NPTEL MOOC,JAN-FEB 2015
Week 1, Module 4

DESIGN AND ANALYSIS OF ALGORITHMS

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Example 3: Document similarity

- Given two documents, how similar are they?
  - Plagiarism detection
  - Checking changes between versions of code
  - Answering web search queries more effectively

Document similarity …

- What is a good measure of similarity?
- **Edit distance**
  - How many changes does one have to make to get from one document to another?
  - What types of changes are allowed?
    - Add or remove a letter
    - Replace one letter by another

Document similarity …

- **Edit Distance**
  - Minimum number of edit operations to transform one document to another
  - How do we compute it?
  - Brute force: try all sequences and choose the best one
    - Delete all of first document, add all of second document
  - Impossibly inefficient!

Decomposing the problem

- Make the first character in both documents the same
  - Explore all possible edit operations that make this possible
- Recursively fix the rest of the documents
- Naive recursion is inefficient
  - Same subproblem solved recursively many times

Naive recursion can be inefficient

- Fibonacci numbers:
  - F(n) = F(n-1) + F(n-2), F(1) = 1, F(2) = 1
    - Sequence is 1,1,2,3,5,8,13,21,….
- Computing recursively
  - F(7) = F(6) + F(5) = (F(5) + F(4)) + (F(4)+F(3)) = (F(4)+F(3)+F(3)+F(2)) + (F(3)+F(2)+F(2)+F(1)) = …

Dynamic Programming

- Making recursive computations efficient
- Ensure that subproblems are computed only once
- How do we store and look up answers to already solved subproblems?

Variations

- Interested only in the meaning of the document
- Focus on words
- Documents are near if they overlap on many words
  - Order in which words occur may not matter
  - Useful for topic based web search
  - Can have dictionary of “similar” words