Instituto de Computação

# Diff

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br

# Exercise

- Conceive an algorithm for identifying the differences among two files without a common ancestor (2-way diff)

# Possible Solution

- Identify the longest common subsequence among both files

# Possible Solution

- Subtract the longest common sequence from both sides to identify what was added/removed

**A**

\- B

**C**

**D**

**E**

\+ G

**F**

# However, how can we find the longest common sequence?

- Possible solution:
  - Generate all subsequences for one of the files
  - Check, for each generated subsequence, if it is also a subsequence of the other file
- Problem:
  - Complexity = $O(2^n n)$

# LCS

- Problem characteristics
  - Can be divided into subproblems
  - The subproblems can repeat during recursion (leading to redundant computation)
- LCS algorithm
  - Longest Common Subsequence
  - Used both in bioinformatics and diff program
  - Adopts Dynamic Programming technique
  - Complexity = $O(n^2)$

# LCS

- Considering the following sequences
  - Xi = (x1, x2, ..., xi)
  - Yj = (y1, y2, ..., yj)
- Algorithm

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & if\ i = 0 \vee j = 0 \\ (LCS(X_{i-1}, Y_{j-1}), x_i) & if\ x_i = y_j \\ longest(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & if\ x_i \neq y_j \end{cases}$$

# LCS

- It can be computed in a bottom-up fashion
  - Using a matrix with all elements of one sequence in the line and all elements of the other sequence in the column
  - Computing line 1 and column 1, then line 2 and column 2, and so on
  - Storing in each cell the length of the sequence and the path to the cells that belong to the LCS

# Longest Common Subsequence
# line & column = 0

| | **Ø** | **A** | **C** | **D** | **E** | **G** | **F** |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | | | | | | |
| **B** | ↑ 0 | | | | | | |
| **C** | ↑ 0 | | | | | | |
| **D** | ↑ 0 | | | | | | |
| **E** | ↑ 0 | | | | | | |
| **F** | ↑ 0 | | | | | | |

# Longest Common Subsequence
# line & column = 1

| | **Ø** | **A** | **C** | **D** | **E** | **G** | **F** |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | ↖ 1 | ← 1 | ← 1 | ← 1 | ← 1 | ← 1 |
| **B** | ↑ 0 | ↑ 1 | | | | | |
| **C** | ↑ 0 | ↑ 1 | | | | | |
| **D** | ↑ 0 | ↑ 1 | | | | | |
| **E** | ↑ 0 | ↑ 1 | | | | | |
| **F** | ↑ 0 | ↑ 1 | | | | | |

# Longest Common Subsequence
# line & column = 2

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | ↖ 1 | ← 1 | ← 1 | ← 1 | ← 1 | ← 1 |
| **B** | ↑ 0 | ↑ 1 | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ |
| **C** | ↑ 0 | ↑ 1 | ↖ 2 | | | | |
| **D** | ↑ 0 | ↑ 1 | ↑ 2 | | | | |
| **E** | ↑ 0 | ↑ 1 | ↑ 2 | | | | |
| **F** | ↑ 0 | ↑ 1 | ↑ 2 | | | | |

# Longest Common Subsequence
# line & column = 3

| | **Ø** | **A** | **C** | **D** | **E** | **G** | **F** |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | ↖ 1 | ← 1 | ← 1 | ← 1 | ← 1 | ← 1 |
| **B** | ↑ 0 | ↑ 1 | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ |
| **C** | ↑ 0 | ↑ 1 | ↖ 2 | ← 2 | ← 2 | ← 2 | ← 2 |
| **D** | ↑ 0 | ↑ 1 | ↑ 2 | ↖ 3 | | | |
| **E** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | | | |
| **F** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | | | |

# Longest Common Subsequence
# line & column = 4

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | ↖ 1 | ← 1 | ← 1 | ← 1 | ← 1 | ← 1 |
| **B** | ↑ 0 | ↑ 1 | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ |
| **C** | ↑ 0 | ↑ 1 | ↖ 2 | ← 2 | ← 2 | ← 2 | ← 2 |
| **D** | ↑ 0 | ↑ 1 | ↑ 2 | ↖ 3 | ← 3 | ← 3 | ← 3 |
| **E** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | ↖ 4 | | |
| **F** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | ↑ 4 | | |

# Longest Common Subsequence
# line & column = 5

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | ↖ 1 | ← 1 | ← 1 | ← 1 | ← 1 | ← 1 |
| **B** | ↑ 0 | ↑ 1 | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ |
| **C** | ↑ 0 | ↑ 1 | ↖ 2 | ← 2 | ← 2 | ← 2 | ← 2 |
| **D** | ↑ 0 | ↑ 1 | ↑ 2 | ↖ 3 | ← 3 | ← 3 | ← 3 |
| **E** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | ↖ 4 | ← 4 | ← 4 |
| **F** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | ↑ 4 | ← 4 ↑ | |

# Longest Common Subsequence
# line & column = 6

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | ↖ 1 | ← 1 | ← 1 | ← 1 | ← 1 | ← 1 |
| **B** | ↑ 0 | ↑ 1 | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ |
| **C** | ↑ 0 | ↑ 1 | ↖ 2 | ← 2 | ← 2 | ← 2 | ← 2 |
| **D** | ↑ 0 | ↑ 1 | ↑ 2 | ↖ 3 | ← 3 | ← 3 | ← 3 |
| **E** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | ↖ 4 | ← 4 | ← 4 |
| **F** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | ↑ 4 | ← 4 ↑ | ↖ 5 |

# Longest Common Subsequence

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | **0** | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 | ← 0 |
| **A** | ↑ 0 | **↖ 1** | ← 1 | ← 1 | ← 1 | ← 1 | ← 1 |
| **B** | ↑ 0 | **↑ 1** | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ | ← 1 ↑ |
| **C** | ↑ 0 | ↑ 1 | **↖ 2** | ← 2 | ← 2 | ← 2 | ← 2 |
| **D** | ↑ 0 | ↑ 1 | ↑ 2 | **↖ 3** | ← 3 | ← 3 | ← 3 |
| **E** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | **↖ 4** | **← 4** | ← 4 |
| **F** | ↑ 0 | ↑ 1 | ↑ 2 | ↑ 3 | ↑ 4 | ← 4 ↑ | **↖ 5** |

# Improvements

- The first implementation of Unix Diff (Hunt & McIlroy, 1976) uses a variation of this LCS algorithm
  - $O(n)$ space complexity
  - $O(n^2 \times \log n)$ time complexity
- The current implementation of Unix Diff (Miller & Myers, 1985) does not fill the whole matrix
  - $O(n)$ space complexity
  - $O(n \times d)$ time complexity, where d is the edit distance

# Miller & Myers algorithm

- bootstrap: Add zeros to the diagonal while the symbols match
- While the lowermost and rightmost cell is empty
  - Rule 1: For each filled cell, inserts its value added by one in the cell in the right
  - Rule 2: For each filled cell, inserts its value added by one in the cell in the bottom
  - Rule 3: For each filled cell, recursively inserts its value in the cell in the diagonal (bottom right) if the symbols in the diagonal match

# Shortest Edit Distance
# d = 0: bootstrap

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | | | | | | |
| **A** | | ↖ 0 | | | | | |
| **B** | | | | | | | |
| **C** | | | | | | | |
| **D** | | | | | | | |
| **E** | | | | | | | |
| **F** | | | | | | | |

# Shortest Edit Distance
# d = 1: rule 1

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 1 | | | | | |
| **A** | | ↖ 0 | ← 1 | | | | |
| **B** | | | | | | | |
| **C** | | | | | | | |
| **D** | | | | | | | |
| **E** | | | | | | | |
| **F** | | | | | | | |

# Shortest Edit Distance
# d = 1: rule 2

| | **Ø** | **A** | **C** | **D** | **E** | **G** | **F** |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 1 | | | | | |
| **A** | ↑ 1 | ↖ 0 | ← 1 | | | | |
| **B** | | ↑ 1 | | | | | |
| **C** | | | | | | | |
| **D** | | | | | | | |
| **E** | | | | | | | |
| **F** | | | | | | | |

# Shortest Edit Distance
# d = 1: rule 3

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 1 | | | | | |
| **A** | ↑ 1 | ↖ 0 | ← 1 | | | | |
| **B** | | ↑ 1 | | | | | |
| **C** | | | ↖ 1 | | | | |
| **D** | | | | ↖ 1 | | | |
| **E** | | | | | ↖ 1 | | |
| **F** | | | | | | | |

# Shortest Edit Distance
# d = 2: rule 1

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 1 | ← 2 | | | | |
| **A** | ↑ 1 | ↖ 0 | ← 1 | ← 2 | | | |
| **B** | | ↑ 1 | ← 2 | | | | |
| **C** | | | ↖ 1 | ← 2 | | | |
| **D** | | | | ↖ 1 | ← 2 | | |
| **E** | | | | | ↖ 1 | ← 2 | |
| **F** | | | | | | | |

# Shortest Edit Distance
# d = 2: rule 2

| | **Ø** | **A** | **C** | **D** | **E** | **G** | **F** |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 1 | ← 2 | | | | |
| **A** | ↑ 1 | ↖ 0 | ← 1 | ← 2 | | | |
| **B** | ↑ 2 | ↑ 1 | ← 2 | | | | |
| **C** | | ↑ 2 | ↖ 1 | ← 2 | | | |
| **D** | | | ↑ 2 | ↖ 1 | ← 2 | | |
| **E** | | | | ↑ 2 | ↖ 1 | ← 2 | |
| **F** | | | | | ↑ 2 | | |

# Shortest Edit Distance
# d = 2: rule 3

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | 0 | ← 1 | ← 2 | | | | |
| **A** | ↑ 1 | ↖ 0 | ← 1 | ← 2 | | | |
| **B** | ↑ 2 | ↑ 1 | ← 2 | | | | |
| **C** | | ↑ 2 | ↖ 1 | ← 2 | | | |
| **D** | | | ↑ 2 | ↖ 1 | ← 2 | | |
| **E** | | | | ↑ 2 | ↖ 1 | ← 2 | |
| **F** | | | | | ↑ 2 | | ↖ 2 |

# Shortest Edit Distance

| | Ø | A | C | D | E | G | F |
|---|---|---|---|---|---|---|---|
| **Ø** | **0** | ← 1 | ← 2 | | | | |
| **A** | ↑ 1 | **↖ 0** | ← 1 | ← 2 | | | |
| **B** | ↑ 2 | **↑ 1** | ← 2 | | | | |
| **C** | | ↑ 2 | **↖ 1** | ← 2 | | | |
| **D** | | | ↑ 2 | **↖ 1** | ← 2 | | |
| **E** | | | | ↑ 2 | **↖ 1** | **← 2** | |
| **F** | | | | | ↑ 2 | | **↖ 2** |

# Diff Algorithms in Git

- Myers
  - Diff algorithm proposed by Myers with speed optimizations that may lead to a non-minimal edit distance
- Minimal
  - Myers with a guarantee of minimal edit distance
- Patience
  - Just considers the unique lines in both files for computing the LCS, potentially leading to a more precise result
- Histogram
  - Extends the Patience algorithm to support low-occurrence common lines instead of just unique lines, potentially leading to faster executions

# References


- Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., 2001. Introduction to Algorithms, 2nd ed., MIT Press.
- Hunt, J., McIlroy, M., “An Algorithm for Differential File Comparison”, Bell Laboratories, 1976.
- Miller, W., Myers, E., “A File Comparison Program”, Software: Practice and Experience, v. 15, n. 11, p. 1025-1040, 1985.

Instituto de Computação

# Diff

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br