Lecture 06

Longest common subsequence problem

Problem Definition: We are given two strings: string *S* of length *n*, and string *T* of length *m*. Our goal is to produce their longest common subsequence: the longest sequence of characters that appear left-to-right (but not necessarily in a contiguous block) in both strings.

Example,

S = ABAZDC

T = BACBAD

– *Dynamic Programming*
1. Initialization
2. Matrix fill (scoring)
3. Traceback (alignment)

Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

Initialization

		B	A	C	B	A	D
		0	0	0	0	0	0
A	0						
B	0						
A	0						
Z	0						
D	0						
C	0						

Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

$M_{i,j}$ = MAXIMUM [$M_{i-1, j-1} + S$, $M_{i-1,j}$, $M_{i,j-1}$]

Initialization
Scoring

		B	A	C	B	A	D
		0	0	0	0	0	0
A	0	0	1	1	1	1	1
B	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Z	0	1	2	2	2	3	3
D	0	1	2	2	2	3	4
C	0	1	2	3	3	3	4

Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

$M_{i,j} = \text{MAXIMUM} [\, M_{i-1,\, j-1} + S \,,\, M_{i-1,j} \,,\, M_{i,j-1} \,]$

Solution???

		B	**A**	**C**	**B**	**A**	**D**
		0	0	0	0	0	0
A	0	0	1	1	1	1	1
B	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Z	0	1	2	2	2	3	3
D	0	1	2	2	2	3	↑4
C	0	1	2	3	3	3	4

Initialization
Scoring
Alignment

Multiple Possibilities

Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

$M_{i,j} = \text{MAXIMUM} [\, M_{i-1,\, j-1} + S \,,\, M_{i-1,j} \,,\, M_{i,j-1} \,]$

		B	**A**	**C**	**B**	**A**	**D**
		0	0	0	0	0	0
A	0	0	1	1	1	1	1
B	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Z	0	1	2	2	2	3	3
D	0	1	2	2	2	3	4
C	0	1	2	3	3	3	4

Implement

Longest common subsequence problem

Dynamic Programming

Example,

S = ABAZDC

T = BACBAD

Space complexity: $O(N^2)$
Time complexity: $O(N^2)$

		B	A	C	B	A	D
		0	0	0	0	0	0
A	0	0	1	1	1	1	1
B	0	1	1	1	2	2	2
A	0	1	2	2	2	3	3
Z	0	1	2	2	2	3	3
D	0	1	2	2	2	3	4
C	0	1	2	3	3	3	4

Sequence Alignment

- Pairwise
  - DOT matrix
  - Dynamic programming
  - Word method (efficient heuristic method; e.g., BLAST)
- Multiple
  - Dynamic programming
  - Progressive method (e.g., CLUSTAL, T-Coffee)
  - Iterative
  - Motif finding