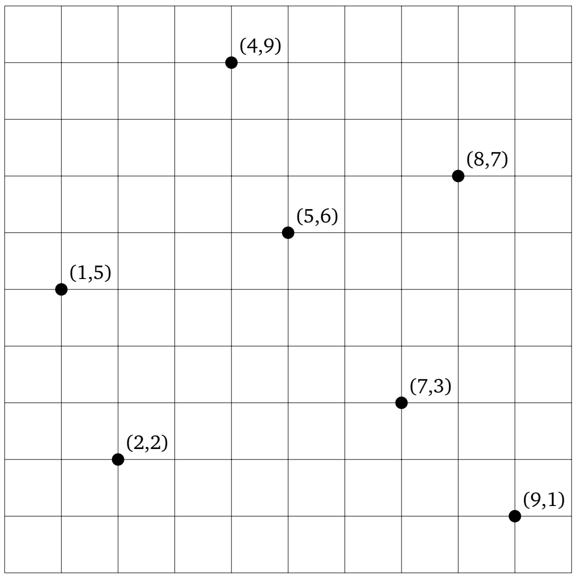
Section 05: k-d Trees and Hashing

1. k-d Trees

(a) Given the points shown in the grid to the right, draw a perfectly balanced k-d tree (where $k = 2$, i.e. a 2-d tree) in the box below. For this tree, first split on the x dimension. The resulting tree should be complete with height 2. Then, draw the corresponding splitting planes on the grid to the right.



(b) Insert the point (6, 2) into the k-d tree you drew below. Then, add that point to the grid and draw the corresponding splitting plane.

(c) Find the nearest point to (3, 6) in your k-d tree. Mark each branch that is not visited (pruned in execution of `nearest`) with an `X` through the branch.

2. Hash Functions

For the following implementations of the `Integer` class's `hashCode()` function, answer whether it is valid or invalid. If it is invalid, explain why. If it is valid, point out a flaw or disadvantage.

Note: The `Integer` class extends the `Number` class, a direct subclass of `Object`. The `Number` class' `hashCode()` method directly calls the `Object` class' `hashCode()` method.

(a) `return -1;`

(b) `return intValue() * intValue();`

(c) `return super.hashCode();`

3. Hashing

(a) Suppose we have a hash table that uses separate chaining and has an internal capacity of 12. Assume that each bucket is a linked list where new elements are added to the front of the list.

Insert the following elements in the EXACT order given using the hash function $h(x) = 4x$ (don't worry about resizing the internal array):

0, 4, 7, 1, 2, 3, 6, 11, 16

(b) Suppose we have a hash table with an initial capacity of 12. We resize the hash table by doubling the capacity. Suppose we insert integer keys into this table using the hash function $h(x) = 4x$.

Why is this choice of hash function and initial capacity suboptimal? How can we fix it?

4. Hashing: Ice Cream

Assume we have a hash table that maps `Characters` to `Integers`, and starts with an internal capacity of 4 and resizes by doubling the capacity if the insert would cause the load factor to exceed 0.75. Also assume that the hash table uses separate chaining with linked lists where new elements are added to the front of the list.

(a) Draw the hash table after inserting the following items in the given order:

```
('i', 1), ('c', 2), ('e', 3), ('c', 4), ('r', 5), ('e', 6), ('a', 7), ('m', 8)
```

Assume the hash code for `'a'` is 0, `'c'` is 2, `'e'` is 4, `'i'` is 8, `'m'` is 12, and `'r'` is 17.

(b) How many hash collisions occur? Do not count the ones that occur during resizing.

(c) How many hash collisions occur during resizing?

(d) What is the current load factor after all insertions?