Analysis

Part A: Asymptotic Analysis

Problem 1: Big Oh and Big Omega Proofs

Function: $f(n) = 3n^3 + 2n^2 - 6n + 3$

(a) Show that $f(n) = O(n^3)$

(b) Prove that $f(n) = \Omega(n^3)$

(c) Conclusion from parts (a) and (b)

Problem 2: Function Growth Ordering

Determine the tight bound (Θ notation) for each function, then order them from fastest (1) to slowest (7) asymptotic class (constant time is fastest, super-exponential is slower).

Rank from 1-7 and give tight bound:

- __ Θ( ) $10n^2 \lg n + 8n^3$
- __ Θ( ) $n! + 2^n$
- __ Θ( ) $20n^2 \log_{16} n$
- __ Θ( ) $100n + 5 \log(n!)$
- __ Θ( ) $16n + 2^n$
- __ Θ( ) $15 \lg(2n) + 30$
- __ Θ( ) $1000n + 10n^2$

Part B: Recurrence Relations

Algorithm Analysis

For each algorithm, write the recurrence relation T(n) and solve it to find Θ(T(n)).

Algorithm 1: exp(a, n)

```
exp(a, n) // calculates a^n for n ≥ 1
{
    if (n == 1)
        return a
    power = exp(a, n/2)
    power = power * power
    if (n %2 == 0)
        return power
    else
        return (power * a)
}
```

Recurrence relation: T(n) =

Show your work: identify a, b, f(n), compare f(n) to n^(log_b a), determine which case applies, and provide the final result

Final answer: T(n) = Θ(   )

Algorithm 2: weirdSum(A, n)

```
weirdSum(A, n) // calculates a weird sum for an array of size n
{
    if (n > 0)
    {
        sum = 0
        for i = 1 to n
            sum = sum + A[i]
        return (sum + weirdSum(A, n-1))
    }
    else return 0
}
```

Recurrence relation: T(n) =

Explain whether Master Theorem applies or if you need to use iteration method. Show your steps to solve.

Final answer: T(n) = Θ(   )

Algorithm 3: something(n)

```
something(n) // does something for a positive integer n
{
    if (n ≤ 1) then
        return
    for j = 1 to sqrt(n)
        x = x+1
    for i = 1 to 10
        something(n/3)
}
```

Recurrence relation: T(n) =

Show your work: identify a, b, f(n), compare f(n) to n^(log_b a), determine which case applies, and provide the final result

Final answer: T(n) = Θ(   )

Direct Recurrence Solving

Solve the following recurrences using the Master Theorem.

(a) $T(n) = 8T(n/2) + 6n^3$

Solution: T(n) = Θ(   )

(b) $T(n) = 4T(n/3) + n^3$

Solution: T(n) = Θ(   )

Part C: Programming Implementation Analysis

Performance Comparison

After implementing the three binomial coefficient methods, run the main method (`gradle run`) provide your empirical analysis:

Timing Results

Record the execution times for your three methods with different input sizes:

n	k	Definition (ms)	Cancellation (ms)	Recursive (ms)
5	2			
10	3			
15	7			
20	10			

Note: You may need to skip larger cases for the recursive method due to exponential time complexity.

Theoretical vs Empirical Analysis

Method 1 (Definition):

- **Theoretical complexity:** O(n)
- **Empirical observations:** Describe what you observed about performance scaling

- **Analysis:** Explain how empirical results match or differ from theory

Method 2 (Cancellation):

- **Theoretical complexity:** O(min(k, n-k))
- **Empirical observations:** Describe what you observed about performance scaling

- **Analysis:** Explain how empirical results match or differ from theory

Method 3 (Recursive):

- **Theoretical complexity:** $O(2^n)$
- **Empirical observations:** Describe what you observed about performance scaling

- **Analysis:** Explain how empirical results match or differ from theory