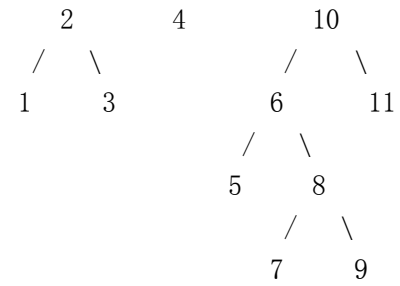
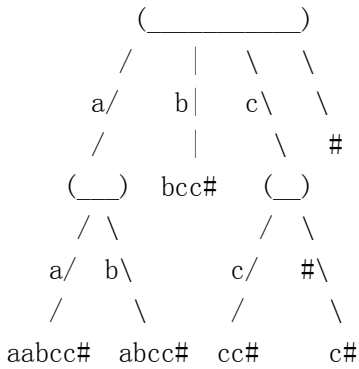
<<< Exam #3 sample solution >>>

1. Splay tree
 : splay --> split

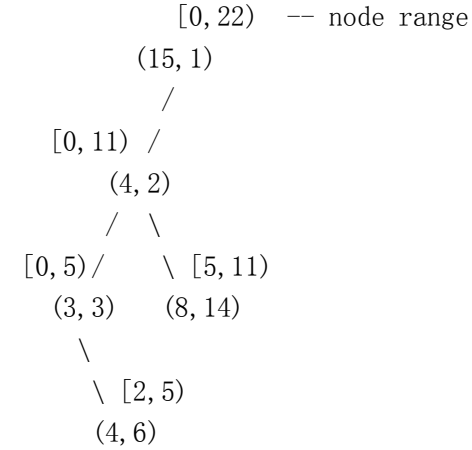


2. suffix tree

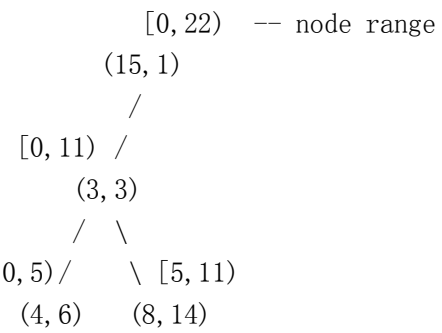


3. min radix priority search tree

o insertions



o delete (4,2)



4. 3D range tree

o Let the 3 keys be x, y, and z. A 3-dimensional range tree is a range tree on z, while each node in that tree is a range tree on y, and finally each node in that is a sorted array on x. A range tree is like a segment tree. The root contains all the records. A delimiter is chosen to be the median of the keys. According to the median key value, the records are split into two equal parts, one containing records no larger than the median goes to the left subtree, the other half goes to the right subtree. And the median is chosen and records are split recursively until the number of records is considerably small.

o Preprocessing time P:
Since we need the sorted array on x at every node, we need to sort the records on x first. The median can be found in linear time, and the splitting is done by scanning through the sorted array of the root, deciding which side a record should go on the fly. So after the splitting, the subarrays created remain sorted on x. Since at each level, all records are present. There are at most logN level. So the splitting on y takes NlogN at evey level. Thus the total time to split on z, containing logN levels as well, takes N(logN)^2. Since it is larger than the sorting of x, NlogN, already, the complexity for the preprocessing is thus O(N(logN)^2).

o space S:
In the range tree on y, all records are present at each level. Thus each tree on y takes NlogN space. But there are total of logN levels of the range tree on z as well. So the total space required is O(N(logN)^2).

o Querying time Q:
Given a query in the form (x1,x2,y1,y2,z1,z2), we first determine which nodes in the range tree of z are contained in the range [z1..z2]. Then at each node we perform another search on y to find out the nodes contained in [y1..y2]. Finally in each node of tree y we use binary search to determine where x1 and x2 are in the sorted array of x. There are logN nodes at tree z that are candidates, and inside each node there are logN nodes in tree y that may contain the answers. Binary search on x takes O(logN) time.
So the total time is O((logN)^3).

But we still need to report the answers, and the answer size might be larger than the search time.
So the final complexity is O((logN)^3+|ans|).

5. quadtree

(a) If the pixel [i,j] is to the NW,NE, SW, or SE of current partitioning line, go to NW, NE, SW, or SE child. repeat this until a leaf node is reached. This leaf node is the corresponding node to the pixel.

(b) You can initialize all the leaf nodes of the quadtree using the scheme in the question (a). This takes O(NlogN) time where N(n^2) is the number of pixels in the image.
Then, all internal nodes can be initialized by just traversing in post-order using black, white, and gray colors which takes O(N).

So, the total time complexity is O(NlogN).

(c) While traversing the tree, you can add the number of pixels in a node if the node stands for white. The number of pixels in the node can be easily computed using tree level (that is, the number of pixels in a node is 4^l where l is tree level).
Of course, if current node is gray, you'll need go down until white node.