Counting Sort & Radix Sort

Counting Sort

• Counting sort is a sorting algorithm which takes the advantage of knowing the range of the numbers in the input array A to be sorted.

• It uses this range to create an array *C* of this length. Each index *i* in *C* is then used to count how many elements in *A* have a value less than *i*. The counts stored in *C* can then be used to put the elements in *A* into their right position in the resulting sorted array.

• If the minimum and maximum values of *A* (range of numbers) are not known, an initial pass of the data will be necessary to find these.

<u>Algorithm:</u> Counting_Sort(A, B, C, k)

Here A[1,…n] is the input array having the range of items 0…k. B[1,…n] is the output array. C[0….k] is the temporary storage.

1. for i = 0 to k do C[i] := 0 [Initialize array C]
2. for j = 1 to length[A] [C[i] contains the no. of elements equal to i]
3. do C[A[j]] := C[A[j]]+1
4. for i = 1 to k [C[i] contains the no. of elements less than or equal to i]
5. do C[i] := C[i]+ C[i-1]
6. for j = length[A] to 1
7. do B[C[A[j]]] := A[j]
8. C[A[j]] := C[A[j]]-1
9. End.

Example

Given A[8] =

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C[5] =

0	1	2	3	4	5
2	0	2	3	0	1

B[5] =

1	2	3	4	5	6	7	8

N = 8 and k = 5

Steps:

1. for j = 1 to 8 do C[A[j]] := C[A[j]]+1.

C[5] =

0	1	2	3	4	5
2	0	2	3	0	1

2. for i = 1 to 5   do C[i] := C[i] + C[i-1]

C[5] =

0	1	2	3	4	5
2	2	4	7	7	8

3. for j = 8 to 1   do B[C[A[j]]] := A[j] and C[A[j]] := C[A[j]]-1

<u>for j = 8</u>

B[5] =

1	2	3	4	5	6	7	8
						3	

C[5] =

0	1	2	3	4	5
2	2	4	6	7	8

<u>for j = 7</u>

B[5] =

1	2	3	4	5	6	7	8
	0					3	

C[5] =

0	1	2	3	4	5
1	2	4	6	7	8

for j = 6

B[5] =

1	2	3	4	5	6	7	8
	0				3	3	

C[5] =

0	1	2	3	4	5
1	2	4	5	7	8

for j = 5

B[5] =

1	2	3	4	5	6	7	8
	0		2		3	3	

C[5] =

0	1	2	3	4	5
1	2	3	5	7	8

for j = 4

B[5] =

1	2	3	4	5	6	7	8
0	0		2		3	3	

C[5] =

0	1	2	3	4	5
0	2	3	5	7	8

for j = 3

B[5] =

1	2	3	4	5	6	7	8
0	0		2	3	3	3	

C[5] =

0	1	2	3	4	5
0	2	3	4	7	8

for j = 2

B[5] =

1	2	3	4	5	6	7	8
0	0		2	3	3	3	5

C[5] =

0	1	2	3	4	5
0	2	3	4	7	7

for j = 1

B[5] =

1	2	3	4	5	6	7	8
0	0	2	2	3	3	3	5

C[5] =

0	1	2	3	4	5
1	2	3	4	7	7

Sorted Output:

1	2	3	4	5	6	7	8
0	0	2	2	3	3	3	5

Problem of Counting Sort

The length of the counting array *C* must be at least equal to the range of the numbers to be sorted (that is, the maximum value minus the minimum value plus 1). This makes counting sort impractical for large ranges in terms of time and memory need.

Complexity of Counting Sort

Counting sort has a running time of $\Theta(n+k)$, where n and k are the lengths of the arrays A (the input array) and C (the counting array), respectively. In order for this algorithm to be efficient, *k* must not be too large compared to n. As long as *k* is $O(n)$, the running time of the algorithm is $\Theta(n)$.

Radix Sort

- A **radix sort** is a sorting algorithm that can rearrange a set of items based on the processing of part of the items' keys in such a way that items are eventually sorted alphabetically or in either ascending or descending order.

- Classifications of radix sort:

1. **Least Significant Digit (LSD) radix sort**

   Start from the least significant digit and move the processing towards the most significant digit.

2. **Most Significant Digit (MSD) radix sort**

   Start from the most significant digit and move the processing towards the least significant digit.

- For decimal number, the radix is 10 i.e. 10 buckets are required.

- For alphabetic information, the radix is 26.

- In radix sort, the total number of passes needed for sorting depends on the maximum number of digits or letters present in the given items. For example, suppose given items are 1020, 3, 22, 393, 200. For rearranging these items, 4 passes are required in radix sort.

- **Example:**

Suppose given items are as follows:

348, 143, 361, 423, 53, 128, 321, 543, 366, 202

Sort the given items using LSD Radix Sort.

<u>Pass 1:</u> The units digits are sorted into bins.

Input	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
348									348	
143				143						
361		361								
423				423						
053				053						
128									128	
321		321								
543				543						
366							366			
202			202							

Pass 2: The tens digits are sorted into bins.

Input	0	1	2	3	4	5	6	7	8	9
361							361			
321			321							
202	202									
143					143					
423			423							
053						053				
543					543					
366							366			
348					348					
128			128							

Pass 3: The tens digits are sorted into bins.

Input	0	1	2	3	4	5	6	7	8	9
202			202							
321				321						
423					423					
128		128								
143		143								
543						543				
348				348						
053	053									
361				361						
366				366						

Sorted Output: 53, 128, 143, 202, 321, 348, 361, 366, 423, 543

Complexity of Radix Sort

Suppose a list of n items A_1, A_2….A_n is given. Let d denote the radix and each item A_i is represented by means of s of digits. That is, $A_i = d_{i1}d_{i2}……d_{is}$.

The radix sort will require s passes. So the number C(n) of comparisons is bounded as follows:

$C(n) <= d * s * n = O(s*n)$……………………………………..(1)

In worst case, s = n, so $C(n) = O(n * n) = O(n^2)$.

In best case, $s = \log_d n$, so $C(n) = O(n\log_d n)$.

<u>Thank You</u>