

Optimization

Optimization problems have the form: Find a solution minimizing/maximizing some value.

Examples:

Multiply a sequence of matrices minimizing the number of multiplications.

From two strings, find a common subsequence of maximum length.

Give change minimizing the number of coins.

Schedule a room to maximize the number of meetings.

Find a character code to minimize the length of a string.

Find a spanning tree with the minimum sum of weights.

Find a path between two vertices of minimum weight.

Find the smallest set of edges covering all vertices.

Dynamic Programming

Dynamic programming is a technique for solving some optimization problems.

A dynamic programming solution to a problem usually involves:

1. Characterize the structure of an optimal solution.
2. Recursively define an optimal solution in terms of optimal solutions for smaller problems.

-
3. Design an algorithm that computes the optimal value for each subproblem once.
 4. Construct an optimal solution.

Notation:

Use $\text{opt}(\dots)$ to denote optimal value.

Use $\text{arg}(\dots)$ to denote optimal parameters.

Matrix Chain Multiplication

Suppose A_1 , A_2 , and A_3 are matrices of size $k \times l$, $l \times m$, and $m \times n$.

$(A_1 A_2) A_3$ results in $klm + kmn$ multiplications.

$A_1 (A_2 A_3)$ results in $lmn + kln$ multiplications.

Structure:

To perform $(A_1 \dots A_k)(A_{k+1} \dots A_n)$, use optimal solutions for $A_1 \dots A_k$ and $A_{k+1} \dots A_n$.

Recursive definition:

Let A_i have dimensions $p_{i-1} \times p_i$.

Let $\text{opt}(i, j)$ be the minimum number of multiplications for $A_i \dots A_j$

if $i = j$, then $\text{opt}(i, j) = 0$

else $\text{opt}(i, j) =$

$$\min_{k=i}^{j-1} (\text{opt}(i, k) + \text{opt}(k + 1, j) + p_{i-1} p_k p_j)$$

Algorithm: $P[0 \dots n]$ are the dimensions.

```

MATRIX-CHAIN-ORDER( $P[0 \dots n]$ )
for  $i \leftarrow 1$  to  $n$  do  $\text{opt}[i, i] \leftarrow 0$ 
for  $j \leftarrow 2$  to  $n$ 
  do for  $i \leftarrow j - 1$  downto  $1$ 
    do  $\text{opt}[i, j] \leftarrow \infty$ 
      for  $k \leftarrow i$  to  $j - 1$ 
        do  $q \leftarrow \text{opt}[i, k] + \text{opt}[k + 1, j]$ 
           $+ P[i - 1]P[k]P[j]$ 
          if  $q < \text{opt}[i, j]$ 
            then  $\text{opt}[i, j] \leftarrow q$ 
               $\text{arg}[i, j] \leftarrow k$ 

```

Construction: $A[i]$ is the i th matrix.

```

MATRIX-CHAIN-MULTIPLY( $A, i, j$ )
  if  $i = j$ 
    then return  $A[i]$ 
  else  $X \leftarrow \text{M.-C.-M.}(A, i, \text{arg}[i, j])$ 
     $Y \leftarrow \text{M.-C.-M.}(A, \text{arg}[i, j] + 1, j)$ 
    return MATRIX-MULTIPLY( $X, Y$ )

```

Longest Common Subsequence

Longest common subsequence (LCS) of “kahijedcdelln” and “hzpidysvdquen” is “hidden”. An example application is diff.

Structure:

Let $X = \langle x_1, \dots, x_m \rangle$ and $Y = \langle y_1, \dots, y_n \rangle$.

Let $Z = \langle z_1, \dots, z_k \rangle$ be an LCS of X and Y .

1. If $x_m = y_n$, then $z_k = x_m = y_n$, and $Z[1 \dots k-1]$ is an LCS of $X[1 \dots m-1]$ and $Y[1 \dots n-1]$.

2. If $x_m \neq y_n$, then $z_k \neq x_m$ or $z_k \neq y_n$, so Z is an LCS of $X[1 \dots m-1]$ and Y , or Z is an LCS of X and $Y[1 \dots n-1]$.

Recursive Definition:

Let $\text{opt}(m, n)$ be the LCS length of $\langle x_1, \dots, x_m \rangle$ and $\langle y_1, \dots, y_n \rangle$

if $m = 0$ or $n = 0$, then $\text{opt}(m, n) = 0$

else if $x_m = y_n$, then

$$\text{opt}(m, n) = 1 + \text{opt}(m-1, n-1)$$

else

$$\text{opt}(m, n) = \max(\text{opt}(m-1, n), \text{opt}(m, n-1))$$

