# Principles of Computer Science II

## Sequence Similarity

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 20

# Edit Distance

- We looked for repeating patterns within DNA sequences.
- How can we measure the similarity between different sequences?
- We use the notion of Vladimir Levenshtein introduced in 1966
- Edit distance – the minimum number of editing operations needed to transform one string into another (insert/delete symbol or substitute one symbol for another).

### Alignment of ATATATAT vs TATATATA

```
A T A T A T A T -
  : : : : : : :
- T A T A T A T A
```

# Sequence Similarity

### Alignment of ATATATAT vs TATAAT

```
A T A T A T A T
  : : : : : : :
- T A T A - A T
```

# Sequence Similarity

### Alignment of TGCATAT vs ATCCGAT

TGCATAT

↓ delete last T

TGCATA

↓ delete last A

TGCAT

↓ insert A at the front

ATGCAT

↓ substitute C for G in the third position

ATCCAT

↓ insert a G before the last A

ATCCGAT

Five operations.

# Sequence Similarity

## Alignment of TGCATAT vs ATCCGAT

TGCATAT
↓ insert A at the front
ATGCATAT
↓ delete T in the sixth position
ATGCAAT
↓ substitute G for A in the fifth position
ATGCGAT
↓ substitute C for G in the third position
ATCCGAT

Four operations.

# Edit Distance

- Vladimir Levenshtein defined the notion of Edit distance
- Did not provide an algorithm to compute it.

# Edit Distance Algorithm using Dynamic Programming

- Assume two strings:
  - $v$ (of $n$ characters)
  - $w$ (of $m$ characters)
- The alignment of $v, w$ is a two-row matrix such that
  - first row: contains the characters of $v$ (in order)
  - second row: contains the characters of $w$ (in order)
  - spaces are interpersed throughout the table.
- Characters in each string appear in order, though not necessarily adjacently.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | T | - | G | T | T | A | T | - |
| A | T | C | G | T | - | A | - | C |

- No column contains spaces in both rows.
- At most $n + m$ columns.

# Edit Distance Algorithm using Dynamic Programming

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | T | - | G | T | T | A | T | - |
| A | T | C | G | T | - | A | - | C |

- Matches – columns with the same letter,
- Mismatches – columns with different letters.
- Columns containing one space are called indels
  - Space on top row: insertions
  - Space on bottom row: deletions

$$\#\text{matches} + \#\text{mismatches} + \#\text{indels} < n + m$$

## Representing the rows

| v | A | T | - | G | T | T | A | T | - |
|---|---|---|---|---|---|---|---|---|---|
| w | A | T | C | G | T | - | A | - | C |

- One way to represent $v$
  - AT-CGTAT-
- One way to represent $w$
  - ATCGT-A-C
- Another way to represent $v$
  - AT-CGTAT-
  - 122345677
  - number of symbols of $v$ present up to a given position
- Similarly, to represent $w$
  - ATCGT-A-C
  - 123455667

## Representing the rows

| v | A | T | - | G | T | T | A | T | - |
|---|---|---|---|---|---|---|---|---|---|
| w | A | T | C | G | T | - | A | - | C |

| v | 1 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| w | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 6 | 7 |

can be viewed as a coordinate in 2-dimensional $n \times m$ grid:

$$\binom{0}{0} \binom{1}{1} \binom{2}{2} \binom{2}{3} \binom{3}{4} \binom{4}{5} \binom{5}{5} \binom{6}{6} \binom{7}{6} \binom{7}{7}$$

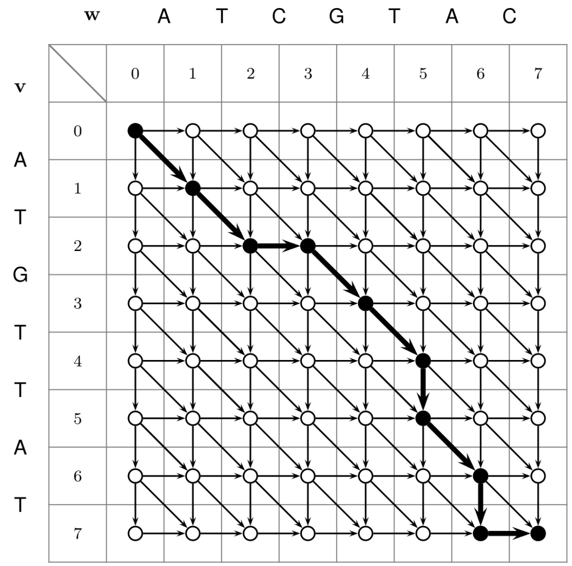
The entire alignment is simply a path:

$(0,0) \rightarrow (1,1) \rightarrow (2,2) \rightarrow (2,3) \rightarrow (3,4) \rightarrow (4,5) \rightarrow (5,5) \rightarrow (6,6) \rightarrow (7,6) \rightarrow (7,7)$

## Edit distance graph

- Edit graph: a grid of $n, m$ size.
- The edit graph will help us in calculating the edit distance.
- Alignment: a path from $(0,0)$ to $(n,m)$.
- Every alignment corresponds to a path in the edit graph.
- Diagonal movement at point $i, j$ correspond to column $\binom{v_i}{w_j}$
- Horizontal movement correspond to column $\binom{-}{w_j}$
- Vertical movement correspond to column $\binom{v_i}{-}$

## Edit distance graph




| ↘ | ↘ | → | ↘ | ↘ | ↓ | ↘ | ↓ | → |
|---|---|---|---|---|---|---|---|---|
| A | T | - | G | T | T | A | T | - |
| A | T | C | G | T | - | A | - | C |

# Profile most-frequent k-mer

```python
def edit_distance(s1, s2):
    m=len(s1)+1
    n=len(s2)+1

    tbl = {}
    for i in range(m): tbl[i,0]=i
    for j in range(n): tbl[0,j]=j
    for i in range(1, m):
        for j in range(1, n):
            cost = 0 if s1[i-1] == s2[j-1] else 1
            tbl[i,j] = min(tbl[i, j-1]+1, tbl[i-1, j]+1,
                tbl[i-1, j-1]+cost)

    return tbl[i,j]
```

# Profile most-frequent k-mer

```python
def levenshteinDistance(s1, s2):
    if len(s1) > len(s2):
        s1, s2 = s2, s1

    distances = range(len(s1) + 1)
    for i2, c2 in enumerate(s2):
        distances_ = [i2+1]
        for i1, c1 in enumerate(s1):
            if c1 == c2:
                distances_.append(distances[i1])
            else:
                distances_.append(1 + min((distances[i1],
                    distances[i1 + 1], distances_[-1])))
        distances = distances_
    return distances[-1]
```

# $4^{th}$ Assignment

- Work in groups of 3 – not the same as $3^{rd}$ assignment
- Implement an algorithm to solve the following *Equivalent Words* problem.
- Implement an algorithm to solve the generalized *Equivalent Words* problem.
- Solve the third problem.
- Email `ichatz@dis.uniroma1.it`
  Subject: [PCS2] Homework 4
  A link to a github repository with your python code.
- **Deadline: 18/January/2018 or 15/February/2018**

# $4^{th}$ Assignment – 1st Problem

## Equivalent Words

Transform one English word $v$ into another word $w$ by going through a series of intermediate English words, where each word in the sequence differs from the next by only one substitution (1 character).

- Given two words $v, w$ and a dictionary, find out whether the words are equivalent.
- Your program should output the series of transformations for $v$ to become $w$
- Use the following dictionary: `https://goo.gl/hBvqqr`
- Example: To transform head into tail one can use four intermediates:
  head → heal → teal → tell → tall → tail

# $4^{th}$ Assignment – 2nd Problem

## Generalized Equivalent Words

Find an algorithm to solve a generalization of the Equivalent Words problem when insertions, deletions, and substitutions are allowed (rather than only substitutions).

- Given two words $v, w$ and a dictionary, find out whether the words are equivalent.
- Your program should output the series of transformations for $v$ to become $w$
- Use the following dictionary: `https://goo.gl/hBvqqr`
- Example: To transform head into tea one can use four intermediates:
  head $\rightarrow$ heal $\rightarrow$ teal $\rightarrow$ tea

# $4^{th}$ Assignment – 3rd Problem

Two players play the following game with a nucleotide sequence of length $n = n_A + n_T + n_C + n_G$, where $n_A, n_T, n_C$, and $n_G$ are the number of $A, T, C$, and $G$ in the sequence. At every turn a player may delete either one or two nucleotides from the sequence. The player who is left with a uni-nucleotide sequence of an arbitrary length (i.e., the sequence containing only one of 4 possible nucleotides) loses. Who will win? Describe the winning strategy for each $n_A$, $n_T$, $n_C$ , and $n_G$.