Dynamic Programming

- Store answer in a table; retrieve from table when needed.
- DP solves each subproblem once.
- Subproblems needed, repeatedly solving common subproblems.
- 
- D&C would do more work than necessary when subproblems are not independent (else divide-and-conquer).
- Applicable when subproblems are applicable to subproblems.
- Solves problems by combining solutions to subproblems.
1. Characterize structure of optimal solution.
2. Recursively define value of optimal solution.
3. Compute value of optimal solution (e.g.: shortest path vs. shortest path length from 3).
4. Construct actual optimal solution (e.g.: shortest path).

To design a DP algorithm:
- Optimal value may not be unique.
- Find solution of optimal value.

DP is usually applied to optimization problems.

(DP (cont.))
Matrix-Chain Multiplication

Order to perform matrix multiplications is minimized.
Operations (additions and multiplications) such that the total number of scalar multiplications is minimized.

Input: Sequence $A_1, A_2, \ldots, A_n$ of matrices.
Output: An order to compute $A_1 A_2 \cdots A_n$.
Want: Fully parenthesized $A_{1\times 2} A_{2\times 3} A_{3\times 4} \ldots A_{n\times 1}$ minimize cost of multip.

- $A_1: p\times l$ size matrix.
  - $A_1 A_{p\times l}$ size matrix.
- $A_{p\times l} B_{l\times q} = C_{p\times q}$ requires \textit{pq} operations.
- The way to parenthesize: big impact on cost of multip.

Example:

\[
\begin{align*}
A_1 (A_2 (A_3 A_4)) &\rightarrow (A_1 A_2) (A_3 A_4) \\
A_1 (A_2 (A_3 A_4)) &\rightarrow (A_1 (A_2 A_3)) A_4 \\
A_1 (A_2 (A_3 A_4)) &\rightarrow (A_1 A_2) (A_3 A_4) \\
A_1 (A_2 (A_3 A_4)) &\rightarrow (A_1 A_2 A_3) A_4 \\
\end{align*}
\]

(multiplication is associative).

No matter the order of multiplications, final result is the same.

- Product of 2 fully parenthesized matrix products surrounded by parentheses.
  - Product of 2 fully parenthesized matrix products surrounded by a single matrix or a single matrix product.
Counting # of parenthesizations

- check all possibilities

- \( P(n) \) : # of parenthesizations

  - for \( k = 1, 2, \ldots, n-1 \) do :
    - split \( A_1 \cdot A_2 \ldots A_n \) at \( k, k+1 \)
    - parenthesize \( A_1 \cdot A_2 \ldots A_k \) independently
      \( A_{k+1} \cdot A_{k+2} \ldots A_n \)

- Recurrence : \( P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k) P(n-k) & \text{if } n \geq 2 \end{cases} \)

- Solution : \( P(n) = C(n-1) \)

  - sequence of Catalan numbers

- \( C(n) = \frac{1}{n+1} \binom{2n}{n} = \Omega \left( \frac{4^n}{n^{3/2}} \right) \)

  \( \rightarrow \) exponential
1. Structure of optimal solution

- **Notation**: $A_{i:j}$ \rightarrow matrix obtained after evaluating product $A_i \cdot A_{i+1} \ldots A_j$

- Optimal solution: split of $A_1 \cdot A_2 \ldots A_n$ at some $k, k+1$ \(1 \leq k < n\):

  - first compute $A_{1..k}, A_{k+1..n}$
  
  \[ \text{\checkmark multiply} \]
  $A_{1..n}$

  - $A_{1..k}, A_{k+1..n}$ must be optimal parenthesisations

  \[ \Rightarrow \text{optimal solution contains within it} \]
  \[ \text{optimal solutions to subproblems} \]

  \[ \downarrow \]

  key ingredient for applying DP!
2. Recursive solution:

- Define value of optimal solution recursively

- Subproblems: Compute minimum cost of
  \[ A_i A_{i+1} \ldots A_j, \quad 1 \leq i \leq j \leq n \]

- Let \( m[i, j] \) be min # scalar multiplications to compute \( A_i \ldots A_j \)
  
  \( m[1, n] \) is optimum for \( A_1 \ldots A_n \)

- \( i = j \Rightarrow m[i, i] = 0 \)

- \( i < j \) to have split at \( A_k, A_{k+1}, \quad i \leq k < j \)
  
  \( \Rightarrow (A_i A_{i+1} \ldots A_k) (A_{k+1} A_{k+2} \ldots A_j) \)

  \( \Rightarrow m[i, j] \) = \( m[i, k] + m[k+1, j] + P_i P_k P_j \)

- But we don't know \( k \)!

- \( j-i \) possible values for \( k \):
  
  \( i, i+1, \ldots, j-1 \)

- Optimal parenthesisation has one of them
  
  \( \Rightarrow \) check all

\[ m[i, j] = \begin{cases} 
0, & \text{if } i = j \\
\min_{i \leq k < j} \{ m[i, k] + m[k+1, j] + P_i P_k P_j \}, & \text{if } i < j
\end{cases} \]
Note: $m[i, j]$ gives cost. To obtain actual parenthesisation define $s[i, j] = k$ for optimal split of $A_i \ldots A_j$.

3. Computing optimal cost

- Recursive algorithm to compute $m[1, n] \rightarrow$ exponential time!

  - Key observation:
    - "Few" subproblems: one for each pair $i, j$, $1 \leq i \leq j \leq n$;
    $$\Rightarrow \binom{n}{2} + n = \Theta(n^2) \text{ subproblems}$$
    - A recursive algorithm may encounter a subproblem many times
    $$\Rightarrow \text{overlapping subproblems}$$
    - Key ingredient for applying DP!
Perform 3rd step of DP: compute optimal cost 
bottom-up

\[ A_i \rightarrow p_{i-1} \times p_i \]

Input: \(<p_0, p_1, \ldots, p_n>\)

Data structures: 
- table \(m[1..n, 1..n]\) to store \(m[i, j]\)
- \(s[1..n, 1..n]\) to store \(s[i, j]\)

**MCO (p)**

\[
\begin{align*}
n &= \text{length}(p) - 1; \\
\text{for } i &= 1 \text{ to } n \text{ do } m[i, i] = 0; \\
\text{for } l &= 2 \text{ to } n \text{ do } \\
    \quad \text{for } i &= 1 \text{ to } n-l+1 \text{ do } \\
    \quad \quad j &= i + l - 1; \\
    \quad m[i, j] &= \infty; \\
    \quad \text{for } k &= i \text{ to } j-1 \text{ do } \\
    \quad \quad z &= m[i, k] + m[k+1, j] + p_{i-1} p_k p_j; \\
    \quad \quad [\text{if } z < m[i, j] \text{ then] } \\
    \quad \quad \quad m[i, j] &= z; \quad s[i, j] = k \\
\end{align*}
\]

I.e.: 
- first \(m[i, i]\), \(i = 1, n\)
- next \((l=2)\): \(m[i, i+1]\), \(i = 1, n-1\)
- next \((l=3)\): \(m[i, i+2]\), \(i = 1, n-2\)
- etc. \(m[i, j]\) depends on already computed \(m[k, k+1, j]\)
Example:

- $A_1: 30 \times 35$
- $A_2: 35 \times 15$
- $A_3: 15 \times 5$
- $A_4: 5 \times 10$
- $A_5: 10 \times 20$
- $A_6: 20 \times 25$

\[ \rho_0 = 30 \]
\[ \rho_1 = 35 \]
\[ \rho_2 = 15 \]
\[ \rho_3 = 5 \]
\[ \rho_4 = 10 \]
\[ \rho_5 = 20 \]
\[ \rho_6 = 25 \]

\[ m[1,3] \rightarrow A_{1..2} \cdot A_3 \]:
\[ 15 \cdot 750 + 30 \cdot 15 \cdot 5 \]
\[ A_1 \cdot A_{2..3} \]:
\[ 2625 + 30 \cdot 35 \cdot 5 \]

\[ m[2,5] = \min \{ \begin{cases} m[2,2] + m[3,5] + \rho_1 \rho_2 \rho_5 = 13000 \\
 m[2,3] + m[4,5] + \rho_1 \rho_3 \rho_5 = 7125 \\
 m[2,4] + m[5,5] + \rho_1 \rho_4 \rho_5 = 11375 \end{cases} \} \]

4. Constructing optimal solution to have value from step 3

- How to multiply matrices?!

  - Use $S[1..n, 1..n]$
  - $S[i,j]$ records $k$, $i \leq k \leq j$, for optimal parenthesisation of $A_{i..j}$
  - Final multiplication in $A_{1..n} = A_{1..\rho_1, n} A_{\rho_1, n+1..n}$