Solution

**Q1) (10 points) Asymptotic notation**

Explain whether the statement, "The running time of algorithm A is at least $O(n^2)$" makes sense or not.

*Big O notation provides an idea about the worst running time of an algorithm (worst case). Therefore, the statement should be organized as follows:*

*"The running time of algorithm A is at* ***most*** *$O(n^2)$"*

*or*

*"The running time of algorithm A is at least* $\boldsymbol{\Omega}(n^2)$*"*

**Q2) (20 pts) Insertion sort / Mergesort**

Although merge sort runs in $\Theta(n\lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small $n$. Thus, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length $k$ are sorted using insertion sort and then merged using the standard merging mechanism, where $k$ is a value to be determined.

**a) (10 pts)** Calculate the worst-case running time for sorting the n/k sublists, each of length $k$, and sorted by insertion sort.

Insertion sort takes $\Theta(k^2)$ time per $k$-element list in the worst case. Therefore, sorting $n/k$ lists of $k$ elements each takes $\Theta(k^2 n/k) = \Theta(nk)$ worst-case time.

**b) (10 pts)** Calculate the worst-case running time to merge the sublists.

Just extending the 2-list merge to merge all the lists at once would take $\Theta(n \cdot (n/k)) = \Theta(n^2/k)$ time ($n$ from copying each element once into the result list, $n/k$ from examining $n/k$ lists at each step to select next item for result list).

To achieve $\Theta(n\lg(n/k))$-time merging, we merge the lists pairwise, then merge the resulting lists pairwise, and so on, until there's just one list. The pairwise merging requires $\Theta(n)$ work at each level, since we are still working on $n$ elements, even if they are partitioned among sublists. The number of levels, starting with $n/k$ lists (with $k$ elements each) and finishing with 1 list (with $n$ elements), is $\lceil\lg(n/k)\rceil$. Therefore, the total running time for the merging is $\Theta(n\lg(n/k))$.

Sol

**Q3) (20 pts) Probabilistic Analysis and Randomized Algorithms / Indicator Random Variables**

Calculate the expected running time of RAND-SELECT.

$$T(n) = \begin{cases} T(\max\{0, n-1\}) + \Theta(n) & \text{if } 0 : n-1 \text{ split,} \\ T(\max\{1, n-2\}) + \Theta(n) & \text{if } 1 : n-2 \text{ split,} \\ \vdots & \\ T(\max\{n-1, 0\}) + \Theta(n) & \text{if } n-1 : 0 \text{ split,} \end{cases}$$

$$= \sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n)).$$

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right]$$

Take expectations of both sides.

$$\begin{aligned} E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right] \\ &= \sum_{k=0}^{n-1} E[X_k (T(\max\{k, n-k-1\}) + \Theta(n))] \\ &= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max\{k, n-k-1\}) + \Theta(n)] \\ &= \frac{1}{n}\sum_{k=0}^{n-1} E[T(\max\{k, n-k-1\})] + \frac{1}{n}\sum_{k=0}^{n-1} \Theta(n) \\ &\le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \Theta(n) \end{aligned}$$

Upper terms appear twice.

Sol

(But not quite as hairy as the quicksort one.)

$$E[T(n)] = \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \Theta(n)$$

**Prove:** $E[T(n)] \le cn$ for constant $c > 0$.

- The constant $c$ can be chosen large enough so that $E[T(n)] \le cn$ for the base cases.

**Use fact:** $\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \le \frac{3}{8}n^2$ (exercise).

$$\begin{aligned} E[T(n)] &\le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \\ &\le \frac{2c}{n}\left(\frac{3}{8}n^2\right) + \Theta(n) \\ &= cn - \left(\frac{cn}{4} - \Theta(n)\right) \\ &\le cn, \end{aligned}$$

if $c$ is chosen large enough so that $cn/4$ dominates the $\Theta(n)$.

Name and Student ID:______________________

**Q4) (25 pts) Recurrences**
Solve the following recurrences by giving tight Θ-notation bounds. You need to provide your justification for your answer. Your solutions should be asymptotically tight.

**a) (5 pts)** Use Master Method to solve the following recursion:

$$T(n) = 2T\left(\frac{n}{2}\right) + n^3$$

$a=2 \quad b=2$

$f(n)=n^3 \quad n^{\log_b a} = n \qquad n^3 = \Omega(n^{\log_2 2+2})$

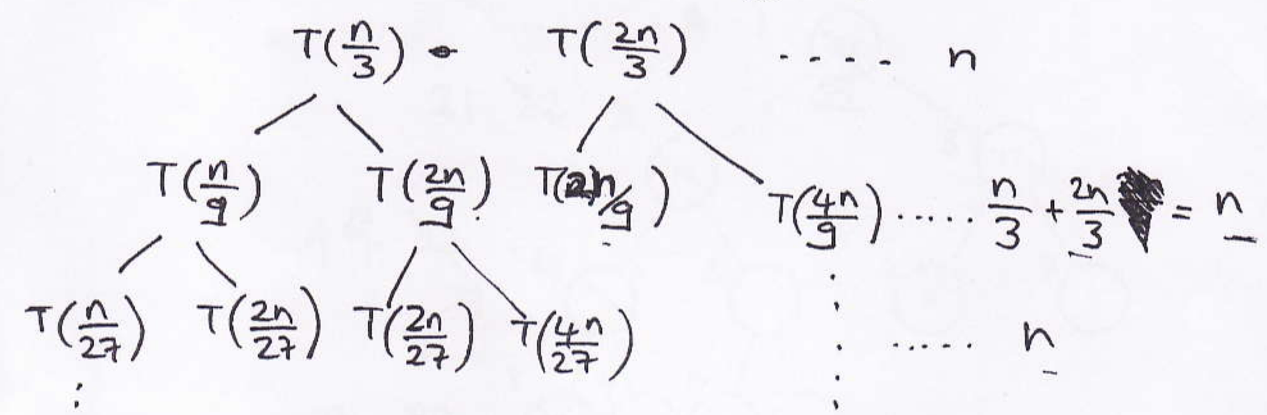
Master 3

$T(n) = \Theta(n^3)$

Gidişat yanlış, sonuç doğru ise $\textcircled{2}$

**b) (10 pts)** Use Recursion Trees Method to solve the following recursion:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$



$T(n/3^i) = 1$ $\qquad$ $T((\frac{2}{3})^i \cdot n) = 1$ $\quad \Theta(n \log n)$

height ⇒ $n = 3^i$ $\quad <$ $\quad <$ height ⇒ $n = (\frac{3}{2})^i$

$\log_3 n$ $\qquad \log_{3/2} n$

**c) (10 pts)** Solve the recurrence $T(n) = 2T(\sqrt{n}) + \log n$ by making a change of variables.

$m = \lg n$

$S(m) = T(2^m)$

$T(2^m) = 2T(2^{m/2}) + m$

$S(m) = 2.S(m/2) + \underset{\log n}{m}$

$m^{\log_2 2}$ $(m)$

Master 2

$\Theta(m) \Rightarrow \Theta(\lg n)$

$\Theta(m \lg m) \rightarrow \Theta(\log n(\log\log n))$

Gidişat doğru, sonuç yanlış ise $\textcircled{7}$

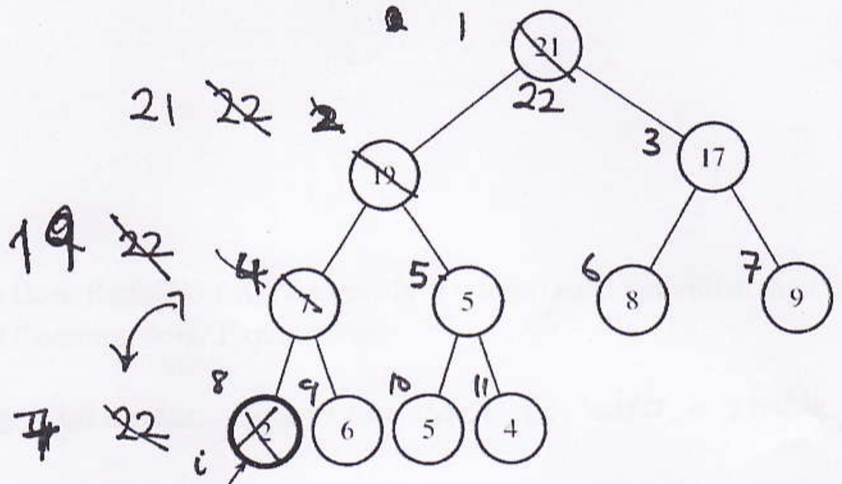
Name and Student ID:________________

Master theorem:
Let a≥1 and b>1 be constants, let f(n) be a function, and let T(n) be defined on the nonnegative integers by the recurrence

$T(n) = aT(n/b) + f(n)$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then *T(n)* can be bounded asymptotically as follows.
1. If $f(n) = O(n^{\log_b a-\epsilon})$ for some constant $\epsilon>0$, then $T(n) = \theta(n^{\log_b a})$.
2. If $f(n) = \theta(n^{\log_b a})$, then $T(n) = \theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon>0$, and if $af(n/b) \le cf(n)$ for some constant $c<1$ and all sufficiently large n, then $T(n) = \theta(f(n))$.

**Q5) (10 points) Heapsort**
For the heap shown below, show what happens when you use HEAP-INCREASE-KEY to increase key 2 to 22. Give justification on why this operation is O(log *n*).



Increase this key to 22

i=8 A[4] < A[8] True
exchage
i=4
A[2] < A[4] True
exchage
i=2
A[1] < A[2] True
exchage
i=1

Because heap increase key goes until i=1 which is the root node.
If the key that is changed is at the longest subtree, then the number of comparisons and exchanges that will be made would be at most O(lgn)
↳ due to height to tree from leaf to root.

**Q6) (15 pts) Radix sort**

**a) (10 pts)** Use radix sort to sort *n* dates. A date is represented as month-day-year and all from 20th century. Analyze the running time.

First sort by day using couting sort with an array of size 31.
Then sort by month " " size 12.
Finally sort by year " " size 100,
where the counter in slot $i$ = year $1900+i$

Running time $\Theta(d(n+k))$.
In this case, $d=3$ $k$ is max 100
so, it's linear time $\Theta(n)$

**b) (5 pts)** Does Radix Sort work correctly if we sort each individual digit using Insertion Sort instead of Counting Sort? Explain why.

→ Yes, because insertion sort is also a stable algorithm.