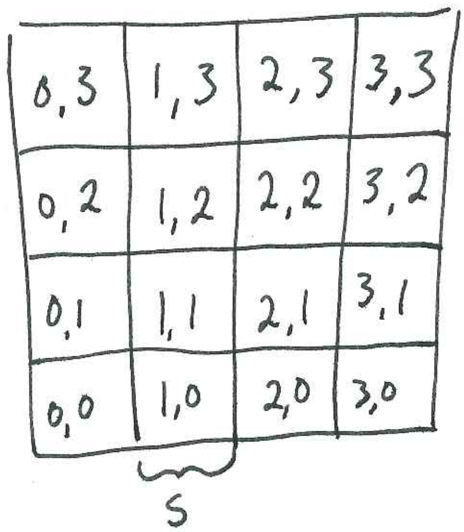
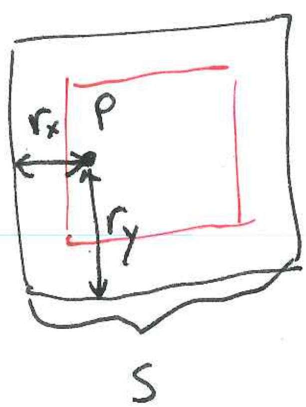


Integer indices in a grid.

$p \in \text{square}(x, y)$

$p_x = xs + r_x \quad 0 \le r_x, r_y < s$

$p_y = ys + r_y$



<u>Def</u> p is central at scale s if $\frac{1}{6}s \le r_x, r_y < \frac{5}{6}s$

The intuitive definition

An equivalent and more handy definition is as follows.

<u>Def</u> p is central at scale s if when we write

$p_x + \frac{1}{6}s = xs + r_x$ for $x, y \in \mathbb{Z}$ and $r_x, r_y \in [0, s)$

$p_y + \frac{1}{6}s = ys + r_y$

we have $r_x, r_y \ge \frac{s}{3}$.

Claim:

Let $p \in [0,1)$ and $s = \frac{1}{2^\ell}$ for some $\ell \in \mathbb{Z}_{\geq 0}$

Let i,j integers s.t. $0 \leq i < j \leq 2$

Then p is central at scale s for either shift i or j.

pf/ Suppose for contr. p is not central for i nor for j.

We can write $p + \frac{1}{6}s + \frac{i}{3} = x_i s + r_i$

and $p + \frac{1}{6}s + \frac{j}{3} = x_j s + r_j$

where $0 \leq r_i, r_j < \frac{s}{3}$. $x_i, x_j \in \mathbb{Z}$

Subtract these equations to get

$$\frac{1}{3}(j-i) = s(x_j - x_i) + (r_j - r_i)$$

$$2^\ell(j-i) - 3(x_j - x_i) = \frac{3}{s}(r_j - r_i) < 1$$

$$[\text{So, } r_j = r_i]$$

$$2^\ell(j-i) = 3(x_j - x_i)$$

So 3 divides $2^\ell(j-i)$, a contradiction.

So for $p \in [0,1)^2$, some shift is "good" for both x and y coordinates, i.e. p is central for some shift.

So, we know that 3 shifts of the QT guarantees that one will return an approximate NN. However, the search could take $O(n)$ time because a compressed QT can have depth $O(n)$.

Now, we want to show a different way of searching a compressed QT. In fact, we will only simulate the search by building a completely different binary search tree.

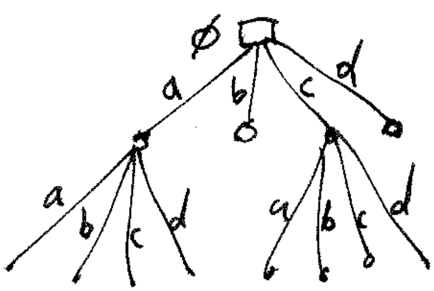
Background:

Tries (pronounced like try with an s at the end)

etymology: reTRIEval. originally pronounced "trees" for this reason, but that is too confusing.

The idea: A data structure for storing strings over a finite alphabet. Store words as nodes in a tree. Every node stores a prefix of its children. Tree has degree equal to the size of the alphabet. Edges labeled with letters. Words are paths from root.

Example Alphabet: $\{a, b, c, d\}$



It looks like a quadtree

aa	ab	b
ac	ad	
ca	cb	d
cc	cd	

Application to QT search:

view boxes as strings

View points as strings.

A point is in a box iff the box string is a prefix of the point string.

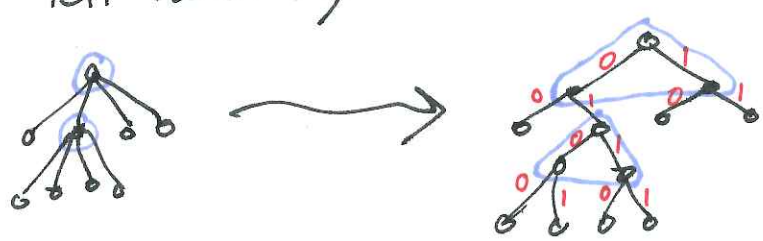
But what are the strings?

Bit Twiddling

Assume points in unit box $[0,1)^2$.

So, every coordinate looks like $0.x_0x_1x_2\cdots x_k$ where $x_0,\dots,x_k$ are the bits in the binary representation. Bit x_0 is most significant.

<u>Recall</u>: We can expand a QT to write it as a binary tree. By convention, split top and bottom first, left and right second.



Label edges 0 and 1 for left and right respectively.

This gives a string for every box.

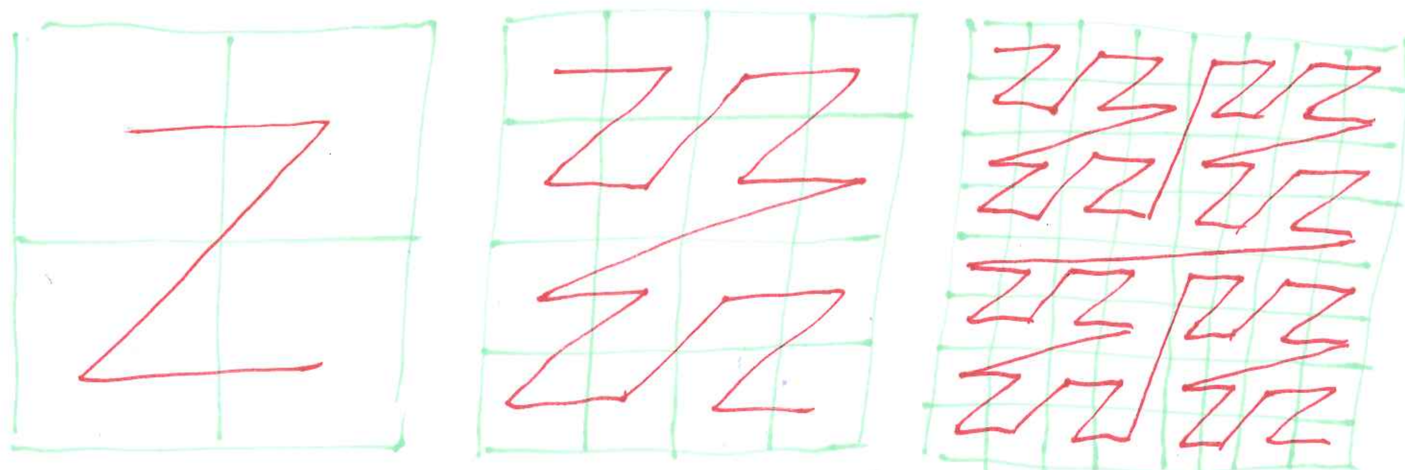
The string for a point is just the search path for that point in an infinite quadtree (the search is finite as long as the point only has finitely many bits).

★ <u>Surprising Fact</u>: This string is obtained by interleaving the bits of the y and x coords.

$$Z\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = 0.y_0x_0y_1x_1\cdots y_kx_k \quad \text{where } x = 0.x_0x_1\cdots x_k \text{ and } y = 0.y_0y_1\cdots y_k$$

The function z maps points in $\mathbb{R}^2$ to numbers.
Thus, it puts an ordering on the points in the plane.
This is called a space filling curve. (SFC)

Here is the SFC at different resolutions



Do you see why I call it z?
This is sometimes called the z-order.

The new data structure

For each $p \in P$ store $z(p)$ in a balanced binary search tree. (BST).

To search for q, find the predecessor and successor of $z(q)$ in the BST.

By construction, one of these points is in the smallest quadtree square that contains both q and a point of P.

★ For 3 shifts, store 3 BSTs.

recall shifts we used were 0, $1/3$, and $2/3$

in binary $\frac{1}{3} = .0101010\overline{1}\ldots$

$$2/3 = .1010101\overline{0}\ldots$$

The z-order makes it more clear why these are good choices. They change the bits at all scales.