Using git and mapping reads

Analysis of High Throughput Sequencing Data
2-7-2017

Agenda

- Git
    - Version control
    - Cloud storage
- Read mapping
    - Aligning short reads to a reference genome
    - Naive string searchign
    - Hashing
- Big O notation
    - You will hate this

Local storage and lazy version control are bad

- Keeping things on just your computer is dangerous!
  - What if your dog eats your computer?
- Losing track of versions is dangerous!
  - Sometimes fixing a bug completely breaks everything
  - Sometimes you want to make a big change

Cloud storage and version control

- Dropbox is pretty common
- The “cloud”
    - A bunch of computers that are redundantly backed up
- Version control
    - Systematically keeping track of updates, and keeping backups of all updates

git and github make this easy!

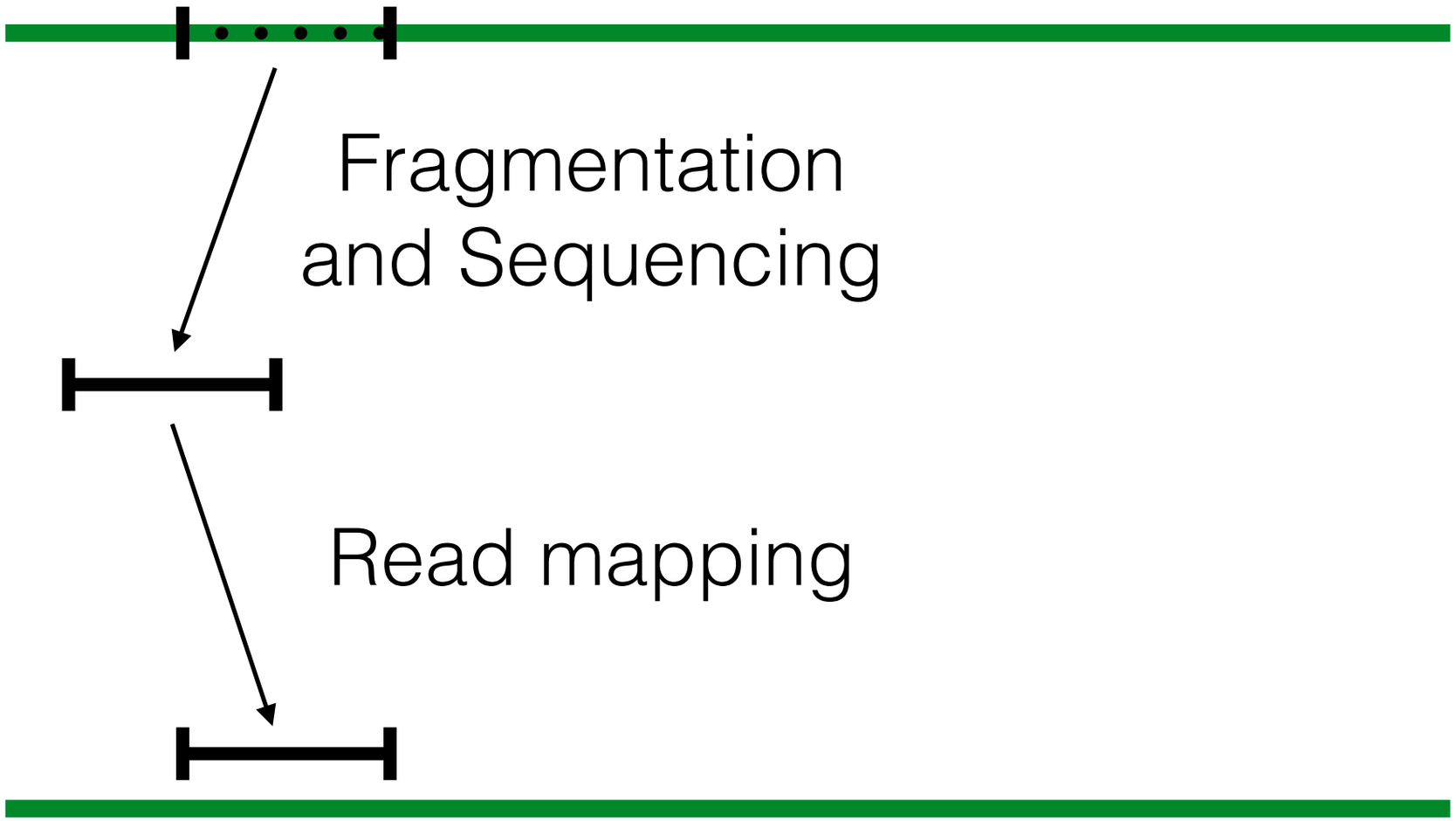
- git is a version control software
  - Repositories
  - Branches
  - Commits
- github is a cloud storage service that uses git

Learning git

- Tutorial in the problems
- Typical workflow
    - git pull
    - do stuff that modifies a file
    - git add file_that_you_changed
    - git commit -m ‘what you changed in the file’
    - git push
- apt-get install on Linux, brew install on MacOS

Read mapping

Can’t see it

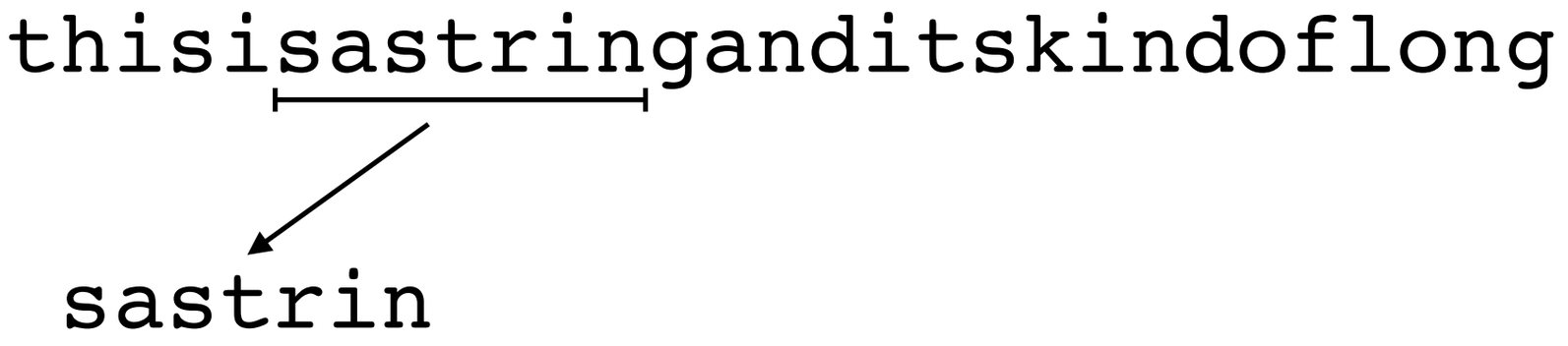


Can see it

Casting as a string searching problem

- Genome is a string of A's, C's, G's and T's
- Read is a substring of the genome
  - Possibly noisy!
- Want to find part of the string where the substring matches

Substring matching



Substring matching

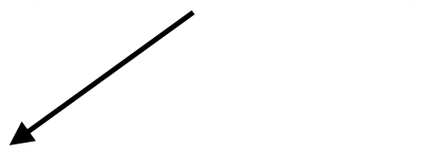
thisisastringanditskindoflong

sastrin

sastrin
thisisastringanditskindoflong

Substring matching

```
thisisastringanditskindoflong
```



```
 sastrin
```

```
 sastrin
thisisastringanditskindoflong
```

Substring matching

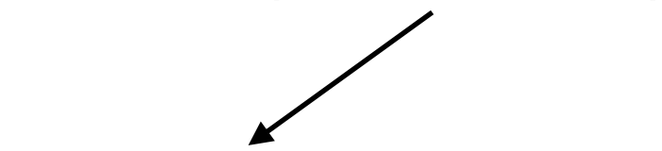
thisisastringanditskindoflong

sastrin

sastrin
thisisastringanditskindoflong

Substring matching

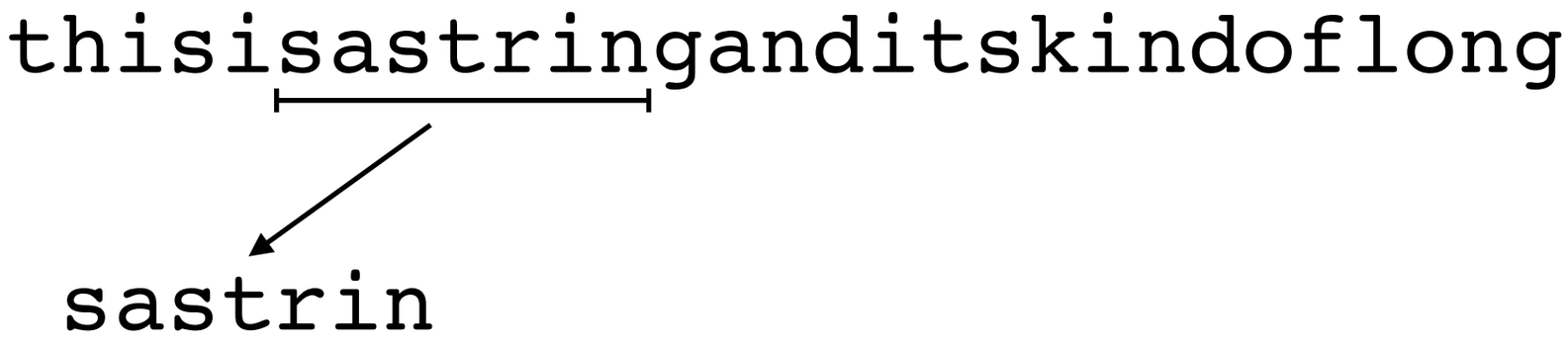
```
thisisastringanditskindoflong
```

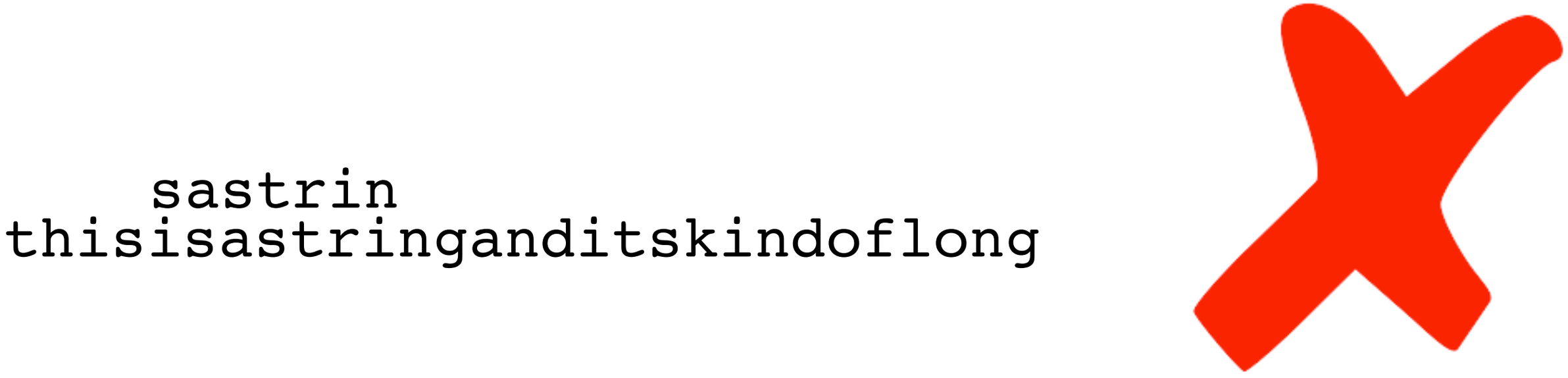


```
sastrin
```

```
   sastrin
thisisastringanditskindoflong
```

Substring matching





Substring matching

```
thisisastringanditskindoflong
```

```
 sastrin
```

```
    sastrin
thisisastringanditskindoflong
```

How long does this take?

- We have a large genome
    - Human genome is 3 billion base pairs
- We have a lot of reads
    - Hundreds of thousands to tens of millions depending on the application
- Want to estimate how long something will take before we start
    - How it will scale as we change the input data

Big O notation

- How the **SLOWEST** part of the algorithm scales with **INPUT** size
- Common examples
  - O(n): linear
  - O(n log n): log-linear
  - O(n^2): quadratic

Complexity of the naive substring searching algorithm

- String of length n
- Substring of length m
- Need to test if the substring matches for all n positions in the string
- For each position in the string need to check if all m positions match
- O(n*m)
- If we do this k times (i.e. we have k reads) then it's O(k*n*m)!

Tips for computing complexity

- Every nested loop that runs for n_i iterations adds a factor of n_i

- For i in range(n1):
    - For j in range(n2):
        - For k in range(n3):
            - ….

- O(n1n2n3)

Hashing provides a far more efficient algorithm

- The slow part is searching the whole genome every single time

- Dictionaries (aka hash tables) provide a fast way to do string matching

    - Looking up a string in a hash table takes O(1) time

- If we has the genome, we can search way faster!

Two step algorithm to search with hashes

- read has length m
- genome has length n
- for i in range(len(n)):
  - add genome[i:(i+m)] as key to dictionary
- for every read:
  - check if the read is in the dictionary

Next time

- The way that this is done in practice

- Actually using bioinformatics software and not writing your own code!