COMP 3711H – Fall 2016
Tutorial 1 – Sketch Solution to Max Gap Problem

In Tutorial 1, you were asked to find a $O(n)$ solution to the max-gap problem, given that you were allowed to use the floor function $\lfloor x \rfloor$ which truncates the fractional part of real number x.
This handout sketches some of the solution details.

- Extra Problem. *The limits of comparison-based lower bounds*
  Let $S = \{x_1, x_2, \ldots, x_n\}$ be a set of integers or real numbers. Let $y_1, y_2, \ldots, y_n$ be the same numbers sorted in increasing order. The *MAX-GAP* of the original set is the value

  $$\texttt{Max}_{1 \le i < n}(y_{i+1} - y_i).$$

  Using a more advanced form of the $\Theta(n \log n)$ proof of the lower bound for sorting it can be proven that calculating MAX-GAP requires $\Theta(n \log n)$ operations if only comparisons and algebraic calculations are used. In this problem, we will see that, if the floor (truncate) operator $\lfloor x \rfloor$ can also be used, the problem can be solved using only $O(n)$ operations!

  1. Find y_1 and y_n, the minimum and maximum values in S.
  2. Let $\Delta = \frac{y_n - y_1}{n-1}$. Let B_i be the half-closed half-open interval defined by

     $$B_i = [y_1 + (i-1)\Delta,\, y_1 + i\Delta)$$

     for $i = 1, 2, \ldots n-1$ and set $B_n = \{y_n\}$.
  3. Prune S as follows. For every B_i throw away all items in $S \cap B_i$ except for the smallest and largest. Let S' be the remaining set.
  4. Find the Max-Gap of S' by sorting S' and running through the items in S' in sorted order. Output this value

  Prove that this algorithm outputs the correct answer and show that every step can be implemented in $O(n)$ time.

First, we show correctness.

Note that $\sum_{i=1}^{n-1}(y_{i+1} - y_i) = y_n - y_i$.
Since there are $n-1$ terms $(y_{+1} - y_i)$, the maximum valued one must have value at least Δ.

If two points x, x' are in the same box B_k then $|x - x'| < \Delta$. So, the two points y_j, y_{j+1} forming the max gap must be in different boxes, with y_j the largest value in its box and y_{j+1} the smallest value in its box (with no points between them).

This means that step 4 of the algorithm will look at the $y_{j+1} - y_j$ after sorting S'. To prove the correctness of the algorithm it only remains to show that non of the new gaps that occur in S' are *larger* than $y_{j+1} - y_j \ge \Delta$.

This is easy to see since the only new gaps that are created are between two points in the same box (after removing points between them) and any gaps between points in the same box are $< \Delta$.

Now we see show the $O(n)$ running time.

Step 1 can be implemented in $O(n)$ time using two linear scans; the first to find y_1 and the second to find y_n.

In linear time, calculate for each item x_i the key

$$k_i = \lfloor \frac{x_i - y_1}{\Delta} \rfloor + 1.$$

By definition $x_i \in B_{k_i}$. **Note that this is the only place in the algorithm we use the floor function.**

Now create two new Arrays of size n, MAX and MIN.
$MAX[k]$ will store the index i of the largest x_i with key k;
$MIN[k]$ will store the index of the smallest x_i with key k.
If there is no item with key k, set $MAX[k] = MIN[k] = 0$.

It is easy to see that these arrays can be filled in properly in $O(n)$ time.

Filling in these arrays actually implements Step 3, since we are throwing away all but the largest and smallest items in each box, i.e., finding S'.

S' can now be sorted in $O(n)$ time as follows. For $i = 1$ to N DO
If $MAX[k] = MIN[k] = 0$, don't write anything.
If $MAX[k] = MIN[k] \neq 0$, write $x_{MIN[k]}$.
If $MAX[k] \neq MIN[k]$ write $x_{MIN[k]}, x_{MAX[k]}$.

Since this writes down the 0,1,2 remaining items in each box in order and the items in each box are written before the items in following boxes, this writes down the items in S' in order.