Dynamic Programming (DP)

All pairs shortest path problem

Used DP: $O(n^3)$

Conceptually: break up the larger problem into smaller "optimal" subproblems

1) Recursive definition
2) Initialization (base case)
3) Recursive function (bigger in terms of smaller) $\leftarrow$ backward
   forward solution (from smaller to larger)

Conceptually: Computational Data implicit

Longest Increasing Subsequence problem (LIS)

Input: sequence of numbers

Positions: $\rightarrow$ 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Values: $\rightarrow$ 9 1 6 4 2 1 5 2 6 2 8 1 5

Subsequence: choose any subset of the positions, but preserve the order

Ordered subset of pos: $<2, 5, 8>$

Subsequence: $1, 2, 3$

Increasing

Talk: What’s the longest increasing subsequence

$\Rightarrow$ length (optimization value)

Actual subsequence (prev)

$<2, 5, 8, 14>$
\[ S = <2, 3, 4, 5, 11> \]

Output the algorithm

\[ \text{len} = 5 \]

\[ |S| = n \]

# of elements or seq length

Very large search space

What is the size?

Assume that the max length is \( k \)

\[
\sum_{i=0}^{k} n^i = \frac{n^{k+1} - 1}{n - 1} \approx O(n^k)
\]

\[ n = |S| \]

\[ k = \text{max subseq length} \]

\[
[1, 2, 3, 4, 5, 7, 8, 9, 10, \ldots, N]
\]

\[ O(n^k) \quad k = n \]

\[ \text{worst case (factorial at least)} \]

DP for LIS
1) \[ L[j] = \text{longest increasing subsequence ending at pos } j \text{ in } S \]

\[ S = \{ 8 \text{ start symbol}, 9, 1, 6, 4, 3, 15, 2, 4, 8, 1, 5, \text{ end symbol} \} \]

2) \[ \text{base case: add a fake value } \& \]
\[ L[0] = 0 \leftarrow \text{always start at pos 0} \]

3) \[ \text{Recursive equation:} \]
\[ L[j] = \max_{i < j} \begin{cases} L[i], & S[i] < S[j] \text{ (value)} \\ \text{Prev}[j], & \text{otherwise} \end{cases} \]

\[ \text{Prev}[j] = \arg \max \{ L[i], \text{ for } i < j \} \]

\[ \text{LIS value: } \max_{j=1, \ldots, n} \{ L[j] \} \leftarrow \text{implicit DAG.} \]

\[ L[n+1] = \text{optimal value LIS} \leftarrow \text{Subtract 1 to get optimal LIS value} \]

4) \[ \text{Forward solution:} \]
\[ S = \{ 8, 9, 1, 6, 4, 3, 15, 2, 4, 8, 1, 5, \text{ end symbol} \} \]

\[ L = \begin{align*}
0 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 4 & 5 & 5 & 6 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{align*} \]

\[ \text{implies optimal LIS } = 6 - 1 = 5 \]

\[ L[3] = \text{LIS upto pos 3} \]

\[ |L| = n + 2 \quad (n + 2) \quad \left| S \right| \quad \left| S \right| \quad 0 \]

\( O(n) \text{ elements to compute} \)

\( \alpha(n) \text{ steps to compute} \)

\( L[i]: \)
\[
\frac{1}{|S_1|} \cdot \frac{2}{|S_2|} \leq \alpha \text{ at most } \alpha n^2 \text{ steps to compute } L[i, j]
\]

\[
O(n) \times O(n) = \left( \sum_{i=1}^{n} \text{ independent of } k \right) O(n^2)
\]

\[
O(n^k) \quad \text{dependent on } \quad k
\]

---

**Edit Distance**

\[ \Rightarrow \text{ global alignment} \]

**Input:** 

\[ \text{BRILLIANT, THRILLING} \]

\[ S_1, S_2 \]

- two sequences over some alphabet \( \Sigma \)

- task: find the least \(# \text{ of change} \) to convert one sequence to the other

- edit distance

  - 1) substitute one letter for another
  - 2) delete/insert one letter

**Alignment/Correspondence**

- put the characters in the two sequences in a one-to-one correspondence, allowing for gaps (allow a special character '-' )

\[
\begin{array}{c}
\text{alignment} \\
\text{disallowed!}
\end{array}
\]

\[ \text{match} \]

\[ \text{mismatch} \]

\[ \text{indel/gap} \]

\[ \text{no-op} \]
\[
\text{edit distance} = \# \text{ mismatches} + \# \text{ gaps} = 2 + 2 = 4
\]

\[S_1 = \text{BRILLIANT} \rightarrow \text{THRILLING} = S_2\]

1. \(B \rightarrow T\)
2. insert \(H\) after pos 1 in \(S_1\)
3. delete \(A\) in \(S_1\)
4. \(T \rightarrow G\)

Task: find the minimum edit distance between \(S_1\) and \(S_2\)

Real-world use case: evolutionary tree (phylogeny)

gene / genome

\[
\text{thrid:} \quad \left[ \begin{array}{c}
\text{BRILLIANT} \\
\text{---} \\
\text{THRILLING}
\end{array} \right] \quad \text{also a valid alignment}
\]

\[
\left[ \begin{array}{c}
\text{BRILLIANT} \\
\text{THRILLING}
\end{array} \right] \quad \text{edit} = 2
\]

Q: How many alignments are there?

\(|S_1| = n \quad |S_2| = m \quad m \leq n\)

Ans: \(\binom{n+m}{m}\), e.g. \(m = n\)

Choose \((2^n)\) possible \((2^n)\) permutations + \(2^n\)
\[
\binom{N+m}{m} = \frac{(N+m)!}{n! \cdot m!}
\]

\[
\text{Vs}\quad DP = O(mn)
\]

\[
\text{polyomial/quadratic}
\]

\[S_1 = \text{SOAP}, \quad S_2 = \text{SOMA}\]

1) \[L(i, j) = \text{least edit distance between } S_1[i, j, \ldots, i] \text{ and } S_2[1, 2, \ldots, j]\]

\[
\text{final answer} = L(n, m)
\]

\[S_1 \sim \text{SOAP}\]

\[S_2 \sim \text{SOMA}\]

2) base case

\[
\begin{align*}
L(0, 0) &= 0 \\
L(0, j) &= j \\
L(i, 0) &= i
\end{align*}
\]

\[\forall 1 \leq j \leq m, \quad \forall 1 \leq i \leq n\]

3) Recursive equation

\[
L(i, j) = \min \left\{ L(i-1, j-1) + \text{edit cost} (S_1[i], S_2[j]), L(i-1, j), L(i, j-1) \right\}
\]
\[
\begin{align*}
L(i, j) &= \min \left\{ \begin{array}{l}
L(i-1, j-1) + \text{edit cost } (S_1[i], S_2[j]) \\
L(i-1, j) + 1 \quad \text{gap} \\
L(i, j-1) + 1 \quad \text{match/mismatch}
\end{array} \right. \\
\text{prev}(i, j) &= \arg \min \left\{ \\
S_1[i] = \text{SOAP} \\
S_2[j] = \text{SOMA}
\right. \\
\end{align*}
\]

\[\text{case to consider}\]

\[
\begin{align*}
\text{put in correspondence} & \quad \text{in } S_1 \\
\text{insertion at } p & \quad \text{delete } p
\end{align*}
\]