Image compression

Given a greyscale image whenre every pixel is between 0 and 1 our objecive is to find a fast and eficient algorithm for compressing the image

step 1: build a complete QuadTree over the image

we build it bottom up and propagate min, max and sum of pixel values

step 2: prune the quadtree

we prune if and only if $\max(\text{blob of pixels}) - \min(\text{blob of pixels}) \leq \theta_1$, where θ_1 is a threshold

step 3: calculate the grayscale of each leaf

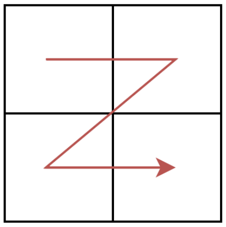
$a = \frac{1}{n}(sum)\ or\ \frac{1}{2}(max - min)$

step 4: encoding

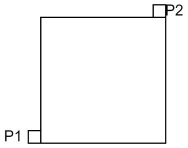
we traverse Q by DFS in a so-called Z-order, being UL-UR-BL-BR

let $\mathbf{s}$ = side length of a leaf and $\mathbf{p}$ a predictor:

$$code(a,s,p,\theta_2) = \begin{cases}(a-p)\frac{s}{\theta_2}\cdot 255 & s < \theta_2\\ (a-p)\cdot 255 & else\end{cases}$$

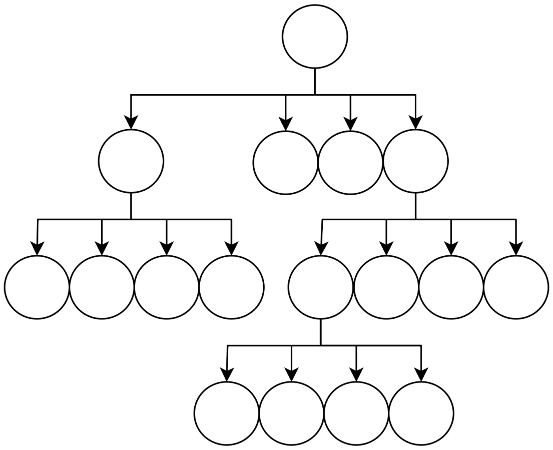


such that $p = \frac{1}{2}(p_1, p_2)$, the **encoded values** of the points outher from the bottom left and top right. this encoding works because of the Z-order, so we always know we visited the predictor’s pixels befor the node we are encoding



step 5: bit encoding

first of all we encode the topology of the quad tree using the **treecode:** we go in Z-order and we store 1 for inner nodes and 0 for leaf nodes, for example this tree:



will be encoded in
110000000110000000

at this point we generate the gray scale values as separates bit streams, using some encodings.

step -1: decoding

to decode is sufficient to backtrack the 5 stepsù:

- build the tree from the treecode
- reconstruct the grayscale valuse: $a = code \cdot max(1, \frac{\theta_2}{s}) + p$
- optionally smooth block artifacts(smoothing operator, interolation, AI)