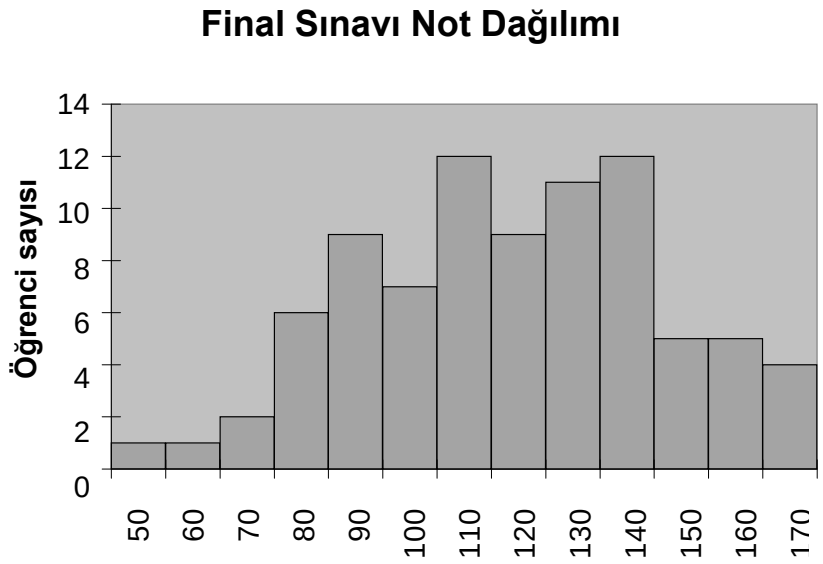
Final Sınavı
Çözümleri



Problem 1. Yinelemeler [15 puan] (3 bölüm)

Aşağıdaki her yinelemenin sıkı asimptotik üst sınırını (O simgelemi) yazın. Cevabınızı açıklamanıza gerek yok.

(a) $T(n) = 2T(n/8) + \sqrt[3]{n}$.

Çözüm: Ana Metodun ikinci durumunu kullanarak

$\Theta(n^{1/3} \lg n)$

(b) $T(n) = T(n/3) + T(n/4) + 5n$

Çözüm:

$\Theta(n)$.

(c) $$T(n) = \begin{cases} 8T(n/2) + \Theta(1) & \text{if } n^2 > M, \\ M & \text{if } n^2 \leq M; \end{cases}$$

M, n'den bağımsız bir değişken iken.

Çözüm:

$\Theta\left(n^3/\sqrt{M}\right)$

Özyineleme ağacı yaklaşık olarak

$$\lg n - \lg \sqrt{M} = \lg\left(n/\sqrt{M}\right)$$

düzeye sahip. Ağaçtaki her iç düğüm, her biri O(1) maliyetli 8 ardıla sahip. Ağacın altında yaklaşık

$$8^{\lg\left(n/\sqrt{M}\right)} = \left(n/\sqrt{M}\right)^3$$

yaprak var. Her yaprak M maliyete sahip olduğundan, toplam maliyet yapraklar tarafından domine edilir. Cevap budur.

$$\Theta\left(M\left(n/\sqrt{M}\right)^3\right) = \Theta\left(n^3/\sqrt{M}\right).$$

Problem 2. Algoritmalar ve koşma süreleri [9 puan]

Aşağıdaki algoritmaları en kötü koşma sürelerinin sıkı asimptotik üst sınırları ile eşleştirin. A'dan I'ya harfleri, algoritmaların yanındaki kutulara yerleştirin. Sıralama algoritmaları için girdi sayısı olarak n kabul edilecektir. Matris algoritmaları için büyüklük nxn olarak kabul edilecektir. Grafik algoritmaları için, köşe sayısı olarak n, kenar sayısı olarak ise Θ(n) kabul edilecektir. Cevaplarınızı açıklamak zorunda değilsiniz. Bazı koşma süreleri birden fazla algoritma için kullanılabileceği gibi, bazıları da hiçbir algoritma ile eşleşmeyebilir. Yanlış cevaptan puan kırılacağı için tahmine dayalı yanıt vermemeniz tavsiye edilir.

☐	Araya yerleştirme sıralama	A: $O(\lg n)$
☐	Yığın Sıralaması	B: $O(n)$
☐	YIĞIN-İNŞASI	C: $O(n \lg n)$
☐	Strassen	D: $O(n^2)$
☐	Bellman-Ford	E: $O(n^2 \lg n)$
☐	Derinliğine arama	F: $O(n^{2.5})$
☐	Floyd-Warshall	G: $O(n^{\lg 7})$
☐	Johnson	H: $O(n^3)$
☐	Prim	I: $O(n^3 \lg n)$

Çözüm: Yukarıdan Aşağıya: D, C, B, G, D, B, H, E, C.

Problem 3. Yerine Koyma Metodu [10 puan]
Aşağıdaki yinelemenin sıkı asimptotik alt sınırını (Ω -simgelemi) bulmak için yerine koyma metodunu kullanın.

$$T(n) = 4T(n/2) + n^2 .$$

Çözüm: Ana Metodumuzdan, $T(n) = \Theta(n^2 \lg n)$ olduğunu biliyoruz. Ayrıca tümevarım hipotezimize göre, m<n olan bütün m'ler için, $T(m) \geq cm^2 \lg m$ 'dir.

$m = 1$ ve c>0 olan bütün c'ler için, taban durumu $T(1) = 1 > c1^2 \lg 1$'dir.

Tümevarım adımı olarak, bütün m<n olan m'ler için, $T(m) \geq cm^2 \lg m$ olduğunu varsayalım. Tümevarım hipotezi,

$$\begin{aligned} T(n) &= 4T(n/2) + n^2 \\ &\geq 4c\left(\frac{n}{2}\right)^2 \lg\left(\frac{n}{2}\right) + n^2 \\ &= cn^2 \lg n - cn^2 \lg 2 + n^2 \\ &= cn^2 \lg n + (1-c)n^2. \end{aligned}$$

denklemini sağlar.

$c < 1$ için, bu değer her zaman $cn^2 \lg n$'den daha büyük olacaktır. Bu nedenle, $T(n) = \Omega(cn^2 \lg n)$'dir.

Problem 4. Doğru veya Yanlış [35 puan] (7 bölüm)

Aşağıdaki ifadelerin doğru veya yanlış olduğunu belirtmek için **D** veya **Y'**yi daire içine alın. Eğer ifade doğru ise, neden doğru olduğunu kısaca açıklayın. Eğer ifade yanlış ise, neden yanlış olduğunu kısaca açıklayın. Ne kadar ayrıntılı içerik sağlarsanız, o kadar yüksek puan alacaksınız, ama cevabınız özet olsun. Açıklamanız, D veya Y seçiminden daha fazla not getirecek.

D Y A_1, A_2 ve A_3, birbirinden farklı, n tane doğal sayının sıralı dizilimleri olsun. Karşılaştırma modelinde, $A_1 \cup A_2 \cup A_3$ kümesinin dengeli bir ikili arama ağacına uyarlanması $\Omega(n \lg n)$ sürealır.

Çözüm: Yanlış. Önce 3 dizilimi, O(n) sürede birleştirin. Birleştirilmiş dizilimden ikili arama ağacı oluşturun: Dizilimin ortancası, ağacın kökü olacak şekilde, özyinelemeli olarak dizilimin birinci yarısından sol alt ağacı, ikinci yarısından sağ alt ağacı oluşturun. Sonuçta oluşan koşma süresi
$T(n) = 2T(n/2) + O(1) = O(n)$'dir.

D Y T, n tane anahtarlı tam bir ikili ağaç olsun. T'nin kökünden verilen bir $v \in T$ köşesine, enine arama kullanarak bir yol bulmak $O(\lg n)$ süreye mal olur.

Çözüm: Yanlış. Enine arama $\Omega(n)$ süreye mal olur. Enine arama, her ağaçtaki her düğümü önce enine göre inceler. v köşesi incelenen son köşe de olabilir. (T'nin sıralı olması gerekmediğini de hesaba katın)

D Y A[1 . . n], n tane tamsayılı sırasız bir dizilim veriliyor. A'nın elemanlarından bir maks-yığın oluşturmak, kırmızı-siyah ağaç oluşturmaktan asimptotik olarak daha hızlıdır.

Çözüm: **Doğru.** Bir yığın oluşturmak O(n) süre alır. Buna karşın, bir kırmızı-siyah ağaç oluşturmak, Ω (n lg n) süre alır, çünkü kırmızı-siyah ağaçtan sıralı bir liste üretmek, sırasıyla geçiş ağaç yürüyüşüyle O(n) süresi gerektirir (ama karşılaştırma modelinde sıralama Ω (n lg n) süre gerektirir).

D Y h kıyım fonksiyonu kullanarak, n tane ayrışık anahtarı, m uzunluğundaki T dizilimine yerleştirdiğimizi düşünelim. Basit kıyım kullanarak, çiftlerin beklenen çarpışma sayısı:
$\Theta(n^2/m)$

Çözüm: **Doğru.** $X_{i,j}$ göstergesel rastgele değişken olsun. Eğer i ve j çarpışırsa 1, çarpışmazsa 0 olsun. Basit kıyımda, i elemanının k yuvasına kıyım olasılığı 1/m'dir. Yani, i ve j'nin aynı yuvaya kıyılma ihtimali $\Pr(X_{i,j}) = 1/m$'dir. Böylece $\mathrm{E}[X_{i,j}] = 1/m$'dir. Şimdi beklenenin doğrusallığını kullanarak, tüm i ve j çiftlerini toplarız.

$$\begin{aligned}
\text{E[çarpışan çiftlerin sayısı]} &= \mathrm{E}\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{i,j}\right] \\
&= \sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathrm{E}\left[X_{i,j}\right] \\
&= \sum_{i=1}^{n}\sum_{j=i+1}^{n} 1/m \\
&= \frac{n(n+1)}{2m} \\
&= \Theta(n^2/m)
\end{aligned}$$

D Y n girdili bütün sıralama ağları Ω (lg n) derinliğe sahiptir.

Çözüm: Doğru. d ağın, derinliği olsun. Ağda, n girdi varken, en fazla nd kadar karşılaştırıcı olabilir. (Buna bakmanın bir şekli de güvercin-deliği prensibidir: Eğer *n* adet tel ve *nd*'den fazla karşılaştırıcı varsa, bazı teller, d tane karşılaştırıcıdan daha fazlasından geçmelidir; böylece derinlik > *d* olur.)

Karşılaştırma tabanlı modelde, aynı anda sadece bir karşılaştırıcı kullanarak sıralama ağını benzetimleyebiliriz. Koşma süresi, karşılaştırıcıların sayısına eşittir (çarpı sabit bir gider). Böylece, her sıralama ağı, karşılaştırma-tabanlı modeldeki sıralama alt sınırı nedeniyle, en az Ω (n lg n) tane karşılaştırıcıya sahip olmak zorundadır.

Sonuçta, nd > Ω (n lg n)'dir ve bu durumda d, Ω (lg n) olur.

D Y Eğer dinamik bir programlama problemi en uygun altyapı özelliğini sağlıyor ise, yerel olarak en uygun çözüm, her yerde en uygundur.

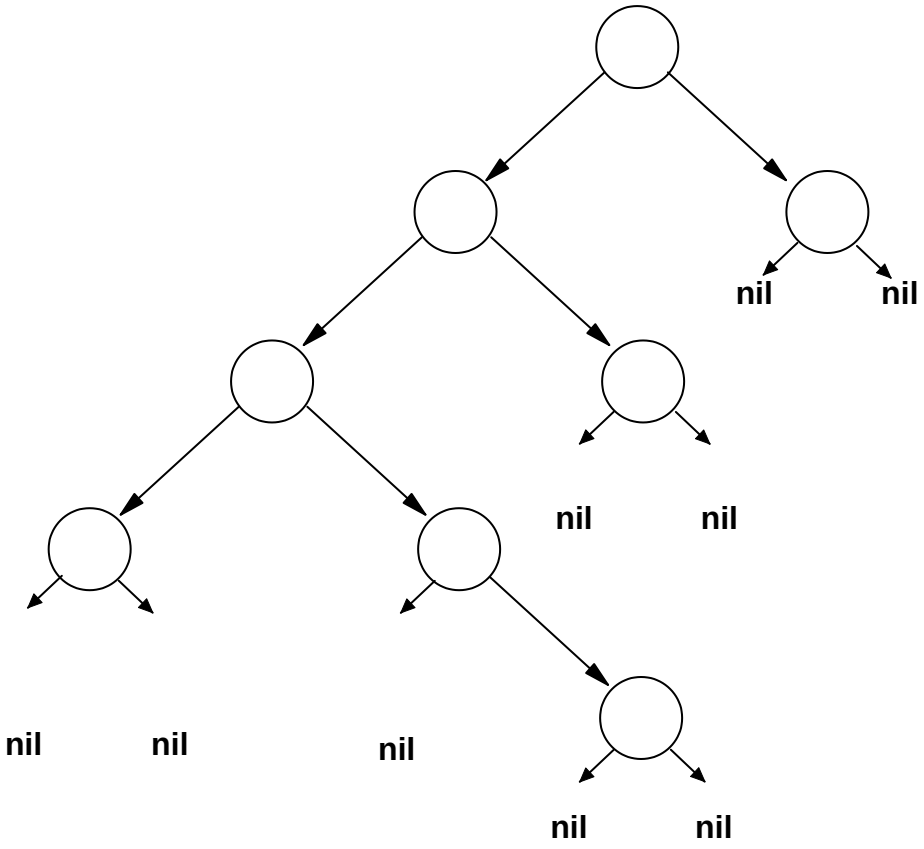
Çözüm: Yanlış. Yerel en uygun çözümün her yerde en uygun olduğunu gösteren özellik, *açgözlü seçim özelliğidir*. "Uzaklık düzenleme" problemini bir karşı örnek olarak düşünün. Bu problem, problem setinde gösterildiği gibi en uygun altyapıya sahiptir ama, yerel olarak en uygun çözüm (bir sonraki düzenlemeyi en iyi yapan) evrensel, yani her yerde en uygun olan çözüm değildir.

D Y $G = (V, E)$, negatif ağırlıklı kenarları olan, ancak negatif ağırlık çevrimi olmayan yönlendirilmiş bir grafik olsun. Öyleyse, $s \in V$ kaynağından tüm $v \in V$'lere bütün en kısa yollar, yeniden ağırlıklandırma tekniğini kullanarak, Bellman-Ford'dan daha hızlı hesaplanabilir.

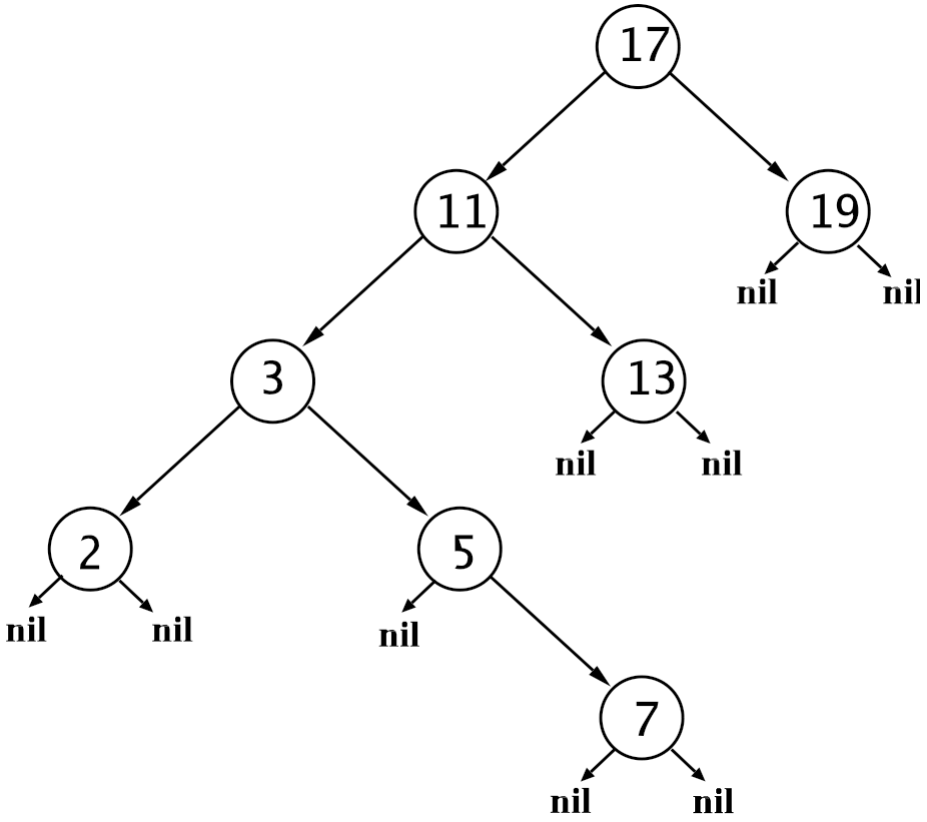
Çözüm: **Yanlış.** Yeniden ağırlıklandırma tekniği, her $v \in V$ köşesine, bir $h(v)$ değeri atayarak en kısa yolu saklar; bu değeri kullanarak kenarların yeni değerini hesaplar: $\tilde{w}(u, v) = w(u, v) + h(u) - h(v)$. Ancak, kenar ağırlıklarının eksi olmadığı durumlarda, $h(v)$ değerlerini bulabilmek için, fark kısıtlarını kullanan Bellman-Ford yöntemini kullanırız. Yeniden ağırlıklandırma tekniği de Bellman-Ford yöntemine dayandığından, daha hızlı koşması mümkün değildir.

Problem 5. Kırmızı-siyah ağaçlar [15 puan] (3 bölüm)

(a) 2, 3, 5, 7, 11, 13, 17, 19 anahtarlarını, aşağıdaki ikili arama ağacına, ikili arama ağacı özelliklerini karşılayacak şekilde yerleştirin.



Çözüm: Doğru cevap için 5 puan verilecektir.



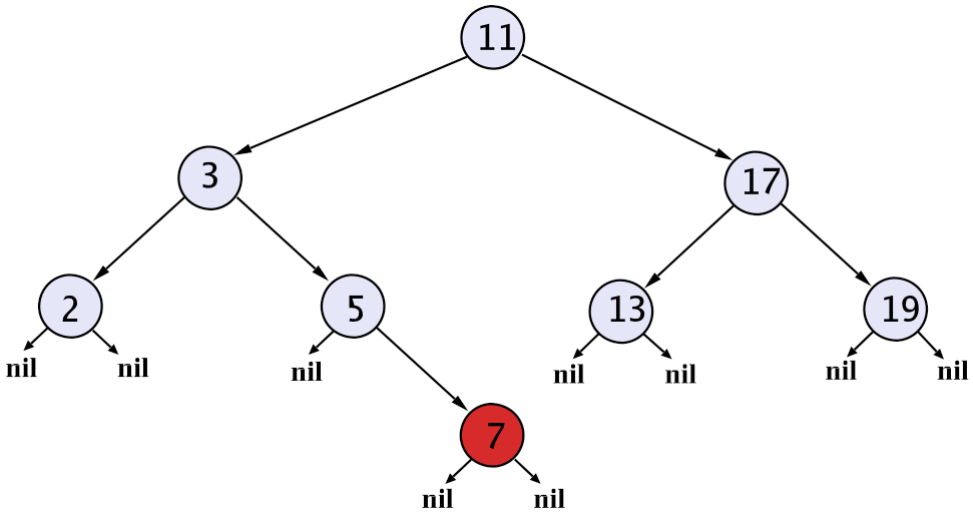
(b) Bu ikili arama ağacının kurallara uygun bir kırmızı-siyah ağaç olarak renklendirilememesinin nedenini açıklayın.

Çözüm: Bunu çelişki ile kanıtlarız. Uygun bir renklendirmenin olduğunu varsayalım. Kırmızı-siyah ağaçlarda, bir düğümden alttaki yapraklara giden bütün yollar eşit sayıda siyah boğuma sahip olmalıdır. (17, 19, NIL) yolu, en fazla 3 siyah düğüm bulundurabilir. Bu nedenle, (17, 11, 3, 5, 7, NIL) yolu da en fazla 3 siyah ve en az 3 kırmızı boğum bulundurmalıdır. Kırmızı-siyah ağaçların özelliklerine göre, kök 17 ve NIL düğümü de siyah olmalıdır. Bunun anlamı yolun üzerindeki 3 düğümün kırmızı olması demektir. (11, 3, 5, 7), ama bu da iki ardışık düğümün kırmızı olması anlamına gelir ve bu da kırmızı siyah ağaçların kurallarına aykırıdır. Bu bir çelişkidir ve bu nedenle bu ağaçta, kurallara uygun bir kırmızı-siyah ağaç renklendirilmesi yapılamaz.

Doğru cevap için 5 puan verilecek. Kırmızı-siyah ağaçların kurallarını yazana ve ağacın neden bu şekilde renklendirilemeyeceğini yazana, ancak neden kurallara uymadığını kanıtlayamayana 2-4 puan verilecek. 1 puan da ağacın dengesiz olduğunu yazana verilecek.

(c) Bir tane döndürme yapılarak ikili arama ağacı bir kırmızı-siyah ağaca dönüştürülebilir. Bunu sağlayan kırmızı-siyah ağacı, her düğümü kırmızı veya siyah olarak etiketleyerek çizin. Bölüm (a)'daki anahtarları kullanın.

Çözüm: Kökten sağa doğru döndürün. Oluşacak ağacı renklendirmenin farklı yolları var. Burada bunlardan biri var. (7 hariç bütün düğümler siyah).



Doğru cevap için 5 puan verilecek. Doğru döndürme ama yanlış renkendirme için 2 puan verilecek.

Problem 6.Wiggly (oynak) dizilimler [10 puan]

A[1 . . 2n + 1] dizilimi, eğer A[1] ≤ A[2] ≥ A[3] ≤ A[4] ≥ . . . ≤ A[2n] ≥ A[2n + 1] ise wiggly yani oynaktır. Sıralı olmayan ve gerçek sayılardan oluşan bir B[1 . . 2n + 1] dizilimi verildiğinde, B'nin bir oynak alt dizilimi olacak şekilde A[1 . . 2n + 1] permutasyonunu çıkaracak verimli bir algoritma tanımlayın.

Çözüm: Bu problemi Θ(n) sürede çözmenin çeşitli yolları var. B'nin ortancası k'yı belirleyici Θ(n) seçme algoritmasıyla bulabiliriz ve B'yi ortancanın etrafında eşit boyutlu $B_{aşağı}$ ve $B_{yukarı}$ olarak 2 parçaya bölüntüleriz. A[1]'e k'yı atarız. Daha sonra, **i** > 1 olan her i için, eğer **i** çift sayı ise, $B_{yukarı}$'dan A[**i**]'ye bir eleman atarız. Aksi takdirde, $B_{aşağı}$'dan A[**i**]'ye bir eleman atarız. $B_{yukarı}$'daki bütün elemanlar, $B_{aşağı}$'daki bütün elemanlardan büyük-eşit olduklarından ve ortanca da $B_{yukarı}$'ya küçük-eşit olduğundan, dizilim wiggly/oynak'dır. Toplam koşma süresi Θ (n)'dir.

Θ (n) doğru çözümü ve çözümlemesi için 10 puan verilecek. Doğru çözüm bulunur ama çözümlemede bir basamak atlanırsa 8-9 puan verilecek.

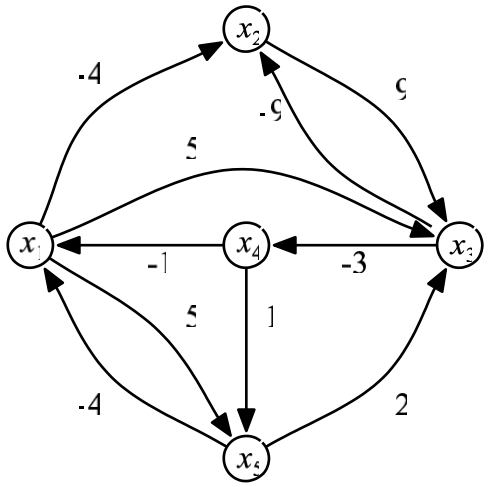
Θ (n lg n) doğru çözümüne ve analizine 5 puan verilecek. Doğru cevap verilir ama çözümleme tam yapılmazsa 2-4 puan verilecek.

Problem 7. Fark Kısıtları [12 puan] (2 bölüm)

Aşağıdaki fark kısıtlarının doğrusal programlama sistemini düşünün. (bir kısıtın eşitlik olduğuna dikkat edin):

$$\begin{aligned} x_1 - x_4 &\le -1 \\ x_1 - x_5 &\le -4 \\ x_2 - x_1 &\le -4 \\ x_2 - x_3 &= -9 \\ x_3 - x_1 &\le 5 \\ x_3 - x_5 &\le 2 \\ x_4 - x_3 &\le -3 \\ x_5 - x_1 &\le 5 \\ x_5 - x_4 &\le 1 \end{aligned}$$

(a) Bu kısıtlar için kısıt grafiğini çizin.



Çözüm: Eşitlik kısıtı,2 eşitsizlik kısıtı olarak yazılabilir, $x_2 - x_3 \le -9$, ve $x_3 - x_2 \le 9$.
Tamamen doğru bir kısıt grafiği, 6 puan alır.
Kenarları yanlış yönde çizerseniz, sadece 4 puan alırsınız. Eşitlik özelliğini doğru çeviremeyenler 3 puan alırlar.

(b) x_1, x_2, x_3, x_4 ve x_5 bilinmeyenleri için çözümü bulun veya neden çözüm olmadığını açıklayın.

Çözüm: Bu sistem için her hangi bir çözüm yoktur, çünkü kısıt tablosunda negatif ağırlıklı bir çevrim var. Örneğin, $x_1 \rightsquigarrow x_3 \rightsquigarrow x_4 \rightsquigarrow x_5 \rightsquigarrow x_1$ 'in ağırlığı $5-3+1-4 = -1$ olur.
Negatif ağırlıklı devreyi içeren tam cevap 6 puan alacak.

Problem 8. Amortize edilmiş artım (increment) [12 puan]
Bitlerden oluşan A[0 . . k − 1] dizilimi (her eleman 0 veya 1) bir ikili x sayısını saklamaktadır

$$x = \sum_{i=0}^{k-1} A[i] \cdot 2^i.$$

(2^k modunda) 1'i x'e eklemek için aşağıdaki yöntemi takip ederiz.

```
INCREMENT(A, k)
1  i ← 0
2  while i < k and A[i] = 1
3        do A[i] ← 0
4           i ← i + 1
5  if i < k
6     then A[i] ← 1
```

Verilen bir x sayısı için, potansiyel $\Phi(x)$'i, x'in ikili yazımındaki 1'lerin sayısı şeklinde tanımlayın. Mesela, $\Phi(19)=3$, çünkü $19 = 10011_2$. Potansiyel fonksiyon tartışmasını, bir artımın amortize maliyetinin, x'in başlangıç değeri x=0 iken, O(1) olduğunu ispatlamak için kullanın.

Çözüm: $\Phi(x)$, 1'lerin sayısından oluşan ve hiçbir zaman negatif olmayan geçerli bir potansiyel fonksiyondur ve tüm x'ler için $\Phi(x) \geq 0$. Sayacımızın başlangıç değeri 0 iken $\Phi_0 = 0$. c_k , INCREMENT(A,k) işlemimizin gerçek maliyeti olsun ve d_k da 3. ve 4. satırlardaki döngümüzün kaç kere çalıştığının sayısı olsun. (yani d_k, x'in ikili gösteriminde, en önemsiz bitten başlayarak ardışık gelen 1'lerin sayısıdır.) Eğer 1, 2, 5 ve 6. satırların hepsinin koşmasının, bir birim maliyeti olduğunu varsayarsak ve döngünün de bir birim maliyet olduğunu düşünürsek, gerçek maliyetimiz;
$c_k = 1 + d_k$ olur.

Potansiyel, her döngüye girdiğimizde 1 azalır. Bu nedenle, potansiyeldeki değişim
$\Delta\Phi = \Phi_k - \Phi_{k-1}$,en fazla
$1 - d_k$ olur. Yani, eğer 6. satırı çalıştırırsak $\Delta\Phi = 1 - d_k$ olurken, çalıştırmazsak, $\Delta\Phi = -d_k$ olur.
Amortize edilmiş maliyet formulünü çağırırsak;

$$\begin{aligned} \hat{c_k} &= c_k + \Delta\Phi \\ &= 1 + d_k + \Delta\Phi \\ &\leq 1 + d_k + 1 - d_k \\ &= 2. \end{aligned}$$

Amortize edilmiş maliyet O(1) olur.

Sadece, gerçek maliyeti hesaplayıp açıklayanlar tam puan alırlar. Başka bir metodu kullanan ya da potansiyel metodu kullanmayanlar en fazla 6 puan alabilecekler.

Φ_0 hesaplanmamışsa ya da potansiyel fonksiyonun neden geçerli olduğu açıklanmamışsa doğru çözümlerden 1-2 not kırılmıştır.

Problem 9. En az yayılan/kapsayan ağaçlar [12 puan]

G = (V, E), w : E → R kenar ağırlık fonksiyonu olan, bağlantılı ve yönsüz bir grafiktir. Bu grafikte bütün kenarların ağırlıklarının farklı olduğunu varsayın. G'deki $(v_1, v_2, \ldots, v_k, v_{k+1})$ çevriminde $v_{k+1} = v_1$ olduğunu ve (v_i, v_{i+1}) kenarının, çevrimdeki en ağır kenar olduğunu varsayın. (v_i, v_{i+1}) kenarının, G'nin en az yayılan ağacı T'ye ait olmadığını ispatlayın.

Çözüm: Çelişki ile ispat. (v_i, v_{i+1}) kenarının en az yayılan T ağacına ait olduğunu varsayalım. (v_i, v_{i+1})'yi T'den ayırmak, bağlı iki P ve Q bileşeni oluşturur ve çevrimin bazı düğümleri P'de, bazıları da Q'da yer alır. Herhangi bir çevrimde en az iki kenar bu kesişimden geçmelidir, yani (v_j, v_{j+1}) gibi bir kenar daha P ile Q'yu bağlar ve bu başka bir **T'** yayılan ağacını oluşturur. (v_j, v_{j+1})'nin ağırlığı (v_i, v_{i+1})'inkinden daha az olacağından, T' nün ağırlığı T den azdır ve T en az yayılan ağaç olamaz. Çelişki.

Problem 10. Çok izlekli çizelgeleme [10 puan]
Aç gözlü bir çizelgeleyici, çok izlekli bir hesaplamayı 4 işlemci ile 260 saniyede, 32 işlemci ile 90 saniyede bitiriyor. $T1 = 1024$ ve kritik yol uzunluğu $T_\infty = 64$ iken
Hesaplamanın çalışma ihtimali var mı? Cevabınızı açıklayın.

Çözüm : Aç gözlü çizelgeleyiciler için,

$$\min\{T_1/P, T_\infty\} \leq T_P \leq T_1/P + T_\infty$$

olduğunu biliyoruz. Yukarıda verilen sayılar bu eşitsizlikleri P'nin her iki değeri için karşılıyor. Dolayısıyla, bu hesaplamanın çalışma ve kritik yolunun doğru olma olasılığı vardır.

Problem 11. Bir matrisin ters çevrilmesi [40 puan] (4 bölüm)
X, N × N boyutlu ve N'nin 2'nin tam kuvvetinde olduğu bir matris olsun. Aşağıdaki kod $Y = X^T$'yi hesaplar:

```
TRANS(X, Y, N)
1  for i ← 1 to N
2       do for j ← 1 to N
3            do Y[j, i] ← X[i, j]
```

Önbelleği hesaba katmayan 2 seviyeli hafıza modelinde, ön belleğin M tane eleman ve B tane blok elemana sahip olduğunu düşünün. X ve Y matrislerinin ikisinin de satır öncelikli düzende düzenlendiğini, X'in doğrusal düzeninin bellekte, X [1, 1], X [1, 2], . . . , X [1, N], X [2, 1], X [2, 2], . . . , X [2, N], . . . , X [N, 1], X [N, 2], . . . , X [N, N], olduğunu ve aynı şeyin Y için de geçerli olduğunu varsayın.

(a) TRANS'daki, N ≫ M iken meydana gelen hafızaya aktarım sayısı $MT(N)$'yi çözümleyin.

Çözüm: Döngü her seferinde X'in bir satırını taradığından, X'e erişim için gereken hafızaya aktarım sayısı $O(N^2/B)$'dir. Buna karşılık, Y'ye erişim için $O(N^2)$ aktarıma ihtiyacımız olur, çünkü Y'de her seferinde bir sütuna erişiriz. Y[**j**, **i**]'yi içeren bloğa eriştiğimizde, N ≫ M olduğundan, Y[**j** + 1, **i**]'e erişmeden i sütununun tamamını taradığımız için Y'ye her erişim hafızaya aktarımı gerektirebilir. Bu nedenle, $MT(N) = O(N^2)$'dir.

Şimdi, ters çevirmeyi yapabilmek için böl ve fethet yöntemini ele alalım:

```
R-TRANS(X, Y, N)
1  if N = 1
2     then Y[1, 1] ← X[1, 1]
3     else  X'i 4 parçaya bölüntüle (N/2) × (N/2) altmatrisler X11, X12, X21, ve X22.
4           Y'i 4 parçaya bölüntüle (N/2) × (N/2) altmatrisler Y11, Y12, Y21, ve Y22.
5           R-TRANS(X11, Y11, N/2)
6           R-TRANS(X12, Y21, N/2)
7           R-TRANS(X21, Y12, N/2)
8           R-TRANS(X22, Y22, N/2)
```

Bölüntülemenin maliyetinin O(1) olduğunu varsayın.

(b) R-TRANS'daki, N ≫ M iken oluşan $MT(N)$ hafızaya taşıma sayısının yinelemesini verin ve çözün.

Çözüm: Eğer, uzun ön bellek varsayımı yaparsak (yani, $M = \Omega(B^2)$), veya matrisin özyinelemeli blok tasarımında saklandığını varsayarsak bu durumda,

$$MT(n) = \begin{cases} 4T(n/2) + \Theta(1) & \text{if } cn^2 > M, \\ M/B & \text{if } cn^2 \le M \end{cases} .$$

Yineleme ağacının yaklaşık

$$\lg N - \lg(\sqrt{M/c})$$

düzeyi vardır. Her düzey bir önceki düzeyin 4 katı düğüme sahip olduğundan, yinelemenin çözümü yapraklar tarafından domine edilir. Ağaç her biri M/B maliyete sahip

$$4^{\lg(N\sqrt{c}/\sqrt{M})}$$

yaprağa sahiptir. Bu nedenle

$$\begin{aligned} MT(N) &= \frac{M}{B}\left(\frac{N\sqrt{c}}{\sqrt{M}}\right)^2 \\ &= \frac{cN^2}{B} \\ &= O\left(\frac{N^2}{B}\right). \end{aligned}$$

Aşağıdaki çokizlekli algoritma matris ters çevirmesini paralel olarak hesaplar.

```
P-TRANS(X, Y, N)
1  if N = 1
2     then Y[1, 1] ← X[1, 1]
3     else  X'i  4'e bölüntüle (N/2) × (N/2) altmatrisler X11, X12, X21, and X22.
4           Y'i  4'e bölüntüle (N/2) × (N/2) altmatrisler Y11, Y12, Y21, and Y22.
5           spawn P-TRANS(X11, Y11, N/2)
6           spawn P-TRANS(X12, Y21, N/2)
7           spawn P-TRANS(X21, Y12, N/2)
8           spawn P-TRANS(X22, Y22, N/2)
9           sync
```

(c) $T_1(N)$ için yinelemeler verin ve çözün ve P-TRANS'ın kritik yol uzunluğu $T_\infty(N)$'yi bulun. Algoritmanın asimptotik paralelliği nedir?

Çözüm:

$$\begin{aligned} T_1(N) &= 4T_1\left(\frac{N}{2}\right) + O(1) \\ &= \Theta(N^2). \\ T_\infty(N) &= T_\infty\left(\frac{N}{2}\right) + O(1) \\ &= \Theta(\lg N). \end{aligned}$$

Algoritmanın paralelliği ise

T_1/T_∞, or $\Theta(N^2/\lg N)$.

Professor Kellogg koduna dikkatsizce, aşağıdaki gibi 2 yeni **sync** komutu ekliyor.

```
K-TRANS(X, Y, N)
 1  if N = 1
 2     then Y[1, 1] ← X[1, 1]
 3     else  X'i 4'e bölüntüle (N/2) × (N/2) altmatrisler X11, X12, X21, and X22.
 4           Y'i 4'e bölüntüle (N/2) × (N/2) altmatrisler Y11, Y12, Y21, and Y22.
 5           spawn K-TRANS(X11, Y11, N/2)
 6           sync
 7           spawn K-TRANS(X12, Y21, N/2)
 8           sync
 9           spawn K-TRANS(X21, Y12, N/2)
10           spawn K-TRANS(X22, Y22, N/2)
11           sync
```

(d) $T_1(N)$'nin işini yapacak yinelemeyi verin ve çözün ve K-TRANS'ın kritik yol uzunluğu $T_\infty(N)$'yi bulun. Algoritmanın asimptotik parallelliği nedir?

Çözüm: Algoritma için yapılan iş değişmez.

$$\begin{aligned} T_1(N) &= 4T_1\left(\frac{N}{2}\right) + O(1) \\ &= \Theta(N^2). \end{aligned}$$

Kritik yol ise büyür, çünkü alt problemlerden birini paralel olarak çözeriz.

$$\begin{aligned} T_\infty(N) &= 3T_\infty\left(\frac{N}{2}\right) + O(1) \\ &= \Theta(N^{\lg 3}). \end{aligned}$$

Algoritmanın parallelliği ise

$$T_1/T_\infty, \text{ or } \Theta(N^{2-\lg 3}).$$

KARALAMA KAĞIDI – Lütfen, sınav sonunda kitapçıktan ayırın.

KARALAMA KAĞIDI – Lütfen, sınav sonunda kitapçıktan ayırın.