Lecture 8: Dynamic Programming II

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

October 5, 2023

Overview

- Longest Common Subsequence
- Minimum Length Triangulation
- Acknowledgements

Longest Common Subsequence (LCS)

- **Input:** Two strings $a_1 a_2 \cdots a_m$ and $b_1 b_2 \cdots b_n$, where $a_i, b_j \in \Sigma$
- **Output:** Largest k such that there is $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ for which $a_{i_\ell} = b_{j_\ell}$ for $\ell \in [k]$
- **Model:** Word RAM model
- **Example:** given two DNA sequences, want to identify common structures.

$$\mathrm{AA}\mathit{ACC}\mathrm{G}\mathit{T}\mathrm{G}\mathit{AG}$$
$$\mathrm{C}\mathit{ACC}\mathrm{CC}\mathit{TA}\mathrm{A}\mathit{G}\mathrm{CC}$$

Longest Common Subsequence (LCS)

- **Input:** Two strings $a_1 a_2 \cdots a_m$ and $b_1 b_2 \cdots b_n$, where $a_i, b_j \in \Sigma$
- **Output:** Largest k such that there is $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ for which $a_{i_\ell} = b_{j_\ell}$ for $\ell \in [k]$
- how to DP this?

Need to find many *overlapping subproblems* (few subproblems)

and

optimal substructure (nice recurrence)

Longest Common Subsequence (LCS)

- **Input:** Two strings $a_1 a_2 \cdots a_m$ and $b_1 b_2 \cdots b_n$, where $a_i, b_j \in \Sigma$
- **Output:** Largest k such that there is $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ for which $a_{i_\ell} = b_{j_\ell}$ for $\ell \in [k]$
- how to DP this?

  Need to find many *overlapping subproblems* (few subproblems)

  and

  optimal substructure (nice recurrence)

- **Idea:** LCS must look at all strings from a certain index until the end. So, try to build solution one index at a time!

Longest Common Subsequence (LCS)

- **Input:** Two strings $a_1 a_2 \cdots a_m$ and $b_1 b_2 \cdots b_n$, where $a_i, b_j \in \Sigma$
- **Output:** Largest k such that there is $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ for which $a_{i_\ell} = b_{j_\ell}$ for $\ell \in [k]$
- how to DP this?

Need to find many *overlapping subproblems* (few subproblems)
and
optimal substructure (nice recurrence)

- **Idea:** LCS must look at all strings from a certain index until the end. So, try to build solution one index at a time!
- $C(i, j)$ = length of LCS of $a_i \cdots a_m$ and $b_j \cdots b_n$

Longest Common Subsequence (LCS)

- **Input:** Two strings $a_1 a_2 \cdots a_m$ and $b_1 b_2 \cdots b_n$, where $a_i, b_j \in \Sigma$
- **Output:** Largest k such that there is $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ for which $a_{i_\ell} = b_{j_\ell}$ for $\ell \in [k]$
- how to DP this?

  Need to find many *overlapping subproblems* (few subproblems)
  and
  optimal substructure (nice recurrence)

- **Idea:** LCS must look at all strings from a certain index until the end. So, try to build solution one index at a time!
- $C(i, j)$ = length of LCS of $a_i \cdots a_m$ and $b_j \cdots b_n$
- want to find $C(1, 1)$

Longest Common Subsequence (LCS)

- **Input:** Two strings $a_1 a_2 \cdots a_m$ and $b_1 b_2 \cdots b_n$, where $a_i, b_j \in \Sigma$
- **Output:** Largest k such that there is $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ for which $a_{i_\ell} = b_{j_\ell}$ for $\ell \in [k]$
- how to DP this?

Need to find many *overlapping subproblems* (few subproblems)
and
optimal substructure (nice recurrence)

- **Idea:** LCS must look at all strings from a certain index until the end. So, try to build solution one index at a time!
- $C(i,j)$ = length of LCS of $a_i \cdots a_m$ and $b_j \cdots b_n$
- **Optimal Substructure:** if $A_i = a_i \cdots a_m$, $B_j = b_j \cdots b_n$ are the partial sequences and $\Gamma(i,j) = c_1 \cdots c_k$ is an LCS of A_i, B_j, then:
  1. $a_i = b_j \Rightarrow c_1 = a_i = b_j$ and $\Gamma(i+1, j+1)$ is LCS of A_{i+1}, B_{j+1}
  2. $a_i \neq b_j$ and $z_1 \neq a_i$ then $\Gamma(i,j)$ is LCS of A_{i+1}, B_j
  3. $a_i \neq b_j$ and $z_1 \neq b_j$ then $\Gamma(i,j)$ is LCS of A_i, B_{j+1}

Recursion for LCS

- **Optimal Substructure:** if $A_i = a_i \cdots a_m$, $B_j = b_j \cdots b_n$ are the partial sequences and $\Gamma(i,j) = c_1 \cdots c_k$ is an LCS of A_i, B_j, then:
  1. $a_i = b_j \Rightarrow c_1 = a_i = b_j$ and $\Gamma(i+1, j+1)$ is LCS of A_{i+1}, B_{j+1}
  2. $a_i \neq b_j$ and $z_1 \neq a_i$ then $\Gamma(i,j)$ is LCS of A_{i+1}, B_j
  3. $a_i \neq b_j$ and $z_1 \neq b_j$ then $\Gamma(i,j)$ is LCS of A_i, B_{j+1}
- Based on optimal substructure, we have:

$$C(i,j) = \begin{cases} 0, \text{ if } i > m \text{ or } j > n \\ C(i+1, j+1) + 1, \text{ if } a_i = b_j \text{ and } i, j \leq n \\ \max\{C(i+1, j), C(i, j+1)\}, \text{ if } a_i \neq b_j \text{ and } i, j \leq n \end{cases}$$

Recursion for LCS

- **Optimal Substructure:** if $A_i = a_i \cdots a_m$, $B_j = b_j \cdots b_n$ are the partial sequences and $\Gamma(i, j) = c_1 \cdots c_k$ is an LCS of A_i, B_j, then:
  1. $a_i = b_j \Rightarrow c_1 = a_i = b_j$ and $\Gamma(i+1, j+1)$ is LCS of A_{i+1}, B_{j+1}
  2. $a_i \neq b_j$ and $z_1 \neq a_i$ then $\Gamma(i, j)$ is LCS of A_{i+1}, B_j
  3. $a_i \neq b_j$ and $z_1 \neq b_j$ then $\Gamma(i, j)$ is LCS of A_i, B_{j+1}
- Based on optimal substructure, we have:

$$C(i,j) = \begin{cases} 0, \text{ if } i > m \text{ or } j > n \\ C(i+1, j+1) + 1, \text{ if } a_i = b_j \text{ and } i, j \leq n \\ \max\{C(i+1, j), C(i, j+1)\}, \text{ if } a_i \neq b_j \text{ and } i, j \leq n \end{cases}$$

- Have $m \cdot n$ subproblems, so bottom up implementation takes $O(mn)$ time.

Recursion for LCS

- **Optimal Substructure:** if $A_i = a_i \cdots a_m$, $B_j = b_j \cdots b_n$ are the partial sequences and $\Gamma(i,j) = c_1 \cdots c_k$ is an LCS of A_i, B_j, then:
  1. $a_i = b_j \Rightarrow c_1 = a_i = b_j$ and $\Gamma(i+1, j+1)$ is LCS of A_{i+1}, B_{j+1}
  2. $a_i \neq b_j$ and $z_1 \neq a_i$ then $\Gamma(i,j)$ is LCS of A_{i+1}, B_j
  3. $a_i \neq b_j$ and $z_1 \neq b_j$ then $\Gamma(i,j)$ is LCS of A_i, B_{j+1}
- Based on optimal substructure, we have:

$$C(i,j) = \begin{cases} 0, \text{ if } i > m \text{ or } j > n \\ C(i+1, j+1) + 1, \text{ if } a_i = b_j \text{ and } i, j \leq n \\ \max\{C(i+1, j), C(i, j+1)\}, \text{ if } a_i \neq b_j \text{ and } i, j \leq n \end{cases}$$

- Have $m \cdot n$ subproblems, so bottom up implementation takes $O(mn)$ time.
- Correctness of solution follows by the correctness of the recurrence.

- Longest Common Subsequence
- Minimum Length Triangulation
- Acknowledgements

Minimum Length Triangulation

- **Input:** n points $P_1, \ldots, P_n \in \mathbb{R}^2$ forming a convex n-gon Γ
- **Output:** a triangulation of Γ such that the perimeters of the $n-2$ triangles is minimized (output sum of perimeters)
- **Model:** unit cost model
- will assume we can compute distance between two points in $O(1)$ time.
- hence can assume we have a function Π which computes the perimeter of a triangle.

Minimum Length Triangulation

- **Input:** n points $P_1, \ldots, P_n \in \mathbb{R}^2$ forming a convex n-gon Γ
- **Output:** a triangulation of Γ such that the perimeters of the $n-2$ triangles is minimized (output sum of perimeters)
- Can we try all possibilities?

Number of triangulations is the $(n-2)^{nd}$ Catalan number:

$$\frac{1}{n-1} \cdot \binom{2n-4}{n-2} = \omega(2^n)$$

Minimum Length Triangulation

- **Input:** n points $P_1, \ldots, P_n \in \mathbb{R}^2$ forming a convex n-gon Γ
- **Output:** a triangulation of Γ such that the perimeters of the $n-2$ triangles is minimized (output sum of perimeters)
- Can we try all possibilities?

  Number of triangulations is the $(n-2)^{nd}$ Catalan number:

$$\frac{1}{n-1} \cdot \binom{2n-4}{n-2} = \omega(2^n)$$

- Can we DP it?

  Need to find *optimal substructure* first, and then look for *overlapping subproblems*.

Recurrence Relation

- **Idea:** which triangle will contain edge $P_n P_1$?
  If we choose index $2 \leq k \leq n-1$ for the third point of the triangle, we have the following perimeters:
  1. Triangle $P_n P_1 P_k$
  2. Polygon with vertices $P_1 P_2 \cdots P_{k-1}$
  3. Polygon with vertices $P_{k+1} \cdots P_n$

Recurrence Relation

- **Idea:** which triangle will contain edge P_nP_1?
  If we choose index $2 \leq k \leq n-1$ for the third point of the triangle, we have the following perimeters:
  1. Triangle $P_nP_1P_k$
  2. Polygon with vertices $P_1P_2 \cdots P_{k-1}$
  3. Polygon with vertices $P_{k+1} \cdots P_n$
- Leads to recurrence:

$$OPT(1,n) = \max_{2 \leq k \leq n-1} \{\Pi(P_1, P_k, P_n) + OPT(1, k-1) + OPT(k+1, n)\}$$

Recurrence Relation

- **Idea:** which triangle will contain edge $P_n P_1$?
  If we choose index $2 \leq k \leq n-1$ for the third point of the triangle, we have the following perimeters:
  1. Triangle $P_n P_1 P_k$
  2. Polygon with vertices $P_1 P_2 \cdots P_{k-1}$
  3. Polygon with vertices $P_{k+1} \cdots P_n$
- Leads to recurrence:

$$OPT(1, n) = \max_{2 \leq k \leq n-1} \{\Pi(P_1, P_k, P_n) + OPT(1, k-1) + OPT(k+1, n)\}$$

- Subproblems: any pair of indices $1 \leq i < j \leq n$ gives us an instance of the problem.

Thus $O(n^2)$ subproblems.

Recurrence Relation

- **Idea:** which triangle will contain edge $P_n P_1$?
  If we choose index $2 \leq k \leq n-1$ for the third point of the triangle, we have the following perimeters:
  1. Triangle $P_n P_1 P_k$
  2. Polygon with vertices $P_1 P_2 \cdots P_{k-1}$
  3. Polygon with vertices $P_{k+1} \cdots P_n$
- Leads to recurrence:

$$OPT(1, n) = \max_{2 \leq k \leq n-1} \{\Pi(P_1, P_k, P_n) + OPT(1, k-1) + OPT(k+1, n)\}$$

- Subproblems: any pair of indices $1 \leq i < j \leq n$ gives us an instance of the problem.

  Thus $O(n^2)$ subproblems.
- Bottom-up approach, takes $O(n)$ time to compute $OPT(i, j)$ if we know optimum for all subproblems. Hence, running time is $O(n^3)$.

Acknowledgement

- Based on [CLRS 2009, Chapter 15] and Prof Lau's notes

  `https://cs.uwaterloo.ca/~lapchi/cs341/notes/L12.pdf`
- Based on Prof. Brown's notes on minimum triangulation

References I

Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford. (2009)

Introduction to Algorithms, third edition.

MIT Press

Kleinberg, Jon and Tardos, Eva (2006)

Algorithm Design.

Addison Wesley