1. Sort the array using any sorting method that runs in nlogn time. This could be mergesort, quicksort, etc... This will result in an array that has each candidate votes all organized next to each other. Simply iterate through the array with a counter to check if that candiates votes is greater then the maximum number of votes. If it is make that the new maximum and return that candiates ID number. This will have a run time of O(nlogn + n) = O(nlogn)

2. In order to get nlogk runtime, we can use a hashmap in which each key represents a candidate and the values in each key represent the number of votes each candidate has. We loop through the array inserting our values into the hashmap, and once finished we simply grab the key with the highest value. Inserting into a hashmap takes O(nlogk) and grabbing the highestr value key is negligible compared to nlogk

3 To achieve an O(N) time complexity, we can use a median of medians approach with by diving the array into groups of 5. Recursively find the medians of each group and partition the array around this pivot. We can than identify the median by counting how many times the median candidate appears. If its greater than (N+1)/2, we know who won with most votes. This gives us O(2N) which equals O(N)

4 Use a median of medians approach to partition the array into two groups ones smaller than N^.5 and ones larger than N^.5. Once completed with O(N) time, we can sort the smaller partition which is a runtime of (N^.5)log(N^.5) However, O(N) is still larger because of the power of the polynomial. Therefore this gives us a runtime of O(N) to seperate and sort elements smaller than N^.5.