Algorithms and Data Structures

COMP261
Tutorial Week 2

Yi Mei

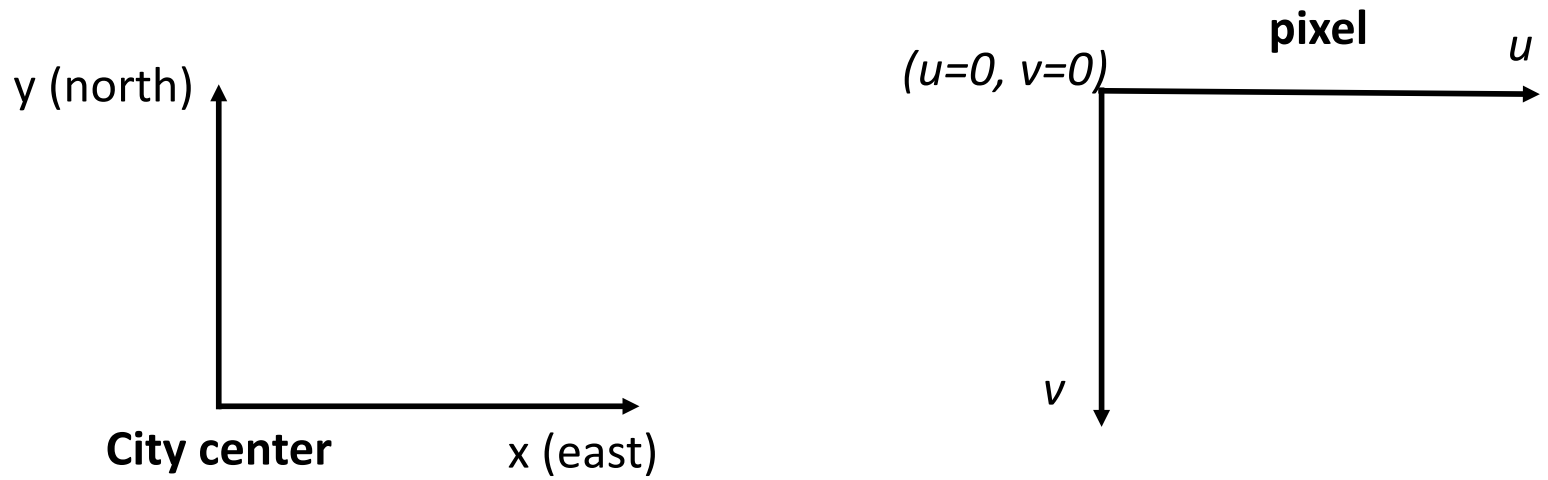
yi.mei@ecs.vuw.ac.nz

Outline

- Graph display
  - Coordinates
  - Redraw under shift and zoom in/out
- Trie
  - Add
  - Get
  - GetAll

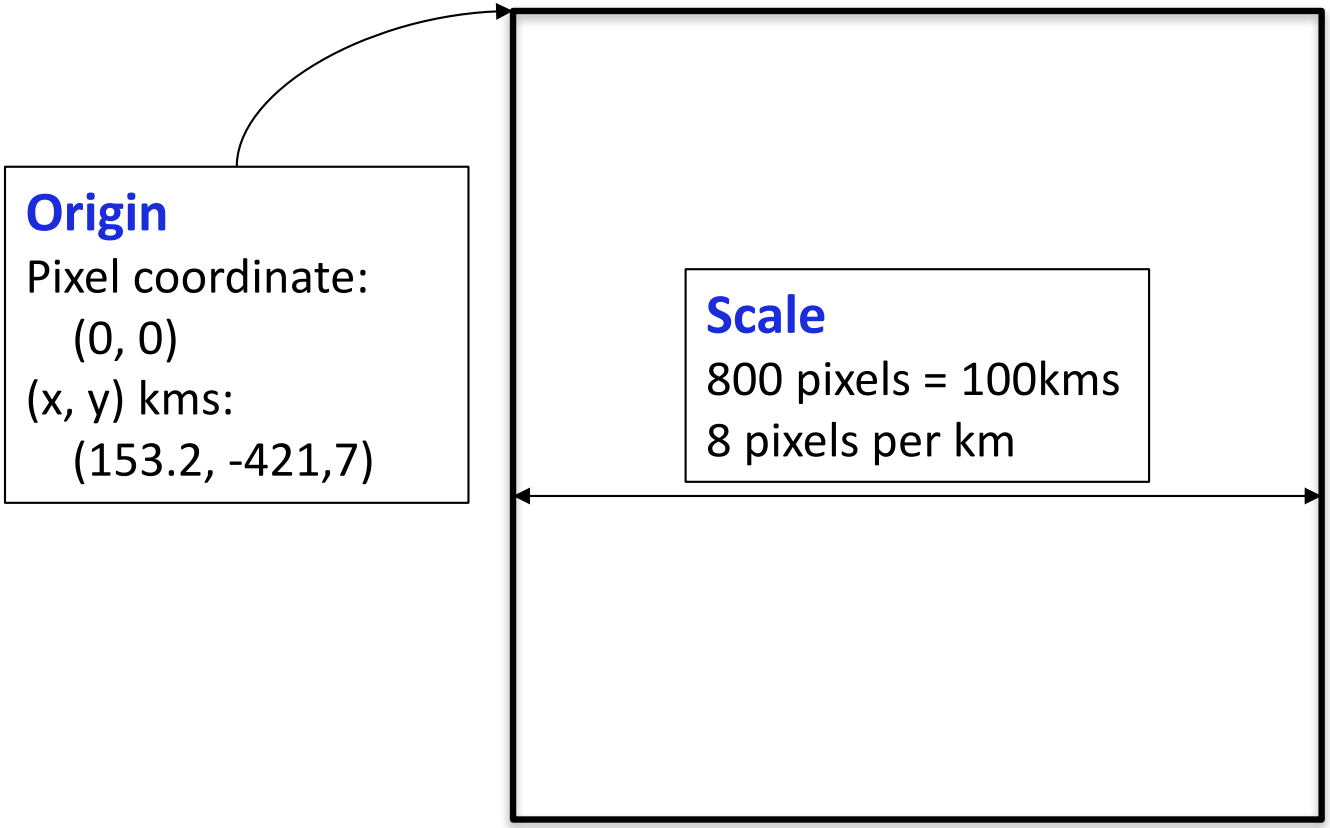
Coordinate Systems for Location

- Coordinate systems to represent locations of nodes
  - Absolute (***fixed***): latitude/longitude
  - Relative (***fixed***): x kms to the east, y kms to the north of the city center
    - Assume a flat map (the earth is a globe actually), but OK
    - Will be useful for shortest path finding
  - Pixel coordinate: for display
    - ***Dynamic***, depends on the area to display



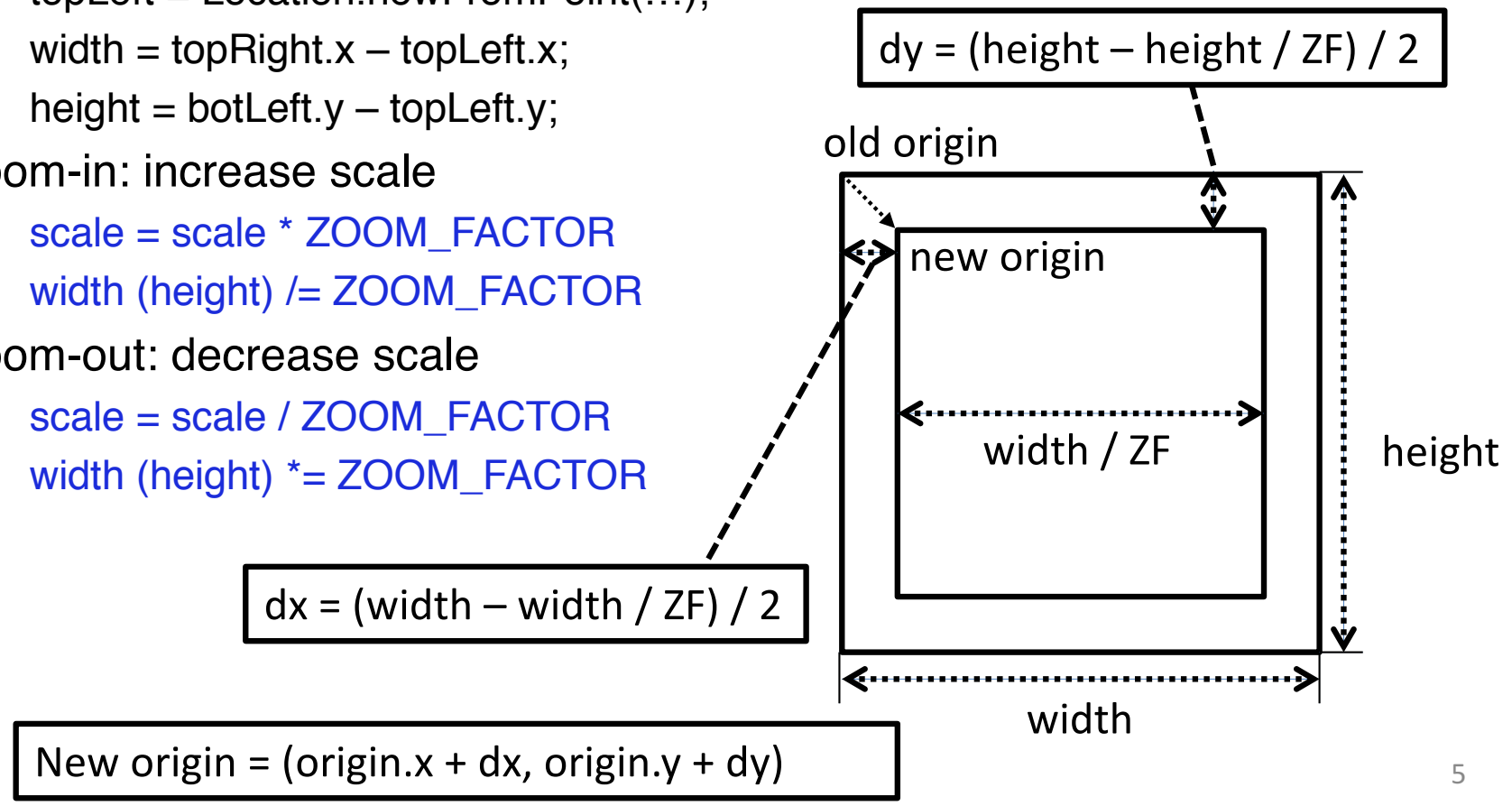
Display Graph

- Size of display screen: number of pixels, e.g. 800 x 800
- The displayed area
  - Origin: e.g. the top-left location
  - Scale: how large the area is covered by the pixels?
    - Number of pixels per kms



Adjust Display Under Movements

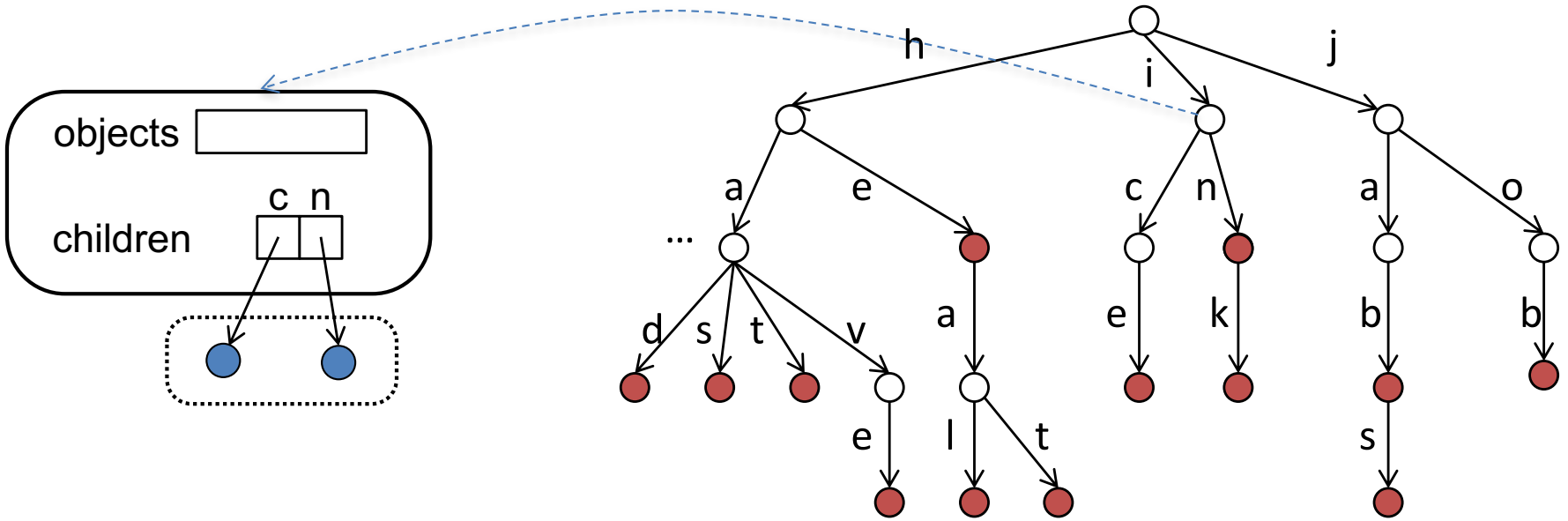
- Shift the displayed area: shift the origin
  - `orig.x = orig.x + dx, orig.y = orig.y + dy;`
- Zoom in/out the displayed area around the current center
  - Change both scale and origin: ZOOM_FACTOR > 1
  - Calculate width and height in kms (using topLeft, topRight, botLeft, botRight)
    - topLeft = Location.newFromPoint(…);
    - width = topRight.x – topLeft.x;
    - height = botLeft.y – topLeft.y;
  - Zoom-in: increase scale
    - scale = scale * ZOOM_FACTOR
    - width (height) /= ZOOM_FACTOR
  - Zoom-out: decrease scale
    - scale = scale / ZOOM_FACTOR
    - width (height) *= ZOOM_FACTOR



Trie

- A trie (prefix tree): an ordered tree data structure
- Each node contains
  - associated objects
  - a set of child nodes (each corresponding to a character)

```
Class TrieNode {
    List<Object> objects;
    HashMap<Character, TrieNode> children;
}
```



Add and Get in a Trie

```
public void add(char[] word, Object obj) {
    Set node to the root of the trie;

    for (c : word) {
        if (node's children do not contain c)
            create a new child of node, connecting to node via c
        move node to the child corresponding to c;
    }

    add obj into node.objects;
}
```

```
public List<Object> get(char[] word) {
    Set node to the root of the trie;

    for (c : word) {
        if (node's children do not contain c)
            return null;
        move node to the child corresponding to c;
    }

    return node.objects;
}
```

Get All in a Trie

```
public List<Object> getAll(char[] prefix) {
    List<Object> results = new ArrayList<Object>();
    Set node to the root of the trie;

    for (c : prefix) {
        if (node's children do not contain c)
            return null;
        move node to the child corresponding to c;
    }

    getAllFrom(node, results);
    return results;
}
```

```
public void getAllFrom(TrieNode node, List<Object> results) {
    add node.objects into results;

    for (each child of node)
        getAllFrom(child, results);
}
```

Example

- Create a trie with these words

had	ice
has	iced
hat	in
hats	ink
have	irk
he	iron
heal	jab
health	jabs
heat	job

- Then
  - `add("inch", o1)`
  - `get("ink")`
  - `getAll("he")`