Discussion today -- please sit near people!

- About 20-30 minutes of our time today will be small group discussion.
- If you intend to participate in discussion, please sit in the **front half of the orchestra** so you can be surrounded by peers who likewise wish to discuss.
- If you are here to collect your pollev points, and would rather not talk, please sit towards the **back of the orchestra or in the mezzanine.**
- I'd encourage you to get out of your comfort zone, I think it will pay off (:

Poll question to confirm understanding of seating arrangements. No correct answer.

PollEv.com/leahp
text **leahp** to **22333**

Data Structures & Social Implications

Cornell CS 2110
Object-Oriented Programming and Data Structures
Leah Perlmutter | Fall 2025

Agenda for Today

1. Graphs & Path Finding Revisited
2. Assumptions of Algorithms
3. Rethinking the Graph Data Structure
4. Reflection on Learning (if time)
5. Gratitude

Announcements

Announcements

- You didn't have to read for today (no lecture notes)
- Friday 12/12 - exam review session in Phillips 101 at 1-4 pm with TAs!
- Thursday 12/18 - final exam 7-9:30 pm
  - location TBD
  - practice exam and ed post about topics coming soon
- Leah has no office hours on Friday, but can meet by appointment on 12/17 (or a different day), just reach out to schedule
- Seating arrangements
  - sit near the front to participate in discussion
  - sit near the back if you don't want to discuss

Graphs & Path Finding Revisited

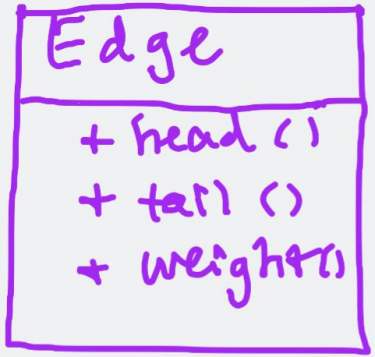
Graph ADT

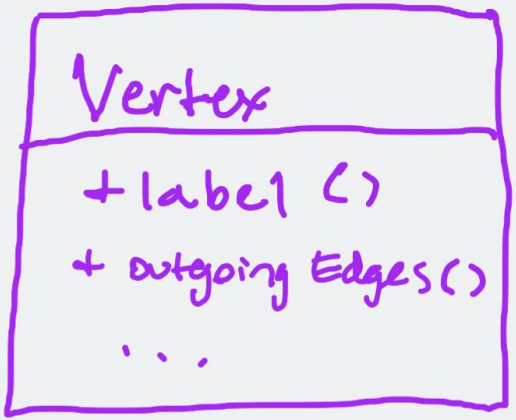
lec21:

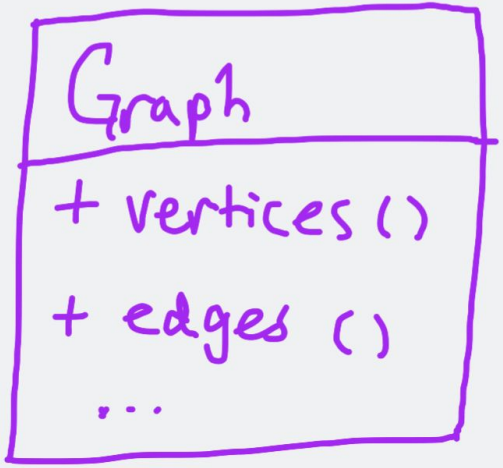
Graph.java

Vertex.java

WeightedEdge.java







Dijkstra's Algorithm (High Level)

Initialize discovered ← {source} frontier ← {source}

while (frontier is not empty) {

v ← frontier vertex with min known distance from s

for each outgoing edge (v, w) {

if w is undiscovered, discover it and add to frontier with best known distance d(s,w) = d(s,v) + weight of edge (v,w)

if w is discovered but we've found a shorter path to it, update its best known distance

}

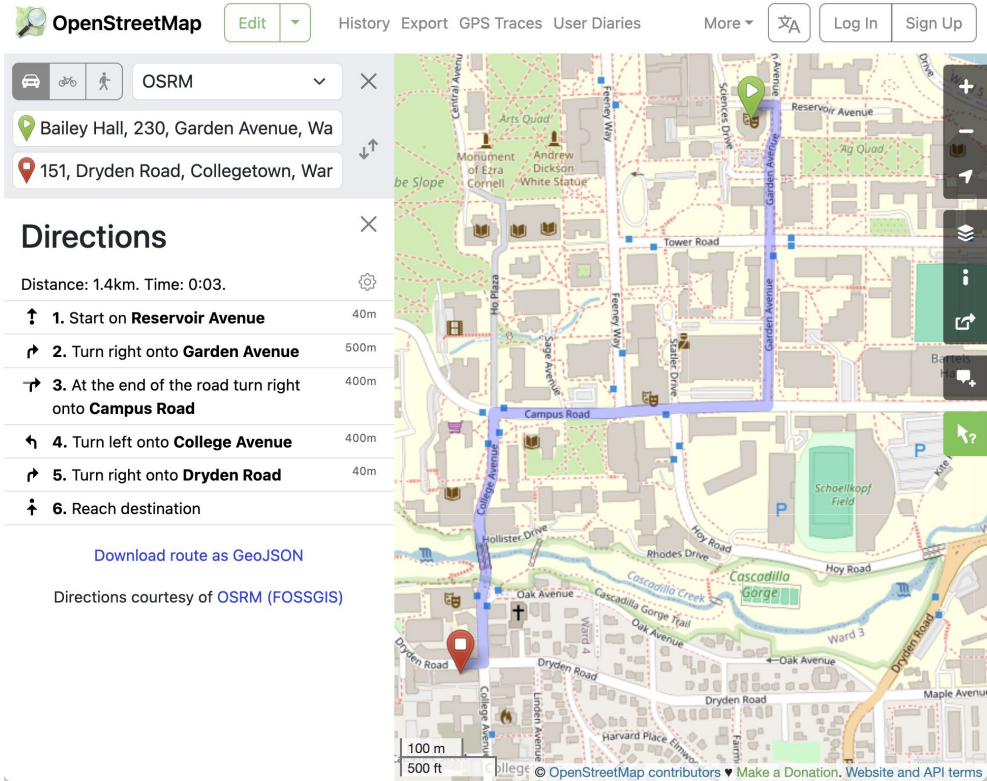
}

```
public static <V extends Vertex<? extends WeightedEdge<V>>>
                    Map<String, PathInfo> dijkstra(V source) {
```

Assumptions of Algorithms

Graph application: Let's plan a route!

- Demo: Open Street Map
  - Think/Pair/Share: Brainstorm assumptions
- Whiteboard: List of assumptions



Assumptions

- driving the speed limit
- not worried about parking
- no traffic or accidents
- Roads open thru winter
- it chooses one of two relatively optimal paths for us
- user prefers efficiency over scenic route
- no public transit
- we are driving

Assumptions (instructor's ideas)

- my car is parked at Bailey (or someone will pick me up)
- going a certain speed (probably the speed limit in the data set)
- the roads on my route will be open when I follow the route
- **going by car**

Rethinking the Graph Data Structure

Activity Overview

- silently identify your group of 3-4
  - while you haven't found your group, raise hand
- silently identify neighboring group of 3-4

When prompted...

- Think silently to yourself
- Discuss with your group of 4
  - designate a speaker to share out
- Share with your neighboring group
- Share out to whole class

Rethinking Path Finding

Old

```
public static <V extends Vertex<? extends WeightedEdge<V>>>
                    Map<String, PathInfo> dijkstra(V source) {
```

New: Supports different modes of transport. Below is one possible way.

```
public static <V extends Vertex<? extends WeightedEdge<V>>>
   Map<String, PathInfo> dijkstra(V source, __ modeOfTransport) {
```

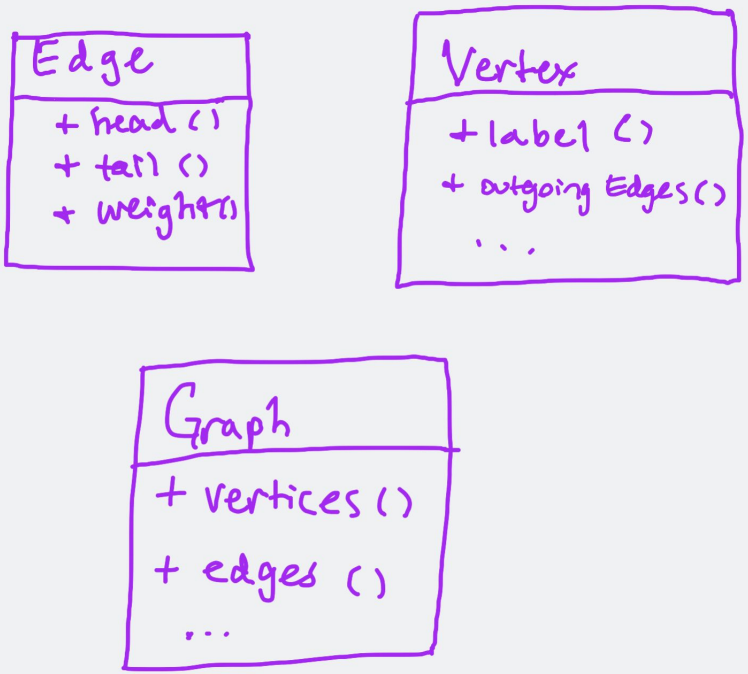
Redesigning the Graph Data Structure

How might we represent a graph data structure so it supports multiple modalities of transport to the same locations?

How will the **data structures** change?

How will the **path finding algorithm** change?

If you finish early, develop ANOTHER way!




```
public static <V extends Vertex<? extends WeightedEdge<V>>>
   Map<String, PathInfo> dijkstra(V source, __ modeOfTransport) {
```

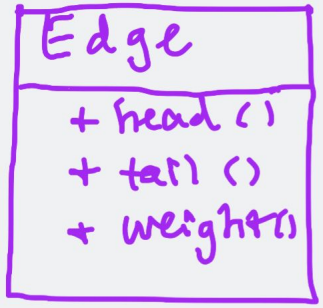
Graph Redesign Ideas

- different graph for each mode of transport
- Edge types for each mode of transport
- Edges say which type(s) of transport they support
- Edges for a given mode of transport store the time to traverse that edge using that mode

Idea 1: Multiple graphs

- Data structures
  - no class redesign, but the application has multiple graphs
    - bikeGraph, carGraph, walkGraph, wheelchairGraph
  - each with its own vertices and edges
- Path finding
  - First identify the correct graph based on the modality
  - Next, path finding as before on that graph

Idea 2: Edges have multiple weight fields

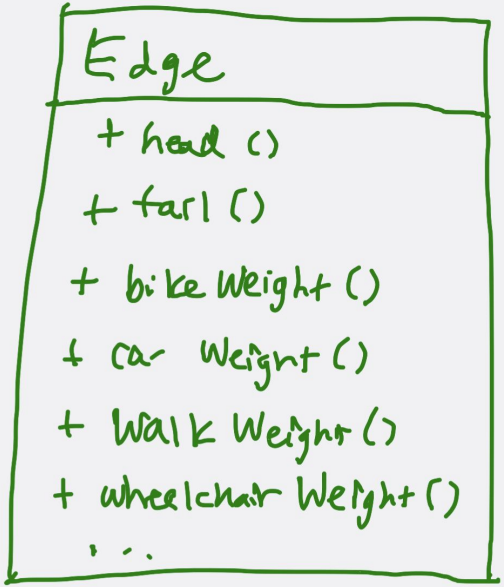


Data structures

- Add fields and methods to Edge

Path finding

- Within the path finding code, refer back to the mode of transport and call the correct method of edge to get the correct weight



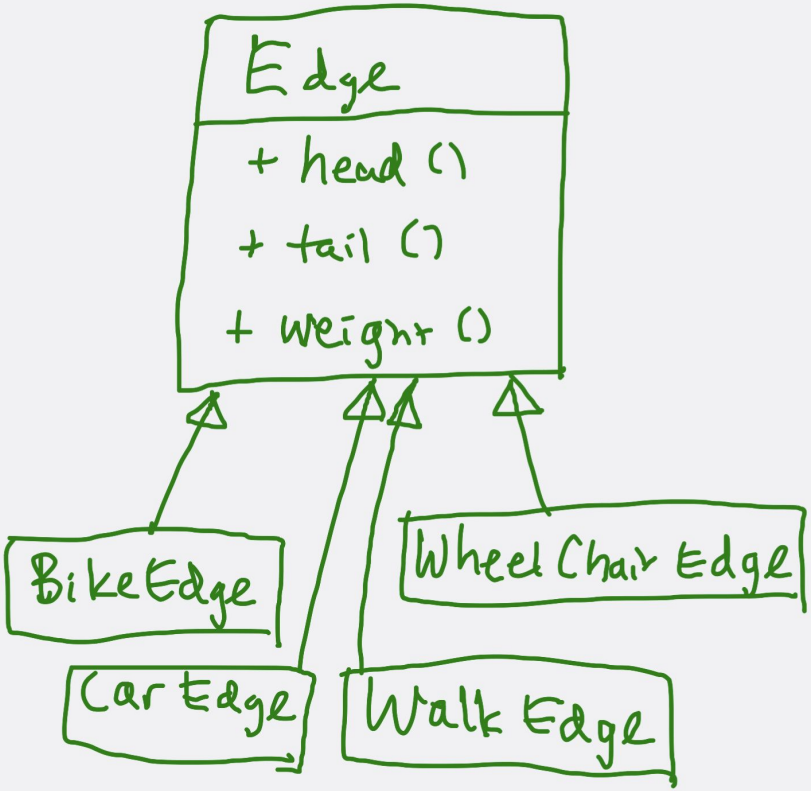
Idea 3: Different Types of Edges

Data structures

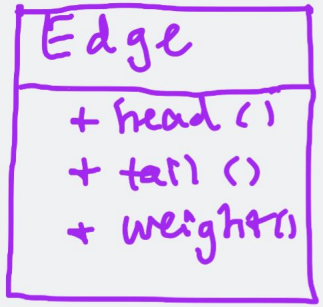
- Vertices can have multiple edges between them. Each edge has a type corresponding to the modality.

Path finding

- After getting the outgoing edges of a vertex, filter them based on the modality we are currently using



Idea 4: Edges have a map of weights

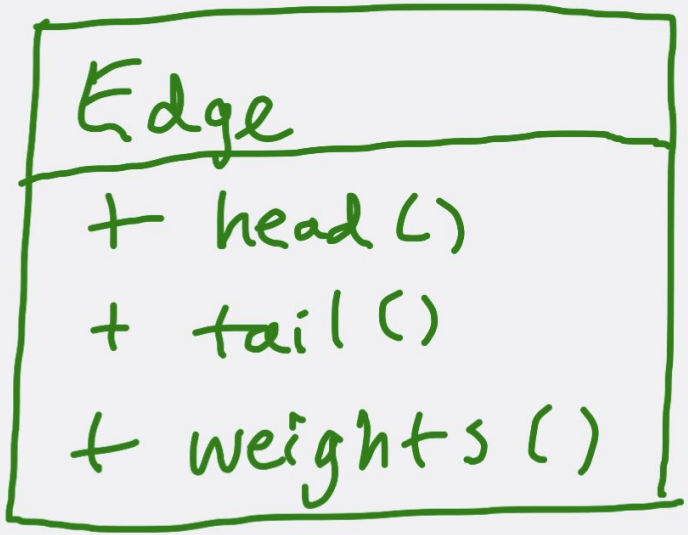


Data structures

- give Edge a weights method that returns Map<Modality, Weight>

Path finding

- Within the path finding code, refer back to the mode of transport and find the correct weight in the map of weights



Pros and cons (if time)

- Idea 1: Multiple graphs
- Idea 2: Edges have multiple weight fields
- Idea 3: Different types of edges
- Idea 4: Edges have a map of weights

Conclusion

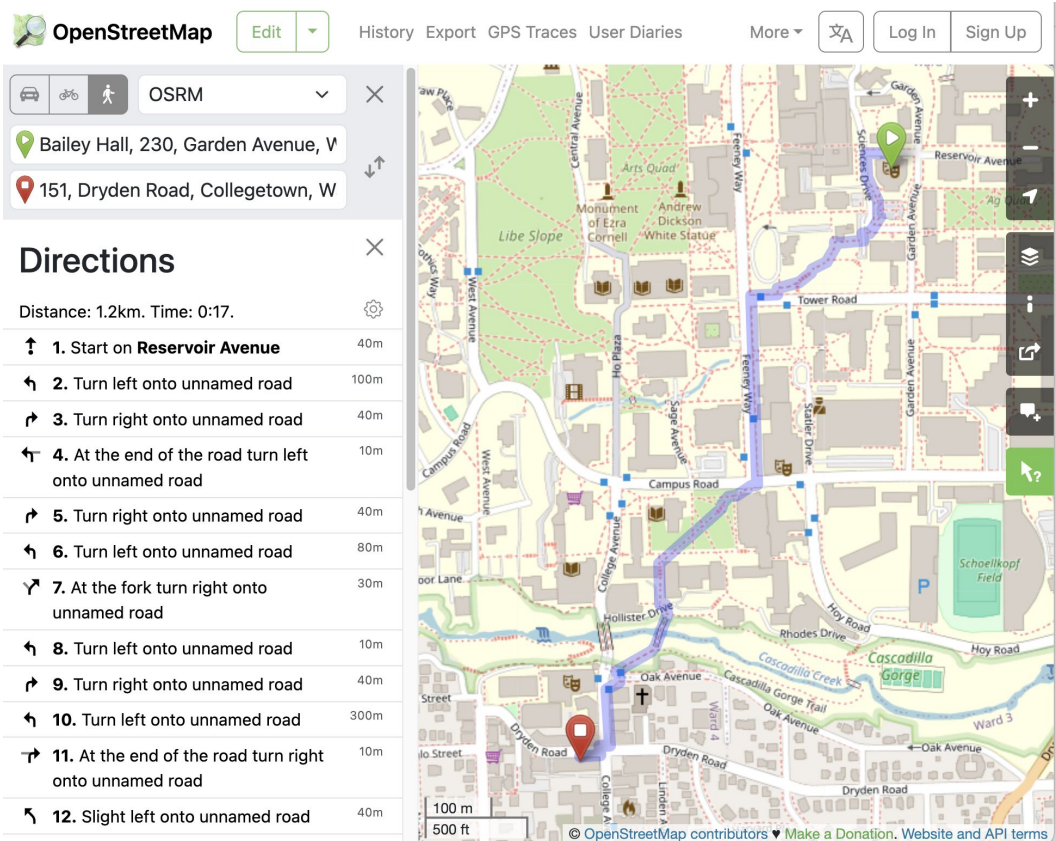
- There's no one answer that is always correct.
- In the real world, we need to be creative with our design of data structures and consider the trade-offs comprehensively based on the specific context.

Assumptions of Algorithms

(continued)

Assumptions of Algorithms (continued)

- Demo: Open Street Map
  - Think/pair/share/ share: Brainstorm more assumptions
- Whiteboard: Assumptions

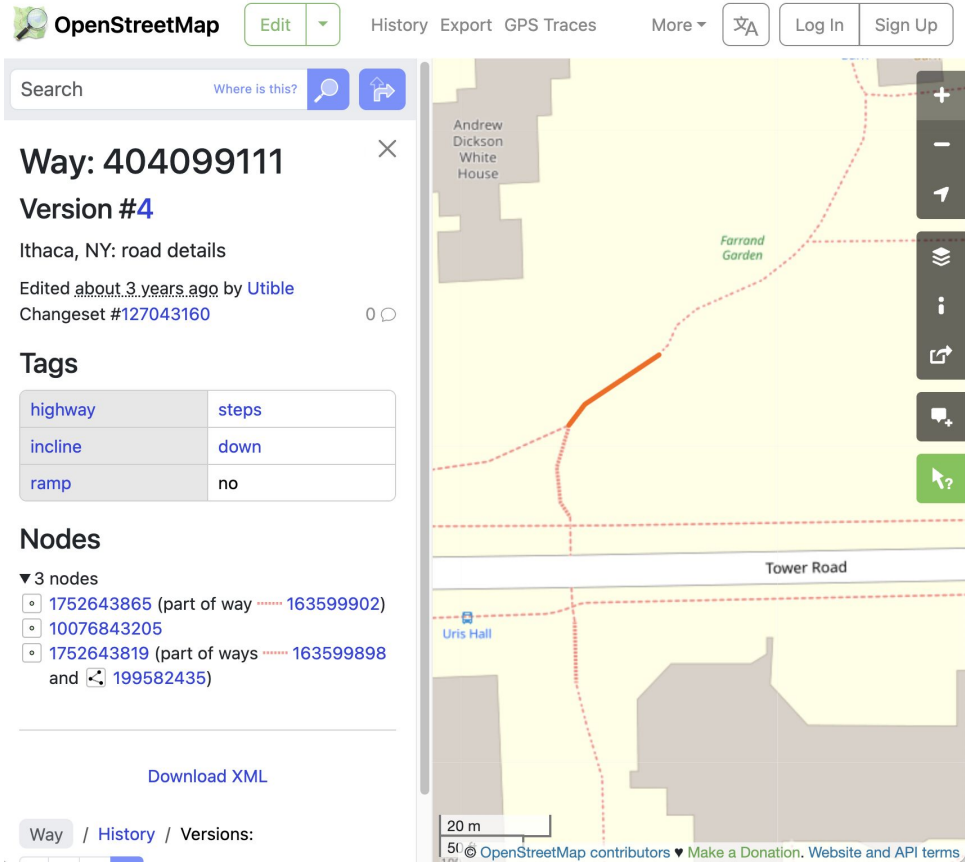


Assumptions (walking directions)

- can't walk thru buildings
- set walking speed
- can go down stairs (!!)
- walking alone and not with a companion who is slower
- trails are maintained (in winter)
- doesn't know about shortcuts
- Leaving Bailey out a certain door
- user can traverse difficult terrain

Assumptions (instructor's ideas)

- walking at a certain speed
- I prefer efficiency (what if I prefer to walk down the gorge?)
- the roads on my route will be traversible when I follow the route
- **I can climb stairs**



What if I'm rolling?

- Wheelchair
- Stroller
- Knee Rover

This photo was taken by Chona Kasinger and published under Creative Commons attribution licensing as part of the Disabled And Here project

Maybe the walk button is only for "Pedestrians"?

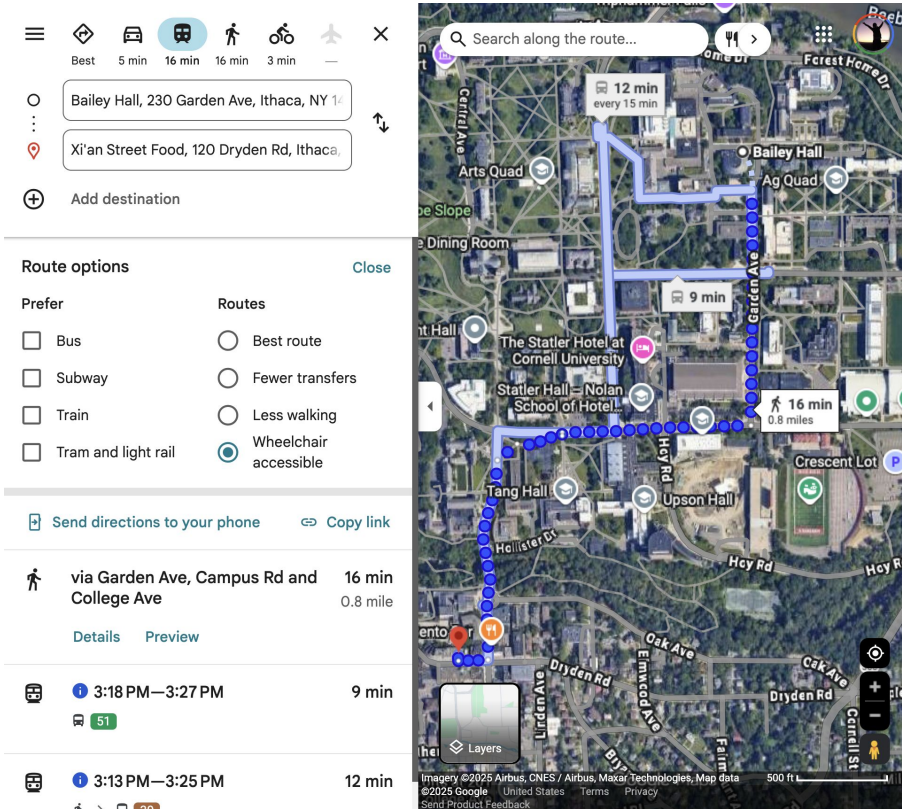
NY State Law Chapter 71, Title 1, Article 1, Section 130:

- **Pedestrian**. Any person afoot or in a wheelchair. [source]

This photo was taken by Chona Kasinger and published under Creative Commons attribution licensing as part of the Disabled And Here project

Assumptions of Algorithms (continued)

- Demo: another map application
- Think/pair/share/share: **How can developers do better? (Process Oriented) What strategies can they incorporate into software development processes in order to be more inclusive?**
- Whiteboard: Your Ideas



How can we do better?

- Beta testing
- Constant communication with communities we want to serve
- Make code extensible

How can developers do better? (instructor ideas)

- Hire wheelchair users to design, build, and test the system
- Perspective taking -- developers place themselves in the shoes of all different kinds of users
- **Get better data**

Collecting map data for accessibility

- Project sidewalk
- Images on the slide are from this paper

Take Away: Accessibility Matters!

To make applications accessible, designers and developers must:

- Take the perspectives of disabled users, ideally by having disabled designers and developers on the team
- Creatively rethink data structures and algorithms
- Thoughtfully collect the right data as the foundation for the data structures being used

Reflection on Learning (if time)

Reflection on learning

Write down your most important take-aways from this class. Make sure to include some content-oriented things and some process-oriented things.

Reflection on learning

Poll: What is one thing from 2110 that you'd like to investigate further? (The first answers to be submitted will be shown on the screen.)

PollEv.com/leahp
text **leahp** to **22333**

Gratitude

Gratitude

Poll: What are you grateful for in CS 2110?

(instructor will share first as a model)

(The first answers to be submitted to the poll will be shown on the screen.)

PollEv.com/leahp
text **leahp** to **22333**

No right or wrong answer.

Wrap Up

See also

- Project Sidewalk
  - Project Sidewalk homepage
  - Visualization of project sidewalk data (external)
- OpenStreetMap - free and open data set that models map data
  - Elements - overview of the components of the data model of OpenStreetMap
  - Map Features - all the kinds of physical things that can be represented in the data model of OpenStreetMap

Announcements

- Friday 12/12 - exam review session in Phillips 101 at 1-4 pm with TAs!
- Thursday 12/18 - final exam 7-9:30 pm
  - location TBD
  - practice exam and ed post about topics coming soon
- Leah has no office hours on Friday, but can meet by appointment on 12/17 (or a different day), just reach out to schedule