Q1. Insertion sort: takes $10n^2$ to sort $n$ items. Merge Sort: takes $100 n \lg n$ to sort $n$ items. Consider a faster computer $A$ running *insertion sort* against a slower computer $B$ running *merge sort*; Both must sort an array of one million ($n = 10^6$) numbers. Suppose computer $A$ executes 10 billion ($10^{10}$) instructions per second. Computer $B$ executes hundred million ($10^8$) instructions per second. So computer $A$ is 100 times faster than computer $B$. Which computer will run the corresponding sorting program faster and by how much faster?

**Solution:**

To sort one million ($n = 10^6$) numbers:

Computer $A$ takes $10 \cdot (10^6)^2 \text{ instr}/(10^{10} \text{ instr/second}) = 1000 \text{ seconds}$

Computer $B$ takes $100 \cdot 10^6 \cdot \lg(10^6) \text{ instr}/(10^8 \text{ instr/second}) \approx 20 \text{ seconds}$

Computer **B** will run faster by $1000/20 = 50$ times

2. Algorithm $P$ with the running time $n^3$ solves an instance of size 500 in 3 seconds. How long will it take for $P$ to solve an instance of size 2000? Show your calculations.

**Solution:**

$500^3$ operations take 3 sec thus $2000^3$ operations will take $3 \cdot 2000^3/500^3 = 3 \cdot 64 = 192$ seconds.
3. Illustrate the operation of MERGE-SORT on the array \( A = \langle 8, 21, 4, 3, 12, 1, 5, 7 \rangle \) (3 points)

6. Illustrate the operation of INSERTION-SORT on the array \( A = \langle 40, 15, 30, 5, 25, 10, 20, 35 \rangle \)
4. Consider the following code fragment:

```c++
for ( int i = 1; i <= n-1; i++ )
    for ( int j = 1; j <= n-1; j++ )
        cout << “Hello” << endl;
```

   a. How many “Hello”s are printed when \( n = 4? \) 9 times
   b. How many “Hello”s are printed in terms of \( n? \) \( (n - 1) * (n - 1) \)
   c. How many “Hello”s are printed in \( O( ) \) notation? \( O(n^2) \)

13. Select the right rank of the following functions in order of growth. That is, find an arrangement \( f_1, f_2, ..., f_5 \) satisfying \( f_1 = O(f_2), f_2 = O(f_3), \) ... and so forth.

   a. \( \frac{1}{n} \leq \lg(n) \leq \lg(lg(n)) \leq \lg^2(n) \leq \frac{1}{(3^n)} \)
   b. \( \frac{1}{(3^n)} \leq \frac{1}{n} \leq \lg(lg(n)) \leq \lg(n) \leq \lg^2(n) \)
   c. \( \frac{1}{n} \leq \lg(lg(n)) \leq \lg(n) \leq \lg^2(n) \leq \frac{1}{(3^n)} \)
   d. \( \frac{1}{(3^n)} \leq \lg(lg(n)) \leq \frac{1}{n} \leq \lg(n) \leq \lg^2(n) \)
Q3. Choose the right answer:

a) \( n^2 \in O(n^3) \)

b) \( n^2 \in O(n) \)

c) \( n^2 \in O(n \lg n) \)

d) \( n^2 \in O(\lg n) \)

1. Solve the following recurrence by using iteration method.

\[
T(n) = \begin{cases} 
1 & n = 0 \\
2T(n-1) & n > 0 
\end{cases}
\]

Solution:

\[
T(n) = 2T(n-1) = 2^2 T(n-2) = 2^3 T(n-3) = \ldots = 2^k T(n-k)
\]

To stop the recursion, we should have \( n - k = 0 \) \( \Rightarrow \) \( k = n \)

\[
T(n) = 2^n T(n-n) = T(n) = 2^n T(0) = T(n) = 2^n
\]

2. Use the Master theorem to solve the this recurrence \( T(n) = 36 \ T(n/6) + n \)

Solution: (5 points)

\[
a = 36 \\
b = 6 \\
f(n) = n \\
n^{\log_b a} = n^2 \\
f(n) = O(n^{2+\varepsilon}) = O(n^{2.5}) = O(n^{1.5}), \quad \text{where} \ \varepsilon = 0.5
\]

Case \( \square \) \( \square \) applies, thus the solution is

\[
T(n) = O(n^2)
\]

6. In a full binary heap with \( n \) nodes, the number of internal nodes (non leaves) is:
a. \( n/2 \)
b. \( (n + 1)/2 \)
c. \( (n - 1)/2 \)
d. \( (n - 1)/2 - 1 \)

8. The following sequence represents a max heap stored in an array: \( \langle 60, 40, 50, 35, 32, 30, 20, 34, 10, 25 \rangle \). What would be the content of the array after the 3rd iteration of the heap-sort algorithm (given below)?

\[
\text{Heapsort}(A)\{
1. \text{Build-Heap}(A)
2. for i \leftarrow \text{length}[A] \text{ downto } 2 \text{ do}
4. heap-size[A] \leftarrow heap-size[A] - 1
5. \text{Heapify}(A, 1)\}
\]

a. 35 34 30 10 32 25 20 40 50 60
b. 35 34 30 10 25 32 20 40 50 60
c. 60 50 40 35 32 30 20 34 10 25
d. 40 10 50 34 60 32 35 25 30 20