**Dynamic Programming**

1. **Memoization**
2. **Fibonacci - interactive**
   1. Base case
   2. Fib () – Fib()
3. **Fibonacci with memorization**
   1. Base case
   2. Fib () – Fib ()
   3. Create static array []
   4. Check if I have element in n position in array
   5. Save the node in array (if we already calculate it)
4. **Longest Increasing Subsequence(LIS)**
   1. Optimal Sub-Structure
   2. Create array with numbers
   3. Create array with current the best solution
   4. Loop the array with numbers
   5. Base case
   6. Loop all previous solution(<current)
   7. Check if current number Is bigger then previous number
5. **Move Down/Right** 
   1. Read Input (rows, cols)
   2. Make matrix with input numbers
   3. Make sums matrix
   4. Make sum for rows and cols for SumMatrix
   5. Looks who of these two numbers in row up and col left is bigger then other
   6. Find the path of largest sum in SumMatrix // start from end index of matrix
   7. Reverse and print
6. **Rod cutting**