CSE530 Algorithms & Complexity
Lecture 3: Dynamic Programming

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

March 7, 2018

Course Organization

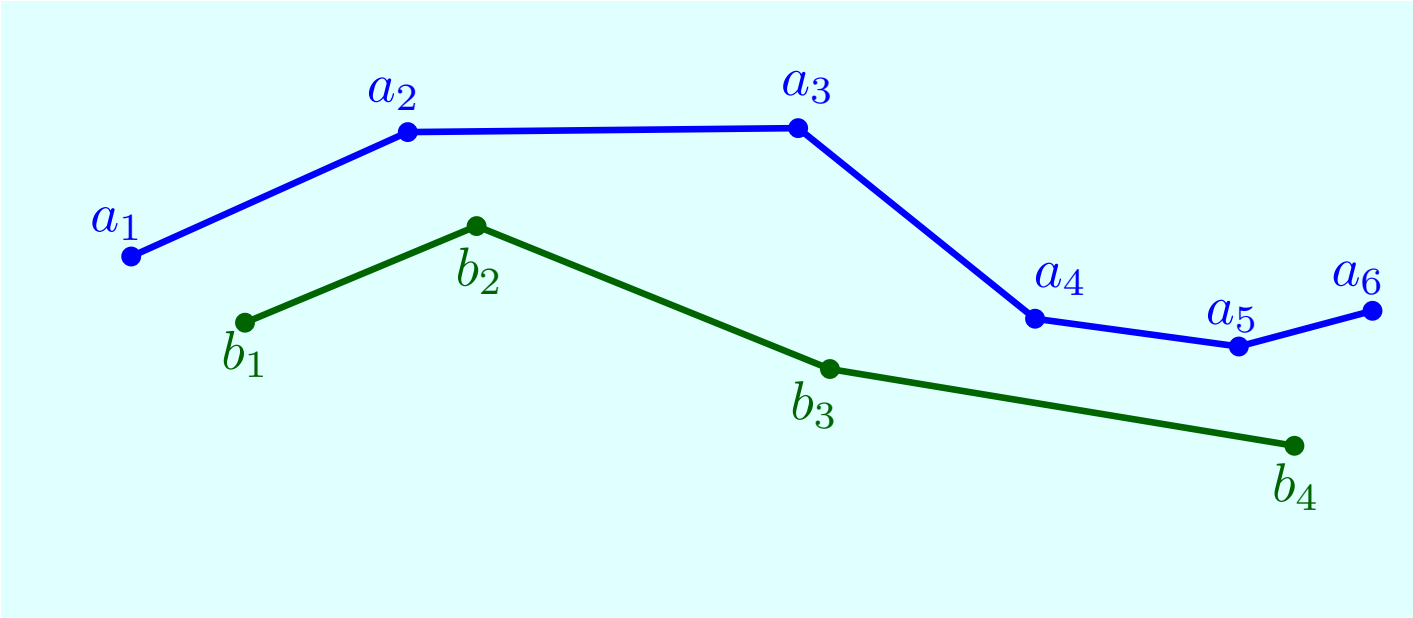
- I will post Assignment 1 on blackboard on Thursday.
- It is due on March 19 (Mon), at the beginning of the lecture.
- Please check the slides on academic integrity at the end of Lecture 0.

- Solutions to exercise set 1 have been posted
- Reminder: Attendance is counted in this course (see Course Information). Please use the electronic attendance system. In particular, try to use the mobile phone app as the old system is no longer maintained.

Introduction

- *Dynamic programming* is an important algorithms design technique, that often yields polynomial-time algorithms.
- It is one of the first techniques to try when you face a nontrivial algorithmic problem.
- This lecture is a review of dynamic programming through two examples:
  - *Dynamic Time Warping* (DTW): a similarity measure for time series.
  - A histogram construction problem.

Time Series



- We are given two sequences of points $A = (a_1, a_2, \ldots, a_m)$ and $B = (b_1, b_2, \ldots, b_n)$.
- Example: two sequences of points in $\mathbb{R}^2$.
- Point sequences are called *time series* in statistics.

Metric Spaces

- Sequences $A = (a_1, a_2, \ldots, a_m)$ and $B = (b_1, b_2, \ldots, b_n)$ come from a *metric space* (M, d), and we know the distance $d(a_i, b_j)$ for all i, j.

Metric Spaces

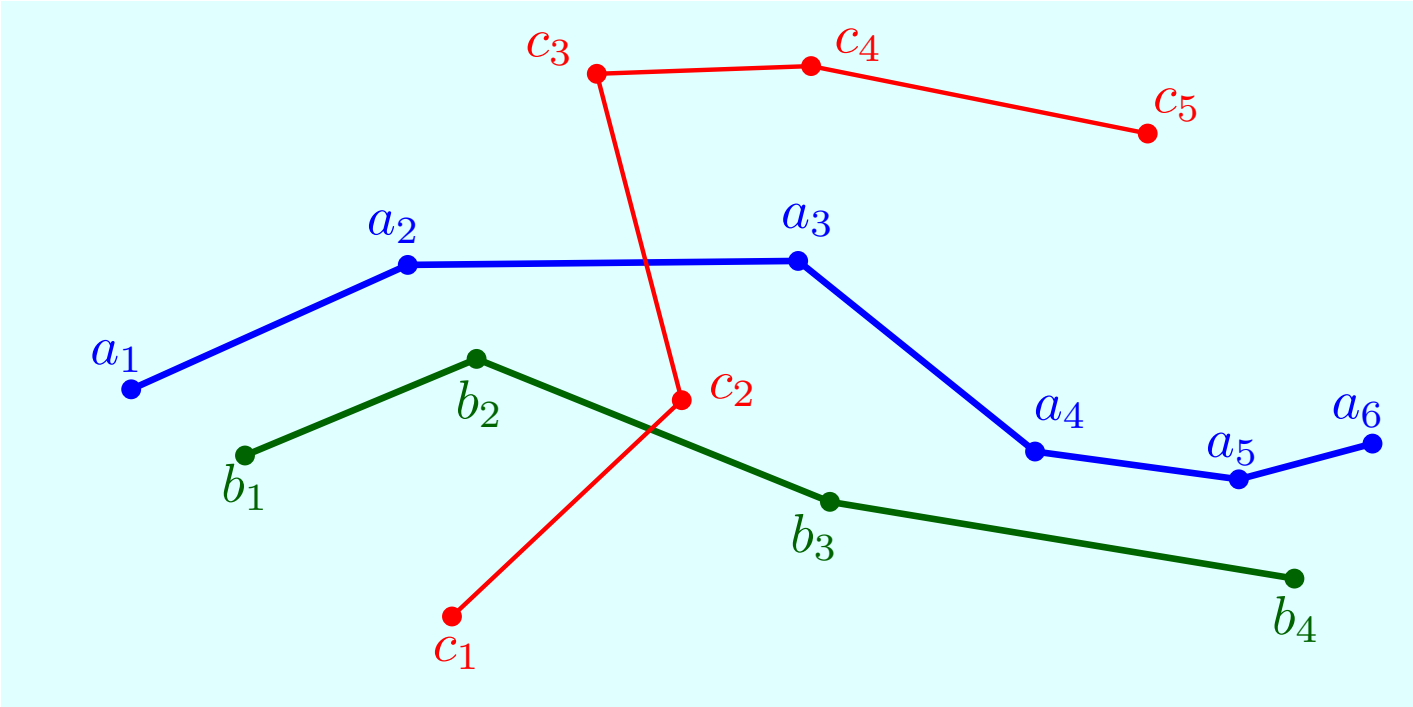
Let M be a set and $d : M \times M \to \mathbb{R}$. We say that (M, d) is a *metric space* if $\forall p, q, r \in M$

- $d(p, q) \geqslant 0$
- $d(p, q) = 0 \;\Leftrightarrow\; p = q$
- $d(p, q) = d(q, p)$ (symmetry)
- $d(p, r) \leqslant d(p, q) + d(q, r)$ (triangle inequality)

- Example of metric space: $\mathbb{R}^d$ with the Euclidean distance.

Similarity Measures for Point Sequences

- Problem: How can we measure similarity between A and B?
- Idea: Find a *similarity measure* $S(\cdot, \cdot) \geqslant 0$ such that $S(A, B)$ is small when A and B are similar.



- In the example above, we would like to have $S(A, B) \leqslant S(A, C)$.

The Euclidean Distance

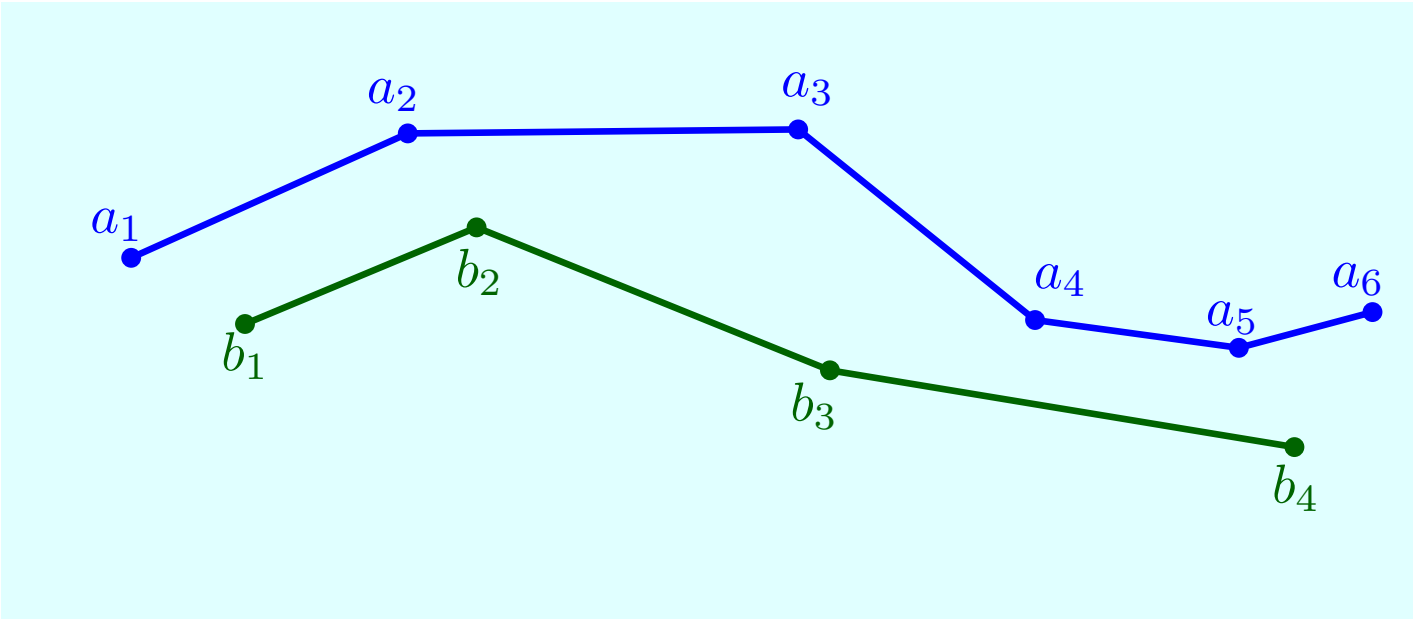
- First approach: The *Euclidean distance*

$$E(A, B) = \sqrt{\sum_{i=1}^{n} d(a_i, b_i)^2}$$

  - This is a metric.
  - Problem: Requires $m = n$.

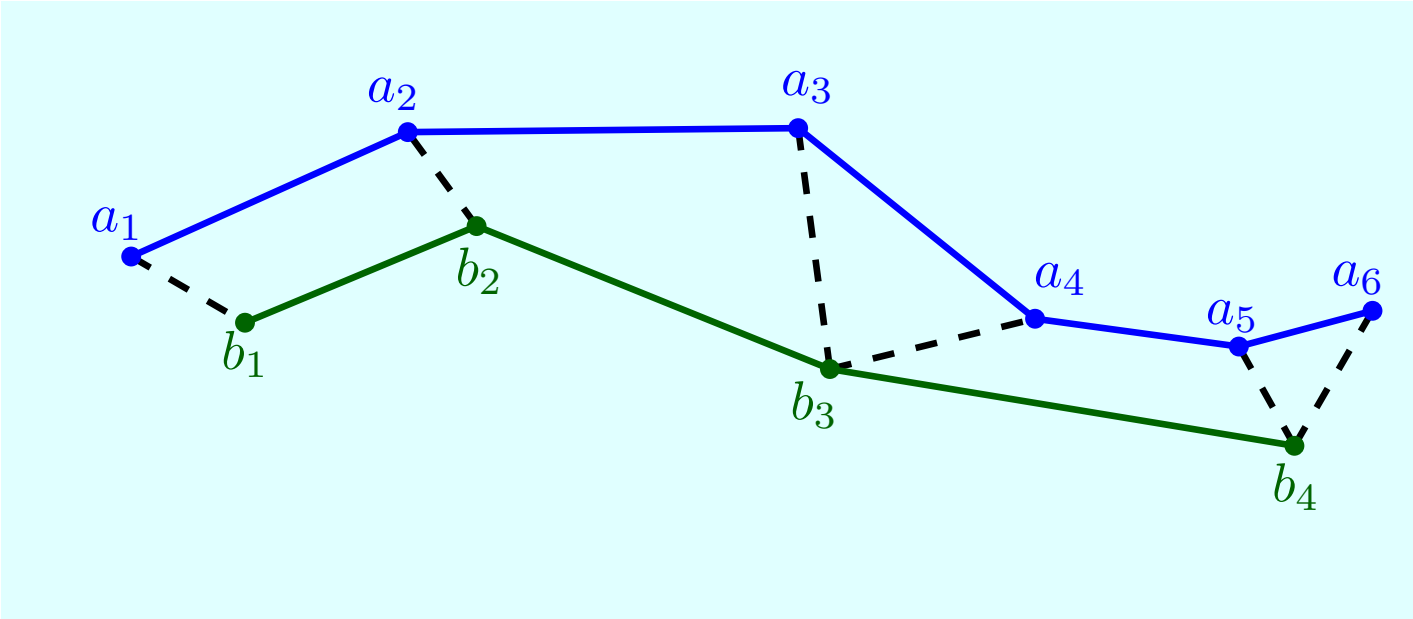
Dynamic Time Warping

- *Dynamic Time Warping (DTW)* allows to measure similarity between two sequences A and B when their lengths m and n differ.



Dynamic Time Warping

- Idea: We find a *coupling* of A and B, and sum up the distances between the pairs in this coupling.



- $d(a_1, b_1) + d(a_2, b_2) + d(a_3, b_3) + d(a_4, b_3) + d(a_5, b_4) + d(a_6, b_4)$.

Sequence Coupling

Definition (Coupling)

An (m, n)-coupling is a sequence (α_k, β_k), $k = 1, \ldots, \ell$ such that:

- $\alpha_1 = \beta_1 = 1$.
- $\alpha_\ell = m$
- $\beta_\ell = n$
- For all $k = 2, \ldots, \ell$,

$$(\alpha_k, \beta_k) = \begin{cases} (\alpha_{k-1} + 1, \beta_{k-1} + 1), \\ (\alpha_{k-1}, \beta_{k-1} + 1), \text{ or} \\ (\alpha_{k-1} + 1, \beta_{k-1}). \end{cases}$$

Dynamic Time Warping

Definition (Dynamic Time Warping)

For any two point sequences $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$, $\mathrm{DTW}(A, B)$ is the minimum over all (m, n)-couplings (α, β) of $\sum_{k=1}^{\ell} d(a_{\alpha_k}, b_{\beta_k})$.

- It satisfies the recurrence relation

$$\mathrm{DTW}(A_i, B_j) = d(a_i, b_j) + \min \begin{pmatrix} \mathrm{DTW}(A_{i-1}, B_{j-1}), \\ \mathrm{DTW}(A_{i-1}, B_j), \\ \mathrm{DTW}(A_i, B_{j-1}) \end{pmatrix}$$

where $A_i = (a_1, \ldots, a_i)$, $B_j = (b_1, \ldots, b_j)$ and $i, j > 1$.

First Algorithm

Algorithm 1: Recursive computation of DTW

1: **function** $\mathrm{DTW}(A_i, B_j)$
2: **if** $i = 1$ and $j = 1$ **then**
3: **return** $d(a_1, b_1)$ ▷ base case
4: **if** $j = 1$ **then**
5: **return** $d(a_i, b_j) + \mathrm{DTW}(A_{i-1}, B_j)$
6: **if** $i = 1$ **then**
7: **return** $d(a_i, b_j) + \mathrm{DTW}(A_i, B_{j-1})$
8: **return** $d(a_i, b_j) +$
9: $\min(\mathrm{DTW}(A_{i-1}, B_{j-1}), \mathrm{DTW}(A_{i-1}, B_j), \mathrm{DTW}(A_i, B_{j-1}))$

Analysis

- Problem: Algorithm 1 runs in *exponential time*.
- More precisely:

Proposition

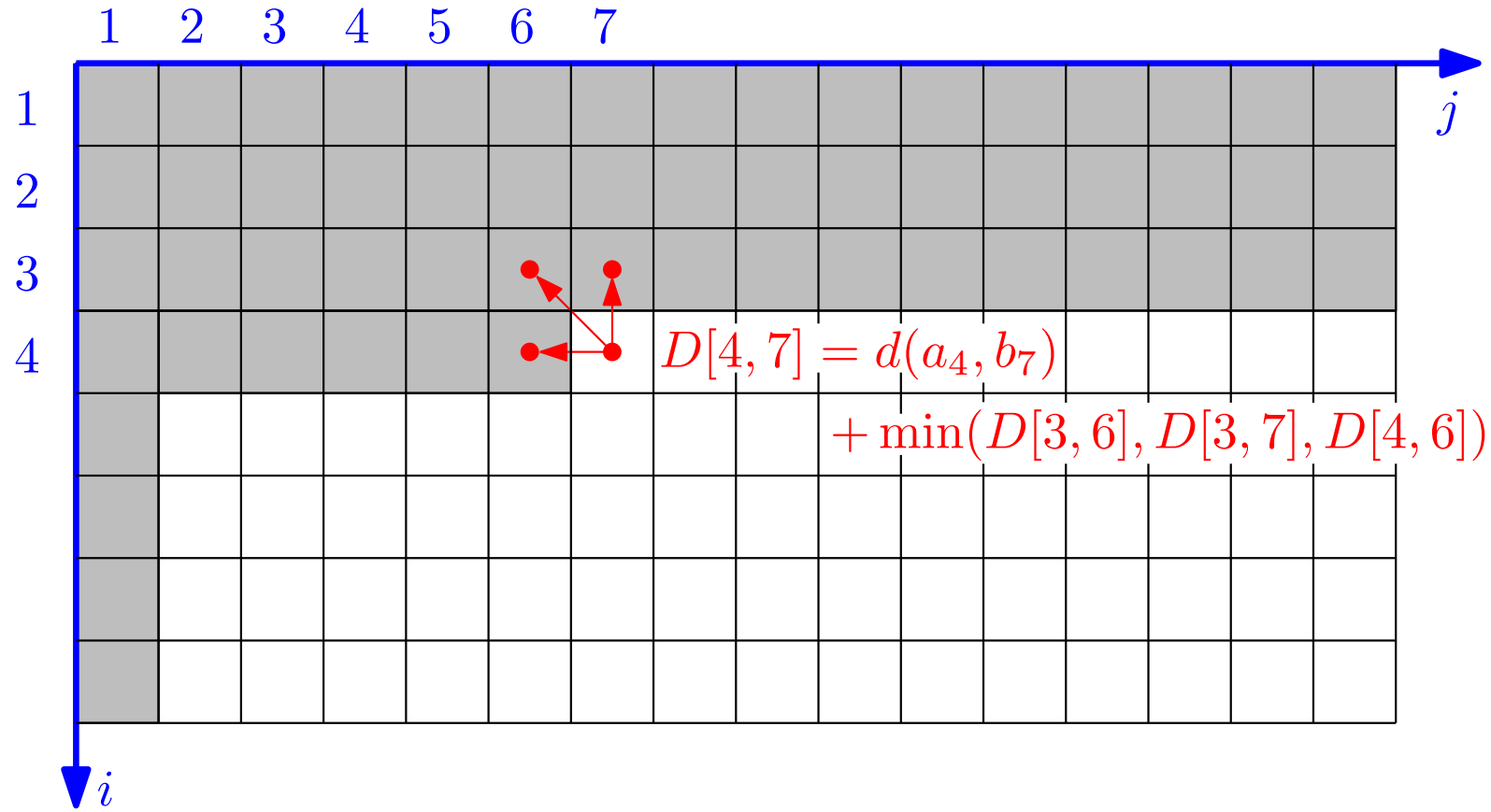
Let $T(m, n)$ be the running time of Algorithm 1 on sequences of lengths m and n. Then $T(n, n) = \Omega(3^n)$.

Proof.

Done in class. □

Better Algorithm

- Idea: Fill a table $D[i, j] = \mathrm{DTW}(A_i, B_j)$ in a bottom-up manner.



Better Algorithm

Algorithm 2

1: **function** $\mathrm{DTW}((a_1, \ldots, a_m), (b_1, \ldots, b_n))$
2: $\quad D \leftarrow$ new $m \times n$ array
3: $\quad D[1,1] \leftarrow d(a_1, b_1)$
4: $\quad$ **for** $i \leftarrow 2, m$ **do**
5: $\quad\quad D[i,1] \leftarrow D[i-1,1] + d(a_i, b_1)$
6: $\quad$ **for** $j \leftarrow 2, n$ **do**
7: $\quad\quad D[1,j] \leftarrow D[1,j-1] + d(a_1, b_j)$
8: $\quad$ **for** $i \leftarrow 2, m$ **do**
9: $\quad\quad$ **for** $j \leftarrow 2, n$ **do**
10: $\quad\quad\quad D[i,j] \leftarrow d(a_i, b_j) + \min(D[i-1,j], D[i-1,j-1], D[i,j-1])$
11: $\quad$ **return** $D[m,n]$

Analysis

Algorithm 2

```
1:  function DTW((a_1, ..., a_m), (b_1, ..., b_n))
2:      D ← new m × n array
3:      D[1,1] ← d(a_1, b_1)
4:      for i ← 2, m do
5:          D[i,1] ← D[i−1,1] + d(a_i, b_1)
6:      for j ← 2, n do
7:          D[1,j] ← D[1,j−1] + d(a_1, b_j)
8:      for i ← 2, m do
9:          for j ← 2, n do
10:             D[i,j] ← d(a_i, b_j) + min(D[i−1,j],
                                           D[i−1,j−1], D[i,j−1])
11:     return D[m,n]
```

l.	cost	times
2	$O(mn)$	1
3	$\Theta(1)$	1
4	$\Theta(1)$	m
5	$\Theta(1)$	$m-1$
6	$\Theta(1)$	n
7	$\Theta(1)$	$n-1$
8	$\Theta(1)$	m
9	$\Theta(1)$	$(m-1)n$
10	$\Theta(1)$	$(m-1)(n-1)$
11	$\Theta(1)$	1

Proposition

Algorithm 2 runs in $\Theta(mn)$ time.

Dynamic Programming

- This is an example of *dynamic programming*.

Dynamic Programming

Dynamic programming consists in:

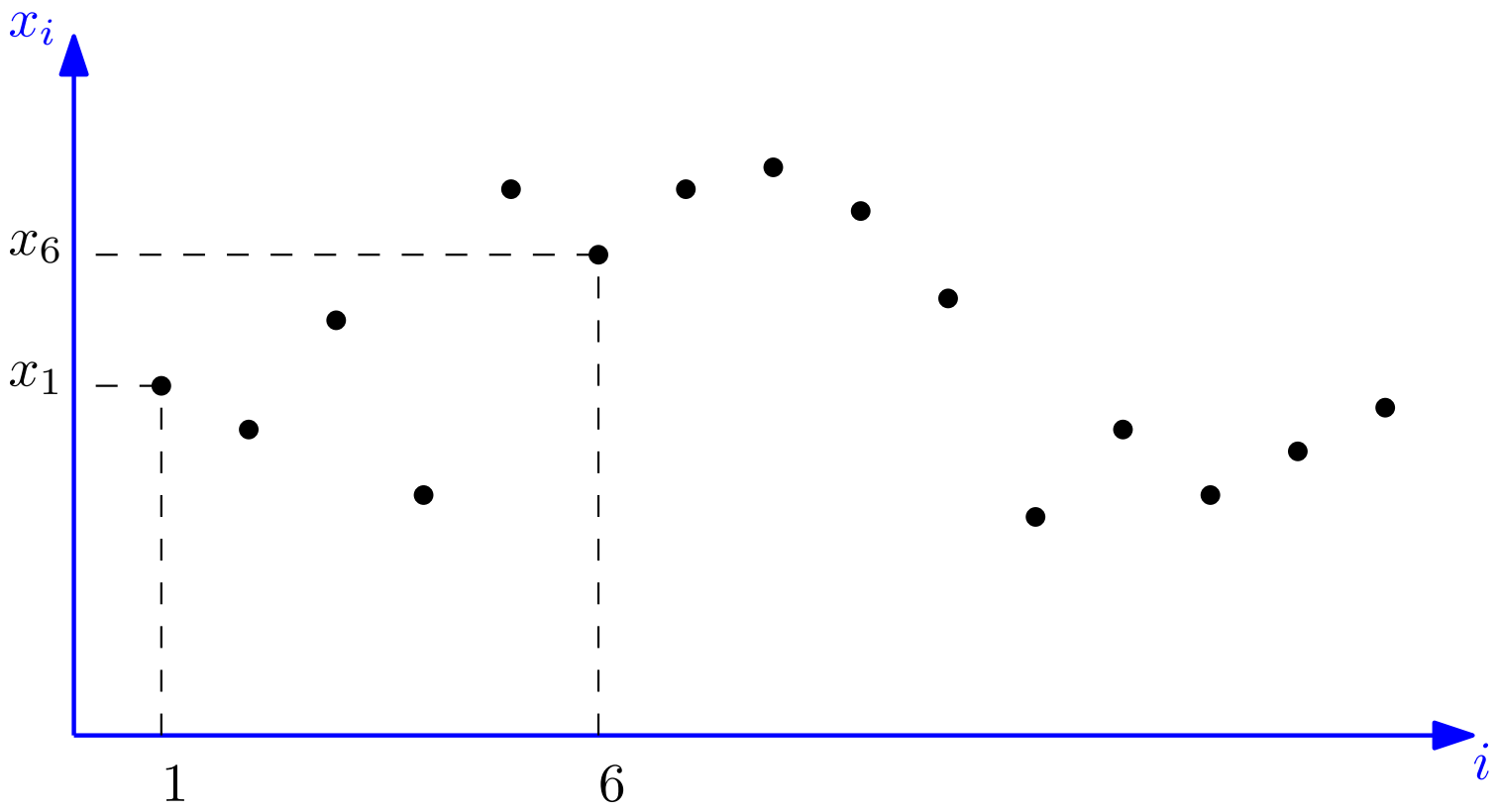
- Breaking the problem into subproblems.
- Solving each subproblem just *once*, and *storing* the result.

- Algorithm 1, on the other hand, solved some subproblems an exponential number of times.

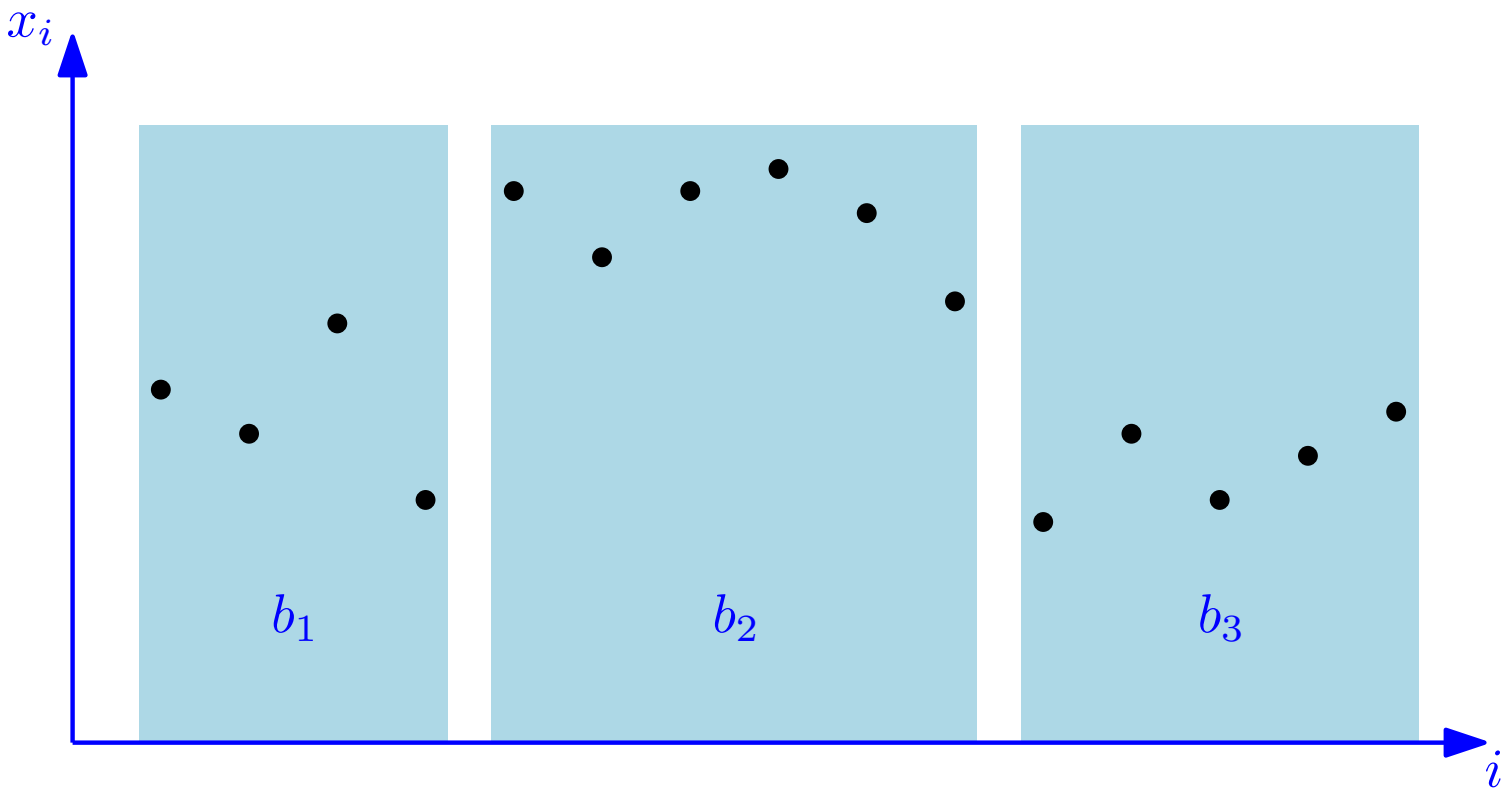
Dynamic Time Warping: Concluding Remarks

- Often used in practice.
- Example: similarity of GPS traces.
- DTW is *not* a metric: does not satisfy triangle inequality
- Complexity:
  - We presented an $O(mn)$ time algorithm.
  - No better algorithm is known.
  - Recent work suggests that it may not be possible to do better.

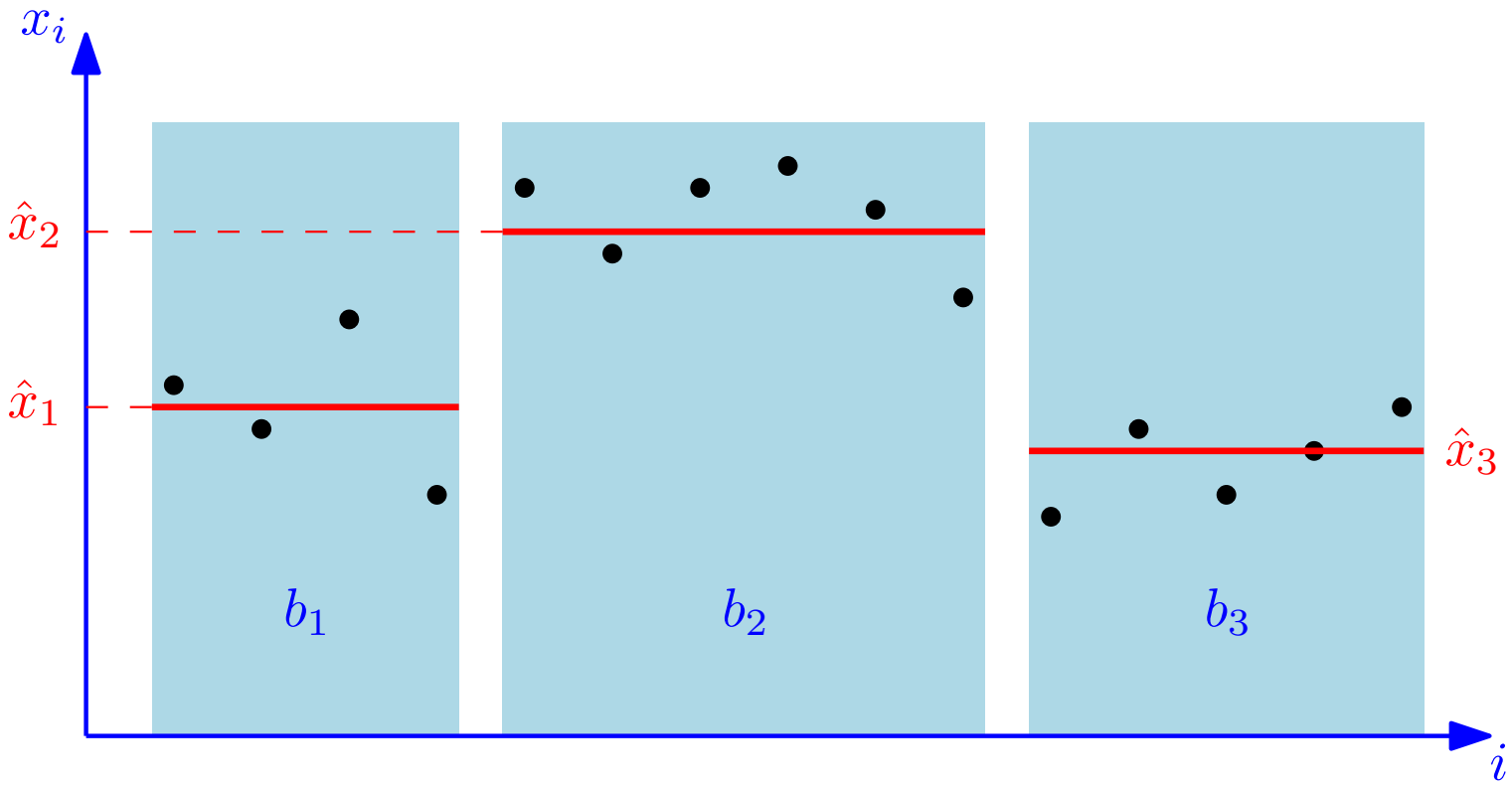
Histogram Construction



Histogram Construction



Histogram Construction



Histogram Construction

- INPUT: integer m, sequence $x_1, \ldots, x_n$
- We will partition it into m *buckets* $b_1, \ldots, b_m$ of consecutive indices such that:
- $\forall p,\ b_p = \{s_p, s_p + 1, \ldots, e_p\}$,
- $\forall p < m,\ e_p = s_{p+1} - 1$
- $s_1 = 1,\ e_m = n$
- **Example**: In previous slide, $s_1 = 1,\ e_1 = 4,\ s_2 = 5,\ e_2 = 10,\ s_3 = 11,\ e_3 = 14$
- Each bucket b_p is represented by a value $\hat{x}_p$.
- The goal is to minimize $\sum_{p=1}^{m} f(s_p, e_p, \hat{x}_p)$ where

$$f(s_p, e_p, \hat{x}_p) = \sum_{i \in b_p} (\hat{x}_p - x_i)^2$$

over all choices of b_p and $\hat{x}_p$, $1 \leqslant p \leqslant m$.

Motivation

- Databases:
  - Summarizing a large dataset using a small histogram that fits in RAM
  - Answering approximate queries
  - Query optimization
- Computer graphics:
  - Curve simplification

Computing x_p^*

Lemma

Let $|b_p|$ denote the number of elements in b_p, and thus $|b_p| = |e_p - s_p + 1|$. Then the optimal value of $\hat{x}_p$ for this bucket is

$$x_p^* = \frac{1}{|b_p|} \sum_{i \in b_p} x_i.$$

Proof.

Done in class. □

Corollary

We denote $f^*(s_p, e_p) = f(s_p, e_p, x_p^*)$. Then we can compute x_p^* and $f^*(s_p, e_p)$ in $O(|b_p|)$ time.

Brute Force Approach

- Brute force:
  - Check all possible ways of partitioning into m buckets.
  - For each bucket b_p, compute x_p^* and $f^*(s_p, e_p)$.

Proposition

- *There are* $\binom{n-1}{m-1}$ *ways of choosing the* m *buckets.*
- $\binom{n}{m} = \Theta(n^m)$ *when* m *is a fixed constant.*
- $\binom{n}{\lfloor n/2 \rfloor} = \Omega(2^{n/2})$

- Conclusion:
  - Running time $O(n\binom{n-1}{m-1})$.
  - For arbitrary m, this is exponential.
  - Only doable for small values of m such as $m = 3$ or 4.

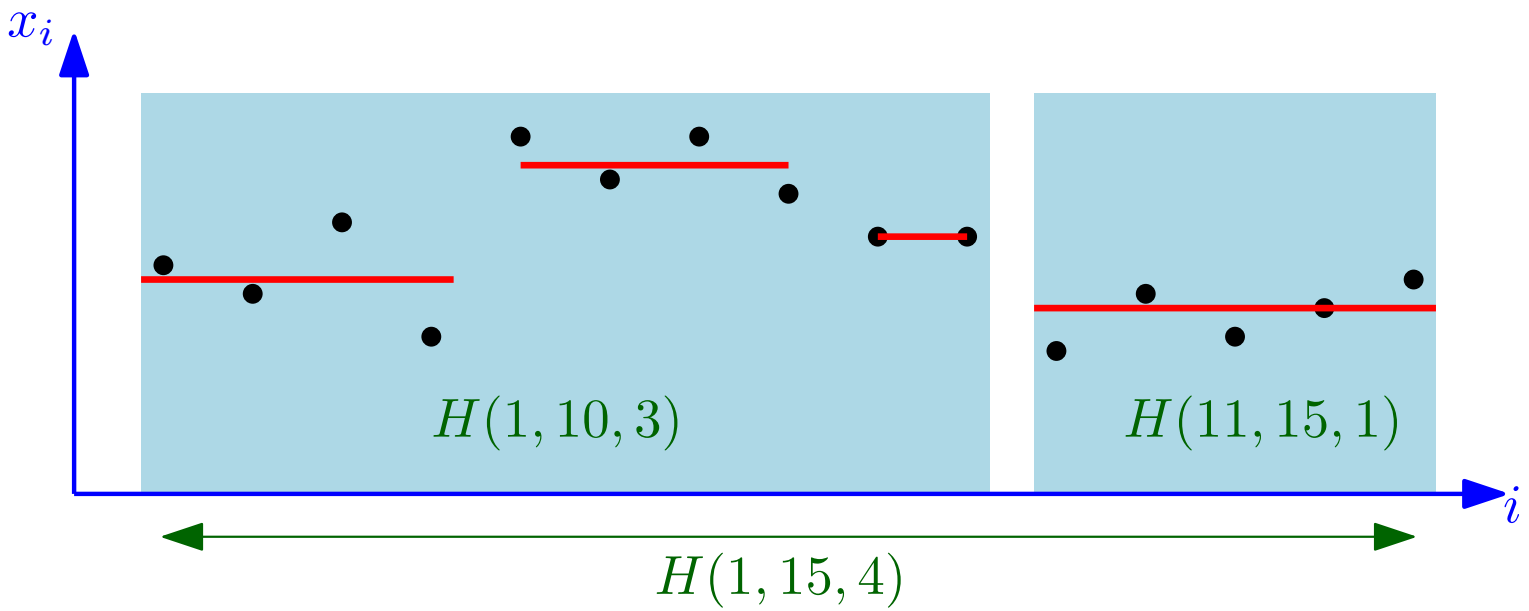
Dynamic Programming Approach

Subproblems

Let $H(i, j, k)$ be the value of an optimal k-buckets histograms for the subsequence $(x_i, x_{i+1}, \ldots, x_j)$.

- Recurrence relation: $\forall 2 \leqslant k \leqslant i \leqslant n$

$$H(1, i, k) = \min_{k-1 \leqslant j \leqslant i-1} \Big(H(1, j, k-1) + H(j+1, i, 1) \Big)$$



Pseudocode

Algorithm 3

1: **function** HISTOGRAM$(n, x_1, \ldots, x_n, m)$
2: **if** $m \geqslant n$ **then**
3: **return** 0
4: $H \leftarrow$ new $n \times n \times m$ array
5: **for** $i \leftarrow 1, n$ **do**
6: **for** $j \leftarrow i, n$ **do**
7: $H[i, j, 1] \leftarrow f^*(i, j)$
8: **for** $k = 2, \ldots, m$ **do**
9: **for** $i \leftarrow k, n$ **do**
10: $H[1, i, k] \leftarrow \min_{k-1 \leqslant j \leqslant i-1} \Big(H[1, j, k-1] + H[j+1, i, 1] \Big)$
11: **return** $H[1, n, m]$

Analysis

Proposition

Algorithm 3 computes the cost of an optimal histogram in $O(n^3)$ time.

Proof.

Line 7 takes $O(n)$ time, so the nested loops 5–7 take $O(n^3)$.
Line 10 takes $O(n)$ time, so the nested loops 8–10 take $O(mn^2)$.
We may assume that $m < n$, as otherwise the solution is trivial. □

- There are at least 3 issues with Algorithm 3. Can you see them?

Concluding Remarks

- The running time and space usage of Algorithm 3 can be improved to $O(mn^2)$ and $O(mn)$, respectively. See lecture notes.
- No better algorithm is currently known.
- If we change the objective function, then better algorithms may exist.
- For instance, if one wants to minimize the maximum error, in other words, we want to minimize

$$\max_{p \in \{1,\ldots,m\}} \max_{i \in b_p} |\hat{x}_p - x_i|$$

then an $O(n)$-time algorithm is known.
(Fitting a Step Function to a Point Set. H. Fournier and A. Vigneron, Algorithmica 60: 95-109, 2011.)