**AP Exp 7**

**Name: Deepanshu Negi**

**Uid: 22BCS16773**

**Section: 613 - B**

**Easy**

Q1. Climbing Stairs

class Solution {

public:

    int climbStairs(int n) {

        if (n <= 2) return n;

        vector<int> dp(n + 1, 0);

        dp[1] = 1;

        dp[2] = 2;

        for (int i = 3; i <= n; i++) {

            dp[i] = dp[i - 1] + dp[i - 2];

        }

        return dp[n];

    }

};

A screenshot of a computer

AI-generated content may be incorrect.

Q2 Best Time to Buy and Sell a Stock

class Solution {

    public int maxSubArray(int[] nums) {

        int maxSum = Integer.MIN\_VALUE;

        int currentSum = 0;

        for (int i = 0; i < nums.length; i++) {

            currentSum += nums[i];

            if (currentSum > maxSum) {

                maxSum = currentSum;

            }

            if (currentSum < 0) {

                currentSum = 0;

            }

        }

        return maxSum;

    }

}

A screenshot of a computer

AI-generated content may be incorrect.

Q3. Maximum SubArray

class Solution {

public:

    int maxProfit(std::vector<int>& prices) {

        int buy = prices[0];

        int profit = 0;

        for (int i = 1; i < prices.size(); i++) {

            if (prices[i] < buy) {

                buy = prices[i];

            } else if (prices[i] - buy > profit) {

                profit = prices[i] - buy;

            }

        }

        return profit;

    }

A screenshot of a computer

AI-generated content may be incorrect.};

Q4. House Robber

class Solution {

public:

    int rob(vector<int>& nums) {

        int n = nums.size();

        if (n == 1) {

            return nums[0];

        }

        vector<int> dp(n, 0);

        dp[0] = nums[0];

        dp[1] = max(nums[0], nums[1]);

        for (int i = 2; i < n; i++) {

            dp[i] = max(dp[i - 1], nums[i] + dp[i - 2]);

        }

        return dp[n - 1];

    }

A screenshot of a computer

AI-generated content may be incorrect.};

**Medium**

Q5. Jump Game

class Solution {

public:

    bool canJump(vector<int>& nums) {

        int r = 0;

        for (int i = 0; i < nums.size(); i++) {

            if (i > r) {

                return false;

            }

            r = max(r, i + nums[i]);

            if (r >= nums.size() - 1) {

                return true;

            }

        }

        return false;

    }

};

A screenshot of a computer

AI-generated content may be incorrect.

Q6. Unique Paths

class Solution {

public:

int uniquePaths(int m, int n) {

vector<vector<int>> dp(m, vector<int>(n));

return dfs(dp, 0, 0);

}

int dfs(vector<vector<int>>& dp, int i, int j) {

if(i >= size(dp) || j >= size(dp[0])) return 0; // out of bounds - invalid

if(i == size(dp)-1 && j == size(dp[0])-1) return 1; // reached end - valid path

if(dp[i][j]) return dp[i][j]; // directly return if already calculated

return dp[i][j] = dfs(dp, i+1, j) + dfs(dp, i, j+1); // store the result in dp[i][j] and then return

}

};

A screenshot of a computer

AI-generated content may be incorrect.

Q7. Coin Change

class Solution {

public:

int coinChange(vector<int>& coins, int amount) {

vector<int> minCoins(amount + 1, amount + 1);

minCoins[0] = 0;

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

for (int j = 0; j < coins.size(); j++) {

if (i - coins[j] >= 0) {

minCoins[i] = min(minCoins[i], 1 + minCoins[i - coins[j]]);

}

}

}

return minCoins[amount] != amount + 1 ? minCoins[amount] : -1;

}

};

Screens screenshot of a computer

AI-generated content may be incorrect.

Q8. Longest Increasing Subsequence

class Solution {

public:

    int lengthOfLIS(vector<int>& nums) {

        vector<int> sub;

        for (int x : nums) {

            if (sub.empty() || sub[sub.size() - 1] < x) {

                sub.push\_back(x);

            } else {

                auto it = lower\_bound(sub.begin(), sub.end(), x);

                \*it = x;

            }

        }

        return sub.size();

    }

};

A screenshot of a computer

AI-generated content may be incorrect.

**Hard**

Q9. Maximum Product Subarray

class Solution {

    public int maxProduct(int[] nums) {

        int res = Integer.MIN\_VALUE;

        for (int n : nums) {

            res = Math.max(res, n);

        }

        int curMax = 1, curMin = 1;

        for (int n : nums) {

            int temp = curMax \* n;

            curMax = Math.max(temp, Math.max(curMin \* n, n));

            curMin = Math.min(temp, Math.min(curMin \* n, n));

            res = Math.max(res, curMax);

        }

        return res;

    }

}

A screenshot of a computer

AI-generated content may be incorrect.

Q10. Decode Ways

class Solution {

public:

int numDecodings(std::string s) {

if (s.empty() || s[0] == '0') {

return 0;

}

int n = s.length();

std::vector<int> dp(n + 1, 0);

dp[0] = 1;

dp[1] = 1;

for (int i = 2; i <= n; ++i) {

int oneDigit = s[i - 1] - '0';

int twoDigits = std::stoi(s.substr(i - 2, 2));

if (oneDigit != 0) {

dp[i] += dp[i - 1];

}

if (10 <= twoDigits && twoDigits <= 26) {

dp[i] += dp[i - 2];

}

}

return dp[n];

}

};

A screenshot of a computer

AI-generated content may be incorrect.

Q11. Best time to buy and Sell a Stock with Cooldown

class Solution {

    public int maxProfit(int[] prices) {

    if(prices == null || prices.length <= 1) return 0;

    int b0 = -prices[0], b1 = b0;

    int s0 = 0, s1 = 0, s2 = 0;

    for(int i = 1; i < prices.length; i++) {

        b0 = Math.max(b1, s2 - prices[i]);

        s0 = Math.max(s1, b1 + prices[i]);

        b1 = b0; s2 = s1; s1 = s0;

    }

    return s0;

}

}

A screenshot of a computer

AI-generated content may be incorrect.

Q12. Perfect Squares

class Solution

{

public:

    int numSquares(int n)

    {

        if (n <= 0)

        {

            return 0;

        }

        vector<int> cntPerfectSquares(n + 1, INT\_MAX);

        cntPerfectSquares[0] = 0;

        for (int i = 1; i <= n; i++)

        {

            for (int j = 1; j\*j <= i; j++)

            {

                cntPerfectSquares[i] =

                    min(cntPerfectSquares[i], cntPerfectSquares[i - j\*j] + 1);

            }

        }

        return cntPerfectSquares.back();

    }

};

A screenshot of a computer

AI-generated content may be incorrect.

Q13. Word Break

class Solution:

def wordBreak(self, s: str, wordDict: List[str]) -> bool:

dp=[None for i in range(len(s))]

res=[False]

def fun(index):

if index>=len(s):

res[0]=True

return True

if dp[index]!=None:

return dp[index]

for each in wordDict:

if s[index:len(each)+index]==each:

if (fun(index+(len(each)))):

dp[index]=True

else:

dp[index]=False

fun(0)

return res[0]

A screenshot of a computer

AI-generated content may be incorrect.

Q14. Word Break 2

class Solution {

public:

vector<string> wordBreak(string s, vector<string>& wordDict) {

unordered\_set<string> dict(wordDict.begin(), wordDict.end());

return wordBreakHelper(s, 0, dict);

}

private:

vector<string> wordBreakHelper(const string& s, int start, const unordered\_set<string>& dict) {

vector<string> validSubstr;

if (start == s.length())

validSubstr.push\_back("");

for (int end = start + 1; end <= s.length(); ++end) {

string prefix = s.substr(start, end - start);

if (dict.find(prefix) != dict.end()) {

vector<string> suffixes = wordBreakHelper(s, end, dict);

for (const string& suffix : suffixes) {

validSubstr.push\_back(prefix + (suffix.empty() ? "" : " ") + suffix);

}

}

}

return validSubstr;

}

};

A screenshot of a computer

AI-generated content may be incorrect.