Dynamic Programming II

Inge Li Gørtz

CLRS Chapter 25.0-25.2, KT section 6.6

Dynamic Programming

- Optimal substructure
- Last time:
  - Rod cutting
  - Longest common subsequence
- Today:
  - Sequence alignment
  - All pairs shortest paths

Sequence Alignment

Sequence alignment

- How similar are ACAAGTC and CATGT.
- Align them such that
  - all items occurs in at most one pair.
  - no crossing pairs.
- Cost of alignment
  - gap penalty δ
  - mismatch cost for each pair of letters $\alpha(p,q)$.
- Goal: find minimum cost alignment.

```
A C A A G T C
- C A T G T -
1 mismatch, 2 gaps
```

```
A C A A - G T C
- C A - T G T -
0 mismatches, 4 gaps
```

Sequence Alignment

- Subproblem property.

- $SA(X_i,Y_j)$ = min cost of aligning strings X[1…i] and Y[1…j].

- Case 1. Align x_i and y_j.
  - Pay mismatch cost for x_i and y_j + min cost of aligning X_{i-1} and Y_{j-1}.
- Case 2. Leave x_i unaligned.
  - Pay gap cost + min cost of aligning X_{i-1} and Y_j.
- Case 3. Leave y_j unaligned.
  - Pay gap cost + min cost of aligning X_i and Y_{j-1}.

Sequence alignment

$$SA(X_i,Y_j) = \begin{cases} j\delta & \text{if } i=0 \\ i\delta & \text{if } j=0 \\ \min\begin{cases}\alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}),\\ \delta+SA(X_i,Y_{j-1}),\\ \delta+SA(X_{i-1},Y_j)\}\end{cases} & \text{otherwise}\end{cases}$$

		A	C	A	A	G	T	C
C								
A								
T								
G								
T								

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j) = \begin{cases} j\delta & \text{if } i=0 \\ i\delta & \text{if } j=0 \\ \min\begin{cases}\alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}),\\ \delta+SA(X_i,Y_{j-1}),\\ \delta+SA(X_{i-1},Y_j)\}\end{cases} & \text{otherwise}\end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1							
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j) = \begin{cases} j\delta & \text{if } i=0 \\ i\delta & \text{if } j=0 \\ \min\begin{cases}\alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}),\\ \delta+SA(X_i,Y_{j-1}),\\ \delta+SA(X_{i-1},Y_j)\}\end{cases} & \text{otherwise}\end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1							
A	2							
T	3							
G	4							
T	5							

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1						
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1						
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1					
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1					
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2				
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2				
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3			
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4		
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4	5	6
A	2	1	2	1	2	3	4	5
T	3	2	3	2	3	3	3	4
G	4	3	4	3	4	3	4	5
T	5	4	5	4	5	4	3	4

$\delta = 1$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

- Use dynamic programming to compute an optimal alignment.
  - Time: $\Theta(mn)$
  - Space: $\Theta(mn)$
- Find actual alignment by backtracking (or saving information in another matrix).
- Linear space?
  - Easy to compute value (save last and current row)
  - How to compute alignment? Hirschberg. (not part of the curriculum).

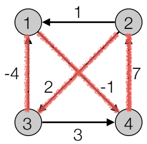
All-Pairs Shortest Paths

All Pairs Shortest Paths

- Applications
  - Distance tables in an atlas
  - Fastest routing in a network
  - Shortest wiring on a circuit board

All-Pairs Shortest Paths

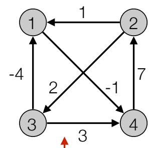
- All-Pairs Shortest Path Problem (APSP)
  - Given directed weighted graph G=(V,E).
  - Weights of edges w_{ij} are real numbers (might be negative).
  - Let n = |V| and m= |E|.
  - Weight of a path is the sum of the weights on its edges.
  - Goal: Compute the shortest path for every pair of vertices.
  - Output: An n x n matrix D = D = (d_{ij}) of shortest path distances.
- Example



	1	2	3	4
1	0	6	8	-1
2	-2	0	2	-3
3	-4	2	0	-5
4	5	7	9	0

All-Pairs Shortest Paths

- All-Pairs Shortest Path Problem (APSP)
  - Given directed weighted graph G=(V,E).
  - Weights of edges w_{ij} are real numbers (might be negative).
  - Let n = |V| and m= |E|.
  - Weight of a path is the sum of the weights on its edges.
  - Goal: Compute the shortest path for every pair of vertices.
  - Output: An n x n matrix D = D = (d_{ij}) of shortest path distances.
- Example

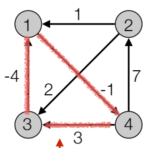


	1	2	3	4
1	0	6	8	-1
2	-2	0	2	-3
3	-4	2	0	-5
4	5	7	9	0

What happens if we change direction on this edge?

All-Pairs Shortest Paths

- All-Pairs Shortest Path Problem (APSP)
  - Given directed weighted graph G=(V,E).
  - Weights of edges w_{ij} are real numbers (might be negative).
  - Let n = |V| and m= |E|.
  - Weight of a path is the sum of the weights on its edges.
  - Goal: Compute the shortest path for every pair of vertices.
  - Output: An n x n matrix D = D = (d_{ij}) of shortest path distances.
- Example



	1	2	3	4
1	0	6	8	-1
2	-2	0	2	-3
3	-4	2	0	-5
4	5	7	9	0

What happens if we change direction on this edge?

Will assume no negative cycles

All-Pairs Shortest Paths

- Assume G given as adjacency matrix of weights, with vertices numbered 1 to n.

$$w_{ij} = \begin{cases} 0 & \text{if } i = j \\ \text{weight of } (i,j) & \text{if } i \neq j, (i,j) \in E \\ \infty & \text{if } i \neq j, (i,j) \notin E \end{cases}$$

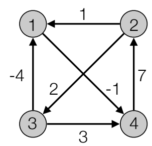
- Dynamic programming.
- Optimal substructure: Subpaths of shortest paths are shortest paths
- Let $l_{ij}^{(m)}$ be the weight of the shortest path from i to j that contains *at most* m edges.
- m = 0: there is a shortest path from i to j with at most 0 edges iff i = j:

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j, (i,j) \end{cases}$$

- m ≥ 1:

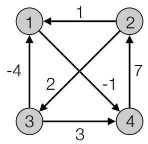
shortest i → k path using at most m-1 edges

shortest i → j path using at most m-1 edges



All Pairs Shortests Paths

$$l_{ij}^{(m)} = \min_{1 \le k \le n} \{ l_{ik}^{(m-1)} + w_{kj} \}$$



$$l_{12}^{(1)} = \min\{0+\infty, \infty+0, \infty+\infty, 7+(-1)\}$$

$L^{(1)}$

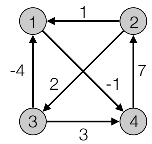
	1	2	3	4
1	0	∞	∞	-1
2	1	0	2	∞
3	-4	∞	0	3
4	∞	7	∞	0

$L^{(2)}$

	1	2	3	4
1	0	6	∞	-1
2	-2	0	2	0
3				
4				

All Pairs Shortests Paths

$$l_{ij}^{(m)} = \min_{1 \le k \le n} \{ l_{ik}^{(m-1)} + w_{kj} \}$$



$L^{(1)}$

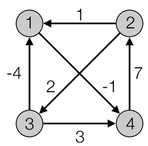
	1	2	3	4
1	0	∞	∞	-1
2	1	0	2	∞
3	-4	∞	0	3
4	∞	7	∞	0

$L^{(2)}$

	1	2	3	4
1	0	6	∞	-1
2	-2	0	2	0
3	-4	10	0	-5
4	8	7	9	0

All Pairs Shortests Paths

$$l_{ij}^{(m)} = \min_{1 \le k \le n} \{ l_{ik}^{(m-1)} + w_{kj} \}$$



$L^{(1)}$

	1	2	3	4
1	0	∞	∞	-1
2	1	0	2	∞
3	-4	∞	0	3
4	∞	7	∞	0

$L^{(2)}$

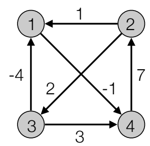
	1	2	3	4
1	0	6	∞	-1
2	-2	0	2	0
3	-4	10	0	-5
4	8	7	9	0

$L^{(3)}$

	1	2	3	4
1	0	6	8	-1
2	-2	0	2	-3
3	-4	2	0	-5
4	5	7	9	0

All-Pairs Shortest Paths

- Running time and space:
  - Space: Θ(n²)
  - Time: Θ(n⁴)



EXTEND(L, W, n)
let $L' = (l'_{ij})$ be a new $n \times n$ matrix
for $i = 1$ **to** n
  for $j = 1$ **to** n
    $l'_{ij} = \infty$
    for $k = 1$ **to** n
      $l'_{ij} = \min(l'_{ij}, l_{ik} + w_{kj})$
return L'

SLOW-APSP(W, n)
$L^{(1)} = W$
for $m = 2$ **to** $n - 1$
  let $L^{(m)}$ be a new $n \times n$ matrix
  $L^{(m)} = $ EXTEND$(L^{(m-1)}, W, n)$
return $L^{(n-1)}$

All-Pairs Shortest Paths and matrix multiplication

- Don't need to compute all n matrices.
- Compute $L^{(1)}$, $L^{(2)}$, $L^{(4)}$, $L^{(8)}$,…, ,$L^{(m)}$ where m smallest power of 2 greater than n-1.
- Know $L^{(m)} = L^{(n-1)}$ for all $m \geq n$.
- Running time: $\Theta(n^3 \log n)$
- Similar to matrix multiplication: Replace "+" with "x" and "min" with "+".

EXTEND(L, W, n)
let $L' = (l'_{ij})$ be a new $n \times n$ matrix
for $i = 1$ **to** n
    for $j = 1$ **to** n
        $l'_{ij} = \infty$
        for $k = 1$ **to** n
            $l'_{ij} = \min(l'_{ij}, l_{ik} + w_{kj})$
return L'

FASTER-APSP(W, n)
$L^{(1)} = W$
$m = 1$
while $m < n - 1$
    let $L^{(2m)}$ be a new $n \times n$ matrix
    $L^{(2m)} = $ EXTEND$(L^{(m)}, L^{(m)}, n)$
    $m = 2m$
return $L^{(m)}$