CSE 331: Software Design & Engineering Fall 2024

Quiz Section 9: Points

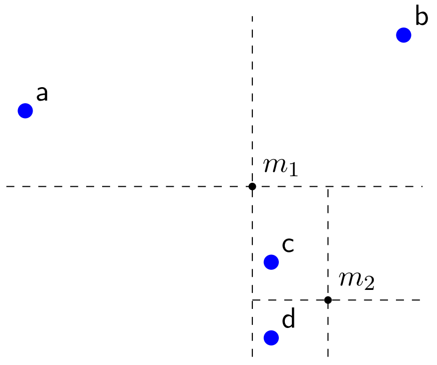
In this assignment, we will work with locations on a 2D map. For the sake of variety, rather than representing these as pairs, we will represent them as records as follows:

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

We will be interested in finding locations that are located within a given region (inside a rectangle). The simplest way to do this would be to iterate through a list of target locations. However, we can do this more efficiently by using a better data structure, namely, a tree, similar to the one we used in HW8, that recursively break up the points into smaller rectangles until each rectangle is either empty or contains a single point.

Trees of Locations

For example, with the four blue points below, we would start by splitting them at point "m_1" (the centroid of those locations). The NW and NE regions then contain only a single point and the SW region is empty. The SE region, however, still contains two points, so we would split them again at point "m_2", which leaves four regions containing at most one point.



We can represent this tree with the following inductive data type:

$$\begin{aligned}\textbf{type } \mathsf{LocTree} \;:=\;& \mathsf{empty}\\ \mid\;& \mathsf{single(loc : Location)}\\ \mid\;& \mathsf{split(at : Location,\ nw : LocTree,\ ne : LocTree,\ sw : LocTree,\ se : LocTree)}\end{aligned}$$

The constructor "empty" represents a region that is empty, while "single(s)" represents a region containing a single location at location s. The final constructor "split(m, nw, ne, sw, se)" represents a region that splits at location m into the four regions to the NW, NE, SW, and SE, respectively.

With those definitions, the example above would be represented with the following tree:

$$\begin{aligned}&\mathsf{split}(m_1, \mathsf{single}(a), \mathsf{single}(b), \mathsf{empty},\\ &\quad \mathsf{split}(m_2, \mathsf{single}(c), \mathsf{empty}, \mathsf{single}(d), \mathsf{empty}))\end{aligned}$$

Regions

Note that, while each node represents some rectangular region in space, the bounds of that region are not recorded in any field. That said, we can easily calculate the bounds of the region as we work down to that node, recursively, through the tree.

Specifically, we can store a rectangular region as an instance of the following type

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

The region R contains the location ℓ iff $R.\text{x1} \leqslant \ell.x \leqslant R.\text{x2}$ and $R.\text{y1} \leqslant \ell.y \leqslant R.\text{y2}$. Thus, the following region includes every point in the plane:

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

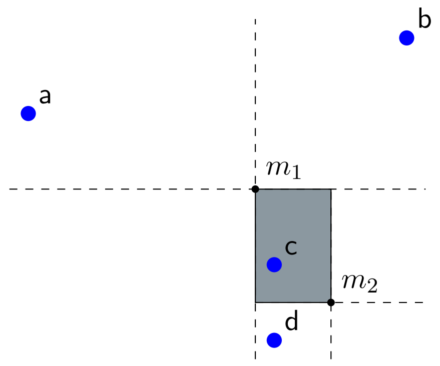
The following functions $\text{NW}, \dots, \text{SE} : (\text{Location}, \text{Region}) \to \text{Region}$ return the intersection of the region passed in with one of the quadrants of a node that was split at the location passed in. For example, $\text{NW}(m, R)$ would return the region that includes only the area that is both inside of R and northwest of m (i.e., an x coordinate that is $\leqslant m.x$ and a y coordinate that is $\leqslant m.y$).

These functions assume that m falls within the region R, and they are defined as follows:

$$\begin{aligned}
\text{NW}(m, R) &:= \{\text{x1}\colon R.\text{x1},\ \text{x2}\colon m.x,\ \text{y1}\colon R.\text{y1},\ \text{y2}\colon m.y\}\\
\text{NE}(m, R) &:= \{\text{x1}\colon m.x,\ \text{x2}\colon R.\text{x2},\ \text{y1}\colon R.\text{y1},\ \text{y2}\colon m.y\}\\
\text{SW}(m, R) &:= \{\text{x1}\colon R.\text{x1},\ \text{x2}\colon m.x,\ \text{y1}\colon m.y,\ \text{y2}\colon R.\text{y2}\}\\
\text{SE}(m, R) &:= \{\text{x1}\colon m.x,\ \text{x2}\colon R.\text{x2},\ \text{y1}\colon m.y,\ \text{y2}\colon R.\text{y2}\}
\end{aligned}$$

For example, $NW(m, R)$ leaves left side of the region at $R.\text{x1}$, but it only allows the region to extend to the right as $m.x$. Any point further to the right than that is outside of the region to the NW of m.

Returning to our example from before



the shaded region, which is the region that lies in the SE quadrant of the split at m_1 and then, within that, the NW quadrant of the split at m_2, can be calculated as $\text{NW}(m_2, \text{SE}(m_1, \text{EVERYWHERE}))$.

Task 1 – Regions in Hot Pursuit!

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.

Task 2 – Tree-search takes root!

(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.

(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 : (LocTree, Region, Region) → List⟨Location⟩ 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.

(c) Prove that if region S contains all locations in the tree T, then fa$(T, S, R) =$ 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 split$(m, \text{nw}, \text{ne}, \text{sw}, \text{se})$, then 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.)