Quad trees, addresses

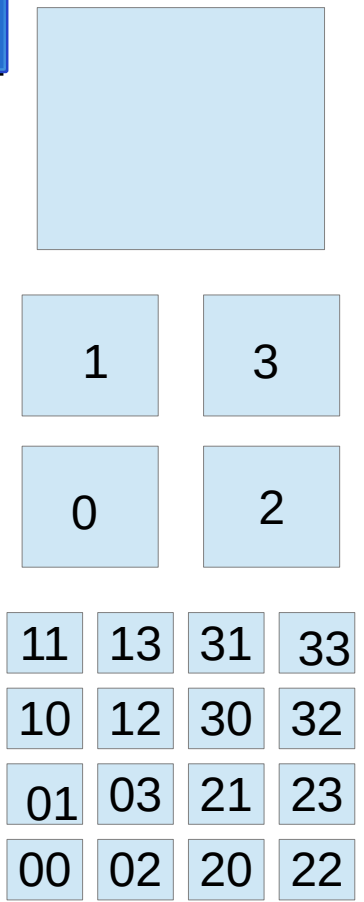
Top level node represents entire space, prefix string is “”

Four nodes on second level each represent 1/4 of total space, Prefix strings are “0”, “1”, “2”, “3”

Sixteen nodes on third level each represent 1/16 of total space, Prefix strings are “00”, “01”, “02”, ... “33”

Can describe rectangular regions by specifying a lower left (LL) address and an upper right (UR) address

E.g. LL of 12 and UR of 31 would be the rectangle covering squares 13, 31, 12, and 30 ... note that it overlaps the two regions covered by the larger squares 1 and 3



Inserting items by address in quadtree

- Area an item covers is rectangle, specified by address of lower left corner and upper right corner, e.g. 010 and 212
- 0,1 are to the left of 2,3
- 0,2 are below 1,3
- If user specifies a partial address for LL then it gets padded with 0’s to a full address length
- If user specifies a partial address for UR then it gets padded with 3’s to a full address length

Item size, address

- Items can be of any size bigger than smallest unit in address space
- e.g. if full address strings are length 5 then a tiny rectangle might have LL 11111, UR 11112 (covering just two of the smallest-size quadrants)
- biggest rectangle has LL 00000, UR 33333 (covering entire space)

Item location in quadtree

- Each node in quadtree has a list of which items that specific nodes stores
- Each node covers specific region of total space
- Quadtree root covers full space (LL-UR 00000-33333)
- Four nodes at second layer each cover 1/4 of total space (00000-03333, 10000-13333, 20000-23333, 30000-33333)
- Sixteen nodes at third layer each cover 1/16 of total space (00000-00333, 01000-01333, ..., 33000-33333)

Addresses, quadrants, tree level

- If we look at the LL/UR addresses of an item, we can identify the smallest node whose region can completely contain the item, e.g. LL 01112, UR 01333: look at the common prefix at the start of the two addresses ***01***112, ***01***333 – i.e. from root go to child node 0, from there go to child node 1
- We use this common prefix to decide which node in the tree will actually store the item
- each node can store multiple items, maintains a list of all the items it is storing

Example, max addr length 2

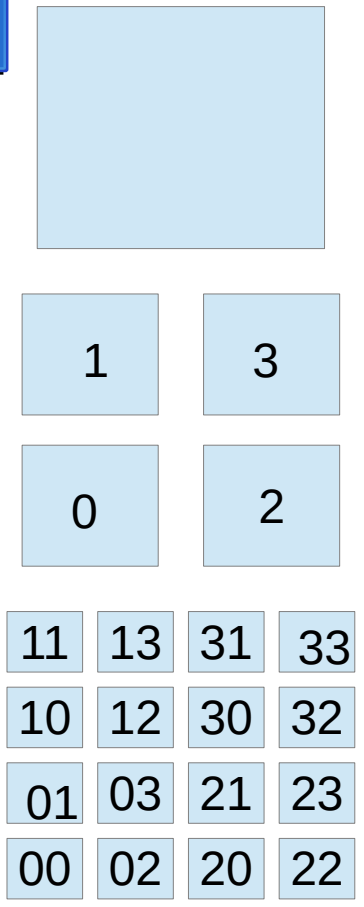
Top level node represents entire space, prefix string is “”

Four nodes on second level each represent 1/4 of total space, Prefix strings are “0”, “1”, “2”, “3”

Sixteen nodes on third level each represent 1/16 of total space, Prefix strings are “00”, “01”, “02”, ... “33”

Suppose we insert item with LL 03, UR 30: it can’t fit cleanly within any single node except the top one

Suppose we insert item with LL 12, UR 13: it fits cleanly in the “1” node



Observations on demo program

- Demo requires you to create a tree before inserting
- For each item it asks you for name, description, and addresses of LL and UR (pads with 0's/3's if you give partial address, max addr length is 10)
- Follows our rules on where to insert the item in the tree
- When searching or printing, lets you search/print an entire subtree, or just a specific node (then asks for the address, i.e. the prefix string, for that node or top node of subtree)