**Examination of Radix Sort Algorithm**

**By**

**Wihar Paladugula**

**Department Of DataScience, University Of Maryland Baltimore County**

**Data Structures and Algorithms in Python**

**December 2023**

**Introduction:**

Radix sort is a non-comparison-based sorting algorithm i.e. it does not compare elements in a list like merge sort or quick sort, rather it sorts based on the digits of each number in the list, by adding the element to the respective bucket based on digit form least significant digit (LSD) to the most significant digit (MSD) or vice versa in an iterative way. It was a long ago famously used in the card-sorting algorithm for processing punched cards.

I chose Radix Sort because it has a unique approach to sorting by considering individual digits, making it efficient for a wide range of input values. It has a linear time complexity and performs well when the range of the input is not significantly larger than the number of elements.

**Algorithm:**

1. **Determine the maximum number of digits:** Initially find the maximum number in the list and we use the number of digits in that number to know the number of passes i.e. number of iterations required for the sorting process.
2. **Perform iterative sorting:** For each digit position starting from the least significant digit (i.e. the digit in ones place e.g. for number 342, 2 is in the ones place) perform the following steps:
   1. Create empty buckets: Create an array of buckets of size 10 each corresponding to a possible digit value(i.e. 0-9)
   2. Distribute elements into buckets: Traverse the input array and place each element into the bucket corresponding to its digit value at the current digit position.
   3. Collect elements from buckets: Iterate through buckets and collect the elements into the original list.
3. **Repeat until all digits are processed:** Continue the iterative sorting process, by increasing the digit position for each pass, until all digits are considered.

**Example:**

Let's consider the following unsorted array of numbers [237,146,259,348,152,163,235,48,36,62]. To sort this array using Radix sort we use the following steps:

Pass 1: Sort by the units or ones digit

* Create 10 empty buckets
* Distribute the elements based on unit digits which will result as follows:
  + Bucket 2: 152, 62
  + Bucket 3: 163
  + Bucket 5: 235
  + Bucket 6: 196, 36
  + Bucket 7: 237
  + Bucket 8: 348, 48
  + Bucket 9: 259
* Collecting elements from the bucket will result in a new list as follows: [152,62,163,235,196,36,237,348,48,259]

Pass 2: Sort the above created new list based on tens digits which will result in buckets as follows:

* Buckets: [235,36,237], [146,348,48], [152,259], [62,163]
* The new list will be [235,36,237,146,348,48,152,259,62,163]

Pass 3: Sort the above created new list based on hundreds digits which will result in buckets as follows:

* Buckets: [36, 48,62], [146,152,163], [235,237,259],[348]
* The final sorted list will be [36, 48,62,146,152,163,235,237,259,348]

**Pros of Radix sort:**

* Effective for handling big datasets: Radix sort works effectively with big integer sets, particularly when the elements have a constant amount of digits.
* Non-comparison-based: Radix sort is more effective for some kinds of data since it does not incur the overhead of comparing items.
* Stable sorting algorithm: Radix sort produces consistent results by maintaining the original order of elements with equal elements.

**Cons of Radix sort:**

* Inefficient for small datasets: Because comparison-based sorting algorithms have constant time complexity, they might be more effective for small datasets.
* Fixed-size keys: Radix sort presumes that the elements have a set number of digits, which might not be appropriate for all kinds of data.

**Computational Complexity:**

* **Time Complexity**: O(d \* (n + k)) = O(n), where n is the number of elements, k is the range of input values, and d is the number of digits in the maximum number.
* **Space Complexity**: O(n + k), where n is the number of elements, and k is the range of input values.

**Pseudo code:**

RadixSort(arr):

Find the maximum number to know the number of digits (d)

for pass in 1 to the number of digits in the maximum number:

Counting Sort the array based on the current digit

Copy the output array to arr

**Reference:**

Gallagher, P., & Jain, S. (2023, October 25). Radix Sort - Data Structures and Algorithms Tutorials. GeeksforGeeks, from <https://www.geeksforgeeks.org/radix-sort/>

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. McGraw-Hill.