Algoritmalara Giriş 18 Mayıs 2003
Massachusetts Institute of Technology 6.046J/18.410J
Profesörler Erik D. Demaine ve Charles E. Leiserson

Pratik Final Sınavı Çözümleri

- Dağıtılan sınav kitapçığını, size söylenene kadar açmayın.
- Final, saat 12:00'de bitecektir.
- Bu sınav kapalı kitap tarzındadır. Sınava el yazımı bir A4 formül kağıdı ile girebilirsiniz.
- Kitapçığın her sayfasına isminizi yazın.
- Cevaplarınızı ayrılan yerde cevaplayın. Daha fazla yere ihtiyacınız olursa, sorunun bulunduğu kağıdın arkasını kullanın. Bir sorunun cevabını başka bir sorunun kağıdına yazmayın, çünkü kağıtlar notlandırma esnasında birbirinden ayrılabilir.
- Sürenizi akıllıca değerlendirin. Bir soruda gereğinden fazla zaman harcamayın. Bütün soruları okuyun, daha sonra en fazla bilgi ve fikir sahibi olduğunuz sorudan cevaplamaya başlayın.
- Gidiş yoluna da puan verileceğinden, cevabınızı açıkça ifade edin. Cevabın doğruluğunun yanında cevabı veriş şekliniz de puanlamaya etki edecektir. Düzenli olun.
- Bir algoritmayı açıklarken, ana fikri Türkçe açıklayın. Sözde kodu, sadece fikrinizi genişletmek için kullanın.
- Bol şanslar!

İsim: _____

Problem 1. Kısa Cevaplar

Aşağıdaki sorulardaki boşlukları doldurun. Birden fazla cevabı olan sorularda, en çok bilinen doğru cevabı tercih edin. Örnek, olarak karşılaştırma metodunu kullanarak n tane elemanı sıralamak için gereken süre [ ] 'ye bilinen en doğru cevap O(n lg n)'dir.

Doğru olan ancak, "en uygun" olmayan yanıtlara kısmi puan verilmeyecektir. Buna karşılık, yanıtınızın açıklamasını yapmak zorunda değilsiniz.

(a) QUICKSORT 'a bir değişiklik yapıldığını, PARTITION her çağrıldığında bölüntü yapılan dizilimin ortancasının bulunduğunu (SELECT algoritmasını kullanarak) ve ortancanın pivot olarak kullanıldığını düşünün.

En kötü durumda algoritmanın koşma süresi [ ] 'dir.

Çözüm: *Θ(n lg n)*'dir. Ortanca, *O(n)* sürede bulunabilir. Dolayısıyla koşma süresi için şu yineleme vardır. *T(n) = 2T(n/2) + O(n)*

(b) Bir veri yapısının **foo** işlemini desteklediğini, *n* **foo**'luk bir dizinin en kötü durumda koşma süresinin *Θ(n lgn)* olduğunu düşünürsek,

Bir **foo** işleminin amortize edilmiş süresi Θ([ ])'dir.

Öte yandan, bir **foo** işleminin gerçek süre maliyeti Θ([ ]) kadar çok olabilir.

Çözüm: Amortize edilmiş süre *Θ(logn)*'dir; en kötü durumda *Θ(n logn)* süreye mal olur.

(c) Yönlendirilmiş bir grafikte, s - t en kestirme yolunu polinomsal sürede bulan bir algoritma var mıdır? [ ] (Evet / Hayır)

Çözüm: Evet, vardır. s - t en kestirme yolu, s - t maksimum akışıyla aynıdır. Ve maksimum akışları polinomsal sürede hesaplayan bir çok algoritma vardır. (Mesela, Edmonds-Karp algoritması)

Problem 2. Doğru veya Yanlış

Aşağıdaki sorularla ilgili, Doğru(D) veya Yanlış(Y) şıklarından birini işaretleyin. Cevaplarınızla ilgili açıklama istenmemekte. Boş veya yanlış cevaplar 0 puan alacaktır.

D Y $T(n) = 3T(n/3) + \log n$ yinelemesinin Ana Metodu kulanarak çözümü, $T(n) = \Theta(n \log n)$'dir.

Çözüm: Yanlış.

D Y Karşılaştırma modelinde çalışan algoritmalarda, herhangi bir n elemanın ortancasını bulmak $\Omega(n \log n)$'e mal olur.

Çözüm: Yanlış.

D Y n düğümlü her ikili ağaç, $O(\log n)$ yüksekliğe sahiptir.

Çözüm: Yanlış.

D Y $G=(V,E)$ gibi kenarlarında maliyet taşıyan bir grafiğimiz ve $S \subseteq V$ kümesi var; (u,v), S'deki herhangi bir köşe ile $V-S$'deki herhangi bir köşe arasındaki en az maliyetli kenar olsun. Bu durumda, G'nin en az yayılan ağacı (u, v) kenarını içermek zorundadır. (Eğer gerekiyorsa, bütün kenarların maliyetinin birbirlerinden farklı olduğunu varsayabilirsiniz.)

Çözüm: Doğru.

D Y T, G'nin en az yayılan ağacı olsun. Herhangi s ve t köşe çifti için, s'den t'ye G'nin içindeki en kısa yol, T'nin içinde s'den t'ye olan yoldur.

Çözüm: Yanlış.

D Y Eğer L_1 problemi polinomsal zamanda L_2 problemine indirgenebiliyorsa ve L_2'nin polinomsal süreli bir algoritması varsa, L_1'in de polinomsal süreli bir algoritması vardır.

Çözüm: Doğru.

Problem 3. Doğru veya Yanlış ve Gerekçeli

Aşağıdaki soruların, doğru mu yanlış mı olduğunu belirtmek için **D** veya **Y'**yi daire içine alın. Daha sonra yanıtınızın gerekçesini kısaca açıklayın. Yanıtlarınız, sadece D veya Y'yi işaretlemenizle değil, gerekçenizi açıklamanızla da değerlendirilecektir.

D Y Dinamik bir setin, INSERT(x,S), DELETE(x,S) ve MEMBER?(x,S) işlemleriyle bakımını yapan ve işlem başına O(1) sürede koşturan bir veri yapısı vardır.

Çözüm: Doğru. Kıyım tablosu bunu destekler.

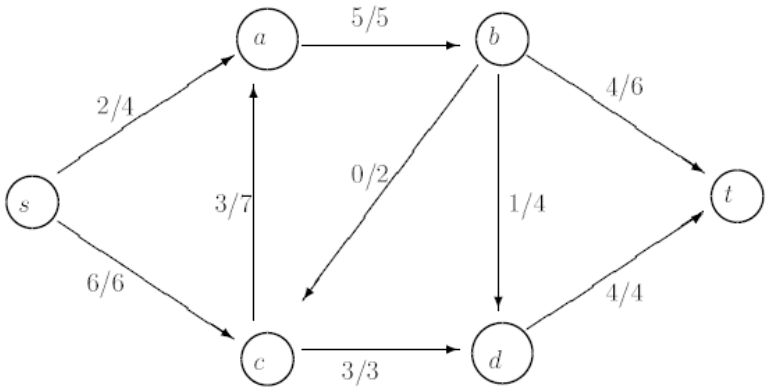
D Y n sayıda işlemin yapıldığı bir dizinin toplam amortize edilmiş maliyeti (her işlemin amortize edilmiş maliyetinin toplamı), toplam gerçek maliyet için bir alt sınır verir.

Çözüm: Yanlış. Bu gerçek maliyet için bir üst sınır verir.

D Y *G = (V,E)* bir ağırlıklı grafik olsun ve M de G'nin en az yayılan ağacı olsun. M'deki her v_1 ve v_2 köşe ikilisi arasındaki yol *G*'deki en kısa yol olmak zorundadır.

Çözüm: **Yanlış.** V = {a,b,c} olan bir grafiğini düşünelim ve (a,b), (b,c) ve (c,a) kenarlarımız olsun. Kenarların ağırlıkları, sırası ile; 3, 3, 4 olur. En az yayılan ağaç, kesinlikle (a,b), (b,c)'dir. Ancak, a'dan c'ye en kısa yol 4 maliyetindedir ve en az yayılan ağaçta göründüğü gibi 6 değildir.

D Y Aşağıdaki şekil, bir akış ağındaki, akışları ifade etmekte. *a / b* simgelemi *b* kapasitesine sahip bir kenarda *a* birim akış olduğunu anlatmakta. Aşağıdaki akış en fazla akıştır.



Çözüm: **Doğru.** Şekilde akış 8 birimdir ve {s,a,c} kesitinin kapasitesi {b,d,t}'ye oranla 8'dir. Kesit, En Fazla Akış, En Küçük Kesit Teoremi uyarınca maksimum olmalıdır.

Problem 4. Karakter Dizimi

Bir metni, düzgün bir şekilde dizgi için hazırlamak istediğimizi varsayalım. Girdimiz, *n* adet $l_1, l_2, \ldots, l_n$ uzunluğunda kelime dizisi (sabit boyutta karakterlerin sayısı olarak ölçülmüş). Her satır en fazla P karakter alıyor, metin sola dayalı ve bir kelime satırlar arasında ikiye bölünmeyecek. Eğer bir satır, i'den j'ye kadar kelimeleri içermekte ise, satır sonundaki boşlukların sayısı;

$$s = P - \sum_{k=i}^{j} l_k - (j - i)$$

Satır sonunda çok fazla beyaz alan kalmasını istemiyoruz. Yani, satır sonlarında kalan boşlukların sayısının karelerinin toplamını en aza indirgemek istiyoruz. Bu sorunu çözmek için verimli bir algoritma yazın. Bu algoritmanın koşma süresi nedir?

Çözüm: Bu soruyu dinamik programlama ile çözeriz. Dizgimiz, sadece *1…j* arasındaki sözcükleri dizgilediğinde (geri kalan sözcükler görmezden gelindiğinde), *A[j]* en uygun maliyet olsun (yani tüm satır sonlarında kalan beyaz boşlukların sayısının karelerinin toplamı). Bu durumda özyinelemeli olarak *A[j]* aşağıdaki gibi gösteririz:

$$A[j] = \min_{i<j : T[j]-T[i] \le P} A[i] + (P - (T[j] - T[i]))^2$$

Burada,

$$T[j] = \sum_{i=1}^{j} l_i$$

T[1,....,n] gibi bir tablodaki değerler, başlangıçta *O(n)* sürede hesaplanabilir. Yukarıdaki eşitlik şunu söylüyor: 1… j'ye kadar olan sözcükleri uygun şekilde dizmek için, önce *i* < *j* olduğunda, 1'den i'ye kadar olan sözcükleri düzgün dizmemiz, sonra da *i+1*, … *j* arasındaki kalan sözcükleri son satıra yerleştirmemiz gerekir.

Dinamik programlama kullanarak, tüm *j = 1….n* için, *A[j]*'nin her değerini sırasıyla hesaplarız. Her değerin hesaplanması *O(n)* süresi gerektirdiğinden, toplam koşma süresi $O(n^2)$ olur. Programın sona ermesinden sonra, *A[n]* en iyi çözümün değerini içerir; dinamik programlamada sıkça kullanıldığı gibi, "backpointers – geriye işaretleyiciler" kullanarak çözümü kendiliğinden yeniden yapılandırabiliriz (yani satır aralarını araya yerleştireceğimiz yerleri).

Problem 5. Algoritmianın Prensesi

Algoritmia'daki politik ayaklanma, Prenses Michelle X. Goewomans'ı sarayını terkedip Nebraska'da bir çiftliğe yerleşmeye zorluyor. Prenses saraydaki bütün eşyalarını Algoritmia'dan Nebraska'ya Ferrari'si ile taşıyacak; taşınma işlemini olabildiğince az gel-git yaparak çözmek istiyor. Basitlik için, prensesin Ferrari'sinin büyüklüğünün 1 olduğunu ve n sayıda x_1, x_2, ..., x_n eşyasının da 0 ile 1 arasındaki gerçek bir sayılar boyutunda olduğunu varsayalım. Problem, prensesin bütün eşyalarını Algoritmia'dan Nebraska'ya taşımak için olabildiğince az arabalık yük şeklinde nasıl bölebileceğimizi bulmak ve Ferrarinin aşırı yüklü olmasından kaçınmaktır. Bu problem NP-zorluk derecesindedir.

Aşağıdaki **ilk-uyan** yaklaştırma algoritmasını düşünelim. İlk olarak arabaya x_1'i koyalım. Daha sonra, i = 2, 3,...,n için, arabaya sığacak x_i eşyalarını yer varsa **sırasıyla** arabaya yerleştirelim; yer yoksa, bir dahaki tura bırakalım. Mesela, x_1=0.2, x_2=0.4, x_3=0.6 ve x_4=0.3 ise, ilk yüklemede x_1 , x_2 olacak, x_3 ikinci yüklemeye bırakılacak ve x_4 ilk yüklemeye dahil edilecektir. Verilen kararların çevrimdışı olduğunu ve prensesin eşyalarını herhangi biri yola çıkmadan partilere böldüğümüzü unutmayalım.

(a) Prensesin kocası Craig, ilk-uyan algoritmasının her zaman yolculuk sayısını en aza indirdiğini iddia ediyor. Karşı bir örnek vererek Craig'in iddiasını çürütün.

Çözüm: n = 4 olsun. x_1=0.3, x_2=0.8, x_3=0.2, x_4=0.7 olsun. En az sefer sayısı 2 olacakken, ilk-uyan algoritması ile en az 3 parti eşya taşınması gerekir.

(b) İlk-uyan algoritmasının oran sınırının 2 olduğunu ispatlayın. (*İpucu*: Kaç parti yük, kapasitenin yarısından az dolu olur?)

Çözüm: Bir *u* yükünün *p* boyutunda olduğunu ve yarıdan az dolu olduğunu düşünün, yani, $p< 0.5$ olsun. x_i bir sonraki v yüklemesinde gidecek ilk eşya olsun. Bu durumda, İlk-uyan algoritmasının doğası gereği, $p+x_i > 1$ ve $x_i > 0.5$'tir. Böylece, v yarıdan daha fazla doludur. Hesaplamada kullanmak için, x_i'nin bir parçasını v'den u'ya taşıyabilir, u'yu tam olarak yarı dolu hale getirebilir, aynı zamanda v'yi de en az yarısı dolu şekilde koruyabiliriz. Bunun nedeni $0.5 - p < x_i – 0.5$ olmasıdır. Bu işlemden sonra, bütün parti yükler en az yarı-doludur. Eğer ilk-uyan algoritması tarafından üretilen parti sayısı h ise,

$$\sum_{i=1}^{n} x_i \geq 0.5h$$

olur. En uygun parti sayısı OPT'nin en az

$$\sum_{i=1}^{n} x_i$$

olduğuna dikkat edin. Bu durumda, $h/OPT \leq 2$'dir ve bu da algoritmanın oran sınırının 2 olduğunu kanıtlar.

Problem 6. Kral Artur'un mahkemesinde gücün dağılımı

Birçok problem, Kral Artur zamanında bile eniyileme (optimizasyon) problemidir. Kral Artur'un mahkemesinde birçok şövalye vardır. Bu şövalyeler, Kral Artur'un problemlerinin çoğunun temel kaynağıydı ve problemlerin bir kısmı hesaplamaya dayalı, bir kısmı ise değildi. Kral Arthur, hesaplamaya dayalı problemleri çözerken, çoğu zaman güçlü büyücüsü Merlin'in sözünü dinlerdi. Rivayete göre, Merlin zaman makinasını kullanarak, 2000 yılındaki 6.046 Algoritmalara Giriş dersini almış, bahsi geçen sorunları da bu derste edindikleri ile çözermiş.

Bu sorunun 2 bölümünde, Merlin'e sorulan problemleri göreceğiz.

(a) Toprağa Hükmetmek

Kral Artur'un mahkemesi *n* tane şövalyeden oluşuyor. Kral Artur *m* tane ülkeye hükmediyor. Her ***i*** şövalyenin yönetebileceği ülke sayısının q_i gibi bir kotası var. Her ***j*** ülkesinin de yönetilmeyi tercih ettiği bir S_j şövalyeler kümesi var. Kral Artur'un, Merlin'e verdiği görev şu; ülkeler, şövalyelere öyle bir dağıtılsın ki, hiçbir şövalye kotasını aşmasın ve her ülke kendi S_j kümesinden bir şövalye tarafından yönetilsin.

(i) Merlin'in bu atamaların hesaplanmasında En Fazla Akış algoritmasını nasıl kullanabileceğini gösterin. Algoritmanın koşma süresini bulun. (v köşesi ve e kenarı olan bir ağda çalışan en-fazla-akış algoritmasının koşma süresi F(v,e) ise, koşma sürenizi bunun cinsinden açıklayabilirsiniz.)

Çözüm: n tane şövalye için n köşe (k_1, ..., k_n), m tane ülke için m köşe (c_1,...,c_m) ve 2 tane de s ve t özel köşelerini kullanarak, toplamda n+m+2 köşeli bir grafik çizeriz.

s'den k_i'ye kapasitesi q_i olan bir kenar koyarız. k_i ile c_j arasında 1 kapasiteli bir kenar varsa, bu j ülkesinin i şövalyesi tarafından idare edilmek istediğini gösterir. c_j'den t'ye de 1 kapasiteli bir kenar çizeriz.

Bu grafikteki en fazla akışı bulalım. Eğer akış, m değerine sahipse, şövalyeleri istenen şekilde ülkelere atama şansı vardır. Bu akış bütünleşik olduğundan, her c_j ülkesinden gelen bir kenar seçecektir. Eğer seçilen kenar k_i şövalyesinden geliyorsa, j ülkesini i şövalyesi yönetiyor demektir. Bu algoritmanın koşma süresi ise;

$$F(n+m+2, \sum_j |S_j|)$$

olur.

(ii) Merlin Şık (i)'deki algoritmayı çalıştırır, ancak, algoritma gereksinimleri karşılayan bir atama yapamaz. Kral Artur, Merlin'den bu konuda bir açıklama ister. Merlin algoritmayı krala açıklar. Kral da algoritmanın doğruluğu konusunda ikna olur. Ancak, Kral Merlin'in algoritmayı problemin verilen biçiminde doğru çalıştırdığına inanmaz. Merlin'den bu özel durum için bir atama olamayacağını kendisine ispat etmesini ister. En Fazla Akış Yöntemi'nden anladığınız kadarıyla hangi ispat yöntemini önerirdiniz? (İspat stratejinizin, bu problemin her örneklemesi için geçerli olması gerektiğini unutmayın.)

Çözüm: *m* boyutunda bir akışımız yoksa, bu grafikte *m*'den az kapasitesi olan bir kesit olmak zorundadır. Kesitin yapısına bakarak, T ülkeler kümesini, j∈T olduğunda, S_j kümelerinin birleşimi olan U verir. Bu durumda,

$\sum_{i \in U} q_i$

|T| 'den küçük olur. (Yani ülkelerin yönetilmek istedikleri şövalyelerin kotaları ülke sayısından az olabilir.) Merlin T ve U kümelerini Artur'a anlatarak, neden T kümesinin şövalyelere atanamadığını anlatabilir.

(b) Anlaşmazlık Çözümü : Artur, yıllık şövalyeler toplantısında bazı şövalyelerin anlaşmazlık yarattığını görür. Gizli ajanlarını kullanarak, bütün muhtemel düello çiftlerini belirler. En az sayıda şövalyeyi sürgüne göndererek, yıllık toplantıda herhangi bir düello çifti kalmamasını sağlamak ister. Merlin hangi şövalyelerin gönderileceğini belirleme noktasında görevlendirilir. Bir mükemmelliyetçi olarak Merlin, en küçük sürgüne gönderilecekler listesi üzerinde çalışmaya başlar. Buna karşın, uzun düşüncelerden sonra, herhangi bir algoritma ile uygun bir çözüm bulamaz.

(i) Neden?
Çözüm: Gönderilecek şövalyeler, uyumsuzluk grafiğinde bir köşe kaplarlar. Böyle bir durumda, en küçük kaplamayı bulmak NP-Tam bir problem olduğundan, Merlin bu sorunu çözemez.

(ii) Merlin gereksinimleri nasıl yumuşatırsa bir çözüm bulabilir?
Çözüm: Merlin Köşe Kaplama için 2-yaklaşma algoritmasını kullanabilir ve en uygun sürgüne gönderilecek şövalye sayısının en fazla 2 katı kadar büyüklükteki bir küme ile sorunu çözebilir.