**Running time complexity while using direct addressing:**

**ADD:**

O(N): N is number of Voter Ids in the file.

CountLines, getMinValue, getMaxValue - all runs in O(N) time so O(N) + O(N) + O(N) = 3 O(N).

Adding into array also runs in O(N) time as N elements being added to the array from the file so 3 O(N) + O(N) = 4 O(N). Constant can be dropped so O(N).

Anyways, in other words, we already know that we are using bin sort which runs in O(N) time.

**FIND:**

O(1): Runs in constant time. Voter Id supposed to be unique and array is sorted using bucket sort. Each element in its proper bucket and can be directly accessed using [voterId - minValue] as index to get the value from the array.

**COUNT**:

O(N): N is number of Voter Ids present in the array.

Need to traverse through whole array for finding a particular candidate id and do a count if a particular value exists.

**Running time complexity while using hash table:**

**ADD:**

O(N): N is number of Voter Ids in the file.

Each add operation runs in constant time O(1) as such but there are N add operations in order to add data from the file so running complexity is O(N) where N is number of Voter Ids in the file. Voter Id supposed to be unique so no chaining required. It can be directly used as keys without a hash function.

**FIND:**

O(1): Runs in constant time. Voter Id supposed to be unique, can be used directly as keys without a hash function and can be directly used to get the value.

**COUNT:**

O(N): N is number of Voter Ids present in the hash table.

Voter Ids are supposed to be unique so number of voter ids would also reflect number of Candidate Ids (number of values). And would need to traverse through all values in the hash table for finding a particular candidate id and do a count if a particular value exists.