Discrete structures

consider a simple problem for this demonstration: calculating the factorial of a non-negative integer n.

1. Iterative Approach:

#include <iostream>

unsigned long long factorialIterative(int n) {

unsigned long long result = 1;

for (int i = 1; i <= n; ++i) {

result \*= i;

}

return result;

}

int main() {

int n = 10; // Example: Calculate 10!

unsigned long long resultIterative = factorialIterative(n);

std::cout << "Iterative result for " << n << "! is: " << resultIterative << "\n";

return 0;

}

1. Recursive Approach:

#include <iostream>

unsigned long long factorialRecursive(int n) {

if (n == 0) {

return 1;

}

return n \* factorialRecursive(n - 1);

}

int main() {

int n = 10; // Example: Calculate 10!

unsigned long long resultRecursive = factorialRecursive(n);

std::cout << "Recursive result for " << n << "! is: " << resultRecursive << "\n";

return 0;

}

1. Iterative Approach:

Number of steps: n (since it performs n multiplications).

Memory usage: Constant (no additional memory is allocated as result is updated in-place).

The iterative approach may be more efficient for some problems due to reduced function call overhead.

1. Recursive Approach:

Number of steps: n (number of recursive calls made).

Memory usage: O(n) (due to the call stack).

The recursive approach is often more concise and easier to understand, but it can lead to stack overflow errors for large inputs (n) due to the depth of recursive calls.

For the iterative approach, you may add counters to track how many times the loop iterates, and for the recursive approach, you can add counters to count how many recursive calls are made. We won't use these counters in this example as we're mostly talking about time complexity. For a more accurate measurement, you can include them in your code.

Due to the call stack, recursive solutions are typically less space-efficient yet frequently offer a more beautiful and natural approach to represent some problems. When performance is an issue, iterative methods are frequently favored since they are more space-efficient. However, the situation and its particular requirements will determine which of these approaches is best.