Name :

Student ID :

Signature

13 May 2020 12:00 – 19 May 2020 23:59 Turkey Time Zone

352 Analysis of Algorithms
MIDTERM EXAM (TAKEHOME)

Bu sayfadaki uyarıları DİKKATLE OKUYUNUZ !!!

Bu ARA SINAV'ın ZAMAN SINIRI bulunmaktadır. Lütfen cevaplarınızı zamanında sisteme yükleyip gönderiniz.

Son gönderim tarihi 19 May 2020 23:59. Bu tarihe kadar gönderenlerin sınavı 100 puan üzerinden değerlendirilecektir. Sistemde yaşanabilecek yoğunlukları veya teknik aksaklıkları düşünerek cevaplarınızı son gün ve son saate bırakmayınız. Herhangi bir nedenden dolayı belirtilen tarihten sonra 20 Mayıs 23:59 'a kadar olan gönderimlerden, yani 19 Mayıs 23:59 - 20 Mayıs 23:59 arası yapılan gönderimlerden, 20 puan kesilecektir. Bu durumda sınavınız 80 üzerinden değerlendirilecktir. *20 Mayıs 23:59 'dan sonra sistem kapanacak ve gönderim yapamayacaksınız.*

Cevaplarınızı A4 kağıtlara elle yazınız. Bilgisayar vb. ortamlarda yazılan çözümler kabul edilmeyecektir.

Algoritma tasarımlarını, aksi belirtilmediği takdirde, sözde kod (pseudocode) olarak yazmanız gerekmektedir. Sözde kod formatı için derste işlediğimiz notasyonu ve formatı takip ediniz. Herhangi bir programlama dilinde yazılan kodlar kabul edilmeyecektir.

Cevaplarınızı okunaklı yazınız. İyi taranmaması veya kötü el yazısı vb. herhangi bir nedenden dolayı okuyamadığım cevaplar değerlendirilmeyecektir.

Çözümlerinizi tarayarak bir şekilde TEK bir PDF dosyasına dönüştürünüz. (örn: PDFill PDF Tools, CamScanner vb. uygulama ile)

Oluşturduğunuz pdf dosyasında cevaplarınızı sırayla yazınız ve her sayfanın üstüne Ad-Soyad, Öğrenci-No ve İmza'nızı eklemeyi unutmayınız.

Dönüştürdüğünüz dosyayı OgrenciNo.pdf olacak şekilde adlandırıp sisteme yükleyiniz. Verilen süre sınav için oldukça uzun bir süredir. Cevaplarınızı yüklemeyi son ana bırakmayınız.

Cevabınızı yükledikten sonra gönder butonuna basmayı unutmayınız.

BAŞARILAR

READ the warnings on this page CAREFULLY !!!

This MIDTERM EXAM has TIME LIMIT. Please upload and send your answers to the system on time.

The submission deadline is 19 May 2020 23:59. If you submit by this deadline, your exam will be evaluated over 100 points. Do not wait for your submission until the last day and the last hour, considering the congestions or technical problems in the system. For any reason, from the submissions made until 20 May 23:59 after the deadline, that is, submissions from 19 May 23:59 to 20 May 23:59, 20 points will be deducted. In this case, your exam will be evaluated over 80 points. *After 20 May 23:59, the system will shut down and you will not be able to submit.*

Write your answers by hand writing on A4 papers. Answers written in computer, etc. environments will not be accepted.

Unless otherwise stated, you should write the algorithm designs as pseudocode. For the pseudocode format, follow the notation and format we have seen in the class. Codes written in any programming language will not be accepted.

Write your answers legibly. Answers that I cannot read for any reason, such as not being scanned well or bad handwriting, will not be evaluated.

Scan your solutions and convert them into a SINGLE PDF file. (i.e. with applications like PDFill PDF Tools, CamScanner etc.)

Write your answers in order in the pdf file you created and do not forget to add your Name-Surname, Student-Number and Signature on each page.

Name the file you converted to StudentNumber.pdf and upload it to the system. The time given is quite a long time for the exam. Do not leave your submission to the last moment.

Do not forget to press the send button after uploading your answer.

GOOD LUCK

Q1 (20 p)	Q2 (10 p)	Q3 (20 p)	Q4 (10 p)	Q5 (15 p)	Q6 (25)	Total 100 p

O : *Big Oh*	Ω : *Big Omega*	Θ : *Big Theta*

$\log_b(a^c) = c \log_b a$	$b^{\log_b a} = a$	$c^{\log_a b} = b^{\log_a c}$
$\log_a(bc) = \log_a b + \log_a c$	$\log_a c \cdot \log_b a = \log_b c$	$\log\log a = \log(\log a)$
$\log_b(1/a) = -\log_b a$	$\log_b a = \dfrac{1}{\log_a b}$	$\log_b a = \dfrac{\log_c a}{\log_c b}$

$\sum_{i=m}^{n} 1 = \underbrace{1+1+\cdots\cdots+1}_{n-m+1\ times} = n-m+1$		$\sum_{i=1}^{n} i = 1+2+\cdots\cdots+n = \dfrac{n\,(n+1)}{2}$
$\sum_{i=1}^{n} i^2 = 1^2+2^2+\cdots\cdots+n^2 = \dfrac{n\,(n+1)\,(2n+1)}{6}$		$\sum_{i=1}^{n} i^k = 1^k+2^k+\cdots\cdots+n^k \approx \dfrac{1}{k+1}\,n^{k+1}$
$\sum_{i=0}^{n} a^i = 1+a+\cdots\cdots+a^n = \dfrac{a^{n+1}-1}{a-1} \quad (a \neq 1);$		$\sum_{i=1}^{n} i\,2^i = 1\cdot 2+1\cdot 2^2+\cdots\cdots+n\cdot 2^n = (n-1)2^{n+1}+2$
$\sum_{i=m}^{n} c\,a_i = c\sum_{i=m}^{n} a_i$	$\sum_{i=m}^{n} a_i \mp b_i = \sum_{i=m}^{n} a_i \mp \sum_{i=m}^{n} b_i$	$\sum_{i=m}^{n} a_i = \sum_{i=m}^{k} a_i + \sum_{i=k+1}^{n} a_i$, *where* $m \leq k < n$

1. (Total 20 pts.) Order the following functions according to their order of growth (from the lowest to the highest). Show your work. Directly giving the answer will not be accepted.

(Aşağıdaki fonksiyonları büyüme hızlarına (order of growth) göre küçükten büyüğe doğru sıralayınız. İşlemlerinizi gösteriniz. Doğrudan cevabı vermeniz kabul edilmeyecektir.)

A	$2^{\log_2 n} \log_2 n$
B	$\left(\sqrt{2}\right)^{\log_2 n}$
C	$(\log_2 n)^2 + 1$
D	$n!$
E	$n\, log_2(\log_2 n)$
F	$\sqrt[3]{n}$
G	$2n \log_2(2n+2)^3$
H	$8^{\log_2 n} + n^2 - 3n$
I	2^{7n}
J	7^{2n}

2. (Total 10 pts.) Suppose you are choosing between the following algorithms: (Aşağıdaki algoritmalar arasında tercih yapacağınızı düşünün):

- The recurrence relation of *Algorithm A* is (A algoritmasının özyineleme bağıntısı)
  $T_A(n) = 3\,T_A(n/2) + 1 \quad T_A(1) = 1 \;\; for \;\; n > 1 \;\; and \;\; n = 2^k$

- The recurrence relation of *Algorithm B* is (B algoritmasının özyineleme bağıntısı)
  $T_B(n) = \; T_B(n-1) + n \quad T_B(0) = 0 \;\; for \;\; n > 1$

- The recurrence relation of *Algorithm C* is (C algoritmasının özyineleme bağıntısı)
  $T_C(n) = \; 5\,T_C(n/4) + n \quad T_C(1) = 1 \;\; for \;\; n > 1 \;\; and \;\; n = 4^k$

Find the complexity of each of these algorithms (in θ notation). Order the complexities in nondecreasing order. Which one would you choose? Prove the ordering using the limit theorem. Your answer will not be accepted if you do not prove it. (Hint: When you find the complexities, you can use master theorem if possible).

(Algoritmaların karmaşıklıklarını bulunuz. Artan sırada sıralayınız. Hangisini seçerdiniz? Sıralamanızı limit teoremini kullanarak ispatlayınız. Cevabınızı ispatlamazsanız sonuç kabul edilmeyecektir. Karmaşıklıkları bulurken, eğer mümkünse, master teoremini kullanabilirsiniz.)

3. (Total 20 pts.) Consider the following algorithm.

```
ALGORITHM Mystery(n)
// Input: An integer n
// Output: ?
1. sum ← 0
2. for i ← 1 to n do
3.     p ← 1
4.     for j ← 1 to i do
5.         p ← 2 * p
6.     p ← p + 2 * i * i
7.     sum ← sum + p
8. return sum
```

3.1) (4 pts) What does this algorithm compute?

(Bu algoritma neyi hesaplıyor?)

3.2) (4 pts) What is the total number of multiplications?

(Toplamda yapılan çarpma sayısı nedir?)

3.3) (2 pts) What is the total number of additions?

(Toplamda yapılan toplama sayısı nedir?)

3.4) (10 pts) Design (write a pseudocode) a more efficient algorithm in terms of number of multiplications. Also find your algorithm's time complexity in terms of number of multiplications and prove that your algorithm is better than the above given algorithm.

(Çarpma sayısı açısından daha verimli bir algoritma tasarlayınız (sözde kod olarak yazınız). Ayrıca, algortmanızın zaman karmaşıklığını çarpma sayısı açısından bulunuz ve yukarıdaki algoritmadan daha iyi olduğunu ispatlayınız.)

4. (Total 10 pts.) Solve the following recurrence relation by backward substitution for $n = 2^k$

(Özyineleme bağıntısını geri yerine yerleştirme metodu ile $n = 2^k$ için çözünüz.)

$T(n) = 7\,T(n/2)\ + n^2 - 1 \qquad for\ \ n > 1 \qquad T(1) = 1$

5. (Total 15 pts.)

Let $A[0 \ldots n-1]$ be an array of n real numbers. A pair $(A[i], A[j])$ is said to be an inversion if these numbers are out of order., i.e. $i < j$ but $A[i] > A[j]$.

($A[0 \ldots n-1]$ n reel sayıdan oluşan bir dizi olsun. $(A[i], A[j])$ çiftine eğer sırasız ise yani $i < j$ iken $A[i] > A[j]$ ise, evirme (inversion) olarak söylenir.)

a) (10 pts) Design (write a pseudocode) an $O(\log n)$ time complexity algorithm for computing the number of inversions.

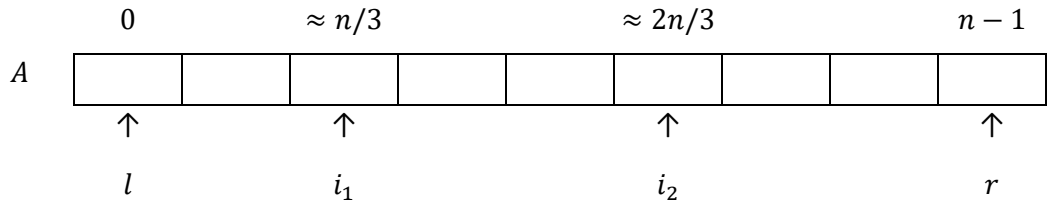
(Evirme sayılarının hesabı için $O(\log n)$ zaman karmaşıklı bir algoritma tasarlayınız (sözde kod olarak yazınız.)

b) (5 pts) Analyze your algorithm. Find the exact equation of the time complexity of the algorithm you design.

(Algoritmanızı analiz ediniz. Tasarladığınız algoritmanın zaman karmaşıklığını tam bir denklem olarak bulunuz.)

6. (Total 25 pts.)

Consider ternary search --- the algorithm for searching in a sorted array $A[0..n-1]$. The algorithm determines two elements (one at $\approx n/3$ and the other is at $\approx 2n/3$ position) within the range of the indices such that all three regions are almost equal to each other. The algorithm searches recursively by comparing K with the element at $\approx n/3$ ($A[\lfloor n/3 \rfloor]$), and if K is larger, compares it with the element at $\approx 2n/3$ ($A[\lfloor 2n/3 \rfloor]$). The algorithm recursively calls itself until it finds the key K or it returns -1 to indicate a failure.



a) (10 pts) Design (write a pseudocode) a recursive algorithm for ternary search.

(Üçlü arama için özyinemeli bir algoritma tasarlayınız (sözde kod olarak yazınız.)

ALGORITHM $TernarySearch\,(A, l, r, K)$
$//\ Implements\ recursive\ ternary\ search$
$//$**Input**$: An\ array\ A[l\,..r]sorted\ in\ ascending\ order\ and\ a\ search\ key\ K$
$//$ **l**$: lower\ index$ **r**$: upper\ index\ of\ the\ array\ A$
$//$**Output**$: An\ index\ of\ the\ array's\ element\ that\ is\ equal\ to\ K$
$//\qquad or\ -1\ if\ there\ is\ no\ such\ element$

b) (5 pts) Set up a recurrence relation $T_T(n)$ for the number of key comparisons in the worst case. (Assume that $n = 3^k$) (En kötü durum için $T_T(n)$ özyineleme bağıntısını kurunuz. $n = 3^k$ olduğunu düşününüz.)

$T_T(n) =$ [ ] $\quad for\ \ n > 1\ \ and\ n = 3^k \qquad T_T(1) =$ [ ]

c) (5 pts) Solve the recurrence $T_T(n)$ by backward substitution for $n = 3^k,\ k \geq 0$.
($T_T(n)$ bağıntısını $n = 3^k,\ k \geq 0$ için geri yerine yerleştirme metodu ile çözünüz.)

d) (5 pts) Compare this algorithm's efficiency with that of binary search. (In binary search, the worst case for the number of key comparisons is $T_B(n) = 2\log_2 n + 2$. You can use limit-based approach.)
(İkili arama algoritması ile verimlilikleri karşılaştırınız. İkili arama için en kötü durumda yapılan karşılaştırma sayısı $T_B(n) = 2\log_2 n + 2$ 'dir. - Limit-tabanlı yaklaşımı kullanabilirsiniz)