1 Sorting, Merge-Sort, and Quick-Sort

1. Let $A$ and $B$ be two sequences of $n$ integers each. Given an integer $x$, describe an $O(n \log n)$-time algorithm for determining if there is an integer $a$ in $A$ and an integer $b$ in $B$ such that $x = a + b$.

2. Given an array $A$, of $n$ numbers in the range from 1 to $n$, describe an $O(n)$ time method for finding the number that occurs most frequently in $A$.


4. Determine the running-time of merge-sort for a) sorted input; b) reverse-ordered input; c) random input; and d) all identical input.

5. Given the array $A$ with elements $[22, 15, 36, 44, 10, 3, 9, 13, 29, 25]$, illustrate the performance of the quick-sort algorithm on $A$.

6. Determine the running-time of quick-sort for a) sorted input; b) reverse-ordered input; c) random input; and d) all identical input.

7. Suppose we modify the quick-sort algorithm so that, instead of selecting the first element as the pivot we choose the element in the middle of the sequence. What is the running time of this version on a sequence that is already sorted?

8. Considering the modified quick-sort of the previous question. Describe the kind of sequence that would cause this version to run in $\Theta(n^2)$ time.

Recurrences

1. Solve each of the following recurrences using the iteration method:

   \begin{align*}
   (a) \quad & T(1) = 1, \quad T(n) = 2T(n/2) + n \\
   & 
   \end{align*}
2. Consider the following C++ code:

```cpp
int foo(int n) {
    int result = 0;
    if (n == 1) {
        return result;
    }
    for (int i = 0 ; i < n ; i++) {
        result += i;
    }
    return foo(n/2) + result + foo(n/2);
}
```

Find and solve the recurrence for the time complexity of `foo`.

Recursion

1. Given the following recursive function. Draw the call tree (graphic representation of all the calls), for an initial call of this algorithm with \( m = 14 \) and \( n = 4 \).

   ```cpp
   // Dijkstra’s Algorithm for GCD (Greatest Common Divisor)
   int gcd(int m, int n) {
       if (m == n) return m;
       else if (m > n) return gcd(n, m - n);
       else return gcd(m, n - m);
   }
   ```

2. Given the C++ function below, what is the returning value of `func(123)`? (don’t use a computer for solving this question)
int func(int n) {
    if (n < 10) {
        return (10 * n) + n;
    } else {
        return (100 * func(n/10)) + func(n%10);
    }
}

3. Given the C++ function below, explain what is the returning value.

int mistery(int *p, int a, int b) {
    if (a > b) {
        return 0;
    } else if (a == b) {
        return p[a];
    } else {
        int m = (a + b) / 2;
        return mistery(p, a, m) + mistery(p, m+1, b);
    }
}

4. Write a recursive function long int tribonacci(int n) that returns the \( n \)-th number in the Tribonacci sequence, where each number in the sequence is the sum of the previous three numbers. Consider the following base cases: \( T_1 = 1, T_2 = 1, T_3 = 1 \).

5. Write a recursive method that accepts an integer and prints that number’s representation in binary to the standard output.

Pointers

1. What is the output of the following C++ code? (don’t use a computer for solving this question)
int *p, *q;
p = new int [10];
q = p + 10;
for (int i = 0 ; i < 10 ; i ++) {
    *(p+i) = i;
}
while (q != p) {
    q --;
    *q *= 2;
}
for (int i = 0 ; i < 10 ; i ++) {
    std::cout << p[i] << " ";
}