2 Chapter 9: Sorting in Linear Time

2.1 Counting Sort

(a) Rank each element

i. Count how many times i occurs

ii. Prefix sum on counter array to find how many items $\leq i$

(b) Place at its location.

Start from right, place item in the output array, decrease its corresponding count.

$W(n) = O(n)$ if range is $O(n)$.

Stable sorts – counting sort, mergesort, Insertion sort, Bucket sort.

Nonstable sorts – Quicksort, heapsort.

2.2 Bucket Sort

k Buckets.

Algorithm:

1. hash items among buckets
2. sort the buckets
3. Combine buckets

Let there be k buckets, n items

1. distribution $O(n)$
2. sort the buckets
   $w(n) = O(n \log n)$
   $A(n) = O(k\frac{n}{k} \log \frac{n}{k}) = O(n \log(n/k))$
3. combine buckets $O(n)$.

Thus, bucket sort is

$$w(n) = O(n \log n)$$

$$A(n) = O(n \log \frac{n}{k})$$

If k is constant,

$$A(n) = O(n \log n - n \log k)$$
$$= O(n \log n)$$

$A(n) = O(n)$ if $k = n/20$, $A(n) = O(n)$

Good when item distribution is known so that bucket get equitable number of keys.

Space Usage

worst-case: each bucket should have space for n key (any allocation)

$\Rightarrow$ total $= O(nk)$

Thus, as k increases, average space increases but so does the space requirement.

If linked allocation is used

Space needed $= O(k) + O(n) = O(n + k) = O(n)$

However sorting each bucket using quicksort, mergesort, and heapsort will be difficult which require array representation.

If insertion sort is used to sort linked list, (buckets),

$$A(n) = O(\frac{n^2}{k^2}) * k = O(\frac{n^2}{k}) = O(n) \text{ for k} = \text{O(n)}.$$

Pseudocode
Input: Array A[1..n] with values in [0, 1)

1. Initialize k empty buckets: B[0], B[1], ..., B[k - 1]

2. // Step 1: Distribute items into buckets
   for i = 1 to n:
      index floor(k * A[i])
      insert A[i] into bucket B[index]

3. // Step 2: Sort each bucket
   for i = 0 to k - 1:
      sort B[i] (e.g., using insertion sort)

4. // Step 3: Concatenate buckets back into A[]
   index 1
   for i = 0 to k - 1:
      for each element x in B[i]:
         A[index] x
         index index + 1

2.3 Radix sort

for $i \leftarrow 1$ to d

stable sort A on digit i.

(a) counting sort can be used

(b) bucket sort can also be used

Pseudocode

Question 1: Give a modified algorithm for Counting Sort which does not use an output array and is able to generate sorted output in the input array itself. You may use a count array and an additional constant number of variables. Additionally, the Counting Sort need not remain stable.

Question 2: Prove that Radix sort, where array is sorted iteratively starting from least significant digit, is correct by induction on the number of digits. You can assume that stable sort employed works correctly.

Question 3: Explain how Bucket Sort achieves () O(n) time complexity when the number of buckets k is proportional to and describe under what assumptions this time complexity holds. Discuss how the choice of sorting algorithm for individual buckets affects the overall time and space complexity.

Question 4: Given a dataset where keys are uniformly distributed across a known range, describe how you would design the bucket sizes and hashing function in Bucket Sort to optimize both time and space usage. What challenges arise if the data distribution is highly skewed?

Question 5: Give a modified algorithm for Bucket Sort that minimizes extra space usage when the number of buckets is large relative to . Explain the trade-offs between time complexity and space efficiency in your design.

Question 6: Prove that Counting Sort works correctly by induction on the size of the input array n, assuming the key range is fixed and small. Show how the prefix sum step guarantees correct placement.

Question 7: Assume you are using Radix Sort to sort alphanumeric strings (letters and numbers) of fixed length. Describe how the choice of base (e.g., base-10, base-26, or base-36) affects the performance and what stable sort would be best suited for each case.

Question 8: For a modified Radix Sort that processes from most significant digit (MSD) first instead of LSD, explain how the stability of the inner sort affects correctness. Provide an example where instability causes failure.