*Owen Lindsey*

*Professor Demland, David*

*CST-201 Exercise 5*

*10/06/2024*

**Exercise 2.3 - 1a Compute the Following Sums**

*Algorithm Explanation*

1. *Identify the sequence:*

*This is an arithmetic sequence with a common difference of 2.*

1. *Use the formula for the sum of an arithmetic sequence:*

*S = n(a1 + an)/2 Where n is the number of terms, a1 is the first term, and an is*  *the last term.*

1. *Calculate n:*

*(999 - 1)/2 + 1 = 500*

1. *Apply the formula:*

*S = 500(1 + 999)/2*

1. *The final sum of S:*

*500(1 + 999)/2 = 250,000*

**Exercise 2.3 - 1b Compute the Following Sums**

*Algorithm Explanation*

*1. Identify the sequence:*

*This is a geometric sequence with first term a = 2 and common ratio r = 2.*

*2. Determine the number of terms:*

*n = log2(1024/2) + 1 = 10*

*3. Apply the formula for the sum of a geometric sequence:*

*S = a(1-r^n)/(1-r)*

*where a is the first term and r is the common ratio.*

*4. The final sum of S:*

*S = 2(1-2^11)/(1-2) = 2(1-2048)/(-1) = 2046*

**Exercise 2.3 - 1c Compute the Following Sums**

*Algorithm Explanation*

1. *Identify the sequence:*

*This is a sum of constant 1's from i=3 to n+1.*

1. *Determine the number of terms:*

*(n+1) - 3 + 1 = n-1*

1. *Apply the formula for the sum of constants:*

*S = k \* number of terms, where k is the constant (1 in this case).*

1. *Simplify:*

*S = 1 \* (n-1) = n-1*

*5. The final sum:*

*n-1*

**Exercise 2.3 - 1d Compute the Following Sums**

*Algorithm Explanation*

1. *Identify the sequence:*

*This is an arithmetic sequence from 3 to n+1.*

1. *Determine the number of terms:*

*(n+1) - 3 + 1 = n-1*

1. *Apply the formula for the sum of an arithmetic sequence:*

*S = ((number of terms)/2) \* (first term + last term)*

1. *Calculate:*

*S = ((n-1)/2) \* (3 + (n+1)) = (n^2 + 3n - 4)/2*

1. *The final sum:*

*(n^2 + 3n - 4)/2*

**Exercise 2.3 - 1e Compute the Following Sums**

*Algorithm Explanation*  
  
 *1. Identify the sequence:*

*This is a sum of i(i+1) from i=0 to n-1.*

*2. Expand the expression:*

*Σ(i^2 + i) = Σi^2 + Σi*

*3. Apply formulas for sum of squares and sum of natural numbers:*

*Σi^2 = n(n-1)(2n-1)/6, Σi = n(n-1)/2*

*4. Combine and simplify:*

*S = n(n-1)(2n-1)/6 + n(n-1)/2 = (n-1)n(n+1)/3*

*5. The final sum:*

*(n-1)n(n+1)/3*

**Exercise 2.3 - 1f Compute the Following Sums**

*Algorithm Explanation*  
  
 *1. Identify the sequence:*

*This is a sum of 3j+1 from j=1 to n.*

*2. Separate the sum:*

*Σ(3j) + Σ(1)*

*3. For Σ(3j): Factor out 3:*

*3Σj*

*4. For Σ(1): add 1, n times:*

*n*

*5. Combine the results: Combine the results:*

*3[n(n+1)/2] + n*

1. *Simplify:*

*(3n^2 + 3n)/2 + n = (3n^2 + 5n)/2*

1. *The final sum:*

*(3n^2 + 5n)/2*

**Exercise 2.3 - 1g Compute the Following Sums**

*Algorithm Explanation*

1. *Identify the sequence:*

*This is a double sum of in from i=1 to n and j=1 to n.*

1. *Solve the inner sum:*

*Σ(j=1 to n) ij = i \* Σj = i \* n(n+1)/2*

1. *Simplify inner sum:*

i \* n(n+1)/2

1. *Solve the outer sum:*

Σ(i=1 to n) [i \* n(n+1)/2]

*5. Factor out:*

n(n+1)/2: [n(n+1)/2] \* Σ(i=1 to n) i

6. Simplify:

[n(n+1)/2] \* [n(n+1)/2] = n^2(n+1)^2/4  
  
 7. The final sum:

n^2(n+1)^2/4

**Exercise 2.3 - 1h Compute the Following Sums**

*Algorithm Explanation*

*1. Identify the sequence:*

*This is a sum of 1/i(i+1) from i=1 to n.*

*2. This is a telescoping series:*

*1/i(i+1) can be rewritten as 1/i - 1/(i+1)*

*3. Write out series:*

*(1 - 1/2) + (1/2 - 1/3) + (1/3 - 1/4) + ... + (1/n - 1/(n+1)*

*4. Observe that all intermediate terms cancel out:*

*All terms except the first (1) and the last (-1/(n+1)) cancel each other out:*

*S = ((n-1)/2) \* (3 + (n+1)) = (n^2 + 3n - 4)/2*

*5. Simplify the result:*

*1 - 1/(n+1) = (n+1-1)/(n+1) = n/(n+1)*

*6. The final sum:*

*n/(n+1)*

**Exercise 2.3 - 10**

*Algorithm Explanation*

1. *Identify the pattern:* 
   1. *Main diagonal: 1, 2, 3, ..., 10*
   2. *Each diagonal above the main: increases by 1 (2 to 11, 3 to 12, etc.)*
   3. *Each diagonal below the main: increases by 1 (9 to 19, 10 to 19)*
2. *Count occurrences of each number:* 
   1. *Numbers 1 to 9 appear 10 times each*
   2. *Number 10 appears 11 times (10 in column, 1 in row)*
   3. *Numbers 11 to 19 appear decreasing times: 9, 8, 7, ..., 1*
3. *Calculate sums for each group:* 
   1. *Sum of 1 to 9: 9 \* 10 \* 5 (average of 1 to 9 is 5)*
   2. *Sum of 10: 10 \* 11*
   3. *Sum of 11 to 19: (119 + 128 + 137 + 146 + 155 + 164 + 173 + 182 + 19\*1)*
4. *Add all sums together*

**Exercise 2.3 - 10**

*Mental Calculation Steps*

1. *Sum of 1 to 9:*

*9 \* 10 \* 5 = 450*

1. *Sum of 10:*

*10 \* 11 = 110*

1. *Sum of 11 to 19: Group pairs:*

*(11+19)\*1, (12+18)\*2, (13+17)\*3, (14+16)4, 155 = 30 + 60 + 90 + 120 + 75 = 375*

1. *Total sum:*

*450 + 110 + 375 = 935*

*5. The final sum:*

*935*

**Exercise 2.4- 10**

*Algorithm S(n): Sum of First n Cubes*

1. *Setting up and solving the recurrence relation:*
2. *Identify the basic operation:*

*The basic operation is the multiplication n \* n \* n (or n^3).*

1. *Set up the recurrence relation:*

*Let T(n) be the number of times the basic operation is executed for input n.*

*T(n) = { 0 if n = 1 T(n-1) + 1 if n > 1 }*

1. *Solve the recurrence relation:*

*T(2) = T(1) + 1 = 0 + 1 = 1 T(3) = T(2) + 1 = 1 + 1 = 2 T(4) = T(3) + 1 = 2 + 1 = 3*

*We can see that T(n) = n - 1 for n ≥ 1*

1. *Verify the solution:*

*Base case (n = 1):*

*T(1) = 0*

*which matches our recurrence Inductive step:*

*T(n) = (n-1) + 1 = n - 1 + 1 = n,*

*which matches T(n+1)*

*Therefore, the basic operation (n^3) is executed n-1 times.*

**Exercise 2.4- 10**

*b. Comparison with non-recursive algorithm:*

1. *Recursive algorithm:* 
   1. *Time complexity: O(n) for the recursive calls*
   2. *Space complexity: O(n) due to the call stack*
   3. *Executes the basic operation n-1 times*
2. *Non-recursive algorthim for comparison.*

*sum = 0*

*for i = 1 to n:*

*sum += i^3*

*return sum*

1. *Time complexity: O(n)*
2. *Space complexity: O(1)*
3. *Executes the basic operation n times*

**Exercise 2.4- 10**

*Comparison:*

1. *Time efficiency:*

*Both algorithms have O(n) time complexity, but the recursive version does one less operation.*

1. *Space efficiency:*

*The non-recursive version is more space-efficient (O(1) vs O(n)).*

1. *Simplicity:*

*The recursive version is more concise and may be easier to understand conceptually.*

1. *Practical considerations:*

*For very large n, the recursive version might cause stack overflow, while the iterative version wouldn't have this issue.*

*While both algorithms are linear in time complexity, the non-recursive version is generally more efficient in practice due to its constant space complexity.*