1. Fibonacci: Iterative

|  |
| --- |
| Fibonacci(x)   1. Arr [x+1] 2. Arr [0] = 0 3. **for** i = 1 **to** x 4. **if** i ≤ 2 5. Arr[i] = 1 6. **else** 7. Arr[i] = Arr[i-1] + Arr[i-2] 8. **return** Arr[x] |

1. Fibonacci: Recursive

|  |
| --- |
| Fibonacci-Rec(n)   1. **if** n = 1 **or** n = 2 2. **return** 1 3. **else** 4. **return** Fibonacci-Rec(n-1) + Fibonacci-Rec(n-2) |

1. Fibonacci: DP
   1. Top-Down

|  |
| --- |
|  |

* 1. Bottom-Up

|  |
| --- |
|  |

1. 2-power-of-n

|  |
| --- |
| two\_power(n)   1. if n = 0 2. return 1 3. else 4. return two\_power (n-1) + two\_power (n-1) |

Make a **memoization** and **look up** strategy to speed up the process

|  |
| --- |
| two\_power(n)   1. **if** n = 0 2. // memoization 3. power\_memo[n] = 1 4. **return** power\_memo[n] 5. // look up 6. **if** power\_memo[n] != 0 7. **return** power\_memo[n] 8. **else** 9. // memoization 10. power\_memo[n] = two\_power (n-1) + two\_power (n-1) 11. **return** power\_memo[n] |