Recursion Problems Solutions

Here is a clean and structured version of your notes with **problem statements, multiple approaches, dry runs, and C++ pseudocode**, ready for interviews or revision:

---

10. Best Time to Buy and Sell Stock

Problem:
Given an array of stock prices, find the maximum profit by choosing one day to buy and one to sell. Return 0 if no profit is possible.

Approach 1: Brute Force

- Try all pairs (buy day, sell day) and track max profit.
- **Time:** O(n^2), **Space:** O(1)

Approach 2: One-Pass (Optimal)

- Track minimum price so far.
- For each price, calculate profit if sold today.
- Update max profit accordingly.
- **Time:** O(n), **Space:** O(1)

Pseudocode:

```cpp
int maxProfit(vector<int>& prices) {
    int minPrice = INT_MAX, maxProfit = 0;
    for (int price : prices) {
        minPrice = min(minPrice, price);
        maxProfit = max(maxProfit, price - minPrice);
    }
    return maxProfit;
}
```

Dry Run:
Input: `[7, 1, 5, 3, 6, 4]`
→ min = 1, max profit = 6 - 1 = **5**

---

11. Move All Negative Elements to One Side

Problem:
Rearrange array so that all negatives come to one side.

Approach: Two Pointers

- Start: `left = 0`, `right = n - 1`
- Swap negatives left, positives right.
- **Time:** O(n), **Space:** O(1)

Pseudocode:

```cpp
void rearrange(vector<int>& nums) {
    int left = 0, right = nums.size() - 1;
    while (left <= right) {
        if (nums[left] < 0) left++;
        else if (nums[right] > 0) right--;
        else swap(nums[left++], nums[right--]);
    }
}
```

Dry Run:
Input: `[-1, 2, -3, 4, 5]`
→ Swap 2 and -3 → `[-1, -3, 2, 4, 5]`

---

12. Check if Array is Sorted and Rotated

Problem:
Check if a sorted array has been rotated.

Approach: Count Decreasing Points

- Count how many times `nums[i] > nums[i+1]` (circularly).
- If count > 1 → not sorted and rotated.
- **Time:** O(n), **Space:** O(1)

Pseudocode:

```cpp
bool isSortedAndRotated(vector<int>& nums) {
    int count = 0, n = nums.size();
    for (int i = 0; i < n; i++) {
        if (nums[i] > nums[(i + 1) % n]) count++;
        if (count > 1) return false;
    }
    return true;
}
```

Dry Run:
Input: `[3, 4, 5, 1, 2]` → Count = 1 → true
Input: `[3, 5, 4, 1, 2]` → Count = 2 → false

---

13. Sum of Two Arrays

Problem:
Each array represents a number (digit by digit), return their sum.

Approach: Elementary Addition

- Reverse arrays, add digit by digit.
- Carry over as needed.
- Reverse result at the end.
- **Time:** O(max(n, m)), **Space:** O(max(n, m))

Pseudocode:

cpp

```cpp
vector<int> sumOfTwoArrays(vector<int>& nums1, vector<int>& nums2) {
    reverse(nums1.begin(), nums1.end());
    reverse(nums2.begin(), nums2.end());
    vector<int> result;
    int carry = 0, i = 0, n1 = nums1.size(), n2 = nums2.size();

    while (i < n1 || i < n2 || carry) {
        int sum = (i < n1 ? nums1[i] : 0) + (i < n2 ? nums2[i] : 0) + carry;
        result.push_back(sum % 10);
        carry = sum / 10;
        i++;
    }

    reverse(result.begin(), result.end());
    return result;
}
```

Dry Run:
Input: `[9, 9, 9]`, `[1]` → Output: `[1, 0, 0, 0]`

2D Array Problems

---

1. Wave Print Pattern

Problem:
Print 2D matrix in wave order column-wise.

Approach:

- Even columns: top → bottom
- Odd columns: bottom → top
- **Time:** O(rows × cols), **Space:** O(1)

Pseudocode:

```cpp
void wavePrint(vector<vector<int>>& matrix) {
    int rows = matrix.size(), cols = matrix[0].size();
    for (int col = 0; col < cols; col++) {
        if (col % 2 == 0)
            for (int row = 0; row < rows; row++)
                cout << matrix[row][col] << " ";
        else
            for (int row = rows - 1; row >= 0; row--)
                cout << matrix[row][col] << " ";
    }
}
```

Dry Run:
Input:

```
1 2 3
4 5 6
7 8 9
```

→ Output: **`1 4 7 8 5 2 3 6 9`**

2. Spiral Matrix Print

Problem:
Print matrix elements in spiral order.

Approach:

- Define **`top`**, **`bottom`**, **`left`**, **`right`**
- Move in spiral: top → right → bottom → left
- **Time:** O(rows × cols), **Space:** O(1)

Pseudocode:

```cpp
void spiralOrder(vector<vector<int>>& matrix) {
    int top = 0, bottom = matrix.size() - 1;
    int left = 0, right = matrix[0].size() - 1;

    while (top <= bottom && left <= right) {
        for (int i = left; i <= right; i++) cout << matrix[top][i] << " ";
        top++;
        for (int i = top; i <= bottom; i++) cout << matrix[i][right] << " ";
        right--;
        if (top <= bottom)
            for (int i = right; i >= left; i--) cout << matrix[bottom][i] << " ";
        bottom--;
        if (left <= right)
            for (int i = bottom; i >= top; i--) cout << matrix[i][left] << " ";
        left++;
    }
}
```

Dry Run:
Input:

```
1 2 3
4 5 6
7 8 9
```

→ Output: **`1 2 3 6 9 8 7 4 5`**

---

3. Rotate Matrix by 90° Clockwise

Problem:
Rotate an N×N matrix 90° clockwise in-place.

Approach: Transpose + Reverse Rows

- **Time:** $O(N^2)$, **Space:** O(1)

Pseudocode:

```
cpp

void rotateMatrix(vector<vector<int>>& matrix) {
    int n = matrix.size();
    for (int i = 0; i < n; i++)
        for (int j = i + 1; j < n; j++)
            swap(matrix[i][j], matrix[j][i]);
    for (int i = 0; i < n; i++)
        reverse(matrix[i].begin(), matrix[i].end());
}
```

Dry Run:
Input:

```
1 2 3
4 5 6
7 8 9
```

→ Output:

```
7 4 1
8 5 2
9 6 3
```

---

4. Search in 2D Matrix

Problem:
Search a target value in a sorted 2D matrix (rows & columns sorted).

Approach: Binary Search (Flattened)

- Treat matrix as 1D.
- **Time:** O(log(n×m)), **Space:** O(1)

Pseudocode:

```cpp
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int rows = matrix.size(), cols = matrix[0].size();
    int low = 0, high = rows * cols - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;
        int midElement = matrix[mid / cols][mid % cols];
        if (midElement == target) return true;
        else if (midElement < target) low = mid + 1;
        else high = mid - 1;
    }
    return false;
}
```

Let me know if you'd like to compile all of this into a printable PDF or add more topics like string, stack, or DP problems. I can also 🔭research this topic if you're preparing for a specific company's DSA rounds.
❞