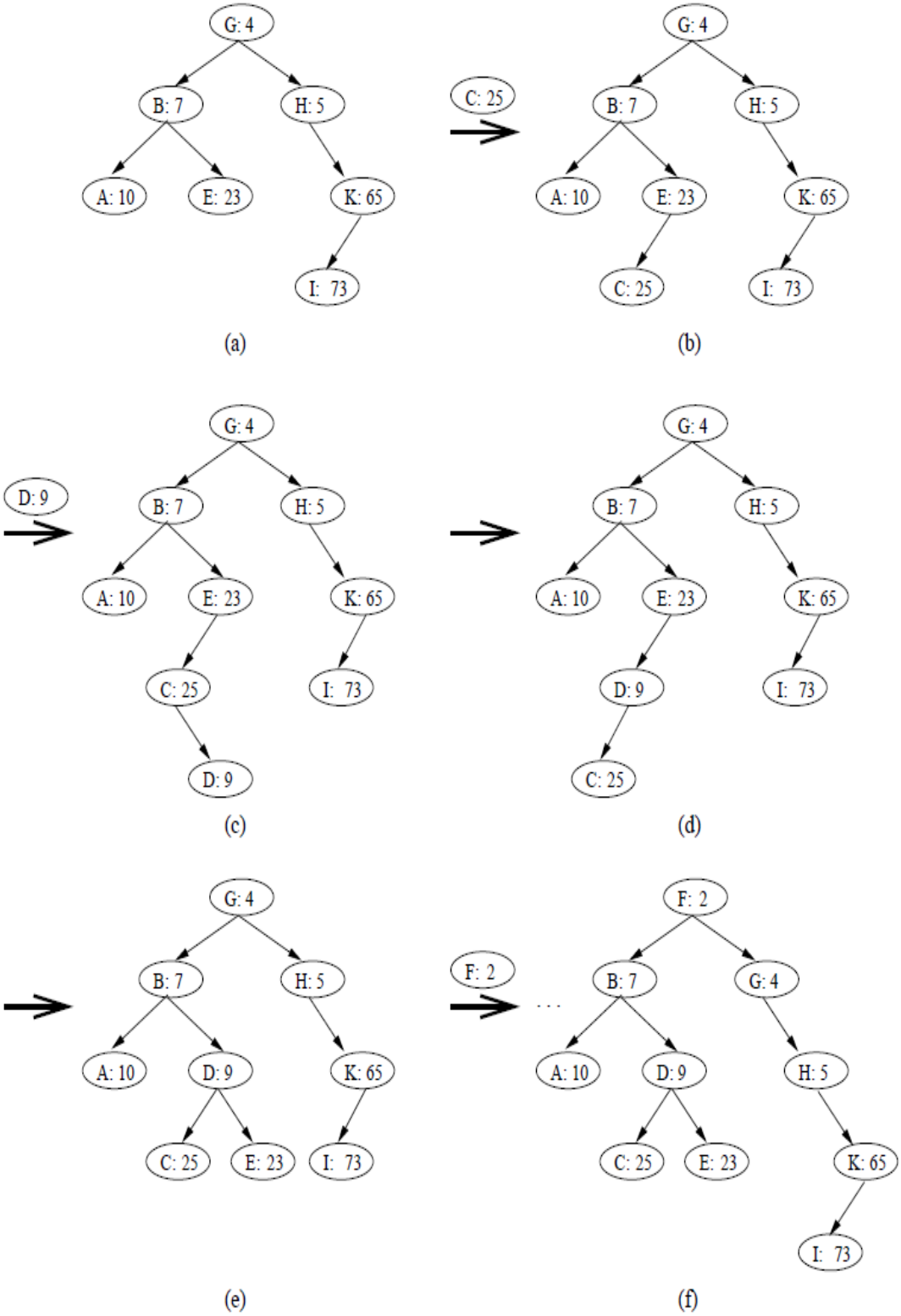
$$X_{i,k} = \begin{cases} 1 & \text{eğer y, x'in sol alt-ağacının sağ omurgasının düğümü ise } (T'\text{de}). \\ 0 & \text{aksi takdirde.} \end{cases}$$

(f) $X_{i,k}$ = 1, eğer ve sadece eğer, (if and only if)

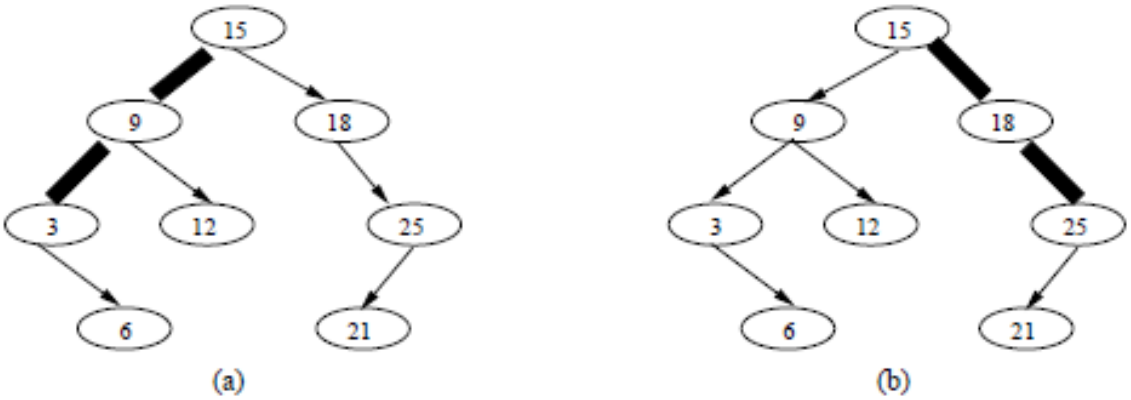
(1) *priority[y]* > *priority[x],*

(2) *key[y]* < *key[x],* ve

(3) her *z* için *key[y]* < *key[z]* < *key[x],* olursa *priority[y]* < *priority[z]* olacağını gösterin.



Şekil 2: TREAP-INSERT' ün çalışması. Şekil 1' de olduğu gibi her *x* düğümü key[x] : *priority[x]* olarak etiketlenmiştir.(a), araya yerleştirme öncesi orijinal treap. (b), anahtarı C ve önceliği 25 olan bir düğümün araya yerleştirilmesi sonrası treap. (c) ve (d), anahtarı D ve önceliği 9 olan bir düğümün araya yerleştirilmesindeki ara evreler. (e), (c) ve (d)' deki yerleştirilmeler tamamlandıktan sonraki treap. (f), anahtarı F ve önceliği 2 olan bir düğümün araya yerleştirilmesi sonrasında treap.



Şekil 3: Bir ikili arama ağacının omurgaları. Sol omurga **(a)**' da kalın çizgiyle işaretlenmiş, sağ omurga da **(b)**' de kalın çizgiyle işaretlenmiş.

(g) Şunu gösterin:

$$\Pr\{X_{i,k} = 1\} = \frac{(k-i-1)!}{(k-i+1)!} = \frac{1}{(k-i+1)(k-i)}.$$

(h) Şunu gösterin;

$$\mathrm{E}[C] = \sum_{j=1}^{k-1} \frac{1}{j(j+1)} = 1 - \frac{1}{k}.$$

(i) Aşağıdakini göstermek için bir simetri argümanı kullanın:

$$\mathrm{E}[D] = 1 - \frac{1}{n-k+1}.$$

(j) Bir treap' e bir düğüm eklenirken yapılan döndürmelerin beklenen sayısının 2' den az olduğu sonucunu çıkarın.

Problem 4-2. Dengeli olmak

Bir ağaç ailesine, ailedeki her ağacın boyu O(lg n) ise ***dengeli*** deyin; burada n, ağaçtaki düğümlerin sayısıdır. (Ağacın *yüksekliğinin* ağacın kökünden yaprağına kadar olan herhangi bir yoldaki kenar sayılarının maksimumu olduğunu hatırlayın. Özellikle tek düğümü olan bir ağacın yüksekliği 0 olur.)

Aşağıdaki her özellik için ikili ağaçlar ailesinin dengeli olma özelliğini taşıyıp taşımadığını belirleyin. Eğer yanıtınız "hayır" ise, bir zıt örnek verin. Eğer yanıtınız "evet" ise kanıtlayın; (ipucu: Kanıtlama tümevarımla yapılmalıdır). Dengeli olmanın *asimptotik* bir özellik olduğunu hatırlayın; bu nedenle zıt örnekleriniz, sadece tek ağacı tanımlamamalı, ailedeki sonsuz sayıda sete yönelik olmalıdır.

(a) Ağacın her düğümü ya bir yapraktır ya da 2 ardılı vardır.

(b) Her alt ağacın boyutu $2^k - 1$ olarak yazılabilir; burada k bir tamsayıdır (her alt ağaç için aynı *değildir).*

(c) Öyle bir $c > 0$ sabiti vardır ki, ağaçtaki her düğüm için, o düğümden çıkan daha küçük alt- ağacın boyutu, daha büyük alt-ağacın boyutunun en az *c* ile çarpımına eşittir.

(d) Öyle bir *c* sabiti vardır ki, ağaçtaki her düğüm için, o düğümün ardıl alt-ağaçlarının yükseklikleri arasındaki fark en çok *c* ' dir.

(e) Bir düğümün ortalama derinliği *O(lg n)*'dir. (Hatırlarsanız, bir *x* düğümünün derinliği, ağacın kökünden *x* düğümüne kadar olan yoldaki kenarların sayısıydı.)