Lecture 26: Dynamic Programming III

2023/10/18

詹博华（中国科学院软件研究所）

Longest common subsequence

- Given a sequence, a subsequence is defined to be original sequence with some elements left out.
- For example: the sequence

$$\langle A, B, C, B, D, A, B \rangle$$

has subsequence

$$\langle B, C, D, B \rangle$$

since

$$\langle A, {\color{red}B}, {\color{red}C}, B, {\color{red}D}, A, {\color{red}B} \rangle$$

but $\langle B, C, A, D \rangle$ is not a subsequence.

Longest common subsequence

- Given two sequences X and Y, find the longest sequence Z such that Z is a subsequence of both X and Y.
- Applications in biology: given two DNA strands, the longest common subsequence gives an indication of their similarity.
- For example:

$$S_1 = \text{ACCGGTCGAGTGCGCGGAAGCCGGCCGAA}$$
$$S_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA}$$

what is the longest subsequence shared by the two strands?

Longest common subsequence

- Answer:

$$S_1 = \text{ACCGGTCGAGTGCGCGGAAGCCGGCCGAA}$$
$$S_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA}$$

$$LCS = \text{GTCGTCGGAAGCCGGCCGAA}$$

Naïve solutions

- Consider each subsequence of X (or Y)?
- If X has length n, then there are about 2^n subsequences to consider (each x_i may be included or not, there may be some repetitions).
- Already an exponential algorithm.

Solution by dynamic programming

- Original problem: let $X = x_1 x_2 \dots x_m$ and $Y = y_1 y_2 \dots y_n$.
- Q: What are the subproblems?
- A: For each $i \leq m$ and $j \leq n$, what is the longest common subsequence of $X_i = x_1 x_2 \dots x_i$ and $Y_j = y_1 y_2 \dots y_j$.
- Q: What is the recurrence relation?
- A: Consider different cases for the longest common subsequence of X_i and Y_j.

Case analysis

- For the longest common subsequence S_{ij} of $X_i = x_1x_2 \dots x_i$ and $Y_j = y_1y_2 \dots y_j$, there are three cases:
- **Case 1:** both x_i and y_j are part of the subsequence. This is possible only if $x_i = y_j$. Then S_{ij} is formed by appending x_i to the longest common subsequence of X_{i-1} and Y_{j-1} (that is, $S_{i-1,j-1}$). So the length of S_{ij} is $1 + \mathrm{len}(S_{i-1,j-1})$.
- **Case 2:** x_i is not part of the subsequence. Then S_{ij} is the longest common subsequence of X_{i-1} and Y_j, that is $S_{i-1,j}$.
- **Case 3:** y_j is not part of the subsequence. Then S_{ij} is the longest common subsequence of X_i and Y_{j-1}, that is $S_{i,j-1}$.

Recurrence relation

- Let $c[i,j]$ denote the length of the longest common subsequence of X_i and Y_j. Then it satisfies the recurrence:

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0\,, \\ c[i-1,j-1]+1 & \text{if } i,j > 0 \text{ and } x_i = y_j\,, \\ \max(c[i,j-1], c[i-1,j]) & \text{if } i,j > 0 \text{ and } x_i \neq y_j\,. \end{cases}$$

- **Note:** here we use the fact that if $x_i = y_j$, then it is always optimal to place x_i, y_j into the longest common subsequence (Why?)

Implementation

- Initialize table $c[i,j]$ for length of LCS, and $b[i,j]$ for the last step taken.
- $b[i,j] = \nwarrow$ if $x_i = y_j$ are in the LCS.
- $b[i,j] = \uparrow$ if x_i is excluded from LCS.
- $b[i,j] = \leftarrow$ if y_j is excluded from LCS.

LCS-LENGTH(X, Y)

```
m = X.length
n = Y.length
let b[1..m, 1..n] and c[0..m, 0..n] be new tables
for i = 1 to m
    c[i, 0] = 0
for j = 0 to n
    c[0, j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i, j] = c[i − 1, j − 1] + 1
            b[i, j] = "↖"
        elseif c[i − 1, j] ≥ c[i, j − 1]
            c[i, j] = c[i − 1, j]
            b[i, j] = "↑"
        else c[i, j] = c[i, j − 1]
            b[i, j] = "←"
return c and b
```

Example

- Compute the LCS of

$$X = \langle A, B, C, B, D, A, B \rangle$$

and

$$Y = \langle B, D, C, A, B, A \rangle$$

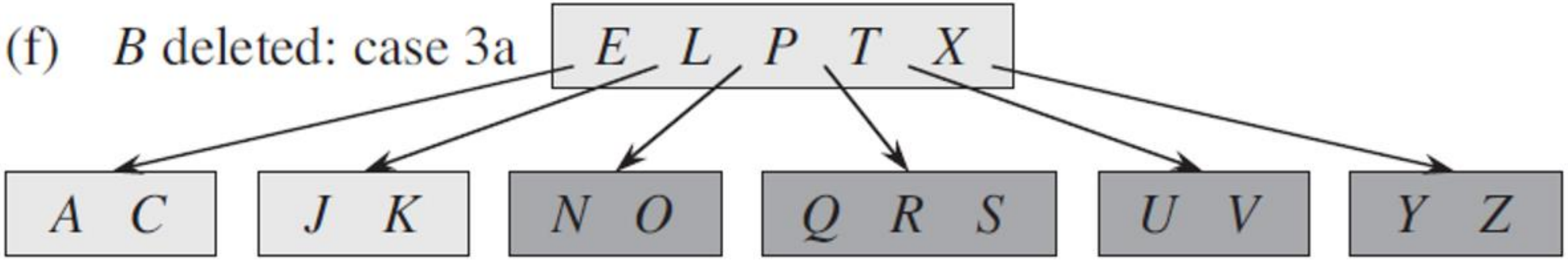
	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

Optimizations

1. It is possible to omit the table $b[i, j]$, and still be able to recover the path.
   - At each location (i, j) where $x_i \neq y_j$, simply compare $c[i-1, j]$ with $c[i, j-1]$, and pick the largest value to proceed.
2. Since the computation of each row depends only on values in the previous row, only two rows need to be stored at a time.

Exercise: deletion in B-trees

- Starting from the following tree, delete J, P, V in order.



Exercise: Longest increasing subsequence

- A sequence $x_1, x_2, \dots x_n$ is increasing if $x_1 < x_2 < \cdots < x_n$.
- Given a sequence X, find longest subsequence of X that is an increasing sequence.
- For example:

$$X = \langle 2, 10, 5, 7, 12, 8, 9 \rangle$$

the longest subsequence is

$$\langle 2, 5, 7, 8, 9 \rangle$$

- Naïve solution: consider all 2^n subsequences – exponential time.

Exercise: Longest increasing subsequence

- Find the longest increasing subsequence in $O(n^2)$ time.
- **Write down each of the following steps:**
  1. What are the subproblems of the main problem?
  2. What is the base case and the recurrence relation among subproblems?
  3. What is an appropriate order for solving the problems?
  4. Write down the pseudocode of the algorithm.
  5. Write an example of running the pseudocode.
- **Challenge:** there is actually an $O(n \log n)$ algorithm, see exercise 15.4-6 in textbook.