Analysis of Recursive Algorithms
Exercises

Write a recurrence relation describing the worst case running time of each of the following algorithms and determine the asymptotic complexity of the function defined by the recurrence relation. Justify your solution using either substitution, a recursion tree or induction. You may NOT use the Master theorem. Assume that all arithmetic operations take constant time.

Simplify and express your answer as $\Theta(n^k)$ or $\Theta(n^k(\log n))$ wherever possible. If the algorithm takes exponential time, then just give exponential lower bounds.

1. **function** func1($A, n$)

```
/* A = array of n integers */
if n ≤ 20 then return (A[1]);
x ← func1(A, ⌊3n/4⌋);
for i ← ⌊n/2⌋ to ⌊n/2⌋ + 8 do
    x ← x + A[i];
endfor
return (x);
```

2. **function** func2($A, n$)

```
/* A = array of n integers */
if n ≤ 12 then return (A[1]);
for i ← 1 to ⌊n/2⌋ do
    for j ← 1 to ⌊n/3⌋ do
        for k ← 1 to n do
            A[k] ← A[k] − A[j] + A[i];
        endfor
    endfor
endfor
y ← func2(A, n − 5);
return (y);
```

3. **function** func3($A, n$)

```
/* A = array of n integers */
if n ≤ 20 then return (A[n]);
x ← 0;
for i ← 1 to 4 do
    for j ← 1 to n − i do
        A[j] ← A[j] − A[n − j];
    endfor
    x ← x+func3(A, ⌊n/4⌋);
endfor
return (x);
```

4. **function** func4$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 3$ **then return** $(A[1])$;
2. **for** $i \leftarrow 1$ **to** $n$ **do**
3.     **for** $j \leftarrow \lfloor n/3 \rfloor$ **to** $\lfloor 2n/3 \rfloor$ **do**
4.         $A[i] \leftarrow A[i] - A[j]$;
5.     **endfor**
6. **endfor**
7. $x \leftarrow$ func4$(A,\ \lfloor 2n/3 \rfloor)$;
8. **return** $(x)$;

5. **function** func5$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 20$ **then return** $(A[n])$;
2. $x \leftarrow$ func5$(A, n-5)$;
3. **for** $i \leftarrow 1$ **to** $\lfloor n/2 \rfloor$ **do**
4.     **for** $j \leftarrow \lfloor n/2 \rfloor$ **to** $n$ **do**
5.         $A[i] \leftarrow A[i] - A[j]$;
6.     **endfor**
7. **endfor**
8. $x \leftarrow x+$ func5$(A, n-8)$;
9. **return** $(x)$;

6. **function** func6$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 20$ **then return** $(A[1])$;
2. $j \leftarrow 3$;
3. **while** $j < n$ **do**
4.     $A[j] \leftarrow A[j-1] + A[j]$;
5.     $j \leftarrow j + \lfloor \sqrt{n} \rfloor$;
6. **endwhile**
7. $y \leftarrow$ func6$(A,\ n-3)$;
8. **return** $(y)$;

(Note: Step 5 uses addition, NOT multiplication.)

7. **function** func7$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 20$ **then return** $(A[1])$;
2. $x \leftarrow 0$;
3. $i \leftarrow n - 3$;
4. **while** $i \geq 6$ **do**
5.     $A[i] \leftarrow A[i] + A[i+1]$;
6.     $x \leftarrow x+$ func7$(A, i)$;
7.     $i \leftarrow i - 4$;
8. **endfor**
9. **return** $(x)$;

(Note: Step 7 uses subtraction, NOT division.)

8. **function** func8$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 10$ **then return** $(A[1])$;
2. $i \leftarrow 3$;
3. **while** $i < n$ **do**
4.     $A[i] \leftarrow A[i] - A[i-1]$;
5.     $i \leftarrow 3 * i$;
6. **endwhile**
7. $x \leftarrow$ func8$(A,\ n-6)$;
8. **return** $(x)$;

9. **function** func9$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 50$ **then return** $(A[n])$;
2. $x \leftarrow$ func9$(A, \lfloor n/4 \rfloor)$;
3. **for** $j \leftarrow 1$ **to** $\lfloor n/3 \rfloor$ **do**
4.     $A[j] \leftarrow A[n-j] - A[j]$;
5. **endfor**
6. $x \leftarrow x+$ func9$(A, \lfloor n/4 \rfloor)$;
7. **return** $(x)$;

10. **function** func10$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 15$ **then return** $(A[n])$;
2. $x \leftarrow 0$;
3. **for** $i \leftarrow 1$ **to** $4$ **do**
4.     **for** $j \leftarrow 1$ **to** $n - i$ **do**
5.         **for** $k \leftarrow 1$ **to** $\lfloor n/2 \rfloor$ **do**
6.             $A[j] \leftarrow A[k] - A[n-j]$;
7.         **endfor**
8.     **endfor**
9.     $x \leftarrow x+$func10$(A, \lfloor n/2 \rfloor)$;
10. **endfor**
11. **return** $(x)$;

11. **function** func11$(A, n)$
/* $A$ = array of $n$ integers */
1. **if** $n \leq 20$ **then return** $(A[1])$;
2. $x \leftarrow 0$;
3. $i \leftarrow 2$;
4. **while** $i \leq n$ **do**
5.     $A[i] \leftarrow A[i] + A[i+1]$;
6.     $x \leftarrow x+$ func11$(A, n-i)$;
7.     $i \leftarrow 3 * i$;
8. **endfor**
9. **return** $(x)$;