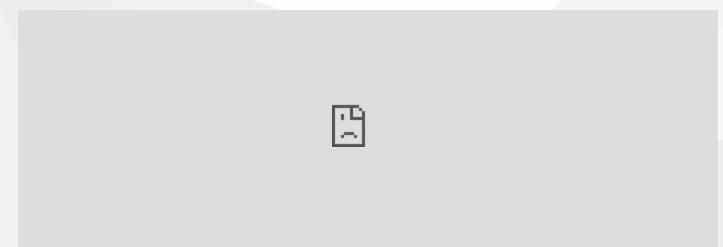
# **CE205 Data Structures**

Week-11

# **String Data Structures**

Download PDF, DOCX, SLIDE, PPTX



#### **Outline**

- Strings
  - -Longest common subsequence problem
    - Longest increasing subsequence
    - Hunt–Szymanski algorithm (Hunt Macllory)
    - Levenshtein distance
    - Wagner–Fischer algorithm
    - String Alignment
      - Needleman Wunsch
      - Smith Waterman
      - Hunt Macllory
    - String Tokenizer
    - String Comparison

# **Strings**

https://www.geeksforgeeks.org/string-data-structure/



- https://ucoruh.github.io/ce100-algorithms-and-programming-II/week-6/ce100-week-6-lcs/?h=lcs#problem-3-longest-common-subsequence
- https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/
- https://www.programiz.com/dsa/longest-common-subsequence



### Longest increasing subsequence

- https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/#:~:text=The Longest Increasing Subsequence (LIS) problem is to find the,50%2C 60%2C 80}.
- https://cp-algorithms.com/sequences/longest\_increasing\_subsequence.html



### Hunt-Szymanski algorithm (Hunt Macllory)

- https://en.wikipedia.org/wiki/Hunt–Szymanski\_algorithm
- https://www.geeksforgeeks.org/python-program-for-longest-commonsubsequence/?ref=gcse
- https://imada.sdu.dk/~rolf/Edu/DM823/E16/HuntSzymanski.pdf
- https://github.com/LetsTrie/Code-Library-Of-Others/blob/master/sgtlaugh/Hunt-Szymanski.cpp



#### Levenshtein distance

- https://en.wikipedia.org/wiki/Levenshtein\_distance
- https://www.geeksforgeeks.org/java-program-to-implement-levenshtein-distance-computing-algorithm/?ref=gcse
- https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0
- https://www.educative.io/answers/the-levenshtein-distance-algorithm



#### Wagner–Fischer algorithm

- https://en.wikipedia.org/wiki/Wagner–Fischer\_algorithm
- https://www.geeksforgeeks.org/java-program-to-implement-wagner-and-fisher-algorithm-for-online-string-matching/



• https://www.geeksforgeeks.org/sequence-alignment-problem/?ref=gcse



#### Needleman Wunsch

- https://en.wikipedia.org/wiki/Needleman–Wunsch\_algorithm
- https://www.geeksforgeeks.org/sequence-alignment-problem/?ref=gcse
- https://berthub.eu/nwunsch/
- http://experiments.mostafa.io/public/needleman-wunsch/index.html
- https://zhanggroup.org/NW-align/



#### **Smith Waterman**

- https://en.wikipedia.org/wiki/Smith–Waterman\_algorithm
- http://jaligner.sourceforge.net/
- http://baba.sourceforge.net/
- https://doc.ugene.net/wiki/display/UUOUM15/Smith-Waterman+Search
- https://www.ebi.ac.uk/Tools/sss/fasta/



## **Hunt Macllory**

- https://en.wikipedia.org/wiki/Hunt–Szymanski\_algorithm
- https://www.geeksforgeeks.org/python-program-for-longest-commonsubsequence/?ref=gcse
- https://imada.sdu.dk/~rolf/Edu/DM823/E16/HuntSzymanski.pdf
- https://github.com/LetsTrie/Code-Library-Of-Others/blob/master/sgtlaugh/Hunt-Szymanski.cpp



# **String Tokenizer**

- https://towardsdatascience.com/tokenization-algorithms-explained-e25d5f4322ac
- https://www.oreilly.com/library/view/applied-natural-language/9781492062561/ch04.html
- https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/?
  ref=gcse
- https://github.com/frohoff/jdk8u-devjdk/blob/master/src/share/classes/java/util/StringTokenizer.java



# **String Comparison**

- https://en.wikipedia.org/wiki/String-searching\_algorithm
- https://www.geeksforgeeks.org/compare-two-strings-in-java/
- https://www.geeksforgeeks.org/comparing-two-strings-cpp/



$$End-Of-Week-11$$

