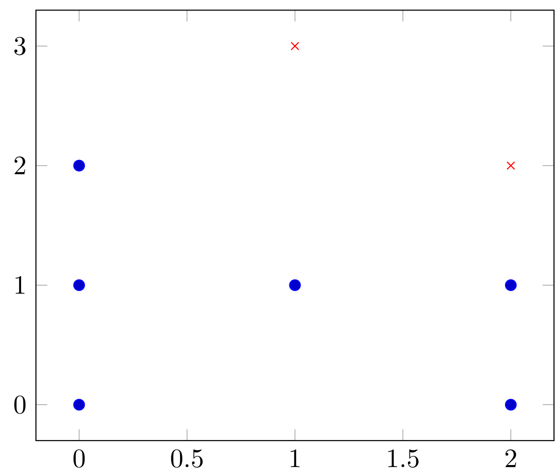
CS 374	Spring 2015

Homework 5

Nikhil Unni (nunni2)

3. 

The red x's represent the undominated points, while the rest are the dominated points.

Let **sortedX** be the sorted P, such that the comparitor compares x values such that the y values are the tie-breakers for two points with equal x values (Alternatively can be implemented by sorted by y values first, then doing a stable sort by x values);
maxY $\leftarrow -1$;
undominated $\leftarrow \emptyset$;
foreach (x_i, y_i) *in sortedX, starting from i:=n to 1* **do**
  if $y_i > maxY$ **then**
    undominated $\leftarrow$ undominated $\cup (x_i, y_i)$;
    maxY $\leftarrow y_i$;
  end
end
output undominated;

Algorithm 1: 3b

The algorithm sorts the points in increasing x order, where y is the tie-breaker for equal values of x. Then, iterating backwards through the sorted list, it keeps track of the current greatest Y value. If the current element has a Y value greater than our previously recorded Y value, it means that its x value is less than the "greatest Y value's" x, but the current element's Y value is greater, so it is an undominated point as well.

Inductively, we can see that if the current element's Y value is not greater than the current Y value, then there must be a value ahead in the sorted list with both a greater X and a greater Y value than the current element, meaning that it is, by definition, a dominated point.

The algorithm runs in O(nlogn) time because we only do a single sort, which is O(nlogn), plus the O(n) time to iterate backwards through the sorted list.