Part of this class is about giving technical presentations, and an important part of giving a technical presentation is the preparation of your slides. You need to make your slides readable, uncluttered, and easy to parse.

There is a tendency to be lazy: To simply cut and paste a graphic from your technical paper, or to shove your data into Excel and then paste the ensuing graph into powerpoint. You need to resist this temptation.

These slides are gleaned from a few sources: Mostly, they are student presentation slides from earlier semesters, but some are from other sources. They illustrate what you are not to do. You don't want your slide here.
Problems:
- Font sizes in the graph too small.
- X axis labels are hard to read – 100,000, 200,000, … would be easier.
- X axis labels are nonsensical: 5.01E+06?

Running Time:
- Inserting into the multimap is $O(n \times \log(n))$.
- `CoverHole` is $O(n)$
- So the program is $O(n^2 \times \log(n))$.
- My solution reaches 2 seconds at 700,000

Source: CS494 talk in 2015.
Experiment

- Times are an average over 100 runs.
- Parameters randomly generated between each run
- Ran on Hydra

Problems:
- Font sizes in the graph too small, again.
- X axis labels can use commas or something to make sense of all of the zeroes..
- Why a legend with only one curve?
Source: CS494 talk in 2015.

Problems:
- What are those values on the Y axis?
- Your axes need units. This is not economics!
Problems:
- Why use a log axis to show a linear curve?
- Grid lines look really cluttered and junky.
- Don't plot values if the specifics don't matter. Let the graph curve speak for itself.
- Why not seconds instead of milliseconds?
Problems:
- OMG – what is the X axis?
  Not Linear
  Not Logarithmic
  ??????
Problems:
- OMG – what are those values on the X axis?
- Why a legend for only one curve?

Source: CS494 talk in 2015.
Problems:
- Y axis labels too small and don't have units.
- If you are plotting percentages, and the values can't go over 100%, don't have 120 on the graph.
- Lazy x axis labels (9:00, 10:00, etc)
Running Time, continued

In practice, the algorithm tends to work on most sets.

- Generating new elements will over-power validation most of the time.

```
<table>
<thead>
<tr>
<th>Array size</th>
<th>Milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.325</td>
</tr>
<tr>
<td>4</td>
<td>0.650</td>
</tr>
<tr>
<td>5</td>
<td>0.975</td>
</tr>
<tr>
<td>6</td>
<td>1.300</td>
</tr>
</tbody>
</table>
```

- **Green line:** N-K = 2
- **Blue line:** N-K = 0

Problems:
- Fonts too small, again.
- Unnatural y axis labels, again.
- Why write out “green line” and “blue line”. Just use a legend.
- Center the axis units.

**MacBook Pro**

2.9 Ghz
Algorithm

- Run through the vector of badgers
  - If badger[i] > 200, increment answer
  - Else, add –badger[i] to a multiset
- While there are still badgers:
  - If the weight of the current bin is 0, pull the first badger off the set and delete it
  - If the weight is greater than 200, increment answer and set bin weight to 0
  - Else we call lower_bound() on –(300 - the current weight of the bin)
    - If lower_bound() returns end, we are either out of elements or the elements left are too big
    - Otherwise we add the badger’s weight and delete it from the set

Problem:
- Too cluttered
- No one wants to read algorithms
Problems:
- All of those fonts are way too small. Again.
- What is on the x axis:
  50,000, 100,000, 500,000, 5,000,000?
- Why so much wasted area?

Performance

Hydra machine

$O(n \log(n))$

Time (seconds)

$n$ (size of vectors)

Handles up to $n = 2^{26}$ before inconsistently returning the correct value. Returns $n = 50$ (constraint) in negligible time ($\sim 0$ seconds).
- A clear screen-shot disaster.
- No x axis values or units.
- Y axis units/label colliding with the text.
- Too many words in the first bullet.
- I wrote jgraph; While I'm happy that you're using it, please don't incorporate it into a disaster...

Running Time cont.

- Tests were run with each Address containing 1 Keyword shared by the previous Address, 1 other Keyword shared by the next Address and 50 unique Keywords. The initial Dangerous Keyword was the "next" Keyword on the last Address. Threshold was 1.

- Solution 2 took ~2.5 seconds at 17000 Addresses, then dumped core.
- Fonts too small. Again.
- OMG, look at that x axis – the labels run into each other, and it's not linear or logarithmic...
- Values are unnecessary, and clutter up the graph.
- Ditto the lines in the graph.
- Fonts too small. Again.
- Why is -0.1 on an axis of running times?
- Use 0 on the y axis.
- “2*O(n)” and “O(n-1)” do not make your CS302 professor happy.

- 2*O(n) first recursive function
- O(n-1) second
- O(n^2) overall

Run on 2.5GHz i5-3210m
Why do we need more computers?

- **Google Deep Learning Study**
  - 16000 core, 1000 machine GPU cluster
  - Trained on 10 million 200x200 pixel images
  - Training required 3 days
  - Training dataset size: no larger than what can be trained in 1 week

- **What would they like to do?**
  - ~2 billion photos uploaded to internet per day (2014)
  - Can we train a deep net on one day of image data?
  - Assume 1000x1000 nominal image size, linear scaling (both assumptions are unrealistically optimistic)
  - **Requires 5 ZettaIPS to train in 3 days** (ZettaIPS=10^{21} IPS; ~5 billion modern GPU cores)
  - World doesn’t produce enough power for this!
  - Data is increasing exponentially with time

- **Need >10^{16}-10^{18} instruction-per-second on 1 IC**
  - Less than 10 fJ per instruction energy budget

Q. Le, IEEE ICASSP 2013

- TOO MUCH STUFF ON THE SLIDE.
- Unreadable pictures.

Source: Presentation at “Neuro Inspired Computing Elements, 2017”