Sequences and Time Series

Introduction to Sequences

K. Selçuk Candan, Professor of Computer Science and Engineering

CASCADE
ASU Center for Assured and SCAlable Data Engineering

Ira A. Fulton Schools of Engineering
Arizona State University

Data is central to smart services

Human-centered IT
Context-aware, location-based services
Accessible work environments
Educational networks

Smart People

Large data in the cloud
Real-time metro-area data analysis
Privacy and access control
Context-aware data warehouses

Smart Governance

Context-aware, location-based services
Navigational guidance
Ambient intelligence and RFID
Real-time sensor data analysis

Smart Living

Smart Environment

Context-aware, location-based services
Real-time sensor data analysis
Sustainability
Resilient

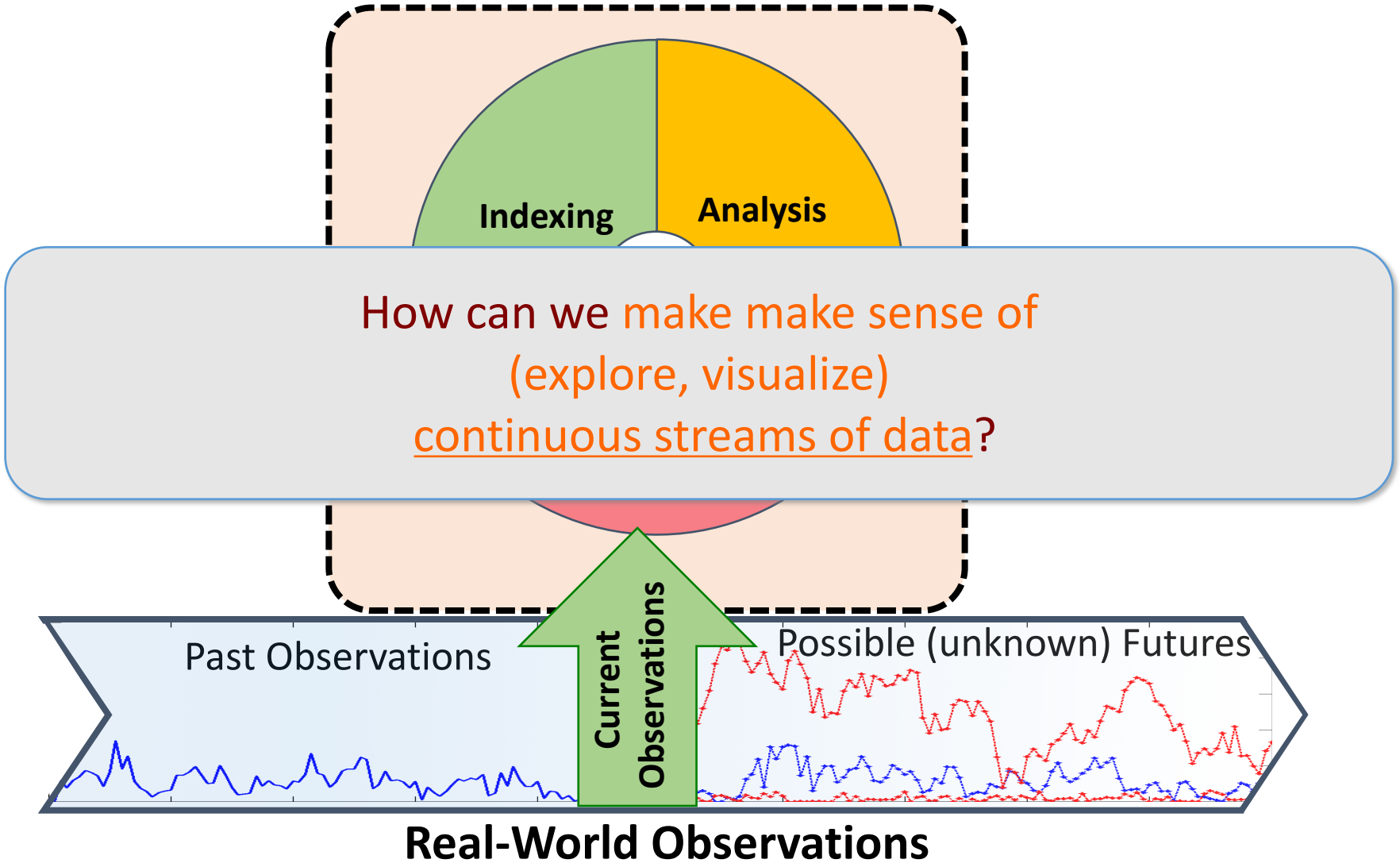
Smart Mobility

Real-time sensor data analysis
Large data in the cloud
Real-time metro-area data analysis
Social networks and evolving communities

Smart Economy

Customer Engagement
Social networks and evolving communities
Recommendation and collaborative filtering
Smart supply-chains

In many applications data is temporal



Common data representations

- Relational/ Object Oriented data
- Vector Space (spatial or high-dimensional) data
- **Strings, sequences, and time series data**
- Trees and graphs
- Fuzzy and probabilistic data

Strings, sequences, time series

- A *string* or *sequence*, $S = (c_1, c_2, ..., c_N)$, *is a finite sequence of symbols.*

abcbbbaabbaabcbbbaaabbc

- A *time series*, $T = (d_1, d_2, ..., d_N)$, *is a finite sequence of data values.*

https://trends.google.com/trends/explore?date=2008-12-19%202018-01-19&q=big%20data

Strings, sequences, time series

- A *string* or *sequence*, $S = (c_1, c_2, ..., c_N)$, *is a finite sequence of symbols.*

abcbbbaabbaabcbbbaaabbc

- A *time series*, $T = (d_1, d_2, ..., d_N)$, *is a finite sequence of data values.*

https://trends.google.com/trends/explore?date=2008-12-19%202018-01-19&q=big%20data

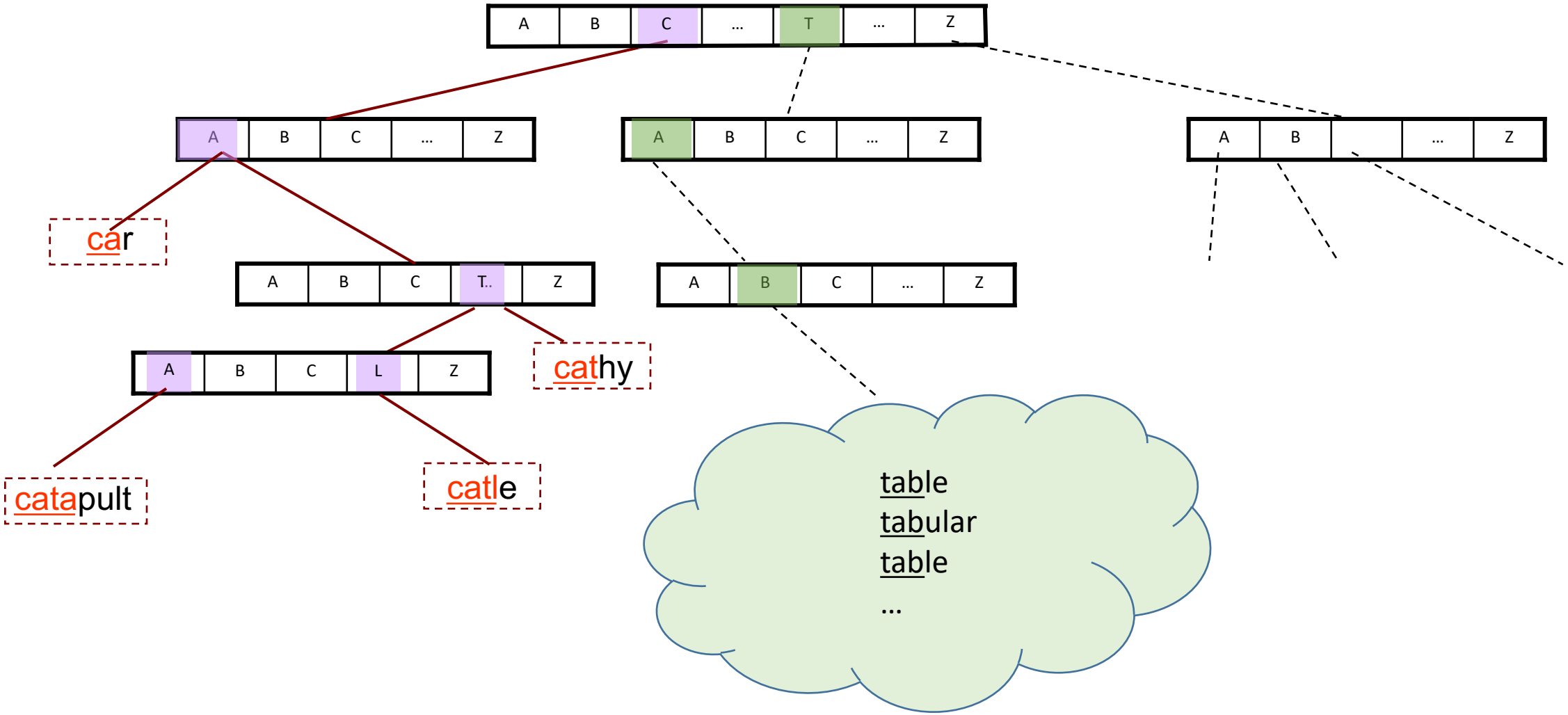
String/sequence matching and search

- Prefix search:
  - Find all strings that start with “tab”:
    - “table”; “tabular”; “tablet”; ….
- Subsequence search:
  - Find all strings that contain the subsequence “ark”:
    - “marketing”; “spark”; “quark”
  - Find all occurrences of “acd”:
    - “aabacdcdabdcababdacddcab.”
- Sequence similarity:
  - “table” vs. “cable”?
  - “table” vs. “tale”?
  - “table” vs. “tackle”?

String/sequence matching and search

- Prefix search:
  - Find all strings that start with “tab”:
    - “table”; “tabular”; “tablet”; ….
- Subsequence search:
  - Find all strings that contain the subsequence “ark”:
    - “marketing”; “spark”; “quark”
  - Find all occurrences of “acd”:
    - “aabacdcdabdcababdacddcab.”
- Sequence similarity:
  - “table” vs. “cable”?
  - “table” vs. “tale”?
  - “table” vs. “tackle”?

Trie data structure



String/sequence matching and search

- Prefix search:
  - Find all strings that start with “tab”:
    - “table”; “tabular”; “tablet”; ….

- Subsequence search:
  - Find all strings that contain the subsequence “ark”:
    - “marketing”; “spark”; “quark”
  - Find all occurrences of “acd”:
    - “aabacdcdabdcababdacddcab.”

- Sequence similarity:
  - “table” vs. “cable”?
  - “table” vs. “tale”?
  - “table” vs. “tackle”?

Subsequence/Pattern Search

data: abcbbbaabbaabcbbbaaabbccbbbaabbaacbbbaabbccbcbbbaabcbbbaabab

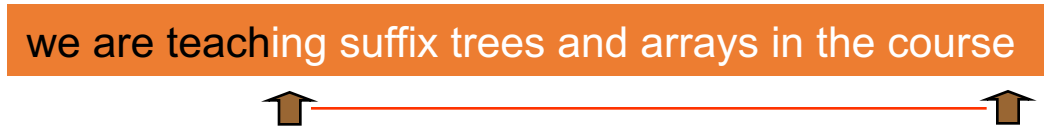
pattern: abbcc

- Brute force approach:
  - scan the sequence, while aligning the pattern for each position in the sequence
- Given a sequence of length N, and pattern of length M
  - Cost: O(NxM)
  - For the above example, cost: 60 x 5

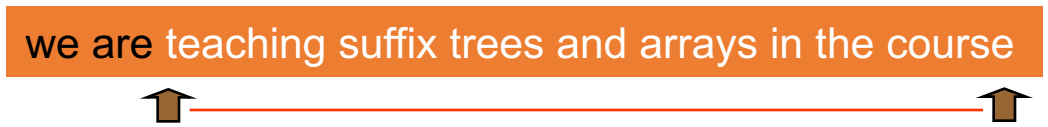
Suffix Trees and Arrays

- Tries work well if we search for a prefix

- Suffix trees and suffix arrays
  - Input text: a single long string
  - each position in the text gives a suffix

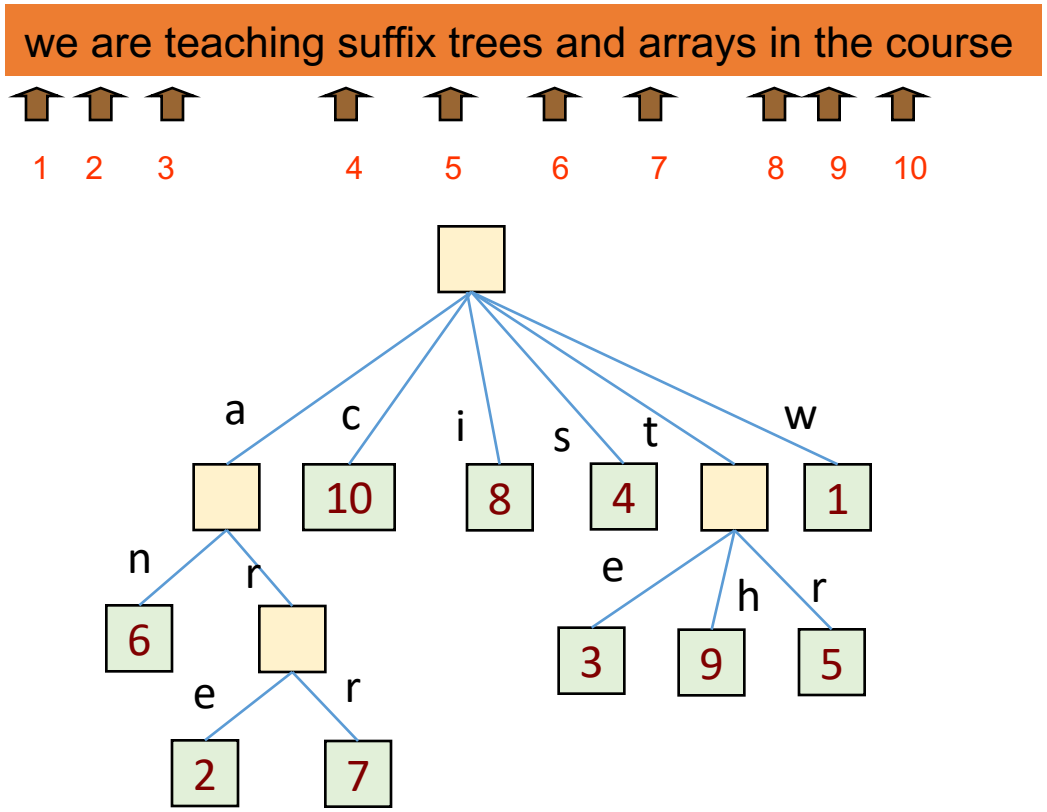


  - alternatively, start of each word in the text gives a suffix



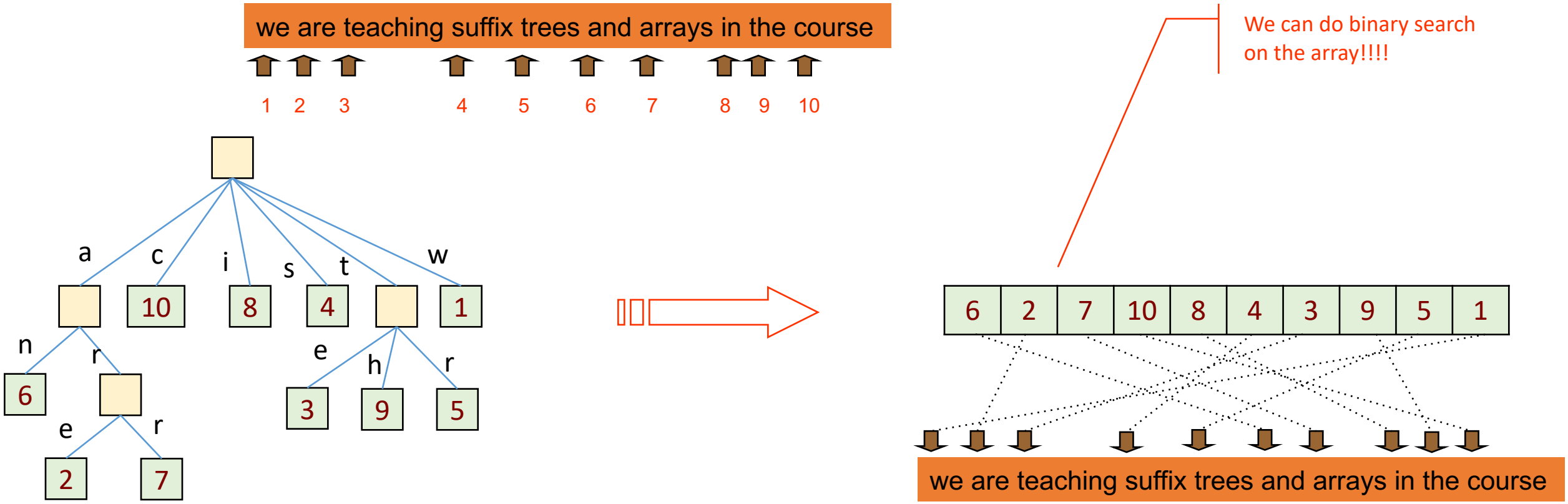
Suffix Trees

- Suffix trees
  - Input text: a single long string
  - each word position in the text gives a suffix



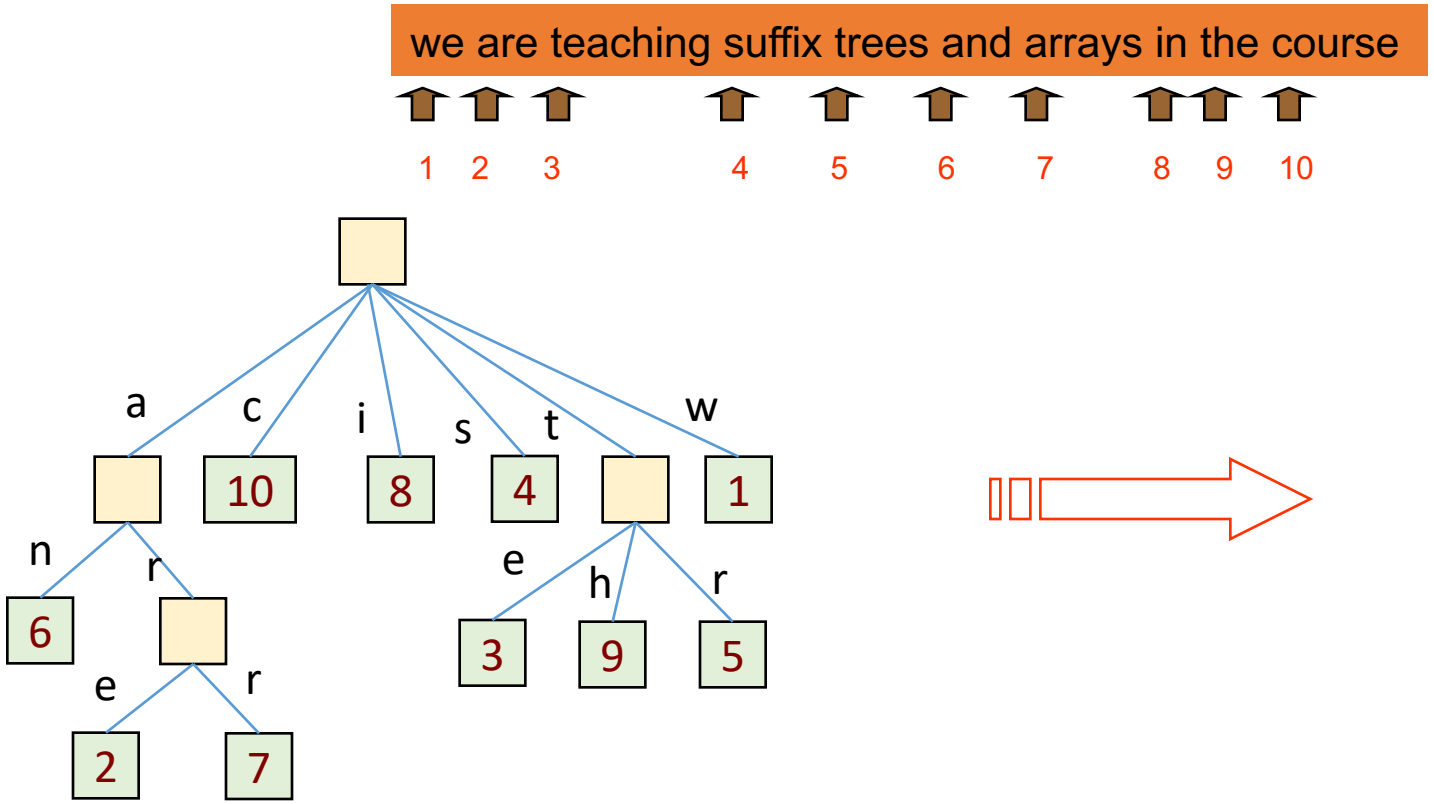
Suffix Arrays

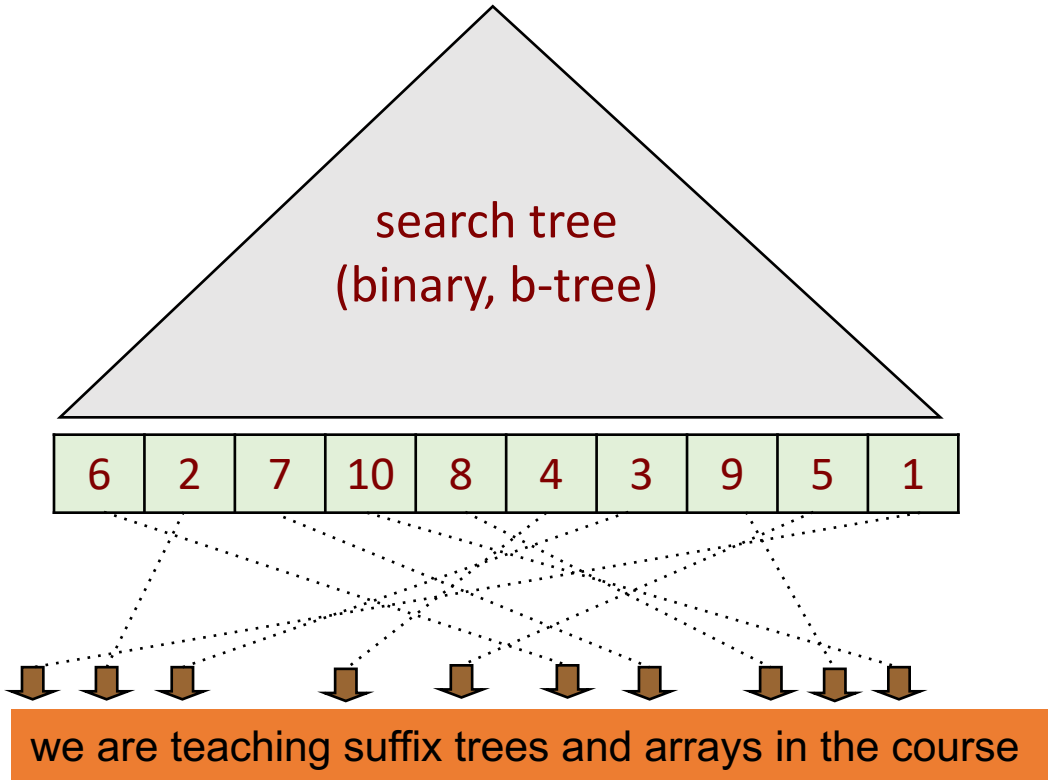
- Suffix arrays
  - Input text: a single long string
  - each word position in the text gives a suffix



Suffix Arrays

- Suffix arrays
  - Input text: a single long string
  - each word position in the text gives a suffix



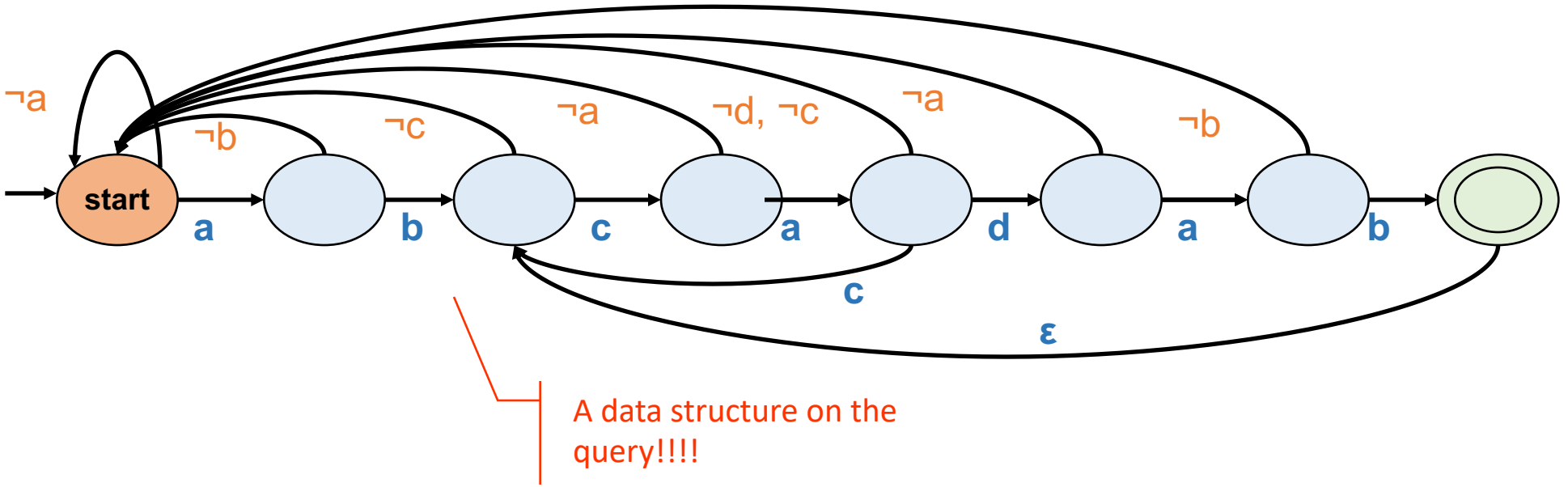


Subsequence/Pattern Search

- What if we are not given the sequence in advance; can we do search without a data structure on the sequence?
  - Yes, scan the sequence…
  - ..but, we have seen that this is expensive
    - Given a sequence of length N, and pattern of length M
    - Cost: O(NxM)
- If we are given the pattern in advance, can we create a data structure on the pattern, instead?

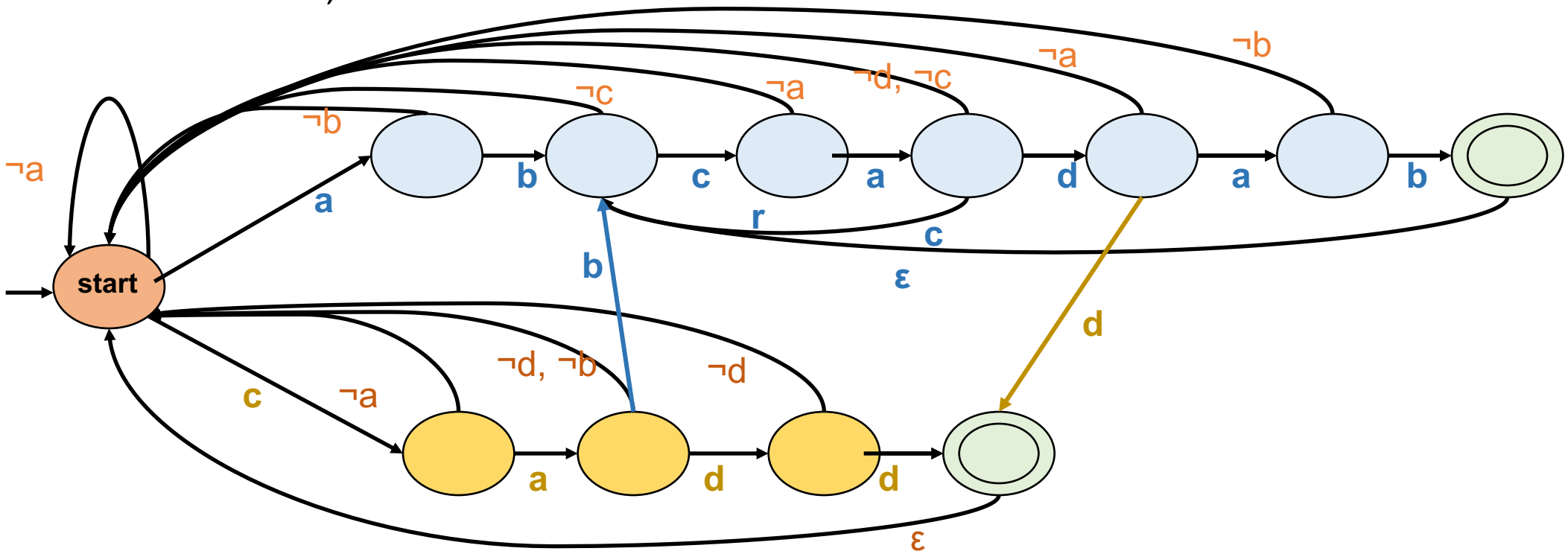
Knuth-Morris-Pratt (KMP)

- Given a sequence of length N, and pattern of length M
- Knuth-Morris-Pratt: O(N)
- Example
  - Pattern: abcadab



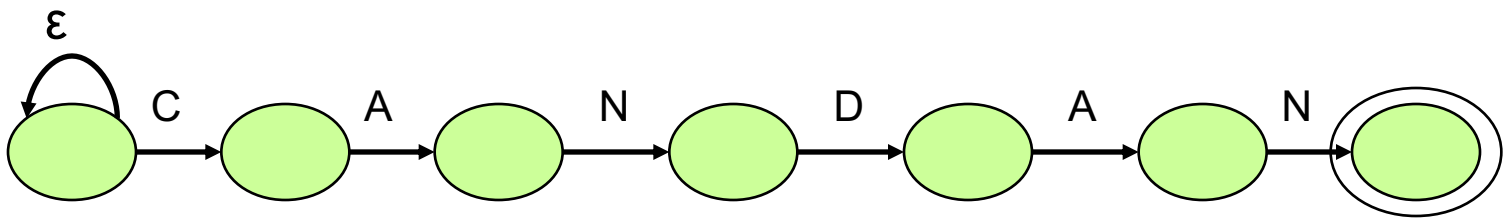
Aho-Corasick Trie

- What is we are given multiple patterns to search simultaneously
- Given a sequence of length N, and patterns of length M_1 and M_2
- Aho-Corasick Trie: O(N)
- Example
  - Patterns: abcadab; cadd

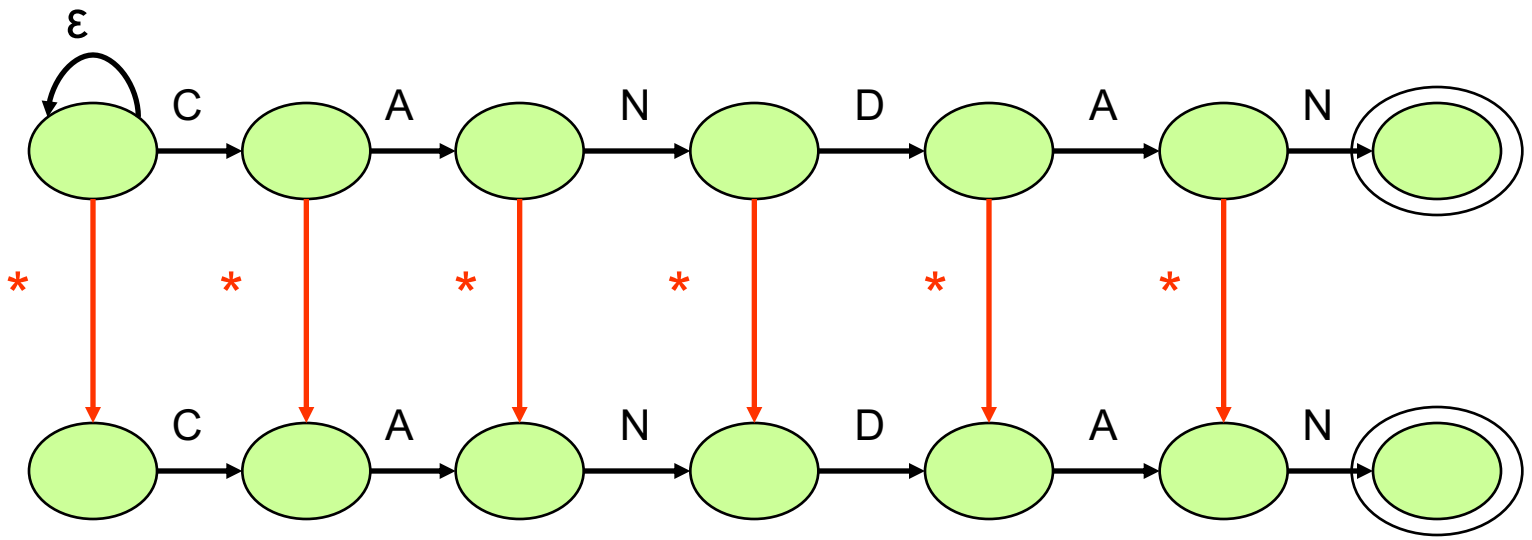


What about approximate matches?

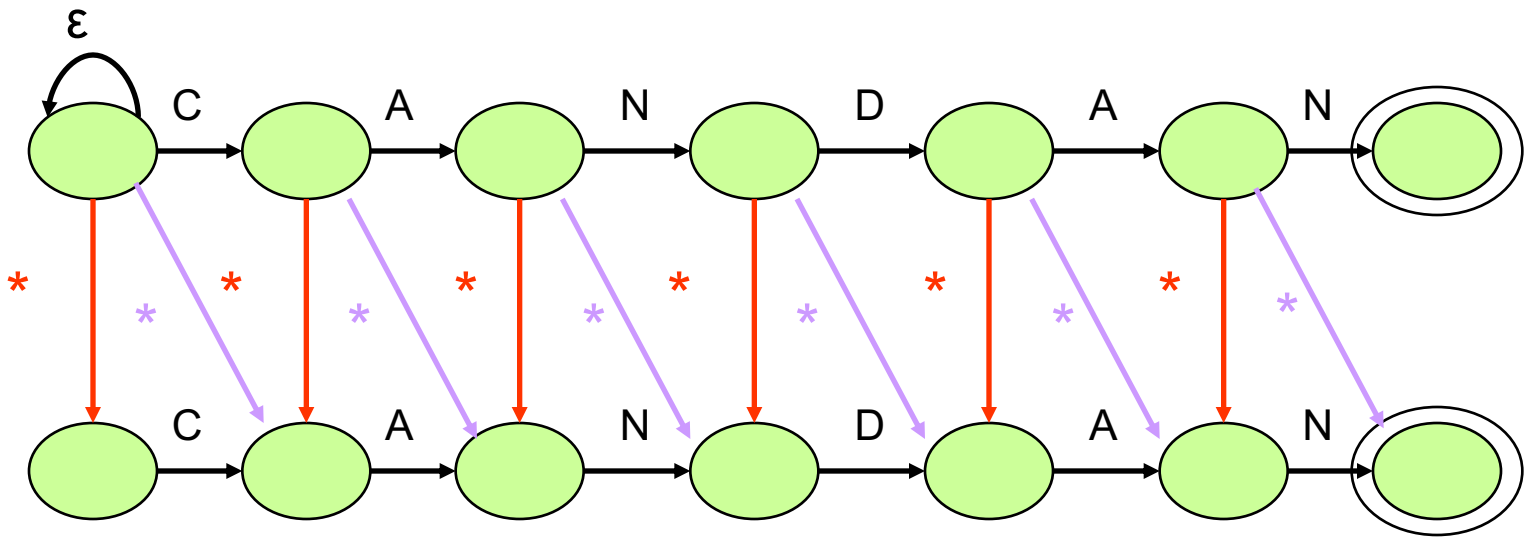
- Example
  - Pattern: CANDAN



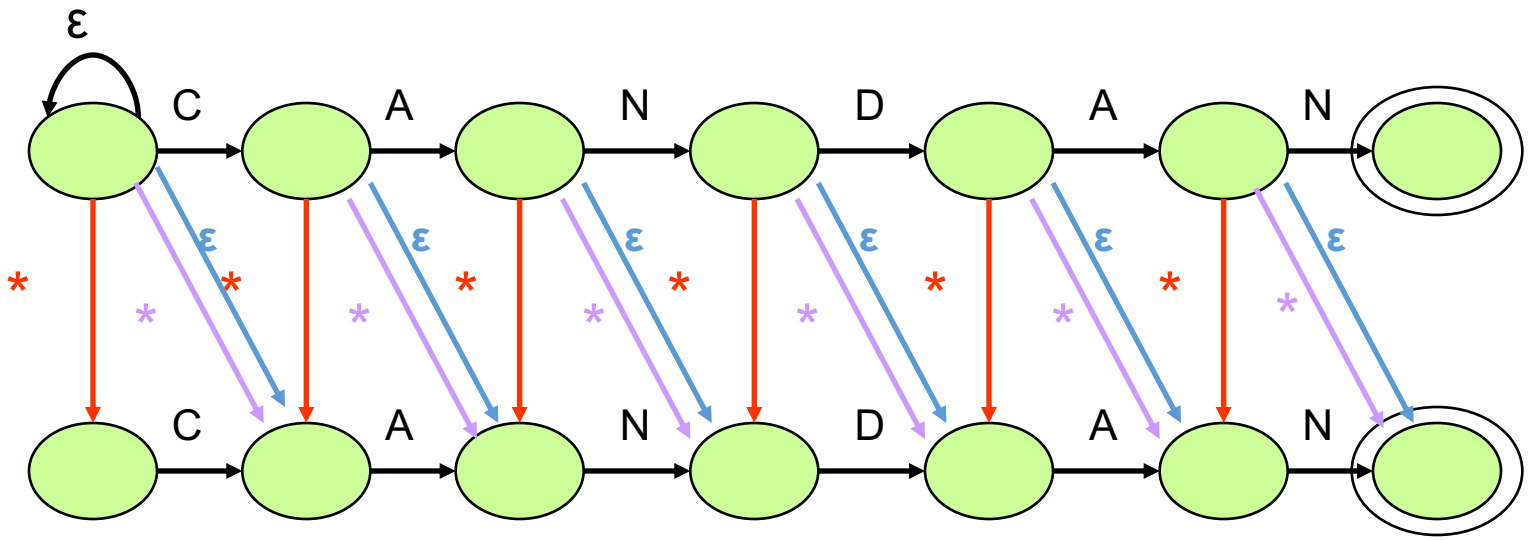
NFA…upto 1 insertion



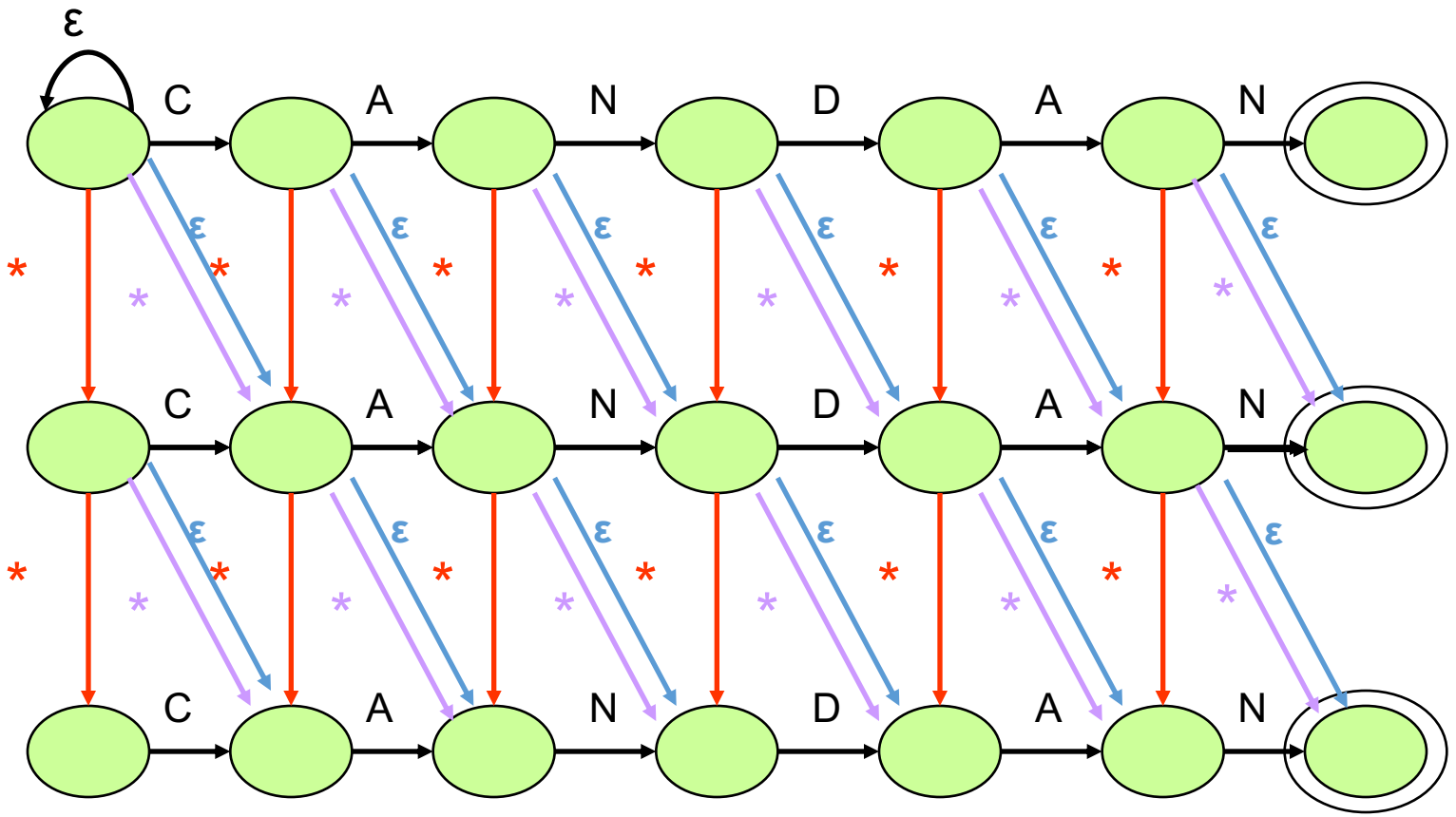
NFA…upto 1 insertion\replacement



NFA…upto 1 ins.\rep.\deletion



NFA…upto 2 ins.\rep.\deletion



Summary

- Prefix based sequence exploration:
  - Trie data structure helps prune the candidate set
- Subsequence search and exploration
  - Suffix trees and suffix arrays helps focus on the part of a long sequence
- Pattern matching
  - Non-deterministic finite automata can be used to support exact and approximate pattern matching