CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 7

For: March 3–March 8

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 ≤ 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.

There are 2 main ways to implement kd-tree.

In one way, internal nodes of the tree corresponds to the horizontal or vertical splitting lines; When you have a region with only one data point, instead of dividing further, simply create a leaf which corresponds to the only data point. Figure 1 is an example of a kd-tree that contains 14 points (internal nodes without any label can be omitted):

Another way to implement kd-tree uses horizontal or vertical lines that pass through data points for splitting. In this case, each tree node (internal and external) represents one data point. Figure 2 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

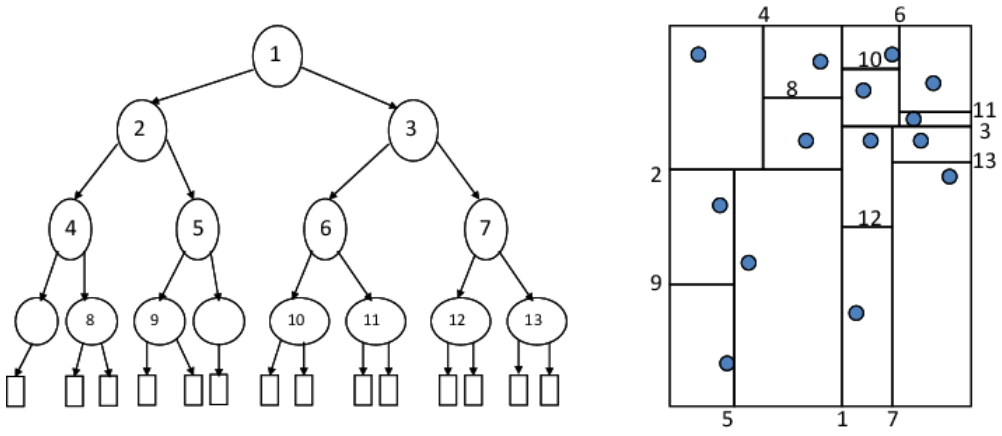

Figure 1: On the left: the points in the input. On the right: how the points are stored in the kd-tree

build a kd-tree? Think about what would keep the tree nicely balanced. What is the running time of the construction algorithm?

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!

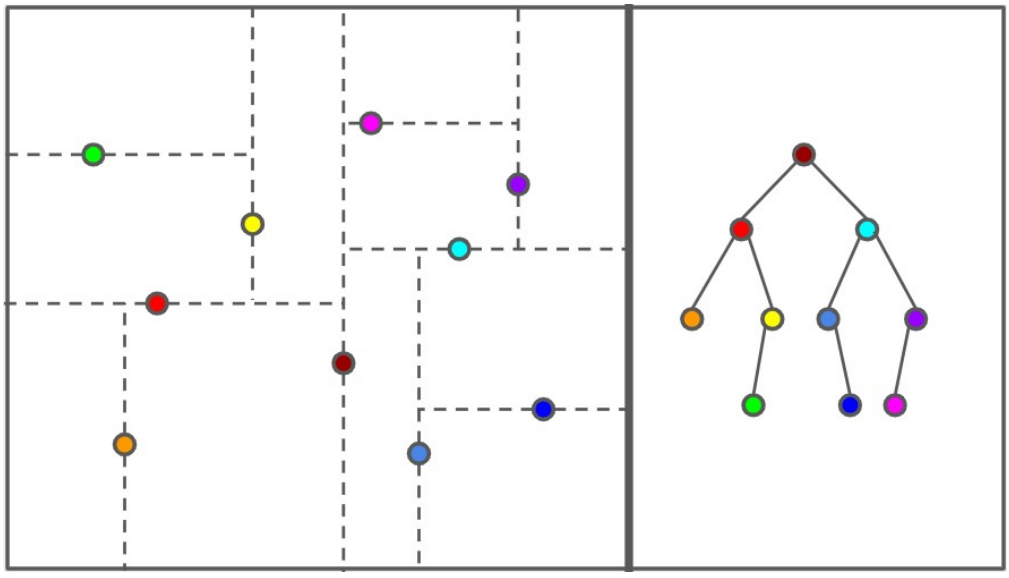

Figure 2: On the left: the points in the input. On the right: how the points are stored in the kd-tree

Problem 4. A Trie Question?

Problem 4.a. Your task is simple. Given an array of 32 bits unsigned positive integers, find 2 numbers such that their XOR is maximum.

- Hint 1: Look at the title of the question.
- Hint 2: Think of numbers bit by bit from the most significant bit.

Problem 4.b. (Bonus, optional, difficult) (Source: https://codeforces.com/contest/1777/problem/F)

Now, we modify the question. You are given an array of 32 bits unsigned positive integers A. Find a subarray of A(l,r) such that the following value is maximised:

$$A_l \oplus A_{l+1} \cdots \oplus A_r \oplus max(A_l, A_{l+1}, \ldots A_r)$$