Dynamic Programming

Ali Akbari

June 2025

Dynamic Programming

- Solves problems by breaking into overlapping subproblems.
- Stores results to avoid redundant computation.
- Examples: Fibonacci, Coin Change, 0/1 Knapsack.

Fibonacci Example

- **Problem**: Compute 6th Fibonacci number: $F(0) = 0$, $F(1) = 1$, $F(n) = F(n-1) + F(n-2)$.
- Result: $F(6) = 8$.
- **Step-by-Step**: Use table to store values.
  1. $F(2) = F(1) + F(0) = 1 + 0 = 1$
  2. $F(3) = F(2) + F(1) = 1 + 1 = 2$
  3. $F(6) = F(5) + F(4) = 5 + 3 = 8$

Coin Change Example

- **Problem**: Minimum coins for amount 7 using coins [1, 2, 5].
- **Step-by-Step**:
  1. Amount 1: 1 coin (1).
  2. Amount 2: 1 coin (2).
  3. Amount 7: Min(1+dp[6], 1+dp[5], 1+dp[2]) = 2 (coin 5 + coin 2).

0/1 Knapsack Example

- **Problem**: Items [(value=60, weight=10), (100, 20), (120, 30)], capacity 50.
- Maximize value without exceeding capacity.
- **Step-by-Step**: Use table to choose items.
  1. Take item 1 (60, 10), item 2 (100, 20): Total value 160.
  2. Add item 3 (120, 30): Exceeds capacity.
  3. Final: Value 220 (items 1 and 2).

Dynamic Programming

- **Fibonacci**: $O(n)$ time, $O(n)$ space.
- **Coin Change**: $O(\text{amount} \times \text{coins})$ time.
- **Knapsack**: $O(n \times \text{capacity})$ time.
- Stores subproblem solutions in tables.