5.1

A possible l can be (maximum of the list - minimun of the list) / (number of numbers in the list- 1).

Proof by contradiction:

If maximum difference < $$l$$, then assume integers in the sorted list is $${a\_1, a\_2,...,a\_n}$$, $$ a\_{i+1} - a{i} < l$$ holds for $$\forall i \isin \{ 1, 2, ..., m-1\} $$

Obviously $$a\_1 $$ = minimun of the list and $$a\_n$$ = maximum of the list, $$l = (a\_n - a\_1) / (n - 1)$$

$$a\_n - a\_1 = (a\_n - a\_{n-1}) + (a\_{n-1} -a\_{n-2}) + ... + (a\_2 - a\_1) < l \* (n-1) = a\_n - a\_1$$ is contradictory

So maximum difference > $$l$$, l is a lower bound for the maximum difference

5.2

Step 0: n = number of elements in the array

Step 1: Scan the whole list to find minimum and maximum value, $$min$$ and $$max$$ respectively;

Step 2: $$l = \lfloor (max - min)/ n \rfloor, m = \lceil n/l \rceil $$, intialize $$m$$ buckets $$\{ bucket\_1, bucket\_2, ..., bucket\_m\}$$

Step 3: For number $$i$$ in the array, store $$i$$ into $$bucket\_{\lfloor (i - min)/l \rfloor + 1}$$

Step 4: Reindex non-empty buckets to be $$\{ bucket\_1, bucket\_2, ..., bucket\_p\}$$ where $$ p <= m $$

Step 5: Scan every buckets to find the minumum and maximum of each buckets, $$min\_i$$ and $$max\_i$$ where $$i \isin \{1,2,...,p \}$$

Step 6: Return $$max\{min\_2-max\_1, min\_3-max\_2, ..., min\_p - max\_{p-1}\}$$

Correctness:

For numbers in the same buckets, they are at most l apart from each other; according to result in 5.1(1), since the maximum difference is larger than l, the two numbers that forming the maximum difference can not from the same bucket (proof by contradiction).

So the two numbers must come from different buckets. However if they do not come from two consecutive buckets, they can not be consecitive numbers in the list when sorted; so they must come from two consecutive buckets, and they are the minimum number of the larger bucket and maximum number of the smaller bucket (proof by contradiction).

So the maximum difference $$\isin \{min\_2-max\_1, min\_3-max\_2, ..., min\_p - max\_{p-1}\}$$

Runtime:

Step 0 takes at most O(n);

Step 1 takes O(n) for a one-through scan and constant time operation (compare with current minimum and maximum value) at each number;

Step 2 takes O(n) since the number of buckets can not exceed n;

Step 3 takes O(n) for a one-through scan and constant time operation (calculate, access bucket, and insert);

Step 4 takes O(n) since the number of buckets can not exceed n;

Step 5 takes O(n) since every number in the list is accessed at most once;

Step 6 takes O(n) to find the maximum value of a list that is smaller than n

The total runtime of this algorithm is O(n)