**CS2040S: Data Structures and Algorithms**

# Discussion Group Problems for Week 7

*For: February 28–March 4*

## 1 Check in

Discuss questions, if you have any, with the tutor and the rest of the class, about the material and content so far.

## 2 Problems

**Problem 1. AVL vs Trie**

Discuss the trade-offs of using AVL and Trie to store strings.

**Problem 2. kd-Trees**

A kd-tree is another simple way to store geometric data in a tree. Let's think about 2-dimensional data points, i.e., points $(x, y)$ in the plane. The basic idea behind a kd-tree is that each node represents a rectangle of the plane. A node has two children which divide the rectangle into two pieces, either vertically or horizontally.

For example, some node $v$ in the tree may split the space vertically around the line $x = 10$: all the points with $x$-coordinates $\leq 10$ go to the left child, and all the points with $x$-coordinates $> 10$ go to the right child.

Typically, a kd-tree will alternate splitting the space horizontally and vertically. For example, nodes at even levels split the space vertically and nodes at odd levels split the space horizontally. This helps to ensure that the data is well divided, no matter which dimension is more important.

All the points are stored at the leaves. When you have a region with only one node, instead of dividing further, simply create a leaf.

Here is an example of a kd-tree that contains 10 points:

**Problem 2.a.** How do you search for a point in a kd-tree? What is the running time?

**Problem 2.b.** You are given an (unordered) array of points. What would be a good way to build a kd-tree? Think about what would keep the tree nicely balanced. What is the running time of the construction algorithm?

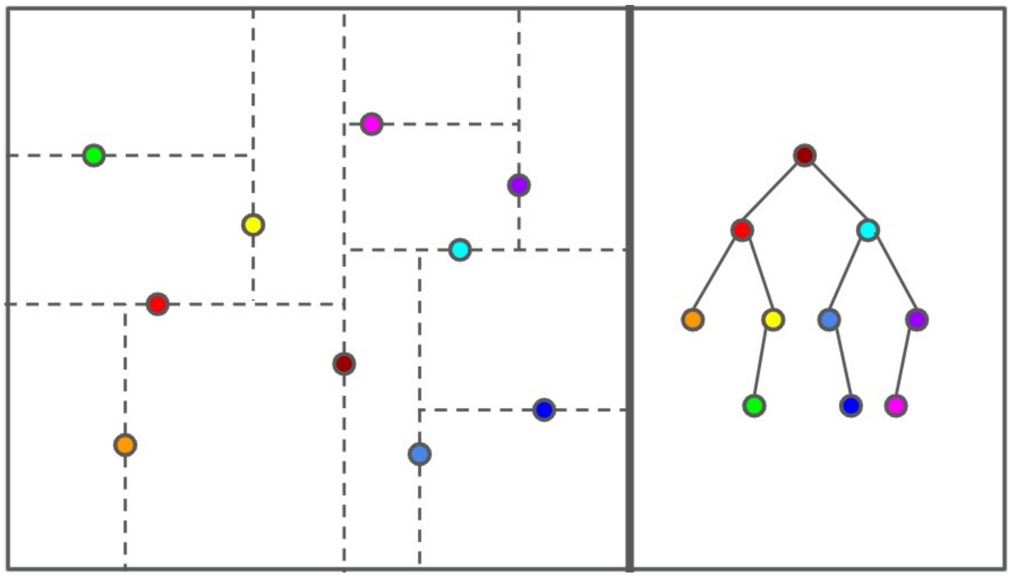

**Figure 1:** On the left: the points in the input. On the right: how the points are stored in the kd-tree

**Problem 2.c.** How would you find the element with the minimum (or maximum) x-coordinate in a kd-tree? How expensive can it be, if the tree is perfectly balanced?

**Problem 3. Tries(a.k.a Radix Trees)**

Coming up with a good name for your baby is hard. You don't want it to be too popular. You don't want it to be too rare. You don't want it to be too old. You don't want it to be too weird.[1]

Imagine you want to build a data structure to help answer these types of questions. Your data structure should support the following operations:

- `insert(name, gender, count)`: adds a name of a given gender, with a count of how many babies have that name.
- `countName(name, gender)`: returns the number of babies with that name and gender.
- `countPrefix(prefix, gender)`: returns the number of babies with that prefix of their name and gender.
- `countBetween(begin, end, gender)`: returns the number of babies with names that are lexicographically after `begin` and before `end` that have the proper gender.

In queries, the gender can be either boy, girl, or either. Ideally, the time for `countPrefix` should not depend on the number of names that have that prefix, but instead run in time only dependent on the length of the longest name.

[1]The website `https://www.babynamewizard.com/voyager` let's you explore the history of baby name popularity!

## 3 Past Midterm Questions

We have posted the CS2040S 2021, 2022, 2023 midterms. We would particularly recommend going over the following questions:

- Sorting Jumble (any of the papers)
- Algorithm Analysis (any of the papers)
- Midterm Sorting (CS2040S 2021, Q8)
- How do they work? (any of the papers)
- 3D Printing (CS2040S 2022, Q9)
- The IntervalRay of Doom (CS2040S 2023, Q7)