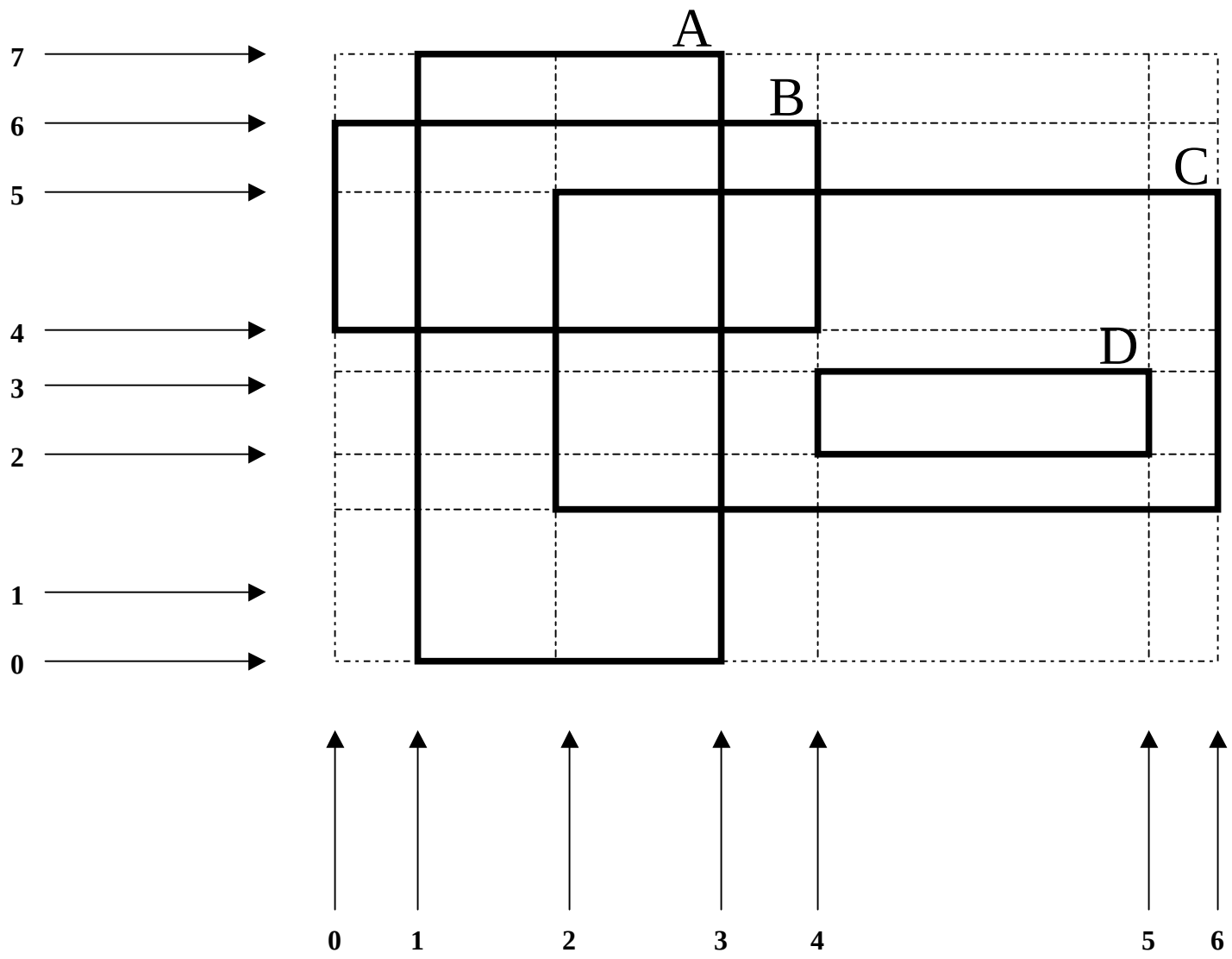
A
B
C
D
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6

Brian Faull
CS-G399 S04

Rectangle cover problem, PS2-3

Consider the rectangle covering problem on 4 rectangles A, B, C, and D, shown in the diagram on the previous page.

In general, this problem can be reduced in complexity significantly by reducing the domain and range of the rectangles to integers in the range [0,2n). The extent of each rectangle can be considered to be described by the height of its top and bottom edges, and by its left and right edges.

Begin by constructing a grid with blocks of unit 1 in the horizontal and vertical directions from the field of rectangles. Begin in the horizontal direction by defining the left-most rectangle to begin at 0 in the horizontal (x) direction. Continue by setting each subsequent (left or right) edge of any rectangle as subsequent integers (these are numbered below the diagram.) If any edges have precisely the same vertical point, they should be given the same value (as the right edge of rectangle B and the left edge of rectangle D in the diagram). Do the same for the vertical (y) direction.

From this, one will obtain a grid of at most (2n - 1)^2 = O(n^2) squares. These squares represent the m points {p1,...,pm} that make up P, the set of points from which to select a covering of these rectangles, such that each rectangle contains at least one point.

Next, for each square create the list of rectangles that overlap that square. This can be done in O(n^3) time by iterating over each rectangle (n), and adding the approprate squares in the matrix (maximum O(n^2)) with the rectangle identifier.

Now, an example of the set-cover problem has been constructed, (in polynomial time of O(n^3)), where the matrix of squares represent the sets (some of which may be empty), and the universe U of elements to cover are the identifiers of the rectangles.

Each iteration, select the set (square) that has the highest cost-effectiveness, defined as the set that has not yet been chosen that covers the highest number of rectangles. This set selection need be performed at most n times (at most n squares (points) are needed to cover the n rectangles), and each search would require no more than n^2 (or a total of n^2*log(n) time for the entire search and selection process if sorting is performed first).

It would be straightforward to select a set of points of these squares output that corresponds with the original, unmodified problem.

The running time is within O(n^3) considering all steps. The approximation guarantee is that which one could obtain from this greedy algorithm is Hn (or anything better that you could come up with for set cover). :)

Comparison to other work:
I found no references to this problem online, the rectangle problems that came up are for different problems. Two of these are

- completely cover an arbitrary polygon with rectangles, approximable to factor O(n log n)
  http://www.nada.kth.se/~viggo/wwwcompendium/node170.html

- cover a rectangular matrix of {0,1} with rectangles, approximable within factor 14.
  http://134.117.206.249:8000/www/resources/np-compendium/wwwcompendium/node169.html