Solve ALL 6 PROBLEMS in the space provided.

Read the Problems CAREFULLY!

There are 5 (FIVE) pages this page included

List of useful formulae

\[
\sum_{i=1}^{n} i = \frac{n(n + 1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n + 1)(2n + 1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n + 1)^2}{4}, \quad n! \approx \left(\frac{n}{e}\right)^n, \quad a^{\log_b n} = n^{\log_b a},
\]

For \( x \neq 1 \), we have that

\[
\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1}, \quad \sum_{i=0}^{n-1} ix^i = \frac{(n-1)x^{n+1} - nx^n + x}{(1-x)^2},
\]

B1. \( f(n) = \Theta(g(n)) \) iff \( \exists \) positive constants \( c_1, c_2, n_0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \ \forall \ n \geq n_0. \)

B2. \( f(n) = \Omega(g(n)) \) iff \( \exists \) positive constants \( c_1, n_0 : 0 \leq c_1 g(n) \leq f(n) \ \forall \ n \geq n_0. \)

B3. \( f(n) = O(g(n)) \) iff \( \exists \) positive constants \( c_2, n_0 : 0 \leq f(n) \leq c_2 g(n) \ \forall \ n \geq n_0. \)

Master Method. \( T(n) = aT(n/b) + f(n) \), such that \( a > 0, b > 1, f(n) > 0. \)

M1 If \( f(n) = O(n^{\log_b a - \epsilon}) \) for some constant \( \epsilon > 0 \), then \( T(n) = \Theta(n^{\log_b a}). \)

M2 If \( f(n) = \Theta(n^{\log_b a} \log^{k+1} n) \), then \( T(n) = \Theta(n^{\log_b a} \log^{k+1} n) \), where \( k \geq 0 \) is a non-negative constant.

M3 If \( f(n) = \Omega(n^{\log_b a + \epsilon}) \) for some constant \( \epsilon > 0 \), and if \( af(n/b) \leq cf(n) \) for some constant \( c < 1 \) and for large \( n \), then \( T(n) = \Theta(f(n)). \)
Problem 1 (30 points)

Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, g_3, \ldots, g_6$ of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $g_3 = \Omega(g_4)$, $g_4 = \Omega(g_5)$, $g_5 = \Omega(g_6)$. Partition your list in equivalence classes such that $f(n)$ and $h(n)$ are in the same class if and only if $f(n) = \Theta(h(n))$. For example for functions $\lg n, n, n^2$, and $2^{\lg n}$ you could write: $n^2, \{n, 2^{\lg n}\}, \lg n$.

$$n^3, (n - 1)!, 8^{\lg n}, n \lg n, \lg (n!), n!.$$ 

Problem 2 (30 points)

Solve the following recurrence relations by providing asymptotically tight bounds. You only need to provide the bound, intermediate derivations are not required. If no boundary case is given, the choice is yours. You may assume that $T(n)$ is positive and monotonically increasing, if you need to do so.

$$T(n) = T(n/4) + n.$$  
$$T(n) = 2T(n/2) + \lg n.$$  
$$T(n) = 2T(n/2) + T(n/4) + n^2.$$
Problem 3 (30 points)
Solve exactly the following recurrence. You may assume that $n$ is a power of two.

$$T(n) = 2T(n/2) + 18n, \quad T(8) = 6.$$ 

Problem 4 (30 points)
Give a \textbf{TRUE} or \textbf{FALSE} for each one of the statements below. Any answer than a full TRUE or FALSE will be considered wrong. All algorithms are the book/notes implementations.

(1) Insertion-Sort sorts in-place.
(2) MergeSort sorts in-place.
(3) On the input sequence $\langle 1, 2, \ldots, n \rangle$, Insertion-Sort is asymptotically faster than MergeSort.
(4) On the input sequence $\langle n, n - 1, \ldots, 1 \rangle$, InsertionSort is asymptotically faster than MergeSort.
(5) On the input sequence $\langle n, n - 1, \ldots, 1 \rangle$, InsertionSort has running time that is $O(n^3)$.
(6) On the input sequence $\langle 1, 2, \ldots, n \rangle$, InsertionSort has running time that is $O(n^2)$. 
Problem 5 (30 points)

Suppose that we are given a Java or C++ class that implements a stack. Suppose however that we want to implement a queue instead. One way to do this is to use two stacks \( A \) and \( B \) to implement a queue. In order to perform \texttt{Enqueue} we push a key into \( A \). In order to perform a \texttt{Dequeue} operation we first check if \( B \) is empty and if so, we "move" \( A \) into \( B \) (that, is we pop off a key from \( A \) and push it to \( B \) immediately and repeat until \( A \) is empty). Then we pop off the key from \( B \) and return (from \texttt{Dequeue}). For example, \texttt{Enqueue(a)}, \texttt{Enqueue(b)}, \texttt{Dequeue()} has the trace.

If \( C=<x \ y] \) \( x \) is on top of the stack \( y \) at the bottom.

<table>
<thead>
<tr>
<th>Operation</th>
<th>( A )</th>
<th>( B )</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enqueue(a)</td>
<td>( A=&lt;a]</td>
<td>( B=&lt;]</td>
<td>1</td>
</tr>
<tr>
<td>Enqueue(b)</td>
<td>( A=&lt;b \ a]</td>
<td>( B=&lt;]</td>
<td>1</td>
</tr>
<tr>
<td>Dequeue()</td>
<td>( A=&lt;]</td>
<td>( B=&lt;a \ b]</td>
<td>'move'</td>
</tr>
<tr>
<td></td>
<td>( A=&lt;]</td>
<td>( B=&lt;b]</td>
<td>'pop'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>return(a) // Dequeue() returns a.</td>
<td>5</td>
</tr>
</tbody>
</table>

Suppose each push and pop costs 1 dollar of work, so that when we move \( n \) keys from \( A \) into \( B \), we pay \( 2n \) dollars since we perform \( 2n \) pop and push operations.

Answer the following questions.
(a) Suppose we start from empty \( A \) and \( B \) and we perform 3 arbitrary \texttt{Enqueue}, then 2 \texttt{Dequeue}, then 3 \texttt{Enqueue}, and finally 2 more \texttt{Dequeue} operations. What is the total cost of these 10 operations and how many elements are in each stack at the end? Explain.
(b) If a total of \( n \) \texttt{Enqueue} and \( n \) \texttt{Dequeue} operations are performed in some order, how large might the running time of one of the operations be? Provide an exact, non-asymptotic answer. Give a sequence of operations that results in such a behavior and indicate which operation (among the \( 2n \) ones of the sequence) has the running time that you specified.
Problem 6 (17 points)


This is the end of the exam.