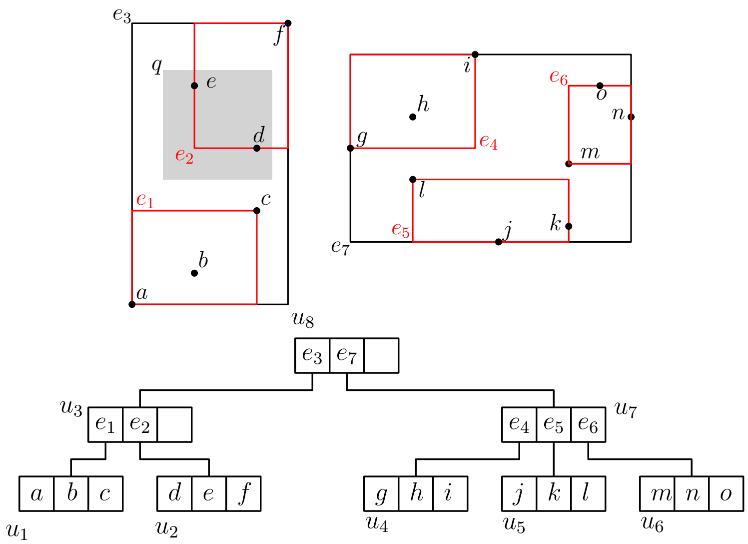
INFS 4205/7205: Exercise Set 1

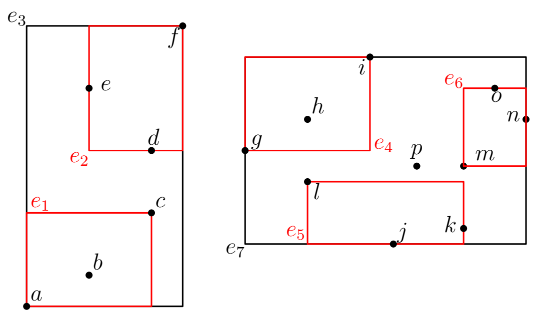
Prepared by Yufei Tao and Junhao Gan

Problem 1. The following figure shows a set of points (labeled $a, b, ..., o$) and also the corresponding R-tree. List all the nodes that need to be accessed in order to answer the range query whose search region is the shaded rectangle.

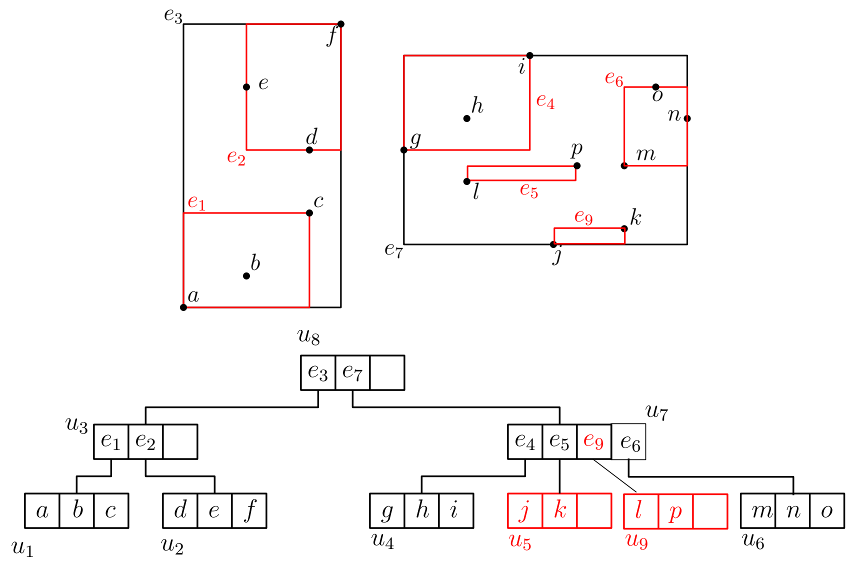


Solution. u_8, u_3, u_2.

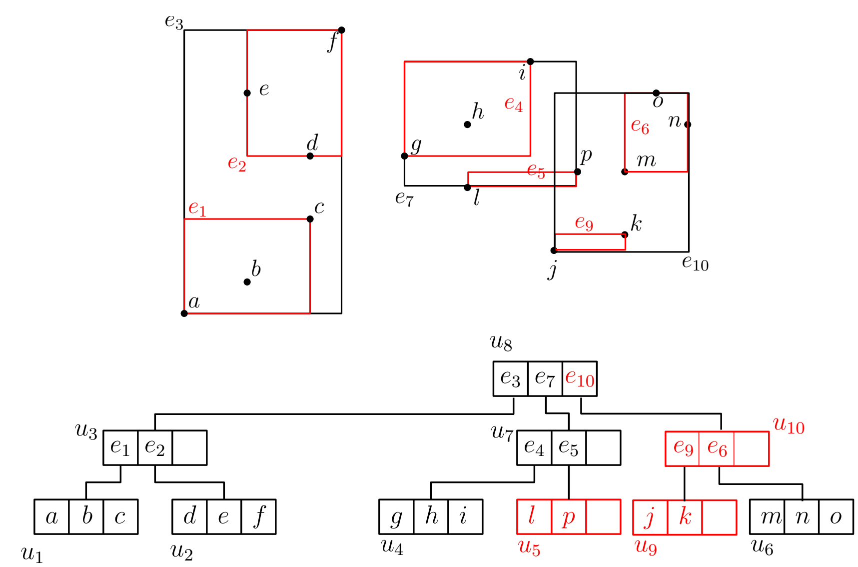
Problem 2. The figure below shows the MBRs of the R-tree in Problem 1 together with a new point p. Draw the MBRs of the R-tree after the insertion of p. Assume that each node can accommodate at most $B = 3$ elements.



Solution. It is easy to verify that the choose-subtree algorithm identifies node u_5 as the leaf node where p should be inserted (see the structure illustrated in Problem 1 for u_5). Adding p to u_5 causes the node to overflow, which is treated by the leaf-split algorithm. The figure below shows the MBRs and the structure of the R-tree after the split. Note that the MBR e_9 of the new node u_9 has been inserted into the internal node u_7.

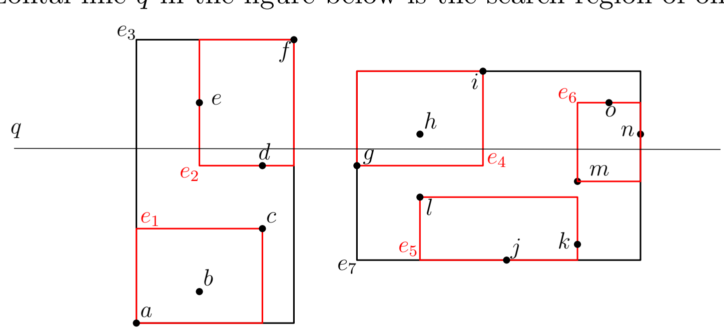


At this moment, however, u_7 overflows, which is treated by the internal-split algorithm. The next figure gives the MBRs and the final structure of the R-tree after the split.



Problem 3. Consider once again the R-tree in Problem 1. Give a range query that returns an empty result but needs to access 6 nodes of the tree.

Solution. The horizontal line q in the figure below is the search region of one such query.



Problem 4. Give an $O(B \log B)$-time implementation for the leaf split algorithm discussed in the class, where B is the maximum number of points in a leaf node.

Solution. Let S be the set of $m = B+1$ points in the (overflowing) leaf node u that the algorithm operates on. Recall that the algorithm processes each of the dimensions in turn. Next, we will describe how to process the x-dimension in $O(m \log m)$ time. The same algorithm works on the other dimensions as well.

As discussed in the class, the algorithm first sorts the points of S by x-coordinate—denote the sorted list by L)—and then, tries all the "possible splits" of L. Specifically, a possible split puts the first i points of L into a set $S_1[i]$, and the other $m-i$ points into another set $S_2[i]$, for every $i \in [\lceil 0.4B \rceil, m - \lceil 0.4B \rceil]$. Then, it obtains the perimeters of $MBR(S_1[i])$ and $MBR(S_2[i])$. It is thus clear that, if we can obtain the two MBRs in constant time, then the whole algorithm runs in $O(m \log m)$ time.

To achieve the purpose, we only need to perform two scans of L before trying all the possible splits. One scan will produce $MBR(S_1[1]), MBR(S_1[2]), ..., MBR(S_1[m])$ in an array, while the other scan will produce $MBR(S_2[1]), MBR(S_2[2]), ..., MBR(S_2[m])$ in another array. We will show that each scan takes only $O(m)$ time, and hence, does not affect the overall $O(m \log m)$ complexity.

Due to symmetry, it suffices to explain how to produce $MBR(S_1[1]), ..., MBR(S_1[m])$. First, $MBR(S_1[1])$ is simply the first point of L (i.e., a degenerated MBR). Inductively, for any $i \geq 2$, $MBR(S_1[i])$ can be obtained from $MBR(S_1[i-1])$ and the i-th point of L in constant time. This completes the description of the entire algorithm for Problem 4.

Remark. The above algorithm works for any constant dimensionality d in $O(m \log m)$ time.

Problem 5. Give a counterexample showing that the leaf split algorithm discussed in the class does not give an optimal split. (Hint: set $B \geq 6$.)

Solution. Set $B = 6$. Consider running the split algorithm on the 7 points below. The figure shows the MBRs of the optimal split, which cannot be discovered by the leaf-split algorithm.