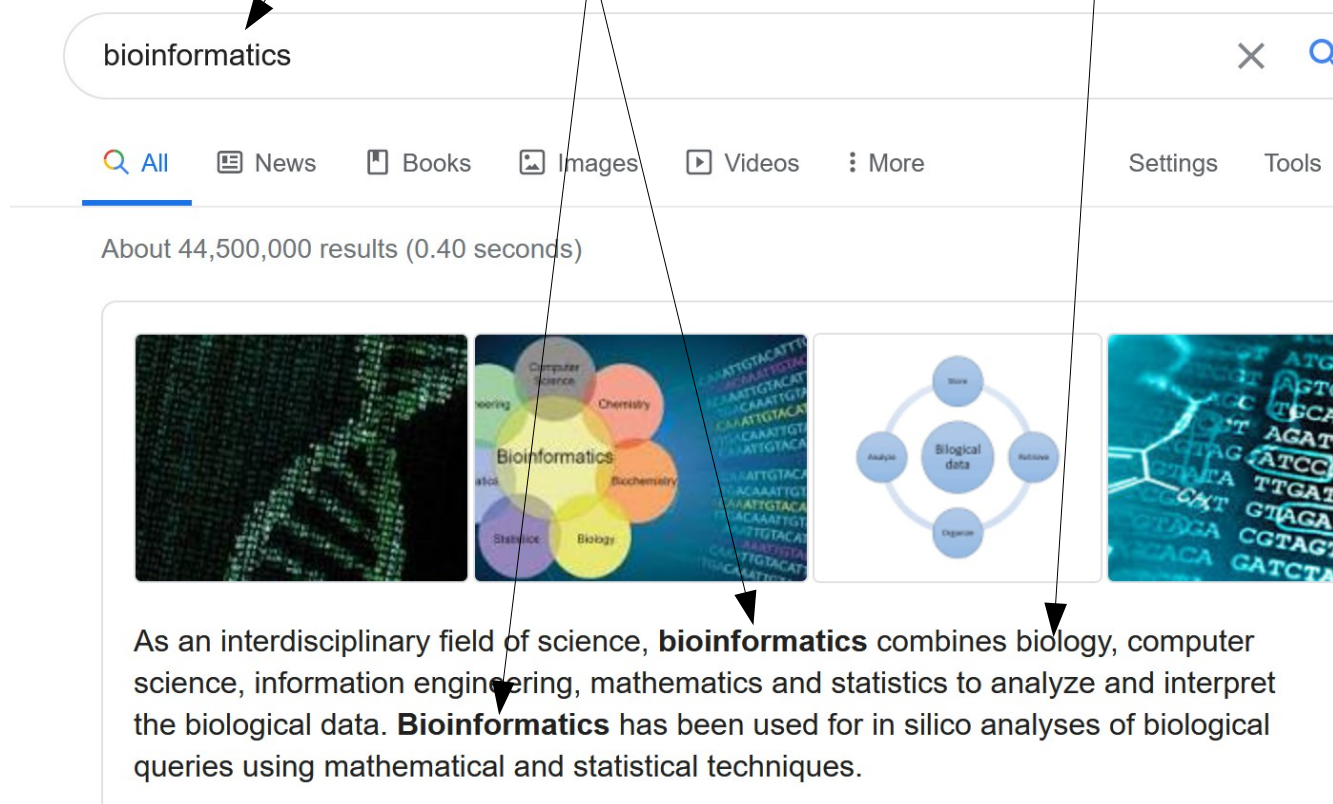


CMSC423: Bioinformatic Algorithms, Databases and Tools

Exact string matching:
Introduction

Exact string matching

- Find a word (**query or pattern**) within a **text**
- Think Google
- The output is a set of **matches** between the pattern and text



The image shows a Google search interface. The search bar contains the text "bioinformatics". Below the search bar, there are tabs for "All", "News", "Books", "Images", "Videos", and "More". The search results show "About 44,500,000 results (0.40 seconds)". Below the results, there are four images: a DNA double helix, a Venn diagram showing the intersection of Computer science, Chemistry, Bioinformatics, Biochemistry, Biology, and Statistics, a circular diagram showing the flow of information from Storage to Analysis, Biological data, and Extraction, and a close-up of a DNA sequence. Arrows point from the search bar to the first image, from the results to the second image, and from the definition text to the third image.

bioinformatics

About 44,500,000 results (0.40 seconds)

As an interdisciplinary field of science, **bioinformatics** combines biology, computer science, information engineering, mathematics and statistics to analyze and interpret the biological data. **Bioinformatics** has been used for in silico analyses of biological queries using mathematical and statistical techniques.

Stop and Think!

How would you write code to find all occurrences of a pattern within a text?

Stop and Think!

How would you solve the pattern matching problem if you were not allowed to use the string functions of your favorite programming language?

Sequence alignment: exact matching

ACAGGTACAGTTCCCTCGACACCTACTACCTAAG
CCTACT
CCTACT
CCTACT
CCTACT

Text
Pattern

```
for i = 0 .. len(Text) {  
  for j = 0 .. len(Pattern) {  
    if (Pattern[j] != Text[i]) go to next i  
  }  
  if we got here pattern matches at i in Text  
}
```

Stop and think?

What is the running time?

```
for i = 0 .. len(Text) {  
  for j = 0 .. len(Pattern) {  
    if (Pattern[j] != Text[i]) go to next i  
  }  
  if we got here pattern matches at i in Text  
}
```

Answer

- Running time = $O(m n)$
 $m = \text{len}(\text{text}); n = \text{len}(\text{pattern})$
- Stop and think: What pattern and text yield the worst-case runtime?
- Stop and think: What is the exact number of comparisons made? (i.e., not approximated by the O-notation)

NEXT: Can we do better?