Pratik Ara Sınav 2

Bu sınav, her biri 25 puan değerinde, 5 sorudan oluşmakta. Toplamda 125 puan toplayabilirsiniz. Her soru, ayrı bir 3-delikli kağıda cevaplanmalı.

Her sayfaya isminizi, 6.046J/18.410J başlığını, problem numarasını, etüt zamanınızı ve araştırma görevlinizi yazın. **Sınavınızı, 29 Nisan saat 09:00 ile 11:00 arasında teslim edin.** Geç teslim edilen sınavlar, dekan onayı olmadan veya etüt öğretmeninizle, öncelikli başka bir ayarlama yapılmadan kabul edilmeyecektir. Sınavı kendi elinizle teslim etmek zorundasınız.

Sınavın çözülmesi için 10 saat yeterlidir. Ancak sizin önünüzde 4 gününüz var. Zamanınızı akıllıca planlayın. Aşırı çalışmayın, uykunuzu alın. İyi yazılmış çözümler yüksek puan alacaktır. Tabii ki asimptotik sınırlarınız ne kadar iyi olursa o kadar yüksek puan alırsınız. Sıra dışı verimli ve seçkin cevaplara Bonus puanlar verilecektir.

Yazım:
Her problem ayrı bir kağıtta veya kağıtlarda çözülmeli. Her sayfanın üzerine adınızı, 6.046J/18.410J'yi, soru numarasını, etüt saatinizi ve araştırma görevlinizin adını yazmayı unutmayın. Herhangi bir soruya cevabınız, çözümünüzün özeti ile başlamalı. Bu özet, çözdüğünüz problemi açıklamalı, kullandığınız teknikleri, önemli varsayımları ve algoritmanızın koşma süresini içermelidir.
Cevaplarınızı bizim anlamamızı kolaylaştırmak için açıkça ve temiz yazın. Koşma süreleri ve algoritmalar ile ilgili kesin ifadeler kullanın. Mesela, sadece "*n sayıyı sıralarız*" demeyin. Onun yerine "Yığın sıralamasını kullanarak n sayıyı, en kötü durumda $O(n \lg n)$ sürede sıralarız"ı tercih edin. Bir algoritmayı açıklarken ana fikri Türkçe yazın. Sözde kodu, sadece çözümünüzü daha açık hale getirmek gerekirse kullanın. Örnekler verin ve şekiller çizin. Çözümlerinizin doğruluğunu ispatlamak için az, öz ve ikna edici ifadeler kullanın. Sınıfta işlenen konuları defalarca tekrarlamayın. Sınıfta, ek derste veya kitapta geçen algoritma ve teoremlere referans vererek çözümünüzü basitleştirin.

Bu sınavın bir amacı da sizin mühendislik mantığınızı sınamaktır. Eğer bir soruyu şüpheli bulduysanız veya açık bulmadıysanız, mantıklı varsayımlar yapın. Ancak ne varsaydığınıza dikkat edin, çünkü eğer bir problemi kolaylaştıran çok güçlü bir varsayımda bulunursanız, o sorudan az puan alırsınız.

Hatalar, vs.: Eğer herhangi bir soruda bir hata bulduğunuzu düşünüyorsanız, bize e-posta atın. Düzeltmeler ve açıklamalar bir e-posta ile ve ağ sayfası üzerinden sınıfa duyurulacaktır. E-postanızı ve ders sayfasını günlük olarak kontrol edin. Böylece önemli olabilecek açıklamaları kaçırmazsınız.

Akademik Dürüstlük İlkesi: Sınav Sınırlı Açık Kitap şeklindedir. Ders notlarınızı, ders kitabınızı, sözlükleri ve dersin sayfasındaki kitapçıkları kullanabilirsiniz. Ancak, bunun haricindeki herhangi bir kaynağı kullanmanız yasaktır. Mesela bu dersin daha önceki dönemlerden gelen ders notlarını veya benzer derslerin ders notlarını veya internetteki içeriği kullanmanız yasaktır. Sınavınızı teslim etmiş olsanız bile, dersin öğretim kadrosu hariç kişilerle sınav ile ilgili iletişim kurmanız 29 Nisan'a kadar yasaktır.

Herhangi bir anda, bu ilkeyi ihlal ettiğinizi düşünürseniz, dersin eğitmenleriyle bir an önce irtibata geçin. Eğer, bir kaynağın kullanılıp kullanılamayacağına dair sorunuz olursa, bu soruyu sınavdan önce dersin eğitmenlerine e-posta ile bildirebilirsiniz.

Anket: Bu sınava ekli olarak bir anket bulacaksınız. Bu anket özellikle akademik dürüstlüğe ilişkin sınav tecrübelerinizle ilgili. Lütfen bu anketi sınav kağıtlarından ayırıp doldurun ve sınavlarla beraber teslim edin. Bu anket isimsiz olarak teslim edilecektir. Anketteki herhangi bir cevaptan ötürü kişilere bir sorumluluk yüklenmeyecektir. Bu bilgiler sınavın faydasını ölçmek için kullanılacak ve istatistiklerin özeti sınıfta duyurulacaktır.

Lütfen Sınav Süresince yukarıdaki açıklamaları her gün tekrar okuyun.
İYİ ŞANSLAR, İYİ EĞLENCELER!

Problem 1. ***Statik Grafik Gösterimi***

$G = (V, E)$ yönlendirilmemiş seyrek bir grafik olsun. $V = \{1, 2, \ldots, n\}$ olsun. $v \in V$ ve $i = 1, 2, \ldots, \text{out-degree}(v)$ için, *v*'nin ***i'ninci komşusunu***, *i*'ninci en küçük *u* köşesi olarak tanımlayın. Öyle ki, $(v, u) \in E$ olduğunda, v'ye komşu olan köşeleri sıralarsak; *u, i*'inci küçük köşe olsun.

Aşağıdaki sorguları sağlayan G grafiğinin bir gösterimini hazırlayın.

- DEGREE(v): v köşesinin derecesini döndürür.
- LINKED(u, v): Eğer bir kenar, *u* ve *v* köşesini bağlıyorsa; DOĞRU döner, aksi halde YANLIŞ döner.
- NEIGHBOR(v, **i**): *v*'nin *i*'inci komşusunu döndürür.

Veri yapınız asimptotik olarak, olabildiğince az yer kaplamalıdır. İşlemler de asimptotik olarak olabildiğince hızlı çalışmalıdır; ancak yer süreden daha önemlidir. Veri yapınızı yer ve süre bakımından analiz edin.

Problem 2. ***Video Oyun Tasarımı***

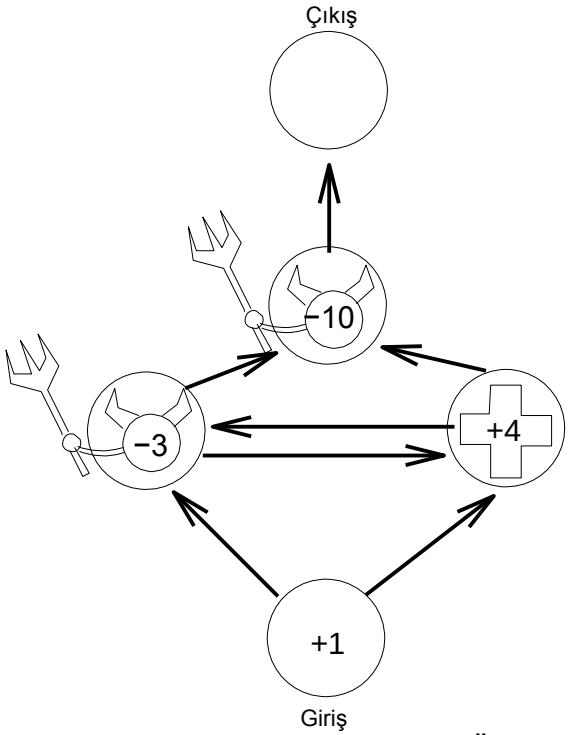
Profesör Cloud, uzun zamandır beklenen ***Take-home Fantasy*** isimli yılın oyununun tasarımına danışmanlık yapıyor. Oyundaki seviyelerden birinde, oyuncular girişten çıkışa gitmek için bir labirentteki birçok odayı gezmek zorunda. Odalarda hayat iksiri veya canavar olabileceği gibi, odalar boş da olabilir. Oyuncu odalarda dolaşırken *L* ***yaşam puanları*** azalır veya artar. Hayat iksiri içmek *L*'yi arttırırken, canavarla savaşmak *L*'yi azaltır. Eğer ***L***, 0'a veya altına düşerse oyuncu ölür.

Şekil 1'de gösterilen bu labirent; köşelerin odaları, tek yönlü kenarların da koridorları simgelediği bir *G=(V,E)* yönlendirilmiş grafiği ile simgelenebilir. Bir $\mathbf{f} : V \rightarrow Z$ köşe-ağırlık fonksiyonu odanın içindekileri simgeler.

- Eğer $\mathbf{f}(v) = 0$ ise oda boştur.
- Eğer $\mathbf{f}(v) > 0$ ise odada hayat iksiri vardır. Oyuncu odaya her girdiğinde *L* yaşam puanı f(v) kadar artar.
- Eğer $\mathbf{f}(v) < 0$ ise odada canavar vardır. Oyuncu odaya her girdiğinde, *L* yaşam puanı |f(v)| kadar azalır. L artı olmazsa oyuncu ölür.

Labirente ***giriş*** odası s Є V, ***çıkış*** odası da t Є V'dir. s'den her v Є V'ye ve her v Є V'den t'ye bir yol olduğunu kabul edin. Oyuncu, $L = L_0$ yaşam puanı ile girişte oyuna başlar, yani L_0 girişteki *f* değeridir. Şekilde $L_0 = 1$'dir.

Profesör Cloud canavarları ve hayat iksirlerini labirente rastgele yerleştirmek için bir program tasarlamış, ama bazı labirentlerde girişten çıkışa ulaşabilmek, $L_0 > 0$ değilse mümkün olmayabiliyor. s'den t'ye giden yolda, eğer oyuncu sağ kalabiliyorsa bu yol "güvenli"dir, yani yaşam-puanları hiç 0 olmuyordur.Oyuncu L_0=r ile başladığında,labirentte güvenli bir yol varsa, o labirenti **r-*girilebilir*** olarak tanımlayın.

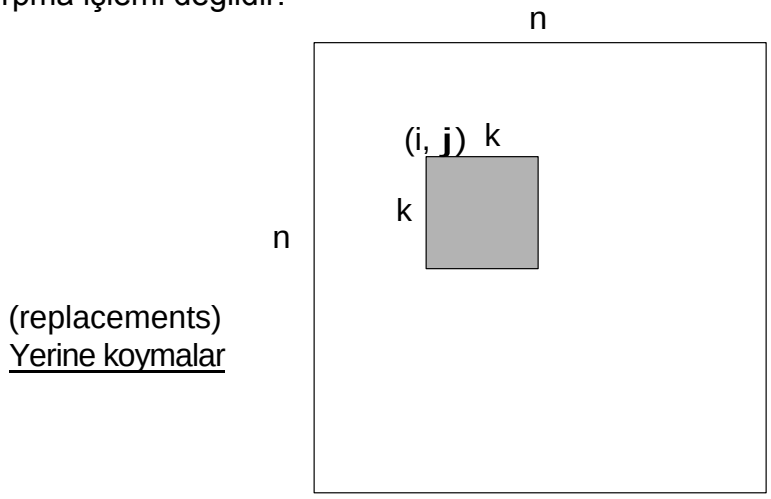


Şekil 1: 1-girilebilir Labirent Örneği

Profesöre *r*'nin en küçük değerini belirlemek için verimli bir algoritma tasarlamasında yardımcı olun; öyle ki, bu algoritma, verilen labirentin *r-girilebilir* olduğunu veya böyle bir *r* olmadığını belirlesin. (Kısmi puan almak için, bir labirentin verilen bir *r* için *r*-girilebilir olup olmadığını belirleme problemini çözün.)

Problem 3. ***Görüntü Filtreleme***

İki-boyutlu filtreleme görüntü ve resim işlemede yaygın bir işlemdir. Bir resim, gerçek sayılardan oluşan *n x n* boyutlu bir matris ile simgelenir. Şekil 2'de görüleceği gibi, *k x k* boyutunda bir pencereyi matris boyunca dolaştırmak ve penceredeki her olası yerleşim için filtrenin penceredeki tüm değerlerin "çarpımını" alması buradaki fikirdir. "Çarpım" tipik bir çarpma işlemi değildir.



Şekil 2: Çıktı elemanı b_{ij} taranmış bölgedeki bütün a'ların çarpımıdır.

Bu problem için, işlemin birleşmeli ve değişimli özelliklere sahip bir ikili bir işlem olduğunu varsayabiliriz. Bu işlemde özdeşlik elemanı ***e*** için;

$$x \otimes e = e \otimes x = x$$

Örneğin, bu işlem, 0 özdeşlik elemanı ile toplama, 1 ile çarpma, ∞ ile **min** v.s. olabilir. Burada önemli olan, bu işlemin – ve + gibi tersinin doğru olduğunu varsaymamanız.

Daha açık olmak için, n x n bir resim verilmekte,

$$A = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0(n-1)} \\ a_{10} & a_{11} & \dots & a_{1(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n-1)0} & a_{(n-1)1} & \dots & a_{(n-1)(n-1)} \end{pmatrix}$$

(k × k)-*filtrelenmiş* görüntü n × n matrisidir.

$$B = \begin{pmatrix} b_{00} & b_{01} & \dots & b_{0(n-1)} \\ b_{10} & b_{11} & \dots & b_{1(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ b_{(n-1)0} & b_{(n-1)1} & \dots & b_{(n-1)(n-1)} \end{pmatrix}$$ i, j = 0,1,...,n-1 için,

$$b_{ij} = \bigotimes_{x=i}^{i+k-1} \bigotimes_{j=y}^{j+k-1} a_{xy}$$

(Kolaylık olsun diye, $x \geq n$ veya $y \geq n$ için $a_{xy} = e$ olduğunu varsayarız.

A girdi matrisinin (k x k) - filtresini hesaplamak için verimli bir algoritma verin. Algoritmanızı çözümlerken, *k*'yı sabit bir değer olarak kabul etmeyin. Koşma sürenizi *n* ve *k* cinsinden ifade edin. (Kısmi puan almak için problemi bir boyutta çözün.)

Problem 4. ***ViTo Tasarımı***

ViTo adlı yeni ve geliştirilmiş bir dijital video kaydedici tasarlıyorsunuz. ViTo yazılımında, bir *i* televizyon gösterisi 3'lü olarak simgelenir. Bu 3'lü, **kanal numarası** c_i, **başlangıç zamanı** s_i ve **bitiş zamanı** e_i'dir. ViTo sahibi, *n* tane izlenecek gösteriyi girdi olarak listeler. $i= 1,2,....,n$ olan her gösteri bir **keyif reytingi** r_i'ye sahiptir. Gösteriler üstüste çakışabileceğinden ve ViTo aynı anda sadece bir gösteri kaydedebileceğinden, ViTo, alınan keyif reytingi en fazla olan gösterilerin alt kümesini kaydetmelidir. ViTo kullanıcısı, bir gösterinin sadece bir bölümünü seyretmekten zevk almadığından, ViTo hiçbir zaman bölüm kaydı yapmaz. ViTo'nun gösterimlerin en iyi alt kümesini seçebilmesi için verimli bir algoritma yazın.

Problem 5. ***Bir Grafiği Geliştirmek***

$G = (V, E)$ yönlendirilmiş grafiğini, dinamik olarak geliştiren bir veri yapısı geliştirmek istiyoruz. Başlangıçta, $V = \{1, 2, \ldots, n\}$ ve $E = \emptyset$'ye sahibiz. Kullanıcı, grafiği aşağıdaki işlemi kullanarak geliştirecek.

- INSERT-EDGE(u, v): u köşesinden v köşesine yönlendirilmiş bir kenar ekler. Öyle ki, $E \leftarrow E \cup \{(u,v)\}$'dir.

Ayrıca, kullanıcı istediği zaman iki köşenin bağlantılı olup olmadığını aşağıdaki işlemle sorgulayabilir.

- CHECK-PATH(u, v): Eğer u köşesinden v köşesine bir yönlendirilmiş yol mevcut ise DOĞRU döndür, aksi takdirde YANLIŞ döndür.

Kullanıcı grafiği tamamen bağlantılı olana kadar geliştirmeye devam eder. Kenarların sayısı tekdüze olarak arttığından ve kullanıcı aynı kenarı hiçbir zaman 2 kere araya yerleştirmediğinden, Toplam INSERT-EDGE işlemlerinin sayısı tam olarak n(n-1)'dir. Kullanıcı grafiği geliştirmeye devam ettiği sürece, *m* kere CHECK-PATH işlemini çağırır ve bu işlemler $n(n-1)$ INSERT-EDGE ile birlikte çalışır. Bu tarz işlemler dizisini verimli bir şekilde destekleyecek bir veri yapısı tasarlayın.