AI NO 02 Lab

HOME TASK / LAB TASK

DATE: ___/___/20__ M T W T F S

Task 1 :- / (Home Task 1)

Time complexity

→ Iterative

The total number of iteration is $n-1$, leading to $O(n)$ iterations. Each iteration perform $O(1)$ so, $O(n) \times O(1) \Rightarrow \boxed{O(n)}$

→ Recursive

R. Formula: $T(n) = T(n-1) + T(n-2) + O(1)$

$T(0) = O(1)$, $T(1) = O(1)$

So, $\Rightarrow \boxed{O(2^n)}$

→ DP (Memoization)

R. Formula: $T(n) = T(n-1) + T(n-2)$

$F(0) = O(1)$, $F(1) = O(1)$

Unique subproblem are n from $F(0)$ to $F(n)$

so, $\Rightarrow \boxed{O(n)}$

→ DP (Tabulation)

array of size $n+1$: $O(n)$

loop runs from 2 to n: $n-1$ iterations

work per iteration: $O(1)$

so, $\Rightarrow \boxed{O(n)}$

DATE: ___/___/20

Task3:

Time Complexity

→ Iterative

It uses two loops, one for length and one for cutting the rod at each possible point so, ⇒ $\boxed{O(n^2)}$

→ Recursive

For each length n, we try every possible cut, which generates two subproblems making the time complexity ~~expoten~~ exponential so, ⇒ $\boxed{O(2^n)}$

→ Memoization

Since each subproblem is solved once and stored in the memo array and for each subproblem, we run a loop of size n, the time complex become quadratic so, ⇒ $\boxed{O(n^2)}$

→ Tabulation

For each length i from 1 to n, we run a nested loop of size i, so, ⇒ $\boxed{O(n^2)}$

DATE: ___/___/20__ M T W T F S

Task2: /(Home Task 3)

Time Complexity:

→ **Iterative**

- Iterate each character in S1: O(m)
- For each character in S1, it iterates through each character in S2 which is O(n).
- The table dp of size m×n is filled which involves as comparison of characters and updating table.

So, $\boxed{O(m \times n)}$

→ **Recursive**

- This approach explores every possible substring and for each comparision it generates three recursive calls. The recursive depth can go up to O(m+n).
- The number of recursive calls grows exponentially, making the total time complexity very inefficient.

So, $\boxed{O(3^{\min(m,n)})}$

DATE: __/__/20 M T W T F S

→ Memoization

The memoized solution ensures that each subproblem is solved only once. The number of unique subproblems is bounded by $O(m \times n)$, where m is length of $S1$ and n is the length of $S2$.

So, $\boxed{O(m \times n)}$

→ Tabulation

Similar to the iterative approach, we fill a table of size $m \times n$. The outer loop runs $O(m)$ times, and for each iteration of the outer loop, the inner loop run $O(n)$ times

So, $\boxed{O(m \times n)}$

DATE: __/__/20__

Home Task 2:

Time Complexity

→ **Iterative**

This case contain loops one is outer loop which run len(coins) times and inner loop that run approx. target times for each coin. So,

$O(len(coin) \times target) = \boxed{O(m \times n)}$

→ **Recursive**

In Recursive technique, we make a recursive call for each coin for the remaining amount (target - coin). At each level of recursion, the problem size decreases by one of the coin denominations, leading to an expotential number of calls So, $\boxed{O(m^n)}$

$m \Rightarrow$ number of coin

$n \Rightarrow$ target amount

→ **Memoization**

In this approach we store the result of eac target in a memoization dictionary, ensuring that each subproblem is solved only once. For each subproblem we loop

DATE: ___/___/20__

over the coin (m coins) and make recursive calls since we only solve each subproblem once and store it in the memo, the number of unique subproblems is n (equal to target value) So, $\boxed{O(m \times n)}$

$m \rightarrow$ number of coins
$n \rightarrow$ Target amount

→ Tabulation

In this approach, we build the Solution from the ground up using a table [dp[ ]] For each amount i from 1 to target we loop through all m coins. The total number of iterations is proportional to m*n (i,e for i we check m coins) So, $\boxed{O(m \times n)}$

Home Task 4:

Time Complexity

→ Iterative

In this approach, there is only one loop that run n times and each iteration contain constant time

So, $\boxed{O(n)}$

DATE: ___/___/20__ (M)(T)(W)(T)(F)(S)

→ Recursive

This approach has an exponential time complexity due to repeated calculations of the same subproblems. The time complexity is $\boxed{O(2^n)}$

→ Memoization

Each subproblem is computed only once and stored in a dictionary.

So, $\boxed{O(n)}$ there are n subproblems.

→ Tabulation

The number of operations is proportional to n, as each subproblem is solved in a loop, Thus time complexity is $\boxed{O(n)}$

Home Task 5:

Time Complexity

→ Iterative

In the iterative approach the time

DATE: ___/___/ 20

Complexity is $\boxed{O(n \times w)}$ because we iterate all over items and all capacities.

→ **Recursive**

The time complexity in recursive technique is exponential $\boxed{O(2^n)}$ because for each item, we have two choices (include or exclude).

→ **Memoization**

The time complexity in memoization is $\boxed{O(n \times w)}$ because we solve each subproblem only once.

→ **Tabulation**

The time complexity is $\boxed{O(n \times w)}$ where n is the number of items and w is the knapsack capacity.

Dry Run

DATE: __/__/20__ M T W T F S

Home Task 1:

Iterative

[0, 1] (Initial fib sequence)

(i=2) [0, 1, 1] (Add 1)

(i=3) [0, 1, 1, 2] (Add 2)

(i=4) [0, 1, 1, 2, 3] (Add 3)

(i=5) [0, 1, 1, 2, 3, 5 (Add 5)

Dynamic Programming (Memo)

fibonacci(5)

fibonacci(4) fibnacci(3) → memo[3]=2

fibonacci(3) fibonacci(2) → memo[2]=

fibonacci(2) → memo[2]=1 fibonacci(1) → 1

DATE: ___/___/20__ M T W T F S

Home Task 2: [CC = Coinchange]

Iterative

coinchange(5)

coinchange(5-1) = coinchange(4)

Coinchange (4-2) = Coinchange (2)

coinchange(5-5) = coinchange(0)

cc(4-1) = cc(3)

cc(3-2) = cc(1)

cc(3-5) [Skipped]

cc(2-1) = cc(1)

cc(2-2) = cc(0)

cc(3-1) = cc(2)

cc(2-2) = cc(0)

cc(2-1) = cc(1)

cc(1-2) [skipped]

cc(1-1) = cc(0)

Dynamic Programming (Memo)

coinchangmemo(5)

dp(5)

dp(-1) x

dp(3)(5-2)

dp(0)(5-5) → 0 coins

dp(2)(4-2) Result = 1

return memo ~~result = 2~~

dp(4)(5-1)

dp(3)(4-1)

dp(0)(2-2) → 0 coins

dp(2)(3-1)

dp(1) (2-1)

dp(-1) x [Invalid]

dp(0)(1-1) → 0 coins.

Home Task 3:

Iterative

Start Matching

Match S1[0] = S2[0] 'a'

Match S1[1] = S2[1] 'b'

Match S1[2] = S2[2] 'c'

No Match S1[3] ≠ S2[3] (Reset)

(No Match on fwith & of later)

Try Match from S1[2] = S2[4] 'c'

compare S1[4] = S2[5] 'e' (Match)

Home Task 4:

Iterative

staircase (n)

n

0 (return 1)   1 (return 1)

n >= 2   Prev1 = 1

prev2 = 1

Loop from 2 to n

i = 2 (curr = 2)   i = 3 (curr = 3)

prev2 = 1   prev2 = 2
prev1 = 2   prev1 = 3

DATE : ___/___/ 20__ (M)(T)(W)(T)(F)(S)

i=4 (curr = 5)

i=5 (curr = 8)

prev2 = 2
prev1 = 3

prev2 = 3
prev1 = 5

Return prev1

Home Task 5:

Iterative

knapsack (values, weight, w)

n=3

dp = [0] * (w+1)

Loop i from 0 to n-1

i=0

i=1

i=2

update dp[]

Update dp[2]

return dp[w]