CS3000: Algorithms & Data
Paul Hand

Lecture 12:

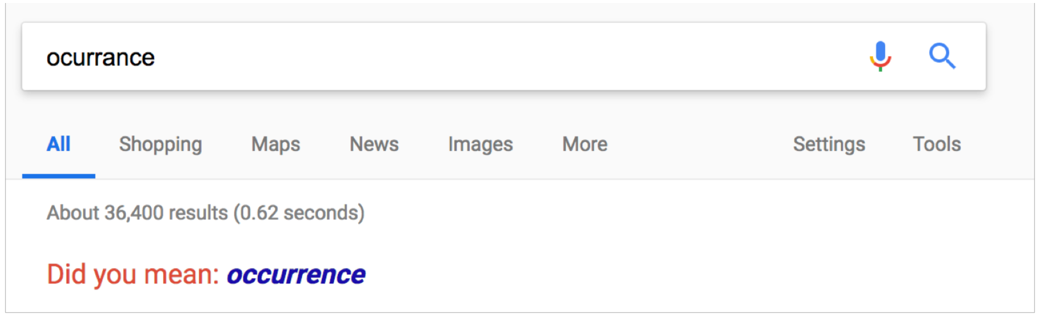
- Dynamic Programming – Sequence Alignment
- Introduction to Graphs

Feb 25, 2019

Sequence Alignments and Edit Distance

Distance Between Strings

- Autocorrect works by finding similar strings



ocurrance

About 36,400 results (0.62 seconds)

Did you mean: ***occurrence***

- `ocurrance` and `occurrence` seem similar, but only if we define similarity carefully

```
ocurrance          oc urrance
occurrence         occurrence
```

Edit Distance / Alignments

- Given two strings $x \in \Sigma^n, y \in \Sigma^m$, the **edit distance** is the number of insertions, deletions, and swaps required to turn x into y.

- Given an alignment, the cost is the number of positions where the two strings don't agree

o	c		u	r	r	a	n	c	e
o	c	c	u	r	r	e	n	c	e

Ask the Audience

- What is the minimum cost alignment of the strings **`smitten`** and **`sitting`**

Activity

- Find two strings where two different alignments (insertions, deletions, replacements) realize the edit distance between them.

Edit Distance / Alignments

- **Input:** Two strings $x \in \Sigma^n, y \in \Sigma^m$
- **Output:** The minimum cost alignment of x and y
  - **Edit Distance** = cost of the minimum cost alignment

Dynamic Programming

- Consider the **optimal** alignment of x, y
- Three choices for the final column
  - **Case I:** only use x ($x_n, -$)
  - **Case II:** only use y ($-, y_m$)
  - **Case III:** use one symbol from each (x_n, y_m)

Dynamic Programming

- Consider the **optimal** alignment of x, y
- **Case I:** only use x ($x_n, -$)
  - deletion + optimal alignment of $x_{1:n-1}, y_{1:m}$
- **Case II:** only use y ($-, y_m$)
  - insertion + optimal alignment of $x_{1:n}, y_{1:m-1}$
- **Case III:** use one symbol from each (x_n, y_m)
  - If $x_n = y_m$: optimal alignment of $x_{1:n-1}, y_{1:m-1}$
  - If $x_n \neq y_m$: mismatch + opt. alignment of $x_{1:n-1}, y_{1:m-1}$

Dynamic Programming

- $\mathbf{OPT}(\boldsymbol{i},\boldsymbol{j})$ = cost of opt. alignment of $x_{1:i}$ and $y_{1:j}$
- **Case I:** only use x ($x_i, -$)
- **Case II:** only use y ($-, y_j$)
- **Case III:** use one symbol from each (x_i, y_j)

Dynamic Programming

- $\mathbf{OPT}(\boldsymbol{i},\boldsymbol{j})$ = cost of opt. alignment of $x_{1:i}$ and $y_{1:j}$
- **Case I:** only use x ($x_i, -$)
- **Case II:** only use y ($-, y_j$)
- **Case III:** use one symbol from each (x_i, y_j)

Recurrence:

$$\mathrm{OPT}(i,j) = \begin{cases} \min\{1+\mathrm{OPT}(i-1,j), 1+\mathrm{OPT}(i,j-1), \quad \mathrm{OPT}(i-1,j-1)\ \} & x_i = y_j \\ \min\{1+OPT(i-1,j), 1+\mathrm{OPT}(i,j-1), 1+\ \mathrm{OPT}(i-1,j-1)\} & x_i \neq y_j \end{cases}$$

Base Cases:

$\mathrm{OPT}(i,0) = i, \mathrm{OPT}(0,j) = j$

Example

x = pert

y = beast

	-	b	e	a	s	t
-						
p						
e						
r						
t						

Finding the Alignment

- $\mathbf{OPT}(\boldsymbol{i}, \boldsymbol{j})$ = cost of opt. alignment of $x_{1:i}$ and $y_{1:j}$
- **Case I:** only use x ($x_i, -$)
- **Case II:** only use y ($-, y_j$)
- **Case III:** use one symbol from each (x_i, y_j)

Edit Distance (“Bottom-Up”)

```
// All inputs are global vars
FindOPT(n,m):
  M[0,j]←j, M[i,0]←i

  for (i= 1,…,n):
    for (j = 1,…,m):
      if ($x_i$ = $y_j$):
        M[i,j] = min{1+M[i-1,j],1+M[i,j-1],M[i-1,j-1]
      elseif ($x_i$ != $y_j$):
        M[i,j] = 1+min{M[i-1,j],M[i,j-1],M[i-1,j-1]}

  return M[n,m]
```

Activity

- Suppose **inserting/deleting costs $\boldsymbol{\delta > 0}$** and **swapping $\boldsymbol{a \leftrightarrow b}$ costs $\boldsymbol{c_{a,b} > 0}$**
- Write a recurrence for the min-cost alignment

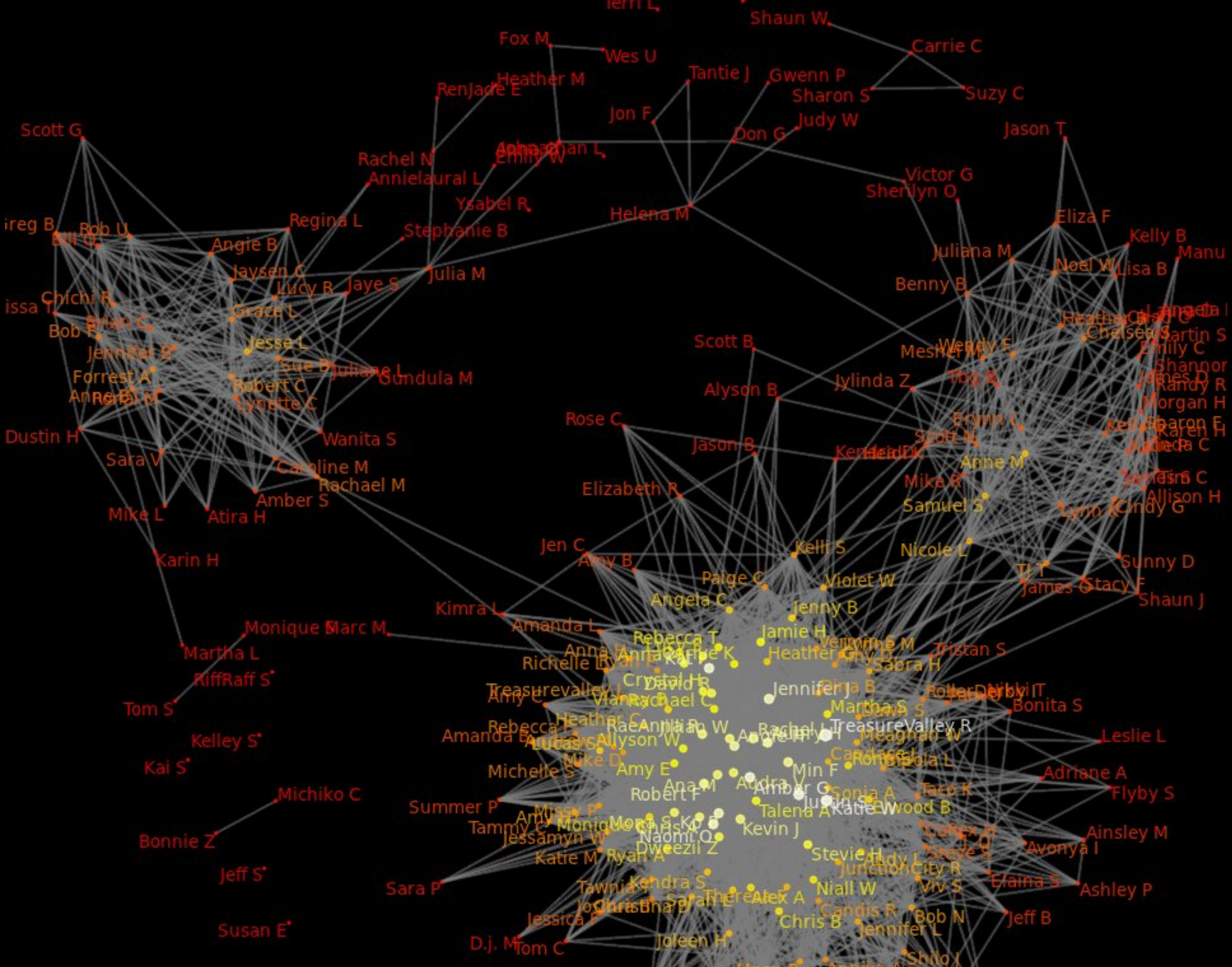
$$\mathrm{OPT}(i,j) = \begin{cases} \min\{1 + \mathrm{OPT}(i-1,j), 1 + \mathrm{OPT}(i,j-1), \quad \mathrm{OPT}(i-1,j-1)\ \} & x_i = y_j \\ \min\{1 + OPT(i-1,j), 1 + \mathrm{OPT}(i,j-1), 1 + \ \mathrm{OPT}(i-1,j-1)\} & x_i \neq y_j \end{cases}$$

Discussion

- Dynamic Programming is a time-space tradeoff. Comment on the tradeoff in the case of edit distance.

$$\mathrm{OPT}(i,j)=\begin{cases}\min\{1+\mathrm{OPT}(i-1,j),1+\mathrm{OPT}(i,j-1),\qquad \mathrm{OPT}(i-1,j-1)\,\} & x_i=y_j\\ \min\{1+OPT(i-1,j),1+\mathrm{OPT}(i,j-1),1+\ \mathrm{OPT}(i-1,j-1)\} & x_i\neq y_j\end{cases}$$

Graphs


Terri L
Shaun W
Fox M
Wes U
Carrie C
Heather M
Tantie J
Gwenn P
RenJade E
Sharon S
Suzy C
Scott G
Jon F
Judy W
Jason T
Don G
Rachel N
Emily W
Annielaural L
Victor G
Sherilyn O
Ysabel R
Helena M
Eliza F
Stephanie B
Regina L
Bob U
Bill O
Angie B
Kelly B
Manu
Juliana M
Noel W
Lisa B
Jaysen C
Julia M
Lucy R
Jaye S
Benny B
Chichi R
Grace L
Brian C
Bob F
Jennifer B
Jesse L
Sue D
Juliane L
Gundula M
Forrest A
Robert C
Lynette C
Scott B
Wendy F
Chelsea S
Martin S
Emily C
Randy R
Morgan H
Alyson B
Jylinda Z
Rose C
Jason B
Karen H
Anne M
Dustin H
Wanita S
Sara V
Caroline M
Rachael M
Amber S
Mike L
Atira H
Elizabeth R
Mike R
Samuel S
Allison H
Cindy G
Karin H
Jen C
Amy B
Nicole L
Kelli S
Paige C
Violet W
Sunny D
TJ T
James G
Stacy F
Shaun J
Angela C
Jenny B
Kimra L
Amanda L
Monique S
Marc M
Rebecca T
Jamie H
Tristan S
Martha L
Sabra H
Richelle L
RiffRaff S
Crystal H
David R
Jennifer J
Treasurevalley J
Amy C
Martha S
Bonita S
Tom S
TreasureValley R
Leslie L
Amanda B
Alyson W
Meaghan W
Kelley S
Lucas S
Mike D
Kai S
Michelle S
Amy E
Min F
Adriane A
Michiko C
Ana M
Audra V
Amber G
Sonia A
Taco K
Flyby S
Robert F
Talena A
Katie W
Ellwood B
Summer P
Missy P
Kevin J
Naomi Q
Tammy C
Ainsley M
Jessamyn W
Bonnie Z
Dweezil Z
Avonya I
Katie M
Ryan A
Stevie H
JunctionCity R
Jeff S
Elaina S
Ashley P
Sara P
Kendra S
Niall W
Viv S
Alex A
Candis R
Bob N
Jeff B
Chris B
Jessica F
Jennifer L
Susan E
D.j. M
Tom C
Joleen H

Graphs Are Everywhere

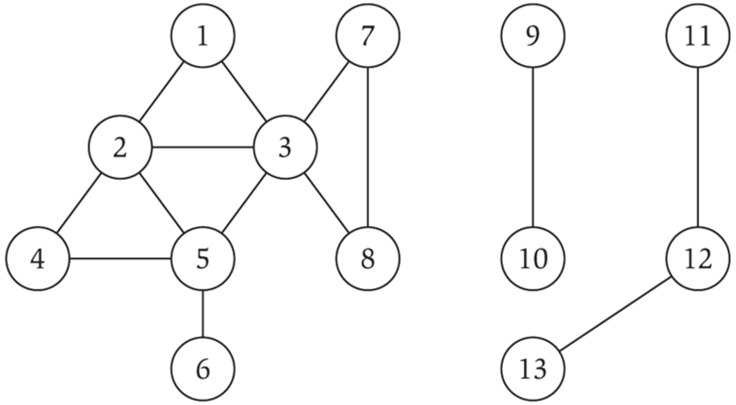
- Transportation networks
- Communication networks
- WWW
- Biological networks
- Citation networks
- Social networks
- …

What’s Next

- **Graph Algorithms:**
  - **Graphs:** Key Definitions, Properties, Representations
  - **Exploring Graphs:** Breadth/Depth First Search
    - Applications: Connectivity, Bipartiteness, Topological Sorting
  - **Shortest Paths:**
    - Dijkstra
    - Bellman-Ford (Dynamic Programming)
  - **Minimum Spanning Trees:**
    - Borůvka, Prim, Kruskal
  - **Network Flow:**
    - Algorithms
    - Reductions to Network Flow

Graphs: Key Definitions

- **Definition:** A directed graph $G = (V, E)$
  - V is the set of nodes/vertices
  - $E \subseteq V \times V$ is the set of edges
  - An edge is an ordered $e = (u, v)$ "from u to v"
- **Definition:** An undirected graph $G = (V, E)$
  - Edges are unordered $e = (u, v)$ "between u and v"
- **Simple Graph:**
  - No duplicate edges
  - No self-loops $e = (u, u)$



Activity

- How many edges can there be in a **simple** directed/undirected graph?

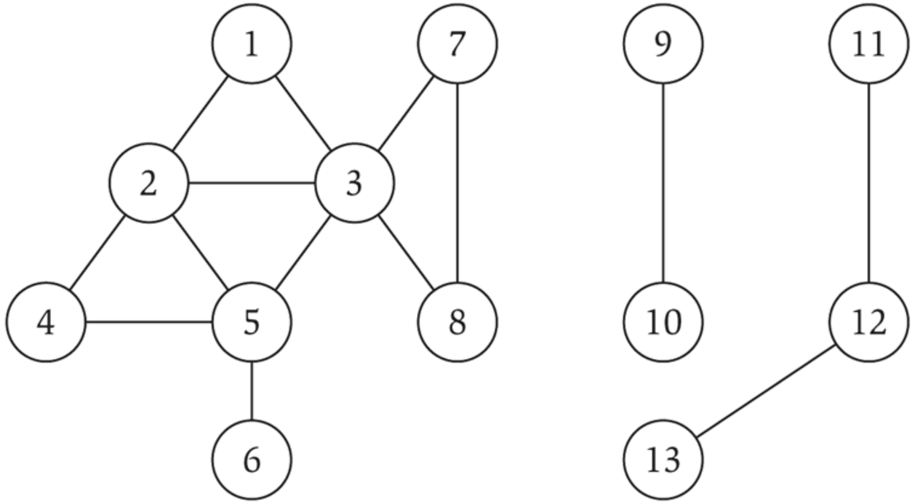
Paths/Connectivity

- A path is a sequence of consecutive edges in E
  - $P = \{(u, w_1), (w_1, w_2), (w_2, w_3), \dots, (w_{k-1}, v)\}$
  - $P = u - w_1 - w_2 - w_3 - \cdots - w_{k-1} - v$
  - The length of the path is the # of edges

- An undirected graph is connected if for every two vertices $u, v \in V$, there is a path from u to v
- A directed graph is strongly connected if for every two vertices $u, v \in V$, there are paths from u to v and from v to u

Cycles

- A cycle is a path $v_1 - v_2 - \cdots - v_k - v_1$ where $k \geq 3$ and $v_1, \ldots, v_k$ are distinct



Activity: how many cycles are there in this graph?

Activity

- Suppose an undirected graph G is connected
  - True/False? G has at least $n - 1$ edges

Activity

- Suppose an undirected graph G has $n - 1$ edges
  - True/False? G is connected