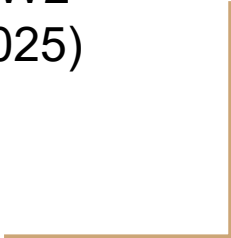




Programming, Problem Solving, and Algorithms

CPSC 203, 2024 W2
(January – April 2025)
Ian M. Mitchell
Lecture 09



Announcements

- Course web page: <https://ubc-cs.github.io/cpsc203/>
 - Weeks 1 – 4 are updated, week 5 is underway.
- Pre-lecture videos: Watch them!
- Assessments:
 - Lab 4 (scavenger hunt) underway: Practice exploring complex data structures
 - POTW 5 due next Sunday
 - Test 2 in CBTF extended until next Monday. (But don't wait that long!)
 - Book for test 3 now!
- Tech stack: Who has got it working? (zoom poll)

Today's Plan...

1. Announcements!
2. Billboard 100 exploration

103 to 203

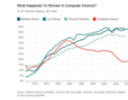
Typical Introductory Data Flow:



.csv file

```
1 if pivot != right:
2     array[right], array[pivot] = ar
3
4     return partition_right(array, left,
5
6 def partition_right(array, left, right,
7     pivot = array[right]
8     mid = left
```

Python problem solution using simple data types and elementary list iteration.



Matplotlib bar or line graph or other summative output illustrating results of computation.

CPSC103++ Data Flow:



Diverse data sources

```
1 if pivot != right:
2     array[right], array[pivot] = ar
3
4     return partition_right(array, left,
5
6 def partition_right(array, left, right,
7     pivot = array[right]
8     mid = left
```

```
1 if pivot != right:
2     array[right], array[pivot] = ar
3
4     return partition_right(array, left,
5
6 def partition_right(array, left, right,
7     pivot = array[right]
8     mid = left
```

data synthesis

```
1 if pivot != right:
2     array[right], array[pivot] = ar
3
4     return partition_right(array, left,
5
6 def partition_right(array, left, right,
7     pivot = array[right]
8     mid = left
```

Analysis and data assembly

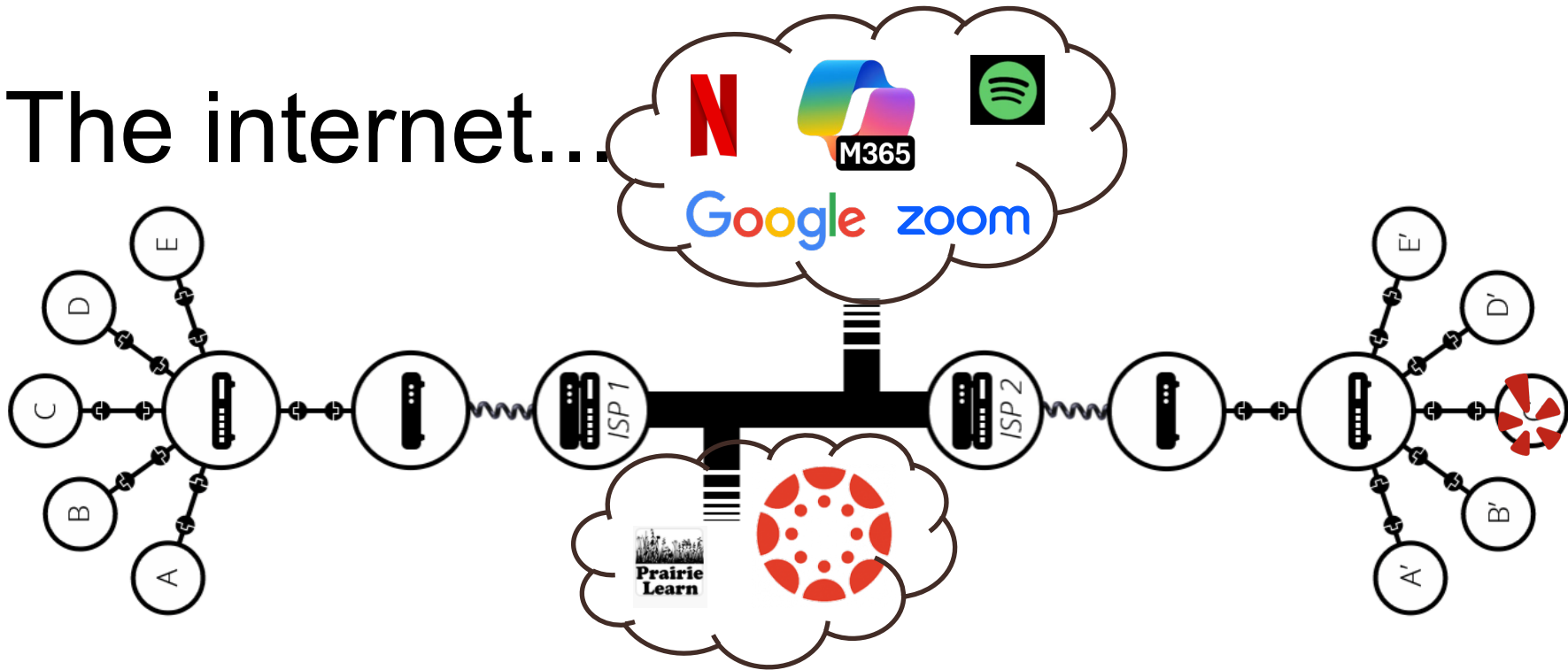


400	50	45
15,290	123,800	3,325
20,091	146,680	3,217
1,065	12,120	0,170
2,173	24,720	0,353
26,124	4700,807	0,257
0,827	68,563	0,378
0,700	0,497	0,304



Diverse outputs

The internet...



- Each type of service on the Internet has a protocol for communicating between client and server (or peers, nodes, caches, ...)
 - For web data, mostly use HTML (hypertext markup language) and related protocols

Anatomy of html...

```
<!DOCTYPE html>
```

```
<html><head><title>The Dormouse's story</title></head>
```

```
<body><p class="title"><b>The Dormouse's story</b></p>
```

```
<p class="story">Once upon a time there were two little sisters.  
Their names were <a href="http://example.com/elsie" class="sister"  
id="link1">Elsie</a>, and <a href="http://example.com/lacie"  
class="sister" id="link2">Lacie</a>, and they lived at the bottom of  
a well.</p>
```

```
</body>
```

```
</html>
```

Billboard Hot 100... page source

- In your browser: View page source (ctrl-u in windows)

```
<div class="chart-list-item piano-content-overlay__gated-item" data-rank="49" data-artist="Taylor Swift" data-title="Lover" data-has-content="true"> <div
class="chart-list-item__first-row chart-list-item__cursor-pointer"> <div class="chart-list-item__position chart-list-item__position--centered"> <div
class="chart-list-item__rank "> 49 </div> <div class="chart-list-item__award"> </div> </div> </div>
class="chart-list-item__image-wrapper"> <div class="chart-list-item__trend-icon"> </div>
```

```
</div>
```

```
<div class="chart-list-item__text-wrapper"> <div class="chart-list-item__text "> <div class="chart-list-item__title">
```

```
<span class="chart-list-item__title-text">
```

Lover

 </div>

<div class="chart-list-item__artist">

Taylor Swift

</div>

<div class="chart-list-item__lyrics ">

Song Lyrics

</div></div></div>

<div class="chart-list-item__chevron-wrapper"><i class="fa fa-chevron-down"></i></div></div>

<div class="chart-list-item__extra-info"><div class="chart-list-item__extra-info-shadow"></div>

<div class="chart-list-item__stats">

<div class="chart-list-item__stats-cell basic-user chart-list-item__stats-cell--first-cell"> <div class="chart-list-item__stats-icon fa fa-arrow-up fa-rotate-45"></div>

<div class="chart-list-item__last-week">23</div>

LAST WEEK </div>

<div class="chart-list-item__stats-cell basic-user "> <div class="chart-list-item__stats-icon fa fa-arrow-up fa-rotate-45"></div>

<div class="chart-list-item__last-week">10</div>

TWO WEEKS AGO</div>

<div class="chart-list-item__stats-cell basic-user "> <div class="chart-list-item__stats-icon fa fa-line-chart"></div>

<div class="chart-list-item__weeks-at-one">10</div>

PEAK POSITION </div>

<div class="chart-list-item__stats-cell basic-user chart-list-item__stats-cell--no-border-right"><div class="chart-list-item__stats-icon fa fa-clock-o"></div>

<div class="chart-list-item__weeks-on-chart">4</div>

WEEKS ON CHART</div></div></div></div>

Beautiful Soup

Reads the html source into a data structure that's easy to query!

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

```
html = simple_get("https://www.billboard.com/charts/hot-100" + '/' + date)
mydivs = html.findAll("div", {"class": "chart-list-item"}) // all the data is here!!

for div in mydivs:
    s = Song(div.attrs['data-title'], div.attrs['data-artist'], int(div.attrs['data-rank']))
```

Still too messy for us! Remedy? <https://github.com/guoguo12/billboard-charts>

demo...

Billboard 100 Demo