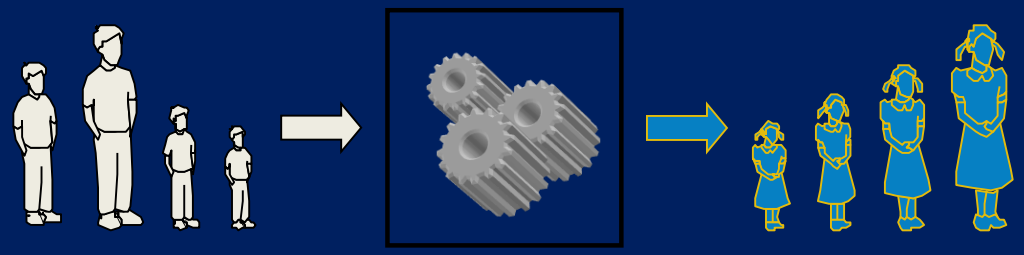
GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC

DESIGN & ANALYSIS OF ALGORITHM (BCSC0012)

Chapter 4: Sorting

Shell Sort



Prof. Anand Singh Jalal

Department of Computer Engineering & Applications

Shell Sort

- Invented by Donald Shell in 1959.
- 1st algorithm to break the quadratic time barrier but few years later, a sub quadratic time bound was proven
- Shell sort works by comparing elements that are **distant** rather than adjacent elements in an array.

- Shell sort uses a sequence $h_1, h_2, \ldots, h_t$ called the ***increment sequence***. Any increment sequence is fine as long as $h_1 = 1$ and some other choices are better than others.

Shell Sort

- Shell sort makes multiple passes through a list and sorts a number of equally sized sets using the insertion sort.
- Shell sort improves on the efficiency of insertion sort by shifting values to their destination.

  Shell sort is also known as ***diminishing increment sort***.

  The distance between comparisons decreases as the sorting algorithm runs until the last phase in which adjacent elements are compared

Empirical Analysis of Shell sort (Advantage)

- Advantage of Shell sort is that its only efficient for medium size lists. For bigger lists, the algorithm is not the best choice. Fastest of all O(N^2) sorting algorithms.
- 5 times faster than the bubble sort and a little over twice as fast as the Insertion sort, its closest competitor.

Empirical Analysis of Shell sort (Disadvantage)

- Disadvantage of Shell sort is that it is a complex algorithm and its not nearly as efficient as the **merge**, **heap**, and **quick sorts**.
- The shell sort is still significantly slower than the merge, heap, and quick sorts, but its relatively simple algorithm makes it a good choice for sorting lists of less than 5000 items unless speed important. It's also an excellent choice for repetitive sorting of smaller lists.

Shell Sort

- Comparison based sorting technique.
- Starts with gap = |_n/2 _| , where n = no. of elements.
- One by one select elements to the right of the gap and place them at their appropriate position.

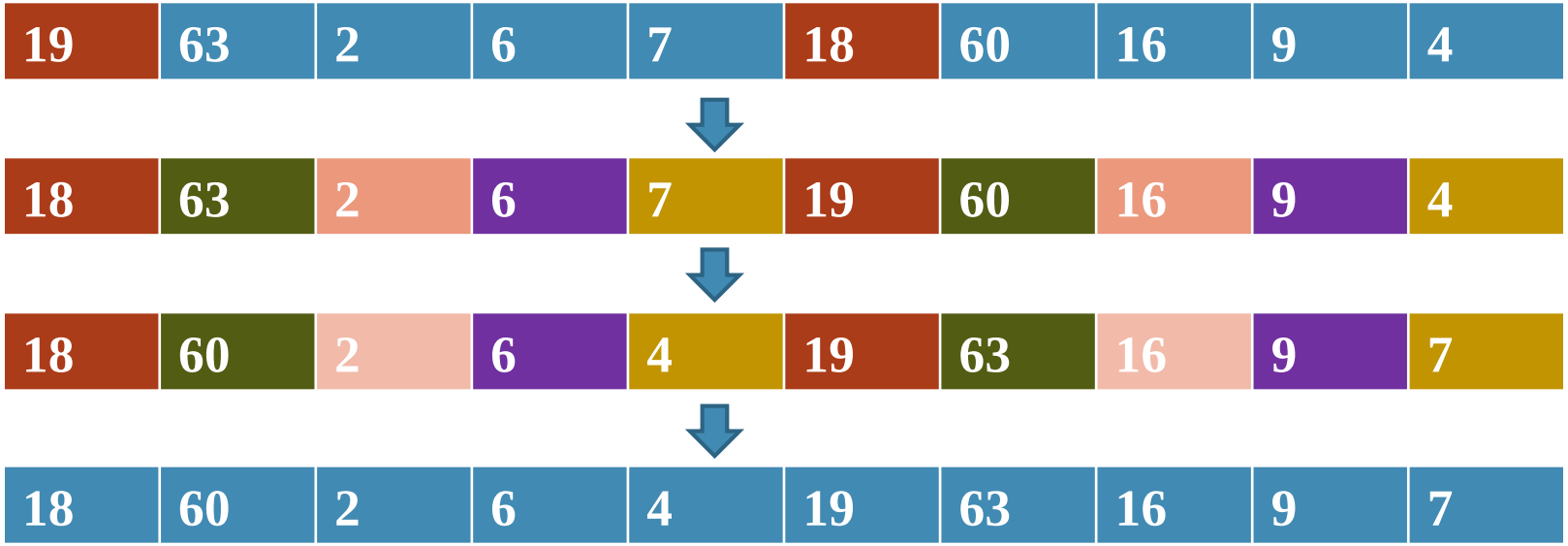
Time Complexity	
Best	O(nlog n)
Worst	$O(n^2)$
Average	O(nlog n)
Space Complexity	O(1)

Shell Sort: Example

19	63	2	6	7	18	60	16	9	4

Solution:
N=10 ; gap = |_ n/2 _| = 5

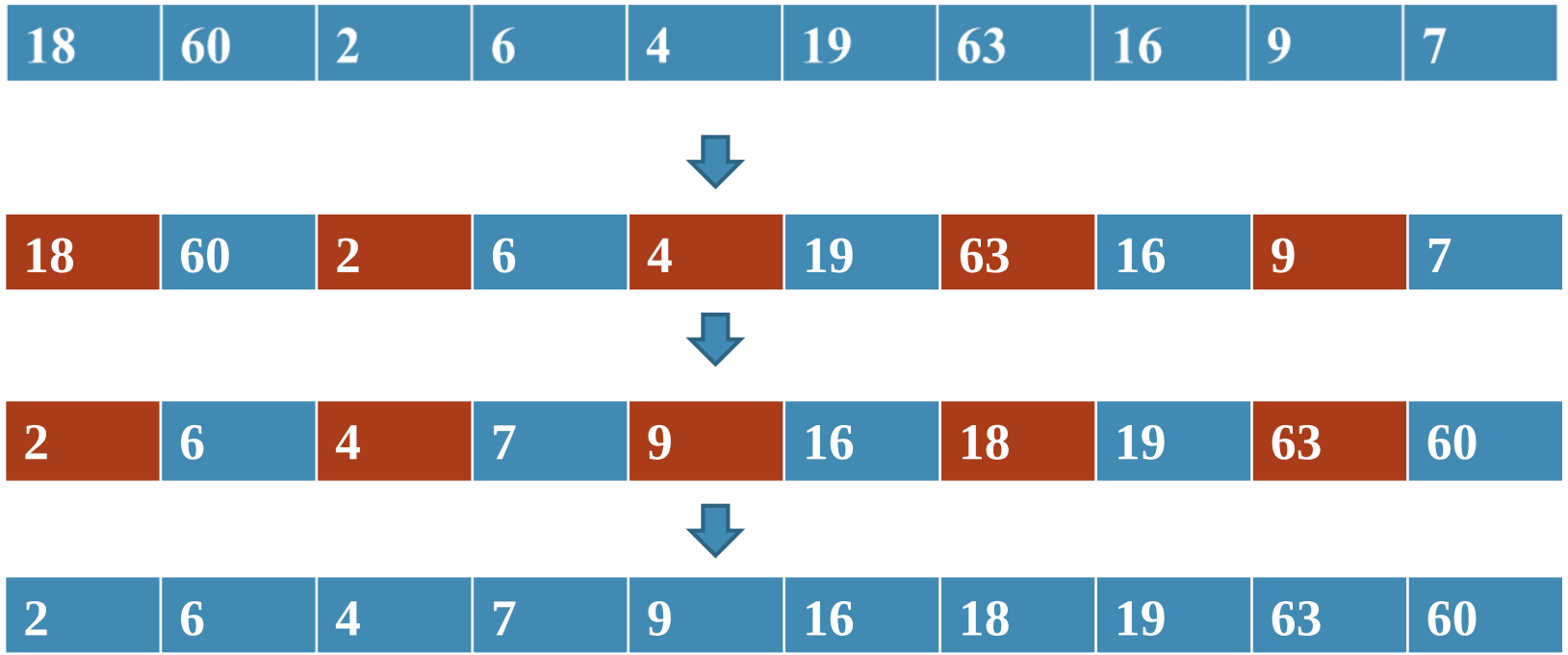
PASS I:



Shell Sort: Example ...

Pass II:

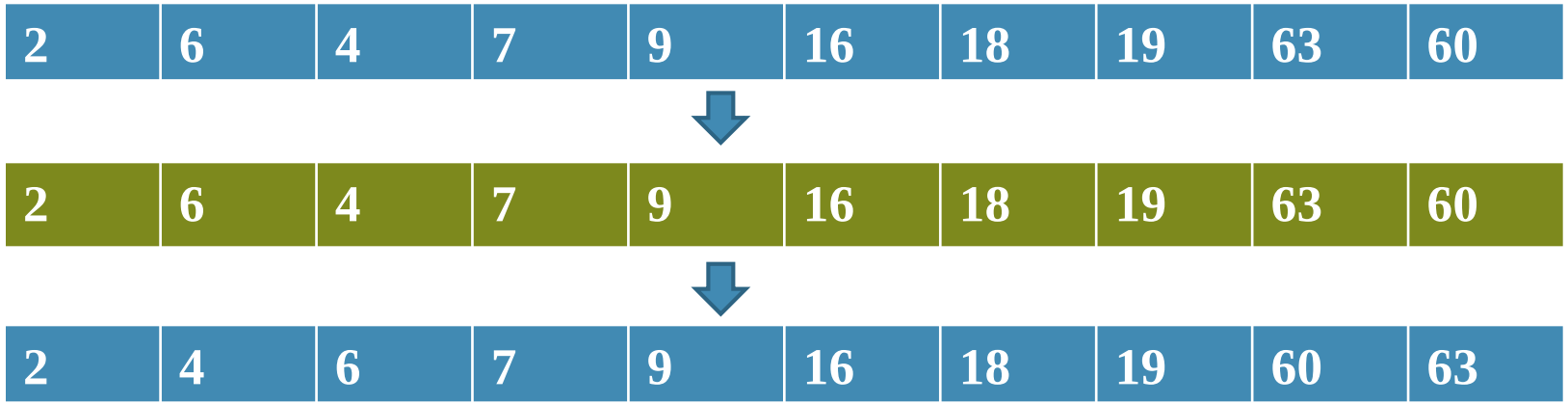
gap = |_ 5/2 _| = 2



Shell Sort: Example ...

Pass III:

gap = |_ 2/2 _| = 1



Shell Sort: Example 2

EXAMPLE 2:

18	32	12	5	38	33	16	2

Shell Sort: Example 2 ...

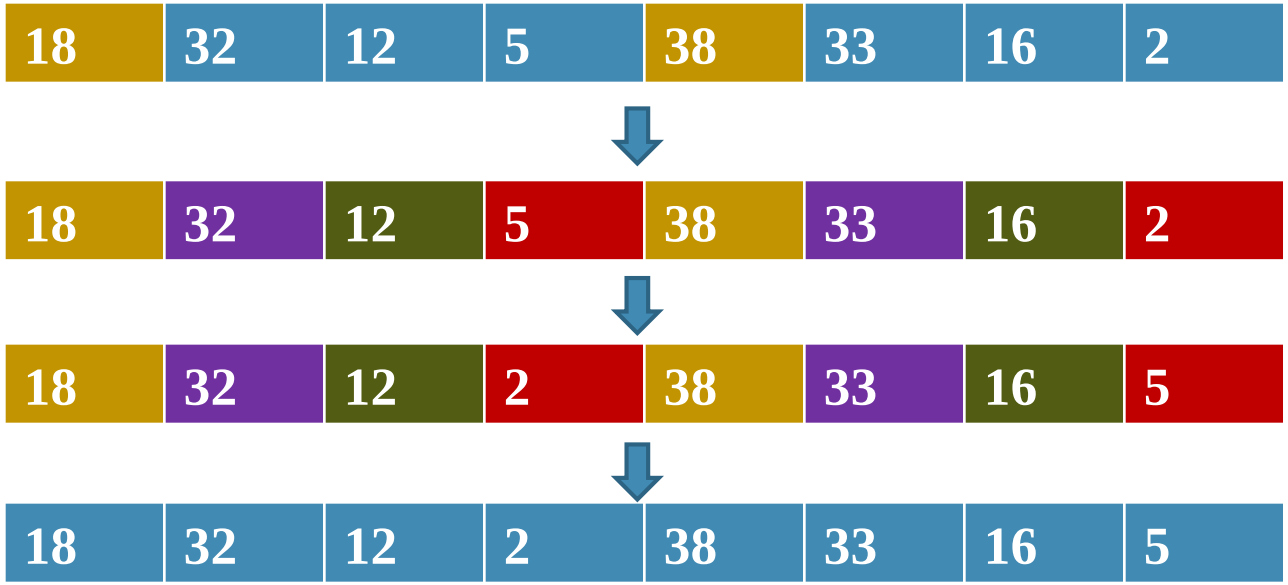
EXAMPLE 2:

18	32	12	5	38	33	16	2

Solution:

N=8 ; gap = |_ n/2 _| = 4

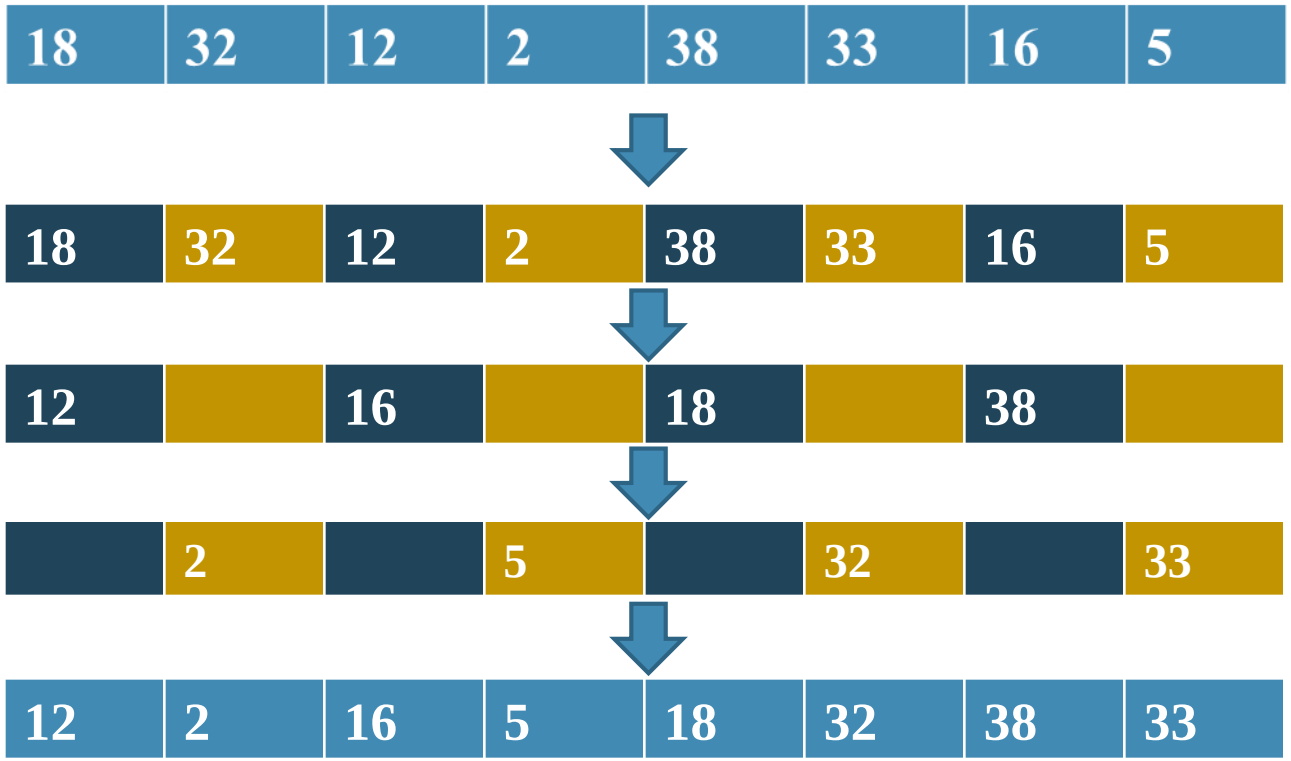
PASS I:



Shell Sort: Example 2 ...

PASS II:

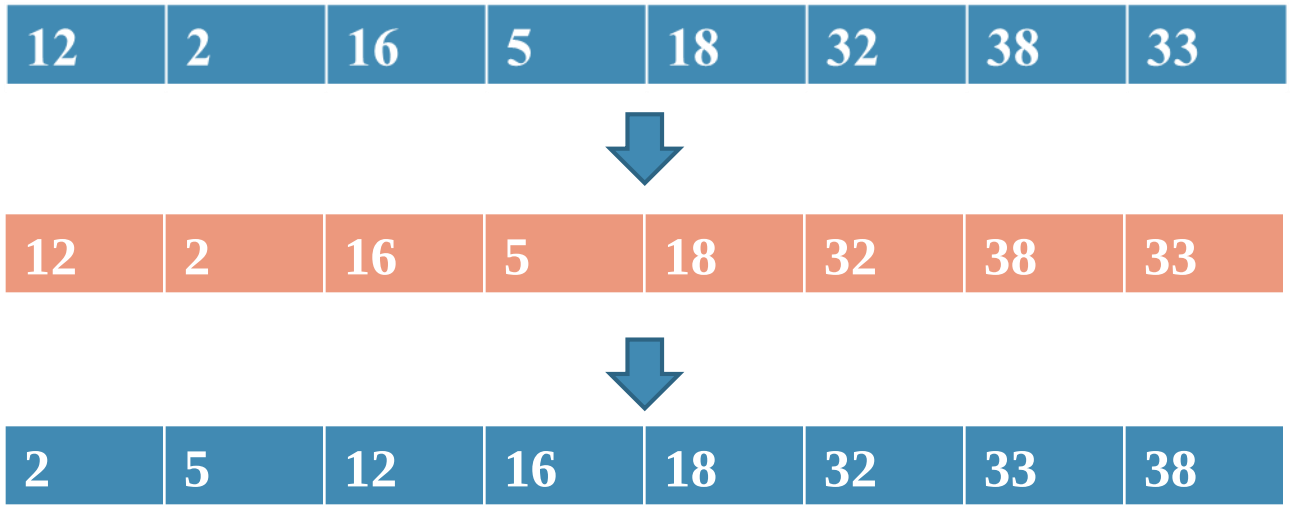
gap = |_ 4/2 _| = 2



Shell Sort: Example 2 ...

PASS III:

gap = |_ 2/2 _| = 1



Shell Sort: Algorithm

Shell Sort(A, n)

```
for( gap = n/2; gap>0; gap = gap/2)
{
   for(i = gap; i < n; i++)
   {
      temp=A[i];
       for(j=i; j>=gap && A[j-ap]>temp; j=j-gap)
       {
          A[j]= A[j-gap]
        }
        A[j]=temp;
     }
}
```

12	34	54	2	3

N=5, gap= |_ 5/2_| = 2

Any Questions ?

Dr. Anand Singh Jalal
Professor
Email: asjalal@gla.ac.in