CSE 331
Software Design & Implementation

Spring 2025

Section 9 – HW 9 Prep

Administrivia

- HW 9 released tonight, due 6/6 (but try to do it earlier because the code is *really*^{2} hard)

- FINAL EXAM 6/10 12:30 pm - 2:20 pm in Kane 130
  - (next section will be primarily exam prep)

Prep for HW 9 / Locations

In HW 9, we will be working with Location objects again:

$$\textbf{type}\ \text{Location} := \{x : \mathbb{R},\ y : \mathbb{R}\}$$

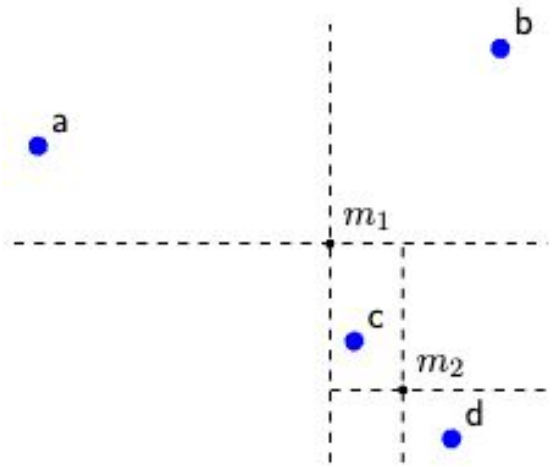
We will be interested in finding all Location points within a given rectangle.

To keep track, we will use trees that recursively split areas on the map until each region only contains a single point

HW 9 Prep & Tips

- HW 9 will be adding functionality to Campus Maps from HW 3
- Section slides and ws designed to introduce you to HW 9 concepts & data structures
- Please take a look at the starter code BEFORE starting the assignment
  - This will allow you to better understand the specifications of the assignments
  - We also give you many helper functions and definitions so this will also prevent you from reinventing the wheel (i.e. calculating the length of a list)

LocTree



- To represent the points in the image on the left, we would start by keeping track of one rectangle that represents the entire map
- We would then split that rectangle into 4 rectangles at m_1
- And then split the lower right rectangles into 4 rectangles at m_2.

m_1 is the average location of all 4 points, m_2 is the average location of points c and d. We call this average point the “centroid” !

Tree Type

This is the inductive type we will use to represent the location tree:

```
type LocTree := empty
              | single(loc : Location)
              | split(at : Location, nw : LocTree, ne : LocTree, sw : LocTree, se : LocTree)
```

Note that nw, ne, sw, and se are lowercase! Uppercase relates to a different function.

Regions

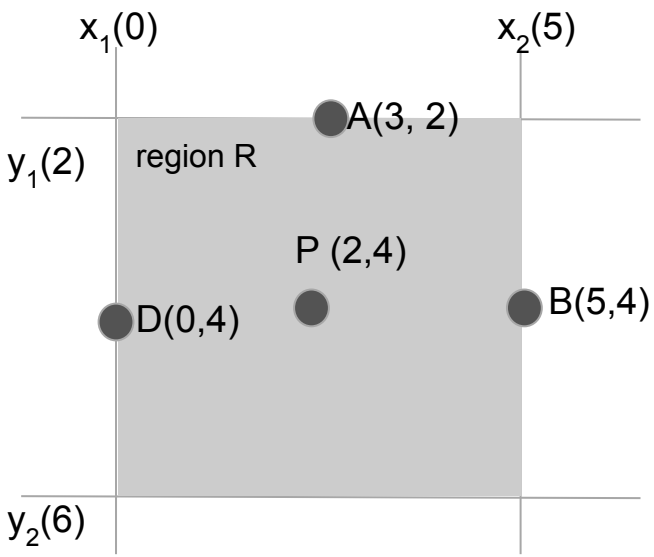
We will keep track of the corners for regions we are interested in. The Region object will be as follows:

$$\textbf{type}\ \text{Region} := \{\text{x1}: \mathbb{R},\ \text{x2}: \mathbb{R},\ \text{y1}: \mathbb{R},\ \text{y2}: \mathbb{R}\}$$

The region *R* contains the location ℓ if and only if *R.x1* <= ℓ.x <= *R.x2* and *R.y1* <= ℓ.*y* <= *R.y2.* Thus, the following region includes every point in the plane:

$$\text{EVERYWHERE} := \{\text{x1}: -\infty,\ \text{x2}: \infty,\ \text{y1}: -\infty,\ \text{y2}: \infty\}$$

Contains point



$R.x_1 = 0$
$R.x_2 = 5$

$R.y_1 = 2$
$R.y_2 = 6$

So region R = {x1: 0, x2: 5, y1: 2, y2: 6}

Looking only at x and y, we can see for point P that:

For the x values: 0 <= 2 <= 5
For the y values: 2 <= 4 <= 6
So we know the region R contains the point P

Meaning, if the point is on any border, then it is also in the region.

Location, Region, LocTree

- Go over how NW(m, R), NE(m, R), etc. work (picture shown below for reference of what the definition is)

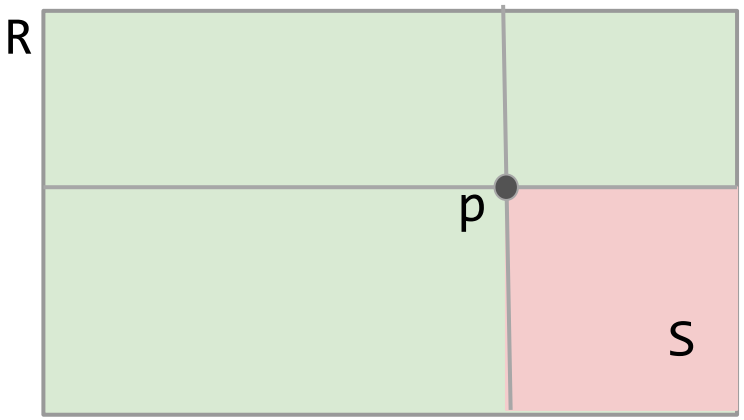
$$\mathsf{NW}(m, R) := \{\mathsf{x1}: R.\mathsf{x1},\ \mathsf{x2}: m.x,\ \mathsf{y1}: R.\mathsf{y1},\ \mathsf{y2}: m.y\}$$
$$\mathsf{NE}(m, R) := \{\mathsf{x1}: m.x,\ \mathsf{x2}: R.\mathsf{x2},\ \mathsf{y1}: R.\mathsf{y1},\ \mathsf{y2}: m.y\}$$
$$\mathsf{SW}(m, R) := \{\mathsf{x1}: R.\mathsf{x1},\ \mathsf{x2}: m.x,\ \mathsf{y1}: m.y,\ \mathsf{y2}: R.\mathsf{y2}\}$$
$$\mathsf{SE}(m, R) := \{\mathsf{x1}: m.x,\ \mathsf{x2}: R.\mathsf{x2},\ \mathsf{y1}: m.y,\ \mathsf{y2}: R.\mathsf{y2}\}$$

Region functions

Functions NW… SW : (Location, Region) → Region

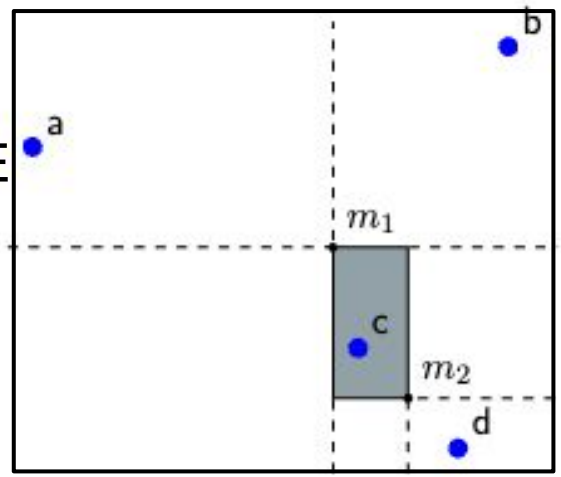
- take a location point and a region rectangle
- return the subregion of parameter that is split at the location point and the in indicated direction

EX: SE(p, R) = S



NW(m2, SE(m1, EVERYWHERE))

Region: EVERYWHERE



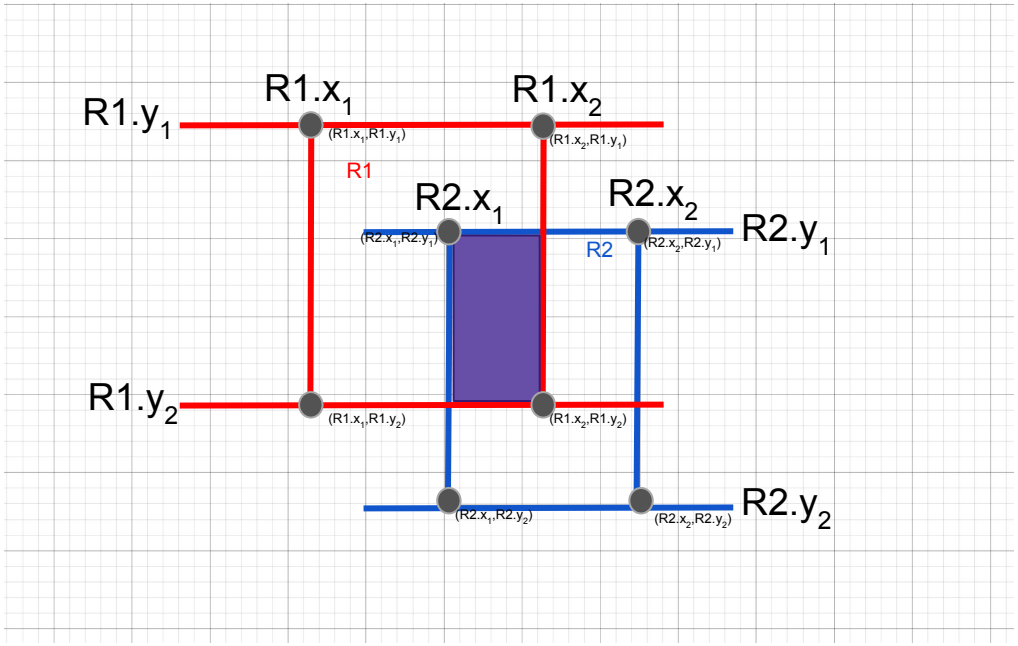
Question 1

We have a function overlap : (Region, Region) → Bool that returns true if two regions overlap. Two regions overlap if they share any area in common. Write an expression that returns true if 2 regions R_1 and R_2 overlap.

type Region := {x1 : ℝ, x2 : ℝ, y1 : ℝ, y2 : ℝ}

Draw out an example!

(R_1.x1 <= R_2.x2) and (R_1.x2 >= R_2.x1) and (R_1.y2 >= R_2.y1) and (R_1.y1 <= R_2.y2)

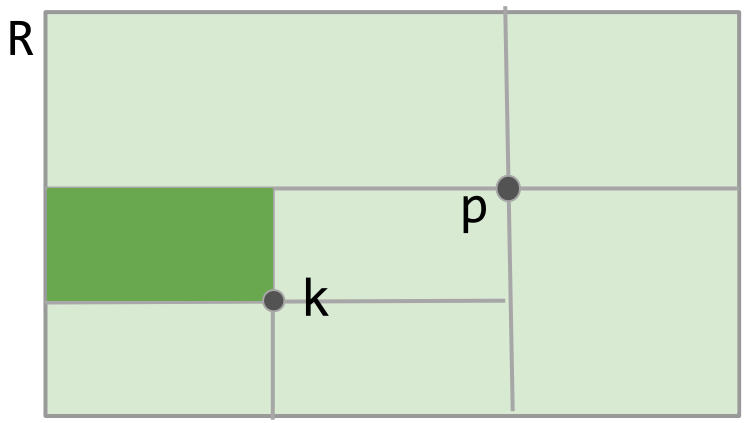


Quick note:

- Remember, x grows to the right, y grows downwards
- Also, this is an arbitrary example. Overlap can happen from any side, the rules here still apply though.

Question …

How can we make a function call that would return us the shaded area?



NW(k, SW(p, R))

- We can see that the area we want is in the SW region of point P
- We can then see that it's in the NW portion of point k
- Putting this all together we get the SW portion at point p of region R, then the NW portion at point k, to get to the highlighted region.

Question 2a

(a) Define a function FindAll : (LocTree, Region) → List⟨Location⟩ that returns a list of all locations in the LocTree that fall within the given region. The order of the locations in the list does not matter but there should be no duplicate entries. Assume we have the following function contains : (Region, Location) → Bool that returns true if the location is within the region.

FindAll(empty, R) := []

FindAll(single(s), R) := [s] if contains(R, s)

FindAll(single(s), R) := [] if not contains(R,s)

FindAll(split(m, nw, ne, sw, se), R) :=
FindAll(nw, R) ++ FindAll(ne, R) ++ FindAll(sw, R) ++ FindAll(se, R)

Question 2b

(b) Improve the algorithm by excluding any quadrants that do not overlap with the region passed in. This will avoid traversing any subtrees that cannot contain any locations in the region. We can do this by defining an improved function fa : $(\text{LocTree}, \text{Region}, \text{Region}) \rightarrow \text{List}\langle\text{Location}\rangle$ that takes in an additional Region parameter. The second Region parameter is the region that we are looking for locations in. The first Region parameter is a region containing all the points in the tree. It will use this region parameter to avoid recursing into quadrants of split nodes when they cannot contain a location in the second Region parameter.

Question 2b

```
fa(empty, S, R)                          := []
fa(single(s), S, R)                      := [s]    if contains(R, s)
fa(single(s), S, R)                      := []     if not contains(R, s)
fa(split(m, nw, ne, sw, se), S, R) := []     if not overlap(R, S)
fa(split(m, nw, ne, sw, se), S, R) :=
    fa(nw, NW(m, S), R) ++ fa(ne, NE(m, S), R) ++ fa(sw, SW(m, S), R)
        ++ fa(se, SE(m, S), R)                   if overlap(R, S)
```

- We can see that our split(m, nw, ne, sw, se) is inside of region S, so if S shares no overlap with region R, then region S contains none of the points in R. Since we only want the points in R, we ignore this region S, and therefore this split node as well.

Question 2c

(c) Prove that if region S contains all locations in the tree T, then $\mathsf{fa}(T, S, R) = \mathsf{FindAll}(T, R)$. Your proof should be by structural induction on T.

Feel free to use the fact that, if S contains all the locations in $\mathsf{split}(m, \mathsf{nw}, \mathsf{ne}, \mathsf{sw}, \mathsf{se})$, then $\mathsf{NW}(m, S)$ contains all the locations in nw and likewise for ne, sw, and se. (This follows from the representation invariant for split nodes and the definitions of these functions.)

$$
\begin{aligned}
\mathsf{FindAll}(\mathsf{empty}, R) &:= [] \\
\mathsf{FindAll}(\mathsf{single}(\mathsf{s}), R) &:= [\mathsf{s}] && \text{if } \mathsf{contains}(R, s) \\
\mathsf{FindAll}(\mathsf{single}(\mathsf{s}), R) &:= [] && \text{if not } \mathsf{contains}(R, s) \\
\mathsf{FindAll}(\mathsf{split}(m, \mathsf{nw}, \mathsf{ne}, \mathsf{sw}, \mathsf{se}), R) &:= \mathsf{FindAll}(\mathsf{nw}, \mathsf{R}) \\
&\quad +\!\!+ \mathsf{FindAll}(\mathsf{ne}, \mathsf{R}) \\
&\quad +\!\!+ \mathsf{FindAll}(\mathsf{sw}, \mathsf{R}) \\
&\quad +\!\!+ \mathsf{FindAll}(\mathsf{se}, \mathsf{R})
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{fa}(\mathsf{empty}, S, R) &:= [] \\
\mathsf{fa}(\mathsf{single}(\mathsf{s}), S, R) &:= [\mathsf{s}] && \text{if } \mathsf{contains}(R, s) \\
\mathsf{fa}(\mathsf{single}(\mathsf{s}), S, R) &:= [] && \text{if not } \mathsf{contains}(R, s) \\
\mathsf{fa}(\mathsf{split}(m, \mathsf{nw}, \mathsf{ne}, \mathsf{sw}, \mathsf{se}), S, R) &:= [] && \text{if not } \mathsf{overlap}(S, R) \\
\mathsf{fa}(\mathsf{split}(m, \mathsf{nw}, \mathsf{ne}, \mathsf{sw}, \mathsf{se}), S, R) &:= \mathsf{fa}(\mathsf{nw}, \mathsf{NW}(m, S), R) \\
&\quad +\!\!+ \mathsf{fa}(\mathsf{ne}, \mathsf{NE}(m, S), R) \\
&\quad +\!\!+ \mathsf{fa}(\mathsf{sw}, \mathsf{SW}(m, S), R) \\
&\quad +\!\!+ \mathsf{fa}(\mathsf{se}, \mathsf{SE}(m, S), R)
\end{aligned}
$$

Question 2c

Introduction: Define P(T) to be the claim that, if region S contains all locations in tree T, then fa(T, S, R) = FindAll(T, R). We will prove this holds by structural induction

Base Cases (T is empty or single):

Note that if S does not contain s, the P is vacuously true

fa(empty, S, R) = [] **def fa**

= FindAll(empty, R) **def of FindAll**

case contains(R,s) is true

fa(single(s), S, R) = [s] **def fa**

= FindAll(single(s), R) **def FindAll**

case contains(R, s) is false

fa(single(s), S, R) = [] **def fa**

= FindAll(single(s), R) **def FindAll**

Question 2c

Inductive Hypothesis: Suppose that P(nw), P(ne), P(sw), P(se) holds for some tree nw, ne, sw, se

Inductive Step (P(R), R = split(m, nw, ne, sw, se)):

Note that if S does not contain m, then P is vacuously true

If overlap(S, R) is false, then we know that no location in the tree can be in region R. Since S contains all the locations in the tree, we know that FindAll(T, R) = [], which is the definition of fa(T, S, R)

Question 2c

We know that NE(m, S) contains all locations in the tree ne. Thus we can see that:

fa(ne, NE(m, S), R) = FindAll(ne, R) **IH (since NE)**

Applying the same logic to NW, SW, SE we get:

fa(nw, NW(m, S), R) = FindAll(nw, R) **IH (since NW)**

fa(sw, SW(m, S), R) = FindAll(sw, R) **IH (since SW)**

fa(se, SE(m, S), R) = FindAll(se, R) **IH (since SE)**

Question 2c

We know that NE(m, S) contains all locations in the tree ne. Thus we can see that:

fa(ne, NE(m, S), R) = FindAll(ne, R) **IH (since NE)**

Applying the same logic to NW, SW, SE we get:

fa(nw, NW(m, S), R) = FindAll(nw, R) **IH (since NW)**

fa(sw, SW(m, S), R) = FindAll(sw, R) **IH (since SW)**

fa(se, SE(m, S), R) = FindAll(se, R) **IH (since SE)**

Substituting these definitions, we can calculate:

fa(split(m, nw, ne, sw, se), S, R)

= fa(nw, NW(m, S), R) ++ fa(ne, NE(m, S), R)
++ fa(sw, SW(m, S), R) ++ fa(se, SE(m, S), R) **Def of fa**

= FindAll(nw, R) ++ FindAll(ne, R)
++ FindAll(sw, R) ++ FindAll(se, R) **Substituting from above**

= FindAll(split(m, nw, ne, sw, se), R) **Def FindAll**

Thus we have proven the claim

Conclusion: P(T) holds for all Trees by structural induction