**UBİ602 Algoritmaların Karmaşıklık Teorisi**

**ÖRNEK FİNAL SINAVI**

**Dilediğiniz 5 soruyu ve Bonus Soruyu yanıtlayınız. Süre 150 dakika**

**1.** Öklidyen Minimum Kapsayan Ağaç (*Euclidean Minimum Spanning Tree*) Problemi: Kartezyen düzlemde *n* nokta verildiğinde, düğümleri verilen noktalar olacak şekilde minimum toplam kenar ağırlığına sahip bir ağaç inşa etme problemidir. Bu problem için **indirgeme (reduction)** yöntemini kullanarak (mümkün olduğunca sıkı (*tight*)) bir zaman karmaşıklığı alt sınırı (lower bound) bulunuz.

**İpucu:** Eleman tekilliği (*Element uniqeness*) problemi “verilen *n* reel sayının içinde birbirinin aynı olan en az bir çift sayı olup olmadığını tespit etme” problemi olup bu problemin Ω(n log n) zaman karmaşıklığı alt sınırına sahip olduğunu biliyoruz.

**2.** Aşağıdaki teoremi kanıtlayınız.

**Teorem**: Eğer P NP ise, gezgin satıcı problemi için *c-*yaklaşık (*c-*approximation) bir algoritma, *sa* yoktur. Bir başka ifade ile P NP ise gezgin satıcı problemini, her girdi için, optimal çözüm *s\** ın bir sabit *c* katı değer ile üstten sınırlı yani *f(sa) ≤ cf(s\**) olacak şekilde çözen polinom zamanlı bir algoritma yoktur.

**3.** *n* elemanlı bir *A* dizisinde en küçük elemanı bulma problemini ele alalım. Aşağıdaki üç farklı algoritma tasarım tekniği için bu problemi çözecek algoritmaları kaba-kod (*pseudo code*) olarak yazınız. Her bir algoritmanın zaman karmaşıklık sınıfını bulunuz.

i) Önsıralama (*presorting*) tabanlı algoritma

ii) Kaba-kuvvet (*brute-force*) algoritma

iii) Böl-ve-fethet (*divide-and-conquer*) algoritma

Not: ***MergeSort***(*A*) fonksiyonunun elinizde olduğunu ve girdi olarak verilen A dizisini küçükten büyüğe sıraladığını varsayınız.

**4.** *Rod-cutting problem* Elimizde *n* birim uzunluğunda bir metal çubuk var. Bu çubuğu *i* (1 ≤ i ≤ n) birimlik parçalara bölerek, her bir paçayı *pi* liraya satabiliyoruz. Örneğin n=3 ve *p1* = 0,9 TL, *p2* = 2,05 TL ve *p3* = 2,90 TL. Toplayabileceğiniz maksimum para miktarını dinamik programlama yöntemini kullanarak bulmanız istenmektedir.

**a)** Problemin optimal çözümünü özyinelemeli bir denklem (*recursive formula*) olarak tanımlayınız.

**b)** Problemi çözen kaba-kod (*pseudo code*) parçasını yazınız.

**c)** Çözümün zaman ve yer (*memory*) karmaşıklık sınıflarını bulunuz.

**5. a)** Bir sayının, *n*, bileşik (*composite*) sayı olup olmadığını bulmak için *n*’i 2’den ’ye kadar olan tüm sayılara sıra ile böleriz. Eğer bu sayılardan herhangi birisi *n*’i tam (kalansız ) bölüyorsa, bu kaba-kuvvet algoritması “evet” (yani sayı bileşik), hiçbiri tam bölmüyor ise “hayır” cevabı döner.

**i)** Bu algoritmanın karmaşıklık sınıfı nedir?

**ii)** Bu algoritma neden bu problemi “class P” olarak sınıflamamız için yeterli değildir?

**b)** A = {2,5,8} ve *d=*10 için altküme toplam (*subset sum*) problemi için geriye-takip (*backtracking*) yöntemini çalıştırınız; yani, verilen problem için tam durum-uzay ağacını (*complete state-space tree*) oluşturunuz.

**6**. **a)** ifadesinin doğruluğunu gösteriniz.( : big-theta)

**b)** Aşağıdaki özyinelemeli fonksiyonunun karmaşıklık sınıfını bulunuz.

T(*n*) = T(*n*/2) + T(*n*/4) + T(*n*/8) + *n*

**Bonus Soru** (10 p)

Dev şirketlerin iş görüşmesinde sordukları en zor sorular

Pozisyon: BitTorrent, Otomasyon Mühendisi

Cücelerden nefret eden bir dev, 10 tane cüceyi en kısadan en uzuna olacak şekilde ard arda dizer. Her cüce önündeki kendinden kısa diğer cüceleri görebilirken, arkasındaki kendinden uzunları göremiyor. Dev her cücenin kafasına, rastgele şekilde siyah veya beyaz bir şapka koyacak ve ilk olarak en uzun cüceden başlayacak şekilde, cüceye kafasındaki şapkanın rengini soracak. Eğer cüce doğru cevap veremezse dev, cüceyi öldürecek. Her cüce arkasındakinin verdiği cevabı duyabilecek ama öldürülüp öldürülmediğini bilemeyecek. Yani doğru mu bildi, yoksa bilemedi mi hiç fikri olmayacak. Dev, cücelere bir fırsat tanıyıp, ölümcül oyun başlamadan birbirleriyle konuşmalarına ve bir strateji belirlemelerine izin veriyor. En az sayıda cücenin ölmesi için, cüceler nasıl bir strateji belirlemeli?