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.

2. *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.

3. *Calculate n:*
    (999 - 1)/2 + 1 = 500

4. *Apply the formula:*

    S = 500(1 + 999)/2

5. *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.

2. *Determine the number of terms:*

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

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

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

4. *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.

2. *Determine the number of terms:*

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

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

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

4. *Calculate:*

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

5. *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:*

   $\Sigma(i^2 + i) = \Sigma i^2 + \Sigma i$

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

   $\Sigma i^2 = n(n-1)(2n-1)/6$, $\Sigma i = n(n-1)/2$

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:*

   $\Sigma(3j) + \Sigma(1)$

3. *For $\Sigma(3j)$: Factor out 3:*

   $3\Sigma j$

4. *For $\Sigma(1)$: add 1, n times:*

   n

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

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

6. *Simplify:*

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

7. *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.

2. *Solve the inner sum:*

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

3. *Simplify inner sum:*

   i * n(n+1)/2

4. *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:*
   a. *Main diagonal: 1, 2, 3, ..., 10*
   b. *Each diagonal above the main: increases by 1 (2 to 11, 3 to 12, etc.)*
   c. *Each diagonal below the main: increases by 1 (9 to 19, 10 to 19)*
2. *Count occurrences of each number:*
   a. *Numbers 1 to 9 appear 10 times each*
   b. *Number 10 appears 11 times (10 in column, 1 in row)*
   c. *Numbers 11 to 19 appear decreasing times: 9, 8, 7, ..., 1*
3. *Calculate sums for each group:*
   a. *Sum of 1 to 9: 9 * 10 * 5 (average of 1 to 9 is 5)*
   b. *Sum of 10: 10 * 11*
   c. *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*
2. *Sum of 10:*

   *10 * 11 = 110*
3. *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*
4. *Total sum:*

   450 + 110 + 375 = 935
5. *The final sum:*

   935

Exercise 2.4- 3

Algorithm S(n): Sum of First n Cubes

a. *Setting up and solving the recurrence relation:*

   1. *Identify the basic operation:*

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

   2. *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 }

   3. *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

   4. *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- 3

b. Comparison with non-recursive algorithm:

1. *Recursive algorithm:*
   a. *Time complexity: O(n) for the recursive calls*

   b. *Space complexity: O(n) due to the call stack*

   c. *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
```

   a. *Time complexity: O(n)*

   b. *Space complexity: O(1)*

   c. *Executes the basic operation n times*

Exercise 2.4- 3

Comparison:

1. *Time efficiency:*

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

2. *Space efficiency:*

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

3. *Simplicity:*

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

4. *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.