Radix sort

1 Description

Radix Sort is a non-comparative sorting algorithm with asymptotic complexity $O(nd)$.

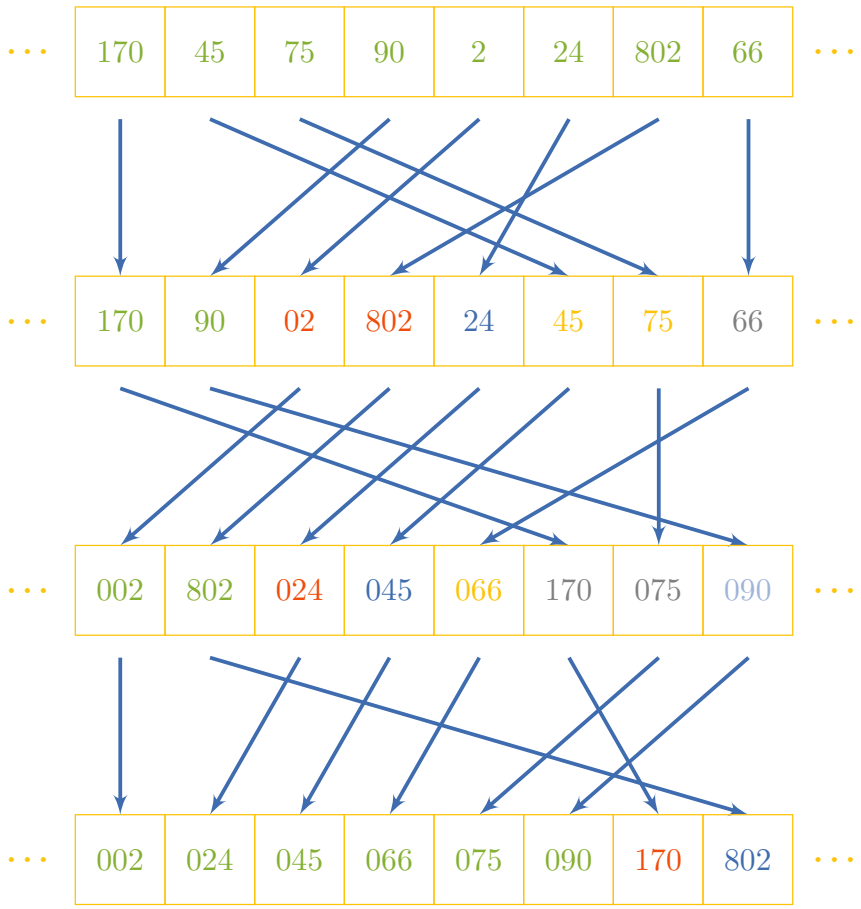
It is one of the most efficient and fastest linear sorting algorithms. Radix sort was developed to sort large integers. As an integer is treated as a string of digits so we can also call it a string sorting algorithm.

1.1 General idea

In radix sort, the elements are first sorted based on the last digit (least significant digit).
Then the result is again sorted by the second digit.
This process is repeated for all digits until the most significant digit is reached.

1.2 An example

We consider the array containing the elements $[170; 45; 75; 90; 2; 24; 802; 66]$
Sorting the array using the *radix sort* algorithm is performed in 3 steps.



1.3 Pseudocode

Algorithme 1 : Radix sort

```
Data : A, d
Result : A
/* It works same as counting sort for d number of passes. Each key in A[1..n] is a d-digit
   integer. (Digits are numbered 1 to d from right to left.)                          */
for j = 1 to p do
    /* A[]- Initial Array to Sort                                                     */
    int count[10] = 0;
    /* Store the count of "keys" in count[] key- it is number at digit place j        */
    for i = 0 to n do
        count[key of(A[i]) in pass j]++
    end
    for k = 1 to 10 do
        count[k] = count[k] + count[k-1]
    end
    /* Build the resulting array by checking new position of A[i] from count[k]       */
    for i = n - 1 to 0 do
        result[ count[key of(A[i])] ] = A[j]
        count[key of(A[i])]-
    end
    /* Now main array A[] contains sorted numbers according to current digit place    */
    for i = 0 to n do
        A[i] = result[i]
    end
end
```

2 Questions

In this set of question, we assume that the GPU has p compute unit and the array to sort contains n elements.

2.1 Design

In the first part, we consider the design aspect of the algorithm in parallel.

- Perform the first stage of the Foster methodology (decomposition).
  - What is the granularity of your decomposition ?
  - How did you select it ?
  - What form of decomposition did you select ?
- Perform the second stage of the Foster methodology (communication).
  - How many exchanges are required ?
  - What is the volume of data to be exchange ?
  - Does it depends on the number of data ?
- Perform the third stage of the Foster methodology (grouping).
  - How did you decide the size of the grouping ?
  - Does the grouping depend on the number od data ?

2.2 GPU consideration

We assume that we have a classical GPU organization of the p cores.

- How will you organize the computation on p compute units organized in warp ?
- What constraints do you need to consider ?