Department of Computer Science and Engineering
University of California at San Diego

CSE 102
Winter 1997

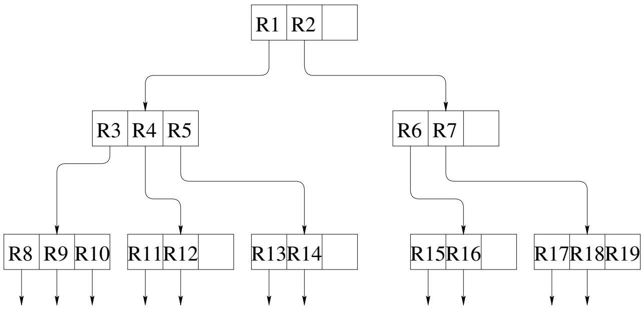
Practice Questions

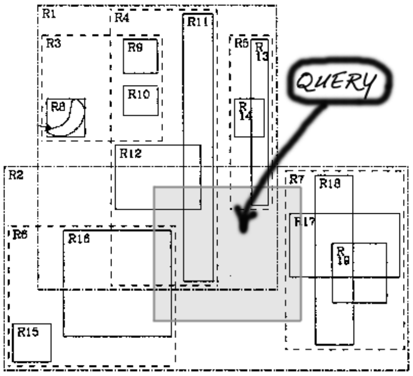
1. Guttman claims that exhaustive search for an optimal split is impractical but you might think that with fast and large central memories, and even faster CPUs you can get away with it. Well... let's see.

   Let us say that we have $m = 8$ and $M = 64$. Your processor can examine one million possible splits per second. Would it be practical to make an exhaustive search? (just to give you a reference: 1 day $\approx 2^{16} sec$, 1 year $\approx 1.8 \times 2^{25} sec$, life expectation at birth of the average American: $1.05 \times 2^{31} sec$, age of the universe: $1.36 \times 2^{62} sec$.)

   Would your answer change if your processor could examine *10 billions (*10^{10}*)* possible splits per second?

2. Consider the following R-tree (fig 3.1 on the paper), and the query region indicated by a gray square.



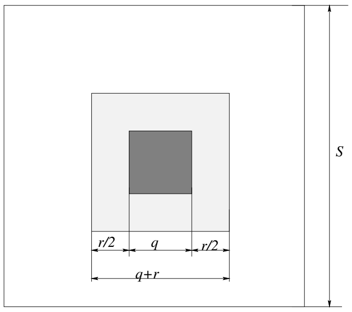


   (a) What regions will the query retrieve?

   (b) What branches of the tree will be traversed, and what regions will be checked for intersection?

3. Suppose that in the same tree we remove the region R8. What tree will we obtain? What intervals will be affected?
4. We have an R-tree with 4096 leaves, m=2, M=16. What are the minimum and maximum heights for the tree?
5. in B-tree, we have an exact criteropn for almost everything: we know how to order the children of a node, how to distribute elements in a node, which children to move and where when we split a node, and whatnot. In R-trees we have no such precise criteria, but only some heuristics which, by definition, sometimes will work, and sometimes won't. The reason, of course, is not that Guttman is not good enough to come up with optimal methods, but has something to do with a fundamental difference between one-dimensional data (like those we have in B-trees) and multidimensional data (like those we have in R-trees). What difference is that?
6. (this is strictly for math lovers). The original objective of R-trees is to minimize the number of regions that intersect a given region (the "query" region), but what Guttman really does is to try to keep regions "small." You are asked to prove that this is a good idea.

   Let us see how. Suppose everything happens inside a square of side S. Your query is a square of side q there in the middle (see Fig.). Let us say for now that the R-tree contains only one region: a



   square of side r that can be placed everywhere inside the square of side S with equal probability. The probability that the query intersect the region is equal to the probability that the center of the region falls within the square of size $q + r$ in the center of the square S (the square shown in light gray in Fig.). What is that probability (remember to assume that the region can be everywhere in S with equal probability).

   What if you have n regions (all of them squares of side r) in the square S? What is the probability that at least one of them intersect the query? (Hint: P(at least one intercept) = 1 - P(no intersection) = 1 - P(All the centers of the regions out of the light gray square in Fig.))

   Based on this, do you think Guttman's strategy is sensible?
7. Given M, what values can m assume to have the R-tree algorithms work properly and avoid performace degradation? What happens if m is too big? What happens if m is too small?
8. Would your answer to the previous question change (and, if yes, how) if instead of R-trees we were considering B-trees?
9. Guttman claims (pag. 51, last paragraph of section 3.3) that insertions can prevent deterioration of the tree structure. Can you explain intuitively why?
10. Suppose you are building an R-tree for a data structure that will never change but will be interrogated very often. Would you use the linear split or the quadratic split?