You may work in pairs for this assignment. If you choose to work with a partner, make sure only one of you submits a solution, and you paste a copy of the Partners Template that contains the names and PIDs of both students at the beginning of the file you submit.

Prepare your answers to the following questions, either in a plain text file or a typed MS Word document. Solutions submitted in other formats will be discarded. Partial credit will only be given if you show relevant work or give a relevant justification for your answer.

---

1. **[15 points]** Consider the algorithm for insertion into a PR quadtree, assuming a bucket size of 1. Suppose that a data object was inserted, and its coordinates matched those of a data object already stored in the tree. What would happen if the insertion algorithm did not reject the insertion of such a duplicate object?

   Suppose we insert a data object B, that matches the coordinates of a data object A that has already been inserted to the tree. The insertion algorithm for the quadtree will attempt to split regions until A and B are in different regions.

   However, that's impossible if A and B are at the same coordinates, so unless the implementation protects against it, the result will be an uncontrolled recursive descent.

2. **[20 points]** Suppose that a PR quadtree, with bucket size 1, represents a region in the *xy*-plane bounded between (0, 0) and (2^10, 2^10). Suppose further that we insert two data objects *A* and *B*, corresponding to locations that are separated by a distance of 5 units. Given only that much information, give the most precise statement you can regarding how many region splits will be required in order to separate *A* and *B*? Justify your answer carefully.

   The minimum number of splits is 1.

   For example, if A and B were at (10, 510) and (10, 515), then the first split would put A into the NW quadrant and B into the NE quadrant.

   The height bound from Samet's paper doesn't help here, since it only potentially applies. The truth is the number of levels depends not only on the minimum distance but also on exactly where the closest pair of points lie in the world region.

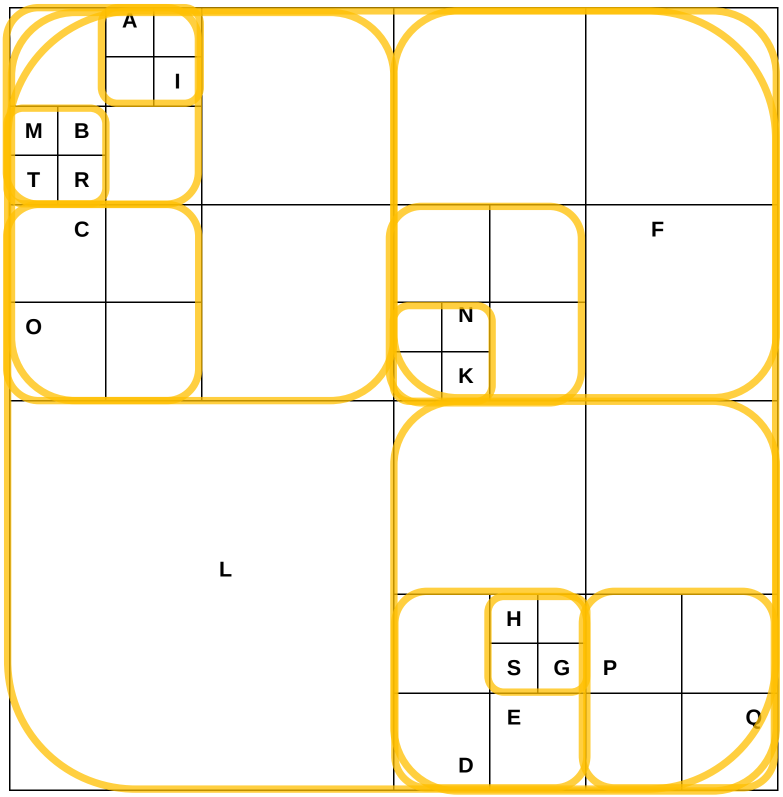
   The maximum number of splits that could be required is 9.

   Each split produces subregions that have sides equal to half that of the region that was split. So, we will get subregions that are 512x512, 256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, and so forth.

   The diagonal of a square of side S equals S*sqrt(2), so a 2x2 square could not contain two points that are 5 units apart, but a 4x4 square could do so.

   The height bound from Samet's paper also does not help with the analysis here, since (at best) that provides a lower bound on the number of levels, and we are concerned with the maximum.

3. The diagram below shows a partitioned world, which can be represented by a PR quadtree, with bucket size 1.



a) **[15 points]** How many leaf nodes (nonempty) would the quadtree have?

Leaf nodes correspond to regions that contain a single data object (since we have a bucket size of 1). There are 19 data objects shown, so the tree would have 19 leaf nodes.

b) **[15 points]** How many internal nodes would the quadtree have?

Internal nodes correspond to regions that have been subdivided; they are highlighted in the diagram above.

Or, you can simply count the number of "centers" where regions have been divided.

Either way, there are 13 internal nodes.

4. Suppose a PR quadtree stores 972 data objects.

a) **[5 points]** If the bucket size is 1, what is the minimum number of levels the tree could have? Why?

The maximum number of nodes in level K is 4^K. The lowest level must have 972 leaf nodes, which would imply $4^K >= 972$, whence K >= 5.

Therefore, there would be at least 6 levels.

b) **[5 points]** If the bucket size is 2, what is the minimum number of levels the tree could have? Why?

With bucket size 2, the maximum number of values stored in level K becomes $2*4^K$. Then, applying the same logic as before, we need to have $2*4^K >= 972$, whence K >= 5.

Therefore, there would be at least 6 levels.

c) **[5 points]** If the bucket size is 4, what is the minimum number of levels the tree could have? Why?

Now, we need $4*4^K >= 972$, whence K >= 4. So there would be at least 5 levels.

d) **[5 points]** If the bucket size is 8, what is the minimum number of levels the tree could have? Why?

Now, we need $8*4^K >= 972$, whence K >= 4. So there would be at least 5 levels.

There is a point here… increasing the bucket size doesn't seem to have much of an effect on the minimum number of levels the quadtree might have.

5. **[15 points]** Haskell Hoo IV, a noted blogger, suggests that if you implement buckets in your PR quadtree leaf nodes, then the bucket for a leaf node should itself be a PR quadtree whose world is simply the region that corresponds to that leaf node. He asserts that's more elegant than using some sort of linear structure for the bucket (which is an entirely subjective claim), and he also asserts that his suggestion will improve efficiency in both memory cost and search cost (which is an objective claim).

Simply from a coding perspective, there's nothing difficult about having a PR quadtree whose leaves also contain PR quadtree objects.

Comment on Hoo's suggestion. Be precise.

To get a perspective on this, it's worth noting that if we use a PR quadtree for the bucket in a leaf, that quadtree will have the same structure as the subtree we would have had if we did not use buckets at all. (So, arguably, Hoo is missing the point here.)

Elegance is in the eye of the beholder, and cannot be assessed rationally.

As for space efficiency, a linear bucket would probably have one pointer per element or else possibly some empty array cells. (We do not count the cost of the data objects since they must be stored in any scheme.) The quadtree bucket must have at least that many pointers, and may have far more, since closely packed data objects would result in many levels. So, it does not seem that Hoo's suggestion is likely to save space.

As for time efficiency, a linear bucket will require a linear traversal, costing O(bucket size). A quadtree bucket would require a computation to determine the direction to take from the first level node; let's say that's about the same cost as a pointer dereference. (It may be a bit more, but both will be O(1).) In the best case, this may beat the linear bucket, but if the quadtree bucket requires much splitting, it will be more expensive to traverse. In fact, that was the motivation for using buckets in the first place…

So, it seems as if Hoo's suggestion is very unlikely to yield better results.