📜 🗓️ DP Practice 1 Summary — June 10, 2025

---

1. Fibonacci Number

- **Problem:** Return the nth Fibonacci number.
- **Approach:** Bottom-up DP with memoization.
- **Code:**

```
vector<int> dp(n+1);
dp[0] = 0;
dp[1] = 1;
for (int i = 2; i <= n; i++) {
   dp[i] = dp[i-1] + dp[i-2];
}
return dp[n];
```

- **Clue:** Classic base-case + recurrence problem.

---

2. House Robber

- **Problem:** Maximize money without robbing adjacent houses.
- **Approach:** DP with recurrence dp[i] = max(dp[i-1], dp[i-2] + nums[i])
- **Fix:** You had dp = list(n) which needed to be dp = [0]*n in Python.
- **Edge Case:** Only one house ⇒ return nums[0].

---

3. Unique Paths

- **Problem:** Count unique paths from top-left to bottom-right.
- **Issue:** Incorrect initialization using [[1]*n]*m (shared rows).
- **Fix:** Use list comprehension: [[1 for _ in range(n)] for _ in range(m)]
- **DP Transition:**

```
dp[i][j] = dp[i-1][j] + dp[i][j-1]
```

---

4. Minimum Path Sum

- ✅ You implemented **two approaches**:

A. Bottom-Up DP

dp[i][j] = grid[i][j] + min(dp[i-1][j], dp[i][j-1]);

B. Dijkstra-Like Approach

priority_queue<tuple<int, int, int>> pq;

dist[x][y] = min cost to reach (x, y)

- **Converted to Python** with a heap (heapq) and explained dry run.

---

5. Maximum Subarray (Kadane's Algorithm)

- **Problem:** Find the subarray with maximum sum.
- **Approach:**

curr = max(nums[i], curr + nums[i])

max_sum = max(max_sum, curr)

- **Clue:** Local vs global max.

---

6. Longest Increasing Subsequence (LIS)

- **Problem:** Length of the longest strictly increasing subsequence.
- **Issue:** Buggy logic with only comparing nums[i] > nums[i-1]
- **Fix:** Use nested loops:

```
for (int i = 1; i < n; ++i)
 for (int j = 0; j < i; ++j)
  if (nums[i] > nums[j])
   lis[i] = max(lis[i], lis[j] + 1);
```

---

7. Edit Distance

- ✅ Implemented correctly in Java
- **Operations Allowed:** Insert, Delete, Replace

- **Transition:**

if (a[i-1] == b[j-1]) → dp[i][j] = dp[i-1][j-1]

else dp[i][j] = 1 + min(replace, insert, delete)

- **Dry run:** horse → ros = 3 operations
- **Table size:** (n+1) x (m+1)

✅ **Concepts Practiced Today**

Concept	Notes
1D/2D DP	Fibonacci, Robber, LIS, Kadane's
Tabulation	Used bottom-up DP extensively
Edge cases	Proper initialization critical
Space Optimization	Yet to be explored
Graph-like DP	Dijkstra for Min Path Sum
String DP	Edit Distance, character matching
Python/C++ mastery	You switched fluently between them

📌 **Suggestions for Next Steps**

Area	Challenge
Space Optimization	Use 1D array when only last row is needed
Backtracking the solution	For Edit Distance, LIS
Memoization	Try top-down recursion with cache
Harder DP Problems	Like Matrix Chain Multiplication, LCS with constraint