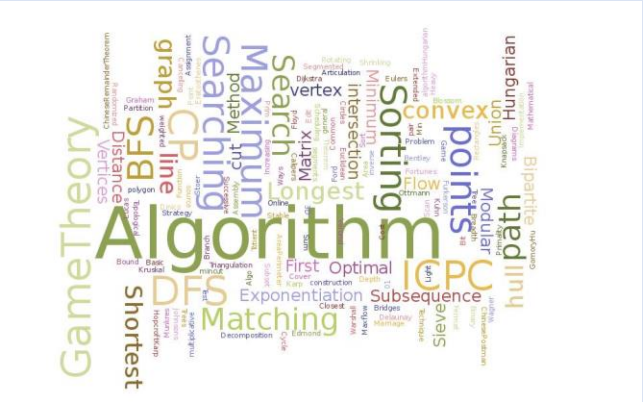
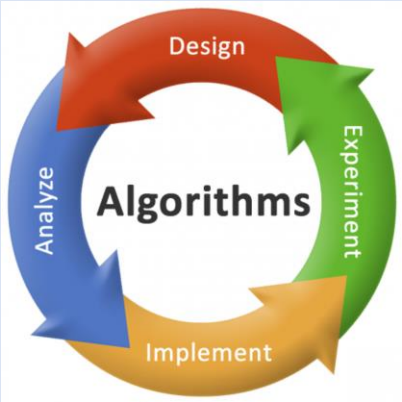
Dynamic Programming Sequence Alignment

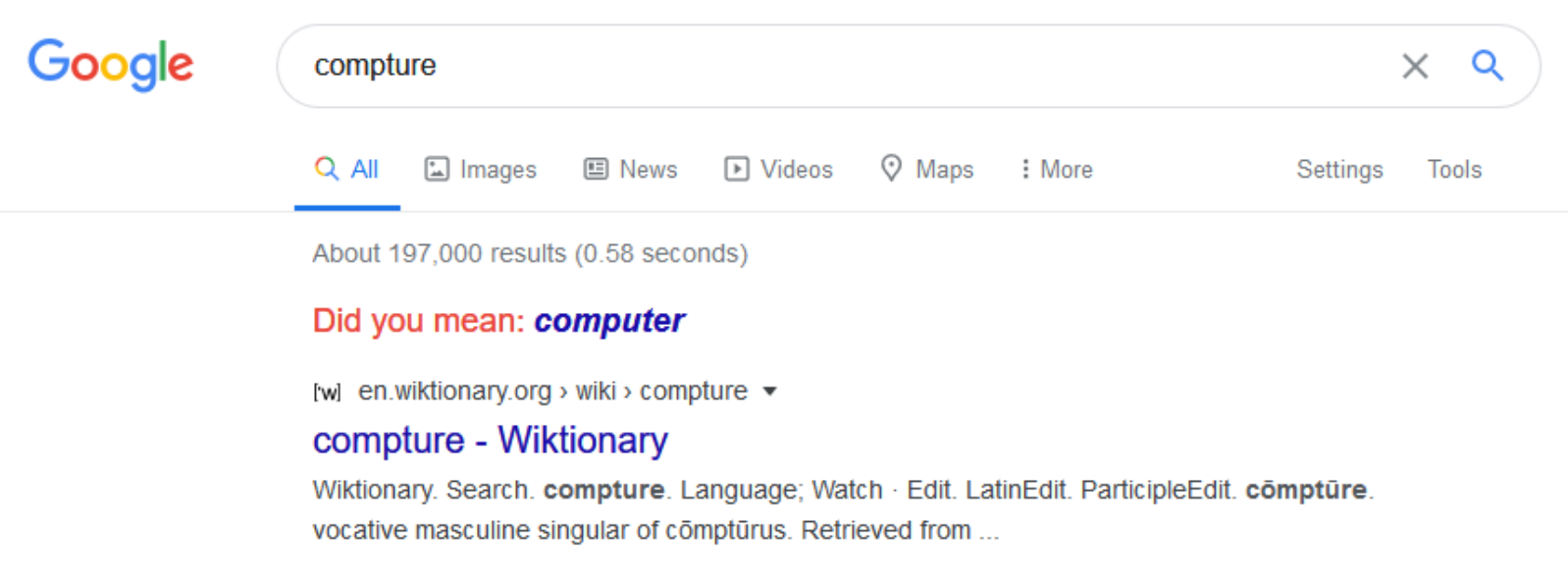
COP 3503
Fall 2021
Department of Computer Science
University of Central Florida
Dr. Steinberg





The Problem

- Dictionaries and websites are getting more and more useful.
- What happens when you type “ocurrance”?
  - “Perhaps you mean occurrence?”
  - The dictionary will search its entries for the word most “similar” to the one typed.



How can we model similarity between two strings?

```
o-currance
occurrence
```

one gap

one mismatch

```
o-curr-ance
occurre-nce
```

three gaps

No mismatch

Another Application – Molecular Biology

- The organism's genome
  - Divided into giant linear DNA molecules known as chromosomes
  - String over the alphabet {A, C, G, T} to determine the properties of the organism
    - Adenine, Cytosine, Guanine, Thymine
- Why is this important?
- Let's say we have two bacteria strains to observe. Sequence alignment allows scientists to observe any possible substrings that have similarities.

The Sequence Alignment Problem Definition

- Let $X = x_1x_2...x_m$ and $Y = y_1y_2...y_n$
- {1, 2, ..., m} and {1, 2, ..., n} represents different positions in strings X and Y
- Matching is a set of ordered pairs with the property that each items occurs in at most one pair.
- We say that a matching M of these two sets is an alignment if there are no "crossing" pairs: if $(i,j), (i',j') \in M$ and $i < i'$ then $j < j'$

Example

- An alignment provides a way to line up two strings

```
stop-
-tops
```

- Corresponds {(2,1)(3,2)(4,3)}

Sequence Alignment Problem

- Suppose M is a given alignment between X and Y
  - *gap penalty*: each gap incurs at cost $\delta > 0$
  - *mismatch cost*: for each pair of letters p, q in the alphabet, there is a mismatch cost α_{pq} for lining up p and q
    - Assumption $\alpha_{pp} = 0$
  - The *cost* of M is the sum of gap and mismatch costs
- Objective: Find an *optimal alignment*, that means an alignment of minimum cost
- Values δ and α_{pq} are provided
- The lower the cost, the more similar X and Y are
- Looking back at *ocurrance* and *occurrence*, the first alignment is better if and only if $\delta + \alpha_{ae} < 3\delta$

Designing the Dynamic Programming Algorithm

- In the optimal alignment M:
  - either $(m,n) \in M$ or $(m,n) \notin M$
- In any alignment
  - if $(m,n) \notin M$, then either the mth position of X or the nth position of Y are not matched in M
  - Can be proven using proof by contradiction
- Property: 1 of the following is true for an aligmnment
  - $(m,n) \in M$
  - The mth position of X is not matched
  - The nth position of Y is not matched

Designing the Dynamic Programming Algorithm

- Let's define OPT(i,j) as the minimum cost of an alignment between $x_1x_2...x_m$ and $y_1y_2...y_n$
- Recursively define OPT(m,n)

Case 1

- Pay α_{xmyn} then optimally align $x_1x_2...x_{m-1}$ and $y_1y_2...y_{n-1}$
  $OPT(m,n) = \alpha_{xmyn} + OPT(m-1,n-1)$

Case 2

- Pay gap cost δ, then optimally align $x_1x_2...x_{m-1}$ and $y_1y_2...y_n$
  $OPT(m,n) = \delta + OPT(m-1,n)$

Case 3

- Pay gap cost δ, then optimally align $x_1x_2...x_m$ and $y_1y_2...y_{n-1}$
  $OPT(m,n) = \delta + OPT(m,n-1)$

Designing the Dynamic Programming Algorithm

- Property: The minimum alignment costs satisfy the following recurrence, for $i \geq 1, j \geq 1$
- $OPT(i,j) = \min(\alpha_{x_i y_j} + OPT(i-1,j-1), \delta + OPT(i-1,j), \delta + OPT(i,j-1))$
- The minimal optimal alignment is achieved through these 3 values.

The Algorithm

<u>Alignment(X,Y)</u>

```
array A[0...m, 0...n]
initialize array A[i,0] = iδ for each i
initialize array A[0,j] = jδ for each j
for j = 1 to n
   for i = 1 to m
      A[i,j] = min{α_xiyj + A[i-1,j-1], δ + A[i-1,j], δ + A[i,j-1]}
return A[m,n]
```

Running Time: O(mn)