SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : **19CS307**- DATA STRUCTURES

II YEAR / III SEMESTER

Unit V- **SORTING AND SEARCHING**

Topic :INSERTION, SHELL AND SELECTION SORT

Insertion , Shell and Selection Sort

Insertion Sort

- Insertion sort is based on the idea that one element from the input elements is consumed in each iteration to find its correct position
- i.e, the position to which it belongs in a sorted array.
- It iterates the input elements by growing the sorted array at each iteration
- It compares the current element with the largest value in the sorted array.

Cont..

- If the current element is greater, then it leaves the element in its place and moves on to the next element else it finds its correct position in the sorted array and moves it to that position.
- This is done by shifting all the elements, which are larger than the current element, in the sorted array to one position ahead

Syntax

- void insertion_sort (int A[ ] , int n)
- {
- for(int i = 0 ;i < n ; i++)
- {
- /*storing current element whose left side is checked for its correct position .*/
- int temp = A[i];
- int j = i;

Cont..

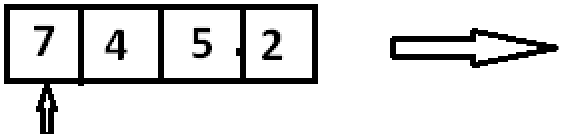
- /* check whether the adjacent element in left side is greater or less than the current element. */
- while(j > 0 && temp < A[j -1])
- {
- // moving the left side element to one position forward.
- A[j] = A[j-1];
- j= j - 1;

Cont..

- }
- // moving current element to its correct position.
- A[j] = temp;
- }
- }

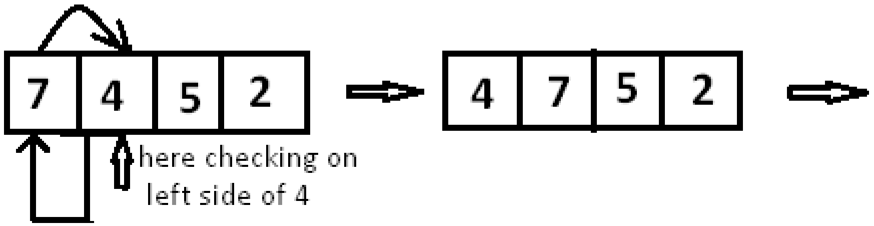
Example

STEP 1.



No element on left side of 7,so no change in its position .

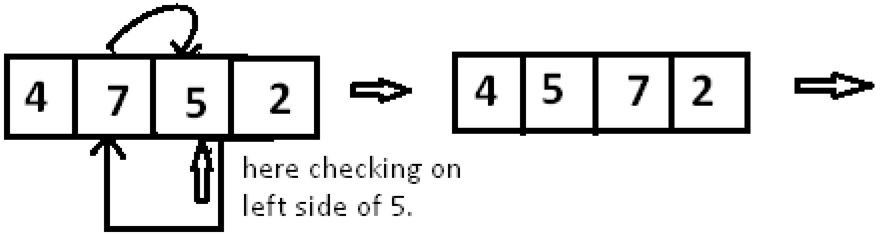
STEP 2.



here checking on left side of 4

As 7>4 ,therfore 7 will be moved forward and 4 will be moved to 7's position.

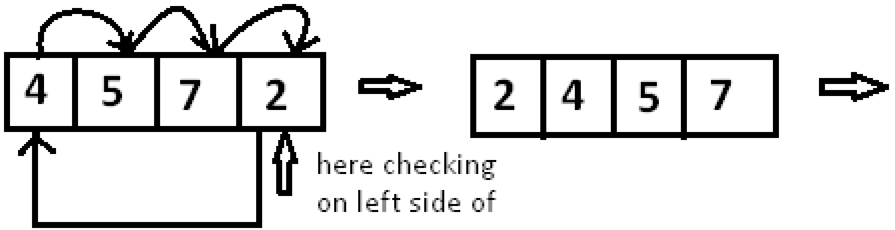
STEP 3.



here checking on left side of 5.

As 7>5,7 will be moved forward,but 4 <5,so no change in position of 4. And 5 will be moved to position of 7.

STEP 4.



here checking on left side of 2

As all the element on left side of 2 are greater than 2,so all the elements will be moved forward and 2 will be shifted to position of 4

Shell Sort

- Shell sort is an algorithm that first sorts the elements far apart from each other and successively reduces the interval between the elements to be sorted.
- It is a generalized version of insertion sort.
- In shell sort, elements at a specific interval are sorted.
- The interval between the elements is gradually decreased based on the sequence used

Shell Sort Algorithm

- shellSort(array, size)
- for interval i <- size/2n down to 1
- for each interval "i" in array
- sort all the elements at interval "i"
- end shellSort

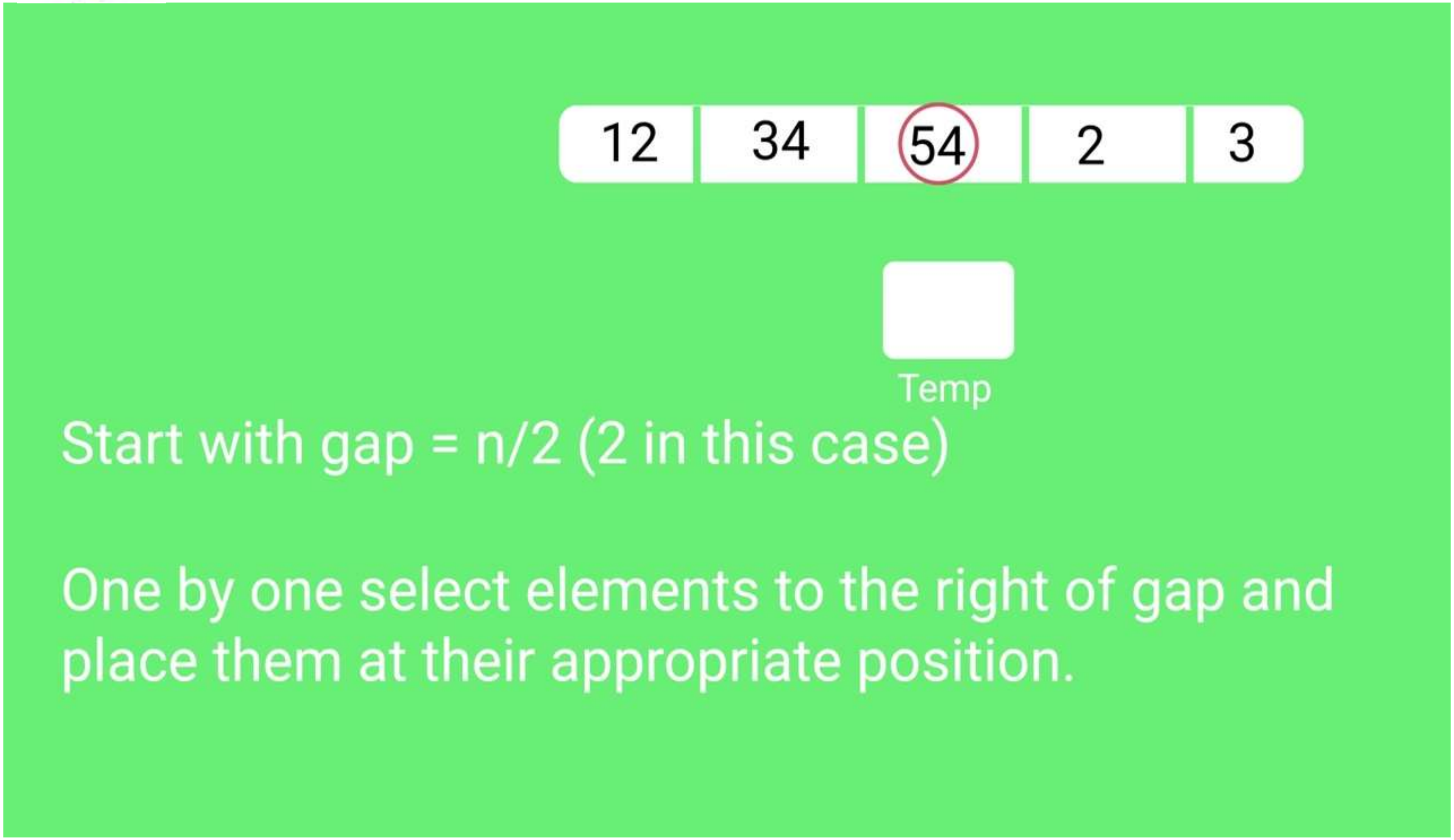
Syntax

```
✓ #include <stdio.h>
✓ void shellSort(int array[], int n){
✓ for (int gap = n/2; gap > 0; gap /= 2){
✓ for (int i = gap; i < n; i += 1) {
✓ int temp = array[i];
✓ int j;
✓ for (j = i; j >= gap && array[j - gap] > temp; j -= gap){
✓ array[j] = array[j - gap];
✓ }
✓ array[j] = temp;
✓ }}}
```

Cont..

```c
void printArray(int array[], int size){
for(int i=0; i<size; ++i){
printf("%d ", array[i]);
}
printf("\n");
}
int main(){
int data[]={9, 8, 3, 7, 5, 6, 4, 1};
int size=sizeof(data) / sizeof(data[0]);
shellSort(data, size);
printf("Sorted array: \n");
printArray(data, size);
}
```

Example

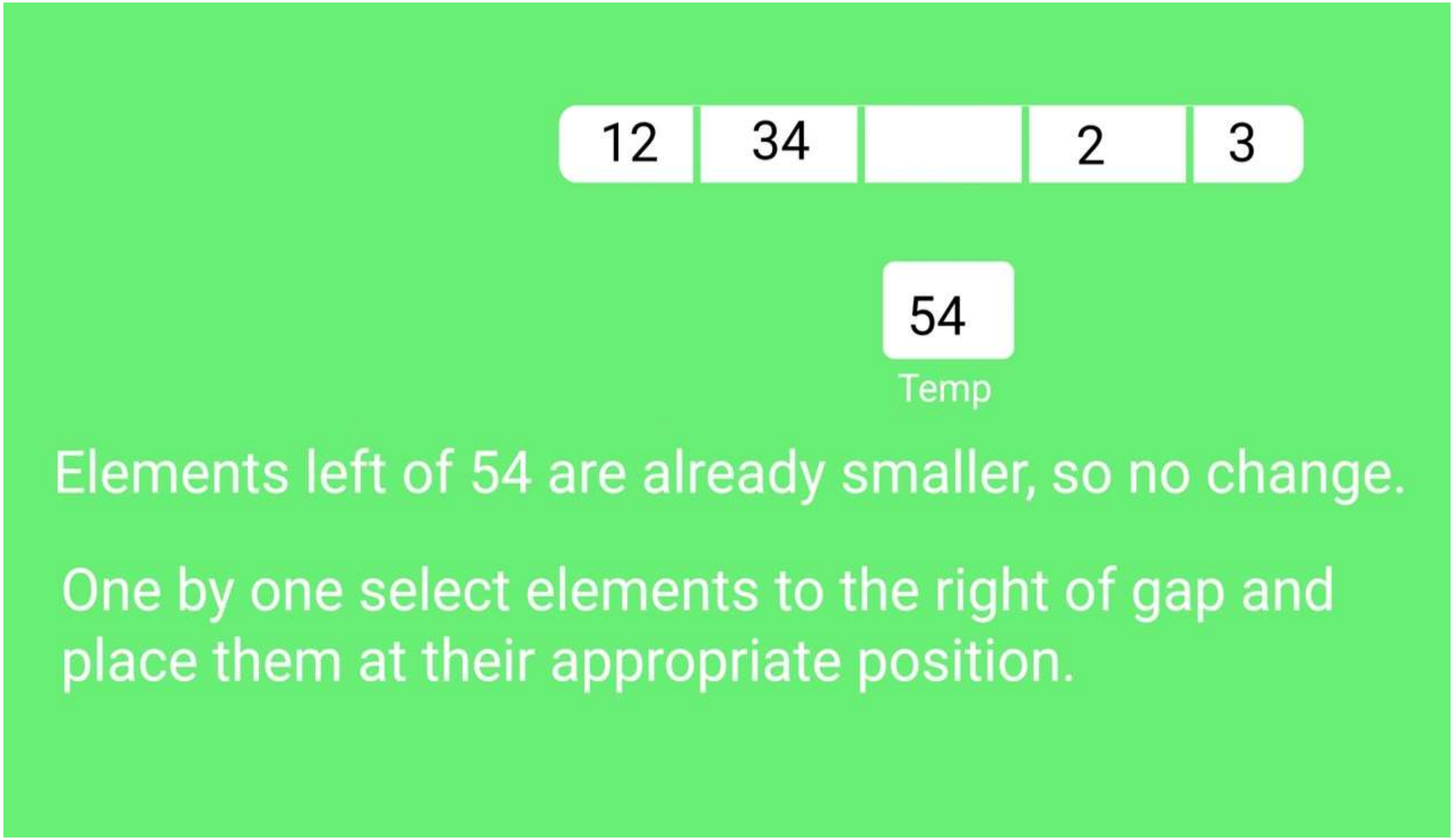


12	34	54	2	3

Start with gap = n/2 (2 in this case)

One by one select elements to the right of gap and place them at their appropriate position.

Cont..



12	34		2	3

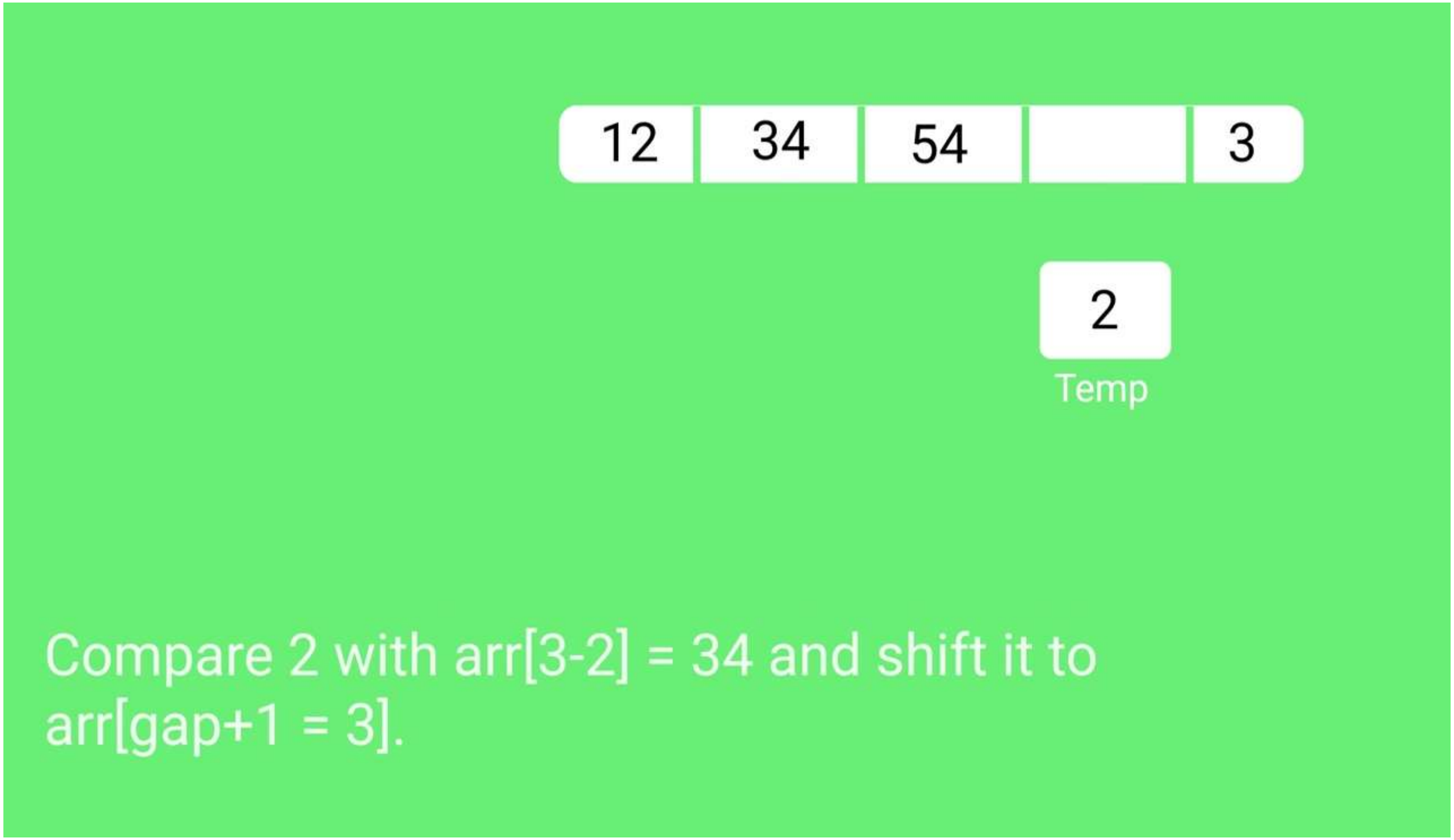
54

Temp

Elements left of 54 are already smaller, so no change.

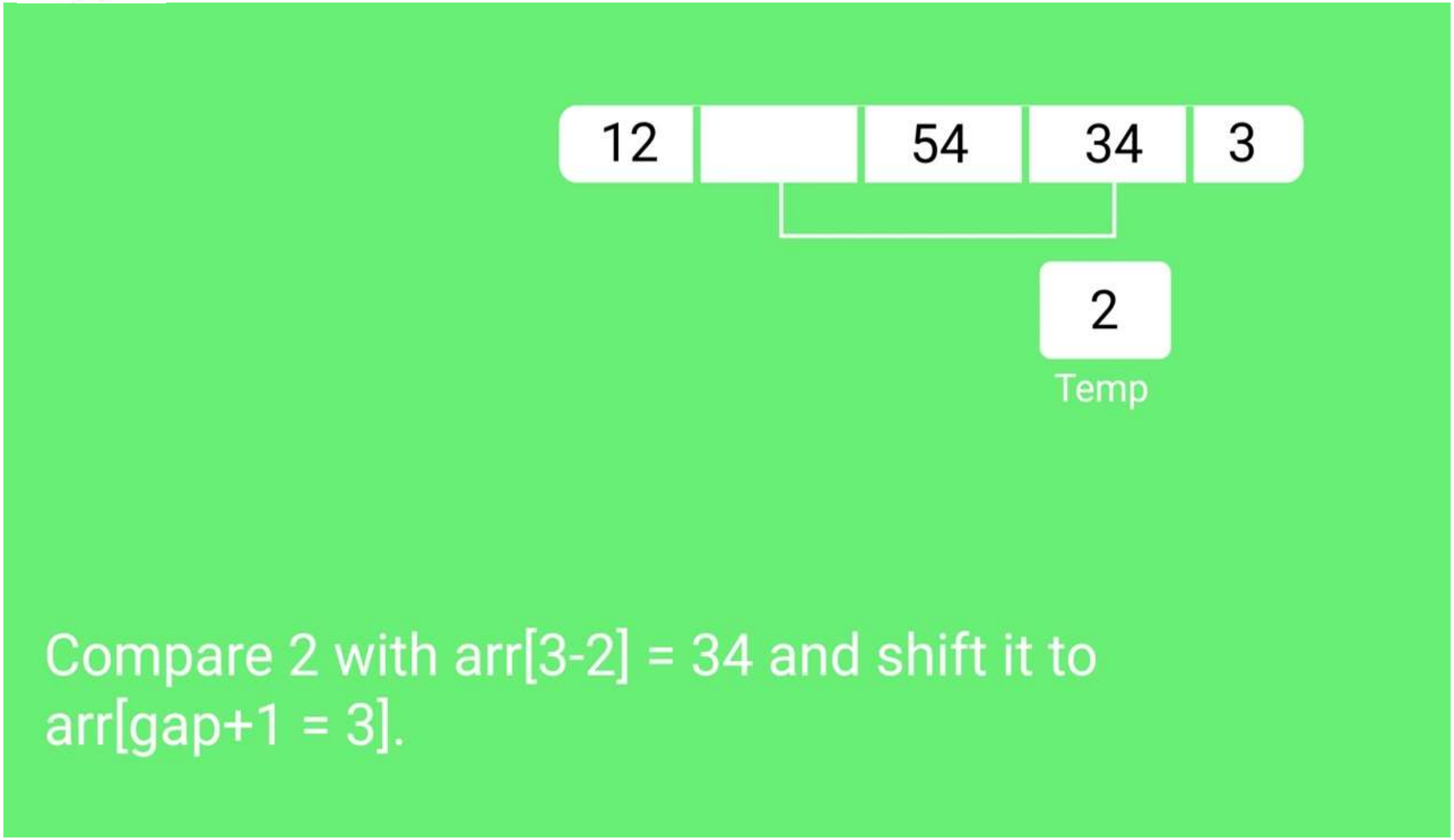
One by one select elements to the right of gap and place them at their appropriate position.

Cont..



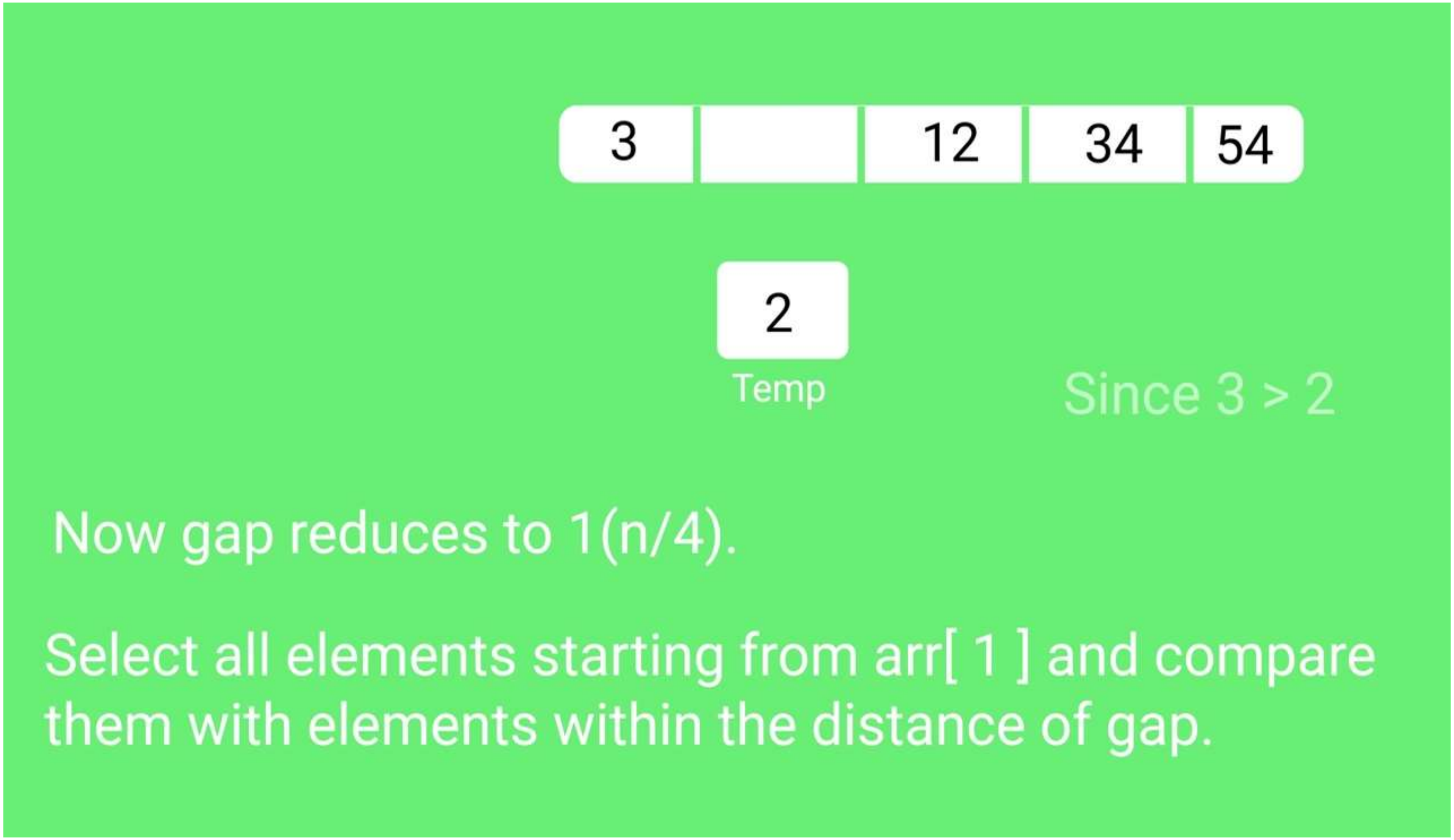
Compare 2 with arr[3-2] = 34 and shift it to arr[gap+1 = 3].

Cont..



Compare 2 with arr[3-2] = 34 and shift it to arr[gap+1 = 3].

Cont..

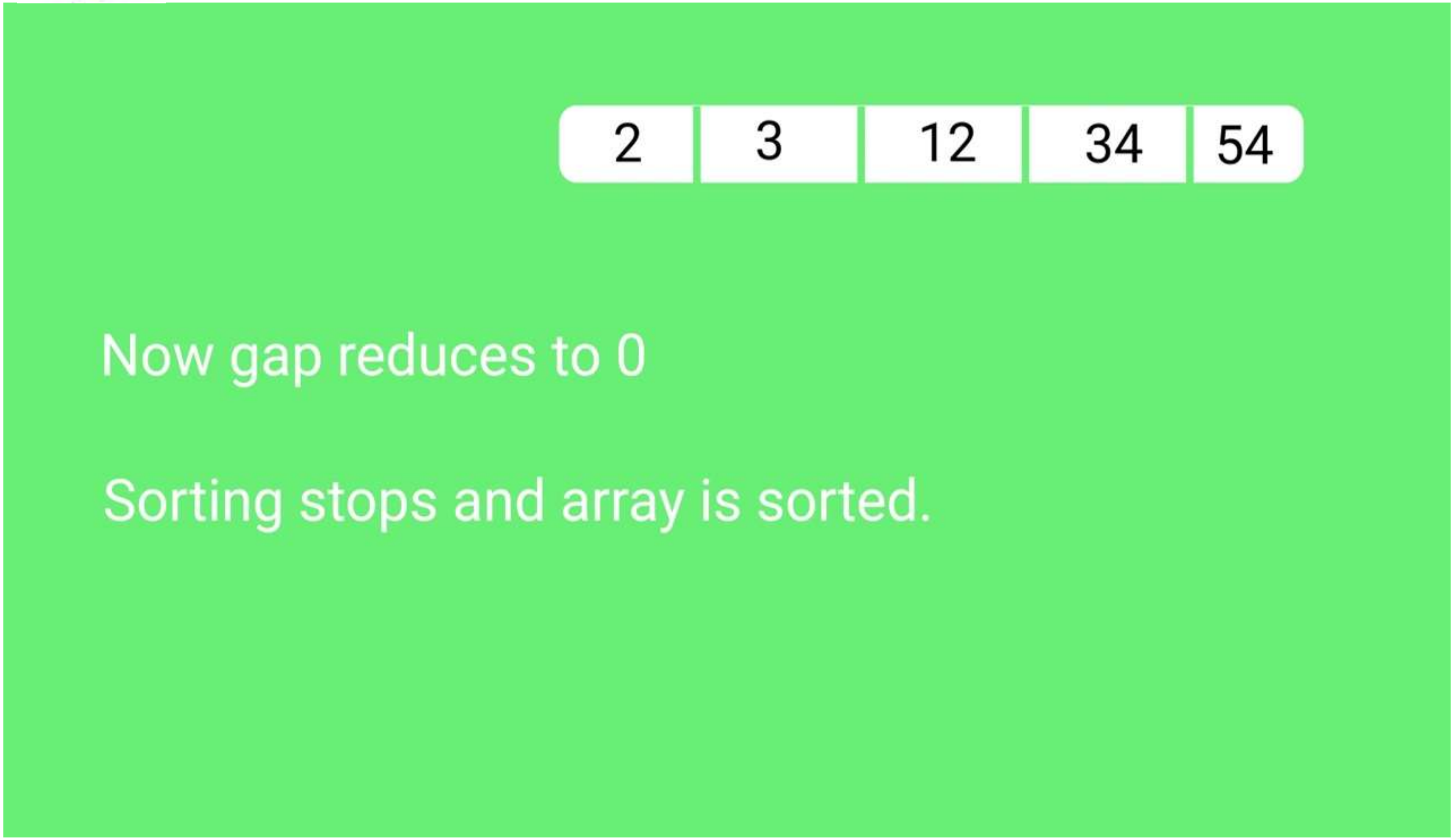


3		12	34	54

Now gap reduces to 1(n/4).

Select all elements starting from arr[1] and compare them with elements within the distance of gap.

Cont..



2	3	12	34	54

Now gap reduces to 0

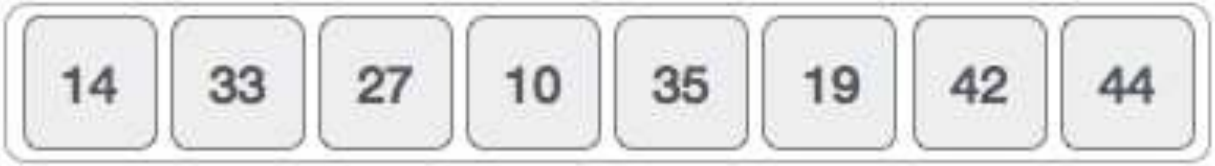
Sorting stops and array is sorted.

SELECTION SORT

- This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end.
- Initially, the sorted part is empty and the unsorted part is the entire list.
- The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array.
- This process continues moving unsorted array boundary by one element to the right.

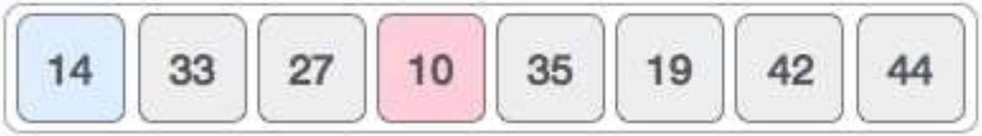
SELECTION SORT –Cont..

- **How Selection Sort Works?**
- Consider the following depicted array as an example.



14	33	27	10	35	19	42	44

- For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

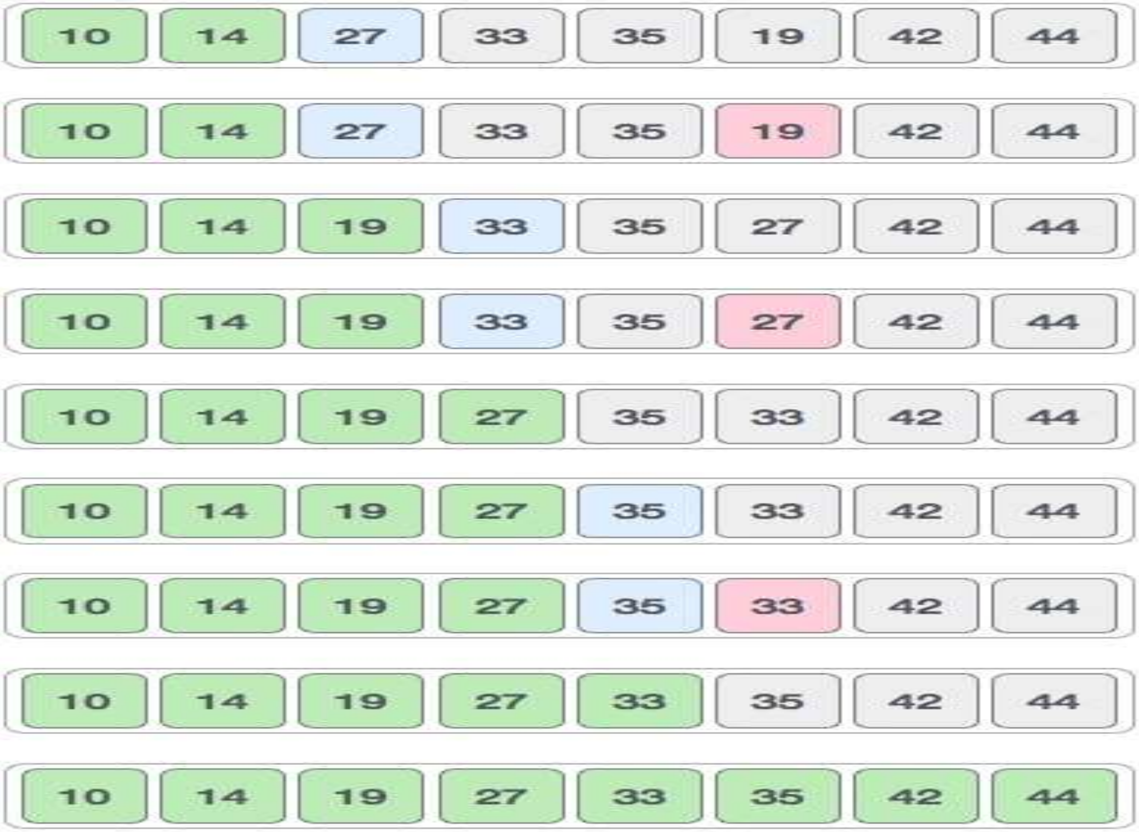


14	33	27	10	35	19	42	44

- So we replace 14 with 10. After one iteration 10, which happens to be the minimum value in the list, appears in the first position of the sorted list.

SELECTION SORT

- **Following is a pictorial depiction of the entire sorting process**



10	14	27	33	35	19	42	44
10	14	27	33	35	19	42	44
10	14	19	33	35	27	42	44
10	14	19	33	35	27	42	44
10	14	19	27	35	33	42	44
10	14	19	27	35	33	42	44
10	14	19	27	35	33	42	44
10	14	19	27	33	35	42	44
10	14	19	27	33	35	42	44

SELECTION SORT -Cont..

- **Algorithm**
- **Step 1** – Set MIN to location 0
- **Step 2** – Search the minimum element in the list
- **Step 3** – Swap with value at location MIN
- **Step 4** – Increment MIN to point to next element
- **Step 5** – Repeat until list is sorted

SELECTION SORT -Cont..

- procedure selection sort
- list : array of items
- n : size of list
- for i = 1 to n - 1
- /* set current element as minimum*/
- min = I
- /* check the element to be minimum */
- for j = i+1 to n
- if list[j] < list[min] then
- min = j;
- end if end for

SELECTION SORT -Cont..

- /* swap the minimum element with the current element*/
- if indexMin != i then
- swap list[min] and list[i]
- end if
- end for
- end procedure

Thank you