# Dynamic Programming (Introduction)

## UCF Programming Team

## Fall 2021

### Algorithm Design Techniques

1. Divide and Conquer
   Examples: Binary search, Merge sort
2. Greedy
   Examples: Huffman coding, Minimum spanning tree,
   Optimal merge patterns
3. Dynamic Programming (DP)

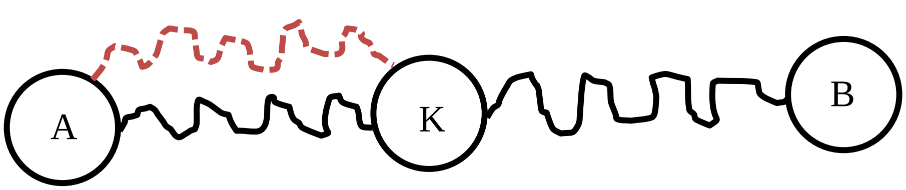
### Dynamic Programming

- Applies to optimization problems.
- Think in terms of a brute force solution but avoid redoing work you have already done.
- Store answers to subproblems you have already solved (usually using tables).
- Usually have the option of ***Iterative DP*** or ***Recursive DP***.
- Principle of Optimality
- Recurrence Relation
- Memoization (Memo"ize")

### Principle of Optimality

Subsequence of the sequence of decisions leading to the optimal solution must also be optimal on the corresponding subproblems.

Example: shortest path from A to B



## Example 1: Longest Decreasing Sequence (LDS)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Values | 5 | 40 | 20 | 30 | 25 | 80 | 10 | 35 |
| LDS | 1 | 1 | 2 | 2 | 3 | 1 | 4 | 2 |

Finding the sequence: find the largest value in LDS, then go backward from there.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| LDS | 1 | **1** | 2 | **2** | **3** | 1 | **4** | 2 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Values | 5 | **40** | 20 | **30** | **25** | 80 | **10** | 35 |

Another example for LDS:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Values | 389 | 207 | 155 | 300 | 299 | 170 | 158 | 65 |
| LDS | 1 | | | | | | | 6 |

Finding the sequence: find the largest value in LDS, then go backward from there.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| LDS | 1 | | | | | | | 6 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Values | 389 | 207 | 155 | 300 | 299 | 170 | 158 | 65 |

## Example 2: Longest (Greatest) Common Subsequence

Common Subsequence: sequence of characters contained in both strings and in the same order

Longest Common Subsequence: longest such string

$String_1$: E **P** F **L** **A** G **Y**

$String_2$: **P** **L** H **A** I **Y** J

| | null | E | **P** | F | **L** | **A** | G | **Y** |
|---|---|---|---|---|---|---|---|---|
| null | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 |
| **P** | 0 | 0 | **1** | **1** | 1 | 1 | 1 | 1 |
| **L** | 0 | 0 | 1 | 1 | **2** | 2 | 2 | 2 |
| H | 0 | 0 | 1 | 1 | **2** | 2 | 2 | 2 |
| **A** | 0 | 0 | 1 | 1 | 2 | **3** | **3** | 3 |
| I | 0 | 0 | 1 | 1 | 2 | 3 | **3** | 3 |
| **Y** | 0 | 0 | 1 | 1 | 2 | 3 | 3 | **4** |
| J | 0 | 0 | 1 | 1 | 2 | 3 | 3 | **4** |

```
If match at i,j
     matrix[i][j] = 1 + matrix[i-1][j-1]

If not match at i,j
     matrix[i][j] = larger of matrix[i-1][j] and matrix[i][j-1]
```

$string_1$: - - - - - . . .     $string_2$: - - - - . . .

To find LCS:

- Move up or left through the matrix until you must change numbers, then move diagonally.
- Each diagonal jump indicates character in the subsequence.
- Once the matrix entry is a zero, stop.

Another example for LCS:

$String_1$: A B C A B B A

$String_2$: C B A B A C

| | null | A | B | C | A | B | B | A |
|---|---|---|---|---|---|---|---|---|
| null | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | | | | | | | |
| B | 0 | | | | | | | |
| A | 0 | | | | | | | |
| B | 0 | | | | | | | |
| A | 0 | | | | | | | |
| C | 0 | | | | | | | 4 |

Two answers: C A B A and B A B A

# Example 3: Matrix Multiplications

Matrix A, Matrix B, Matrix C

$$
\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1q} \\ a_{21} & a_{22} & \dots & a_{2q} \\ \dots & & & \\ a_{p1} & a_{p2} & \dots & a_{pq} \end{pmatrix}
\begin{pmatrix} b_{11} & b_{12} & \dots & b_{1t} \\ b_{21} & b_{22} & \dots & b_{2t} \\ \dots & & & \\ b_{q1} & b_{q2} & \dots & b_{qt} \end{pmatrix}
\qquad
\begin{pmatrix} c_{11} & c_{12} & \dots & c_{1t} \\ c_{21} & c_{22} & \dots & c_{2t} \\ \dots & & & \\ c_{p1} & c_{p2} & \dots & c_{pt} \end{pmatrix}
$$

$p \times q$, $q \times t$, $p \times t$

Matrix A, Matrix B, Matrix C

$$
\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1q} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2q} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3q} \\ \dots & & & & \\ a_{p1} & a_{p2} & a_{p3} & \dots & a_{pq} \end{pmatrix}
\begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} & \dots & b_{1t} \\ b_{21} & b_{22} & b_{23} & b_{24} & \dots & b_{2t} \\ b_{31} & b_{32} & b_{33} & b_{34} & \dots & b_{3t} \\ \dots & & & & & \\ b_{q1} & b_{q2} & b_{q3} & b_{q4} & \dots & b_{qt} \end{pmatrix}
\begin{pmatrix} \dots & \\ \dots & c_{24} \\ & \\ & \end{pmatrix}
$$

```
for i = 1 to number_of_rows[A]
     for j = 1 to number_of_columns[B]

          C[i][j] = 0
          for k = 1 to number_of_columns[A]
               C[i][j] = C[i][j] + (A[i][k] * B[k][j])
          end for

     end for
end for
```

$M_1$: $5 \times 10$

$M_2$: $10 \times 15$

$M_3$: $15 \times 1$

Compute $M_1\ M_2\ M_3$

$(M_1\ M_2)\ M_3$

$(5 * 10 * 15) + 5 * 15 * 1 = 825$ multiplications

$M_1\ (M_2\ M_3)$

$5 * 10 * 1 + (10 * 15 * 1) = 200$ multiplications

Optimization Problem:

- Given n matrices $M_1, M_2, \ldots, M_n$
- Determine the number of multiplications needed in the optimal case to compute $M_1\ M_2 \ldots M_n$

Assume that the dimensions of $M_i$ are $p_i \times q_i$ for $1 \le i \le n$

Let cost[i][j] denote the number of multiplications needed to compute $M_i\ M_{i+1} \ldots M_{j-1}\ M_j$ optimally.

Then,

$$\text{cost}[1][n] = \min_{1 \le k < n} \{\text{cost}[1][k] + \text{cost}[k+1][n] + p_1 * q_k * q_n\}$$

This recurrence corresponds to generating n-1 feasible solutions and then taking the one of minimum cost:

| | |
|---|---|
| $M_1$ | $M_2\ M_3\ M_4\ M_5\ M_6 \ldots M_n$ |
| $M_1\ M_2$ | $M_3\ M_4\ M_5\ M_6 \ldots M_n$ |
| $M_1\ M_2\ M_3$ | $M_4\ M_5\ M_6 \ldots M_n$ |
| $M_1\ M_2\ M_3\ M_4$ | $M_5\ M_6 \ldots M_n$ |
| … | … |
| $M_1\ M_2\ M_3 \ldots M_{n-1}$ | $M_n$ |

Finding the best way to multiply $M_2\ M_3\ M_4 \ldots M_n$

| | |
|---|---|
| $M_2$ | $M_3\ M_4\ M_5\ M_6 \ldots M_n$ |
| $M_2\ M_3$ | $M_4\ M_5\ M_6 \ldots M_n$ |
| $M_2\ M_3\ M_4$ | $M_5\ M_6 \ldots M_n$ |
| $M_2\ M_3\ M_4\ M_5$ | $M_6 \ldots M_n$ |
| … | … |
| $M_2\ M_3\ M_4 \ldots M_{n-1}$ | $M_n$ |

$$\begin{array}{cccc} M_1 & M_2 & M_3 & M_4 \\ p_1 \times q_1 & p_2 \times q_2 & p_3 \times q_3 & p_4 \times q_4 \end{array}$$

| cost[i][j] | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1st | 4th | **6th** |
| 2 | | 0 | 2nd | 5th |
| 3 | | | 0 | 3rd |
| 4 | | | | 0 |

cost[1][2] = cost[1][1] + cost[2][2] + $p_1$ * $q_1$ * $q_2$ $M_1$ $M_2$

cost[2][3] = cost[2][2] + cost[3][3] + $p_2$ * $q_2$ * $q_3$ $M_2$ $M_3$

cost[3][4] = cost[3][3] + cost[4][4] + $p_3$ * $q_3$ * $q_4$ $M_3$ $M_4$

cost[1][3] = $M_1$ $M_2$ $M_3$
min{ (cost[1][1] + cost[2][3] + $p_1$ * $q_1$ * $q_3$) , $M_1$ ($M_2$ $M_3$)
(cost[1][2] + cost[3][3] + $p_1$ * $q_2$ * $q_3$) } ($M_1$ $M_2$) $M_3$

cost[2][4] = $M_2$ $M_3$ $M_4$
min{ (cost[2][2] + cost[3][4] + $p_2$ * $q_2$ * $q_4$) , $M_2$ ($M_3$ $M_4$)
(cost[2][3] + cost[4][4] + $p_2$ * $q_3$ * $q_4$) } ($M_2$ $M_3$) $M_4$

cost[1][4] = $M_1$ $M_2$ $M_3$ $M_4$
min{ (cost[1][1] + cost[2][4] + $p_1$ * $q_1$ * $q_4$) , $M_1$ ($M_2$ $M_3$ $M_4$)
(cost[1][2] + cost[3][4] + $p_1$ * $q_2$ * $q_4$) , ($M_1$ $M_2$)($M_3$ $M_4$)
(cost[1][3] + cost[4][4] + $p_1$ * $q_3$ * $q_4$) } ($M_1$ $M_2$ $M_3$) $M_4$