Worst-Case Analysis for R-trees

(Based on the paper: Optimal Dynamic Range Searching in Non-Replicated Index Structures, by K.V.Ravi Kanth and Ambuj Singh appeared in ICDT 1999.)

Theorem: In an R-tree of 2 n-dimensional points, the worst-case time complexity for an empty range query is O(N/B), where N is the number of points stored in the tree and B is the page capacity.

Proof: Consider an R-tree with minimum page capacity (except the root) m and maximum page capacity M-1. We first note that the time complexity for a range query in an R-tree is proportional to the number of nodes whose corresponding MBRs intersect the query. Since every query that intersects a leaf node also intersects all its ancestors, we will concentrate on the leaf nodes since this give a lower bound on the number of I/O's to answer a range query.
We consider two types of clusters C1 and C2 and we repeatedly insert points from these clusters in an alternate fashion. In the first type, C1, all data points have y-value either 3 or 6. For the cluster C2, all points have y-value either 1 or 8. In particular, for the i-th point in a C1 cluster, using an offset o for the x-coordinate, it has the following coordinates:

(o+ (i-1)/2, 3), if i is even
(o+ i/2, 6), if i is odd.

This point is denote by C1(o, i). Similarly, the i-th point of a C2 cluster is:

(o+ (i-1)/2, 1), if i is even
(o+ i/2, 8), if i is odd.

Now, let I(c) and D(c) denote insertion and deletion of point c. Also, let I(C, o, j, k) denote insertion of a sequence of points from cluster C at offset o, i.e., I(C, o, j, k) = <I(C(o,j), C(o, j+1), …, C(o, k)>.

Let, N=m(2k+1) for some k. We consider the following insertions in an initially empty R-tree:

S = < I(C1, 0, 1, m), I(C2, Lm, 1, M-m), D(C2, Lm, m+1, M-m), I(C1, 2Lm, 1, M-m),
D(C1, 2Lm, m+1, M-m), I(C2, 3Lm, 1, M-m), D(C2, 3Lm, m+1, M-m), …..
I(C2, (2k-1)Lm, 1, M-m), D(C2, (2k-1)Lm, m+1, M-m),
I(C1, 2kLm, 1, M-m), D(C1, 2kLm, m+1, M-m)>

First, a set of m points from cluster C1 at offset 0 is inserted into an empty R-tree. Then, a set of (M-m) points of the cluster C2 at the offset Lm are inserted. The insertion of the last point triggers a split of the only leaf page in the R-tree and the creation of two nodes. The first node (N1) stores m points and corresponds to cluster C1 and the second node (N2) to the cluster C2. Then we delete M-2m points from the leaf node N2. This is followed by an insertion of M-m points from C1 at offset 2Lm. All these insertions occur in the leaf node N2, which is split into two nodes, N2' and N3. These nodes correspond to clusters C2 and C1 respectively. We continue in the same way until we insert all N points. Using induction we can prove that the final R-tree will contain 2k+1 nodes each of m points. The i-th node contains points from the C2 clusters if I is even and from C1 clusters if i is odd.
Now, consider the query: Q=[0, Ln] [4,5]. Since the bounding box of each of the C1/C2 clusters contain the y-interval [4,5] and is inside the [0, Ln] x-interval, it follows that *all* leaf nodes intersect the query. Note also that the query is empty (no point is inside the query). Therefore, the number of leaf nodes retrieved for this query will be O(k)= O(N/m). QED.