Unit-IV

Sorting

- *Sorting is the process of arranging a set of similar information into an increasing or decreasing* order.

Classes of Sorting Algorithms

- There are three general methods for sorting arrays:

➢ Exchange

➢ Selection

➢ Insertion

To understand these three methods, imagine a deck of cards.

- To sort the cards by using *exchange,* spread them on a table, face up, and then exchange out-of-order cards until the deck is ordered.
- Using *selection, spread the cards on the table, select the card of lowest value, take it out of the deck,* and hold it in your hand. Then, from the remaining cards on the table, select the lowest card and place it behind the one already in your hand. This process continues until all the cards are in your hand. The cards in your hand will be sorted when you finish the process.
- To sort the cards by using *insertion, hold all the cards in your hand. Place one card at a time on the table, always inserting it in* the correct position. The deck will be sorted when you have no cards in your hand.

The Bubble Sort

- The bubble sort is an exchange sort. It involves the repeated comparison and, if necessary, the exchange of adjacent elements.

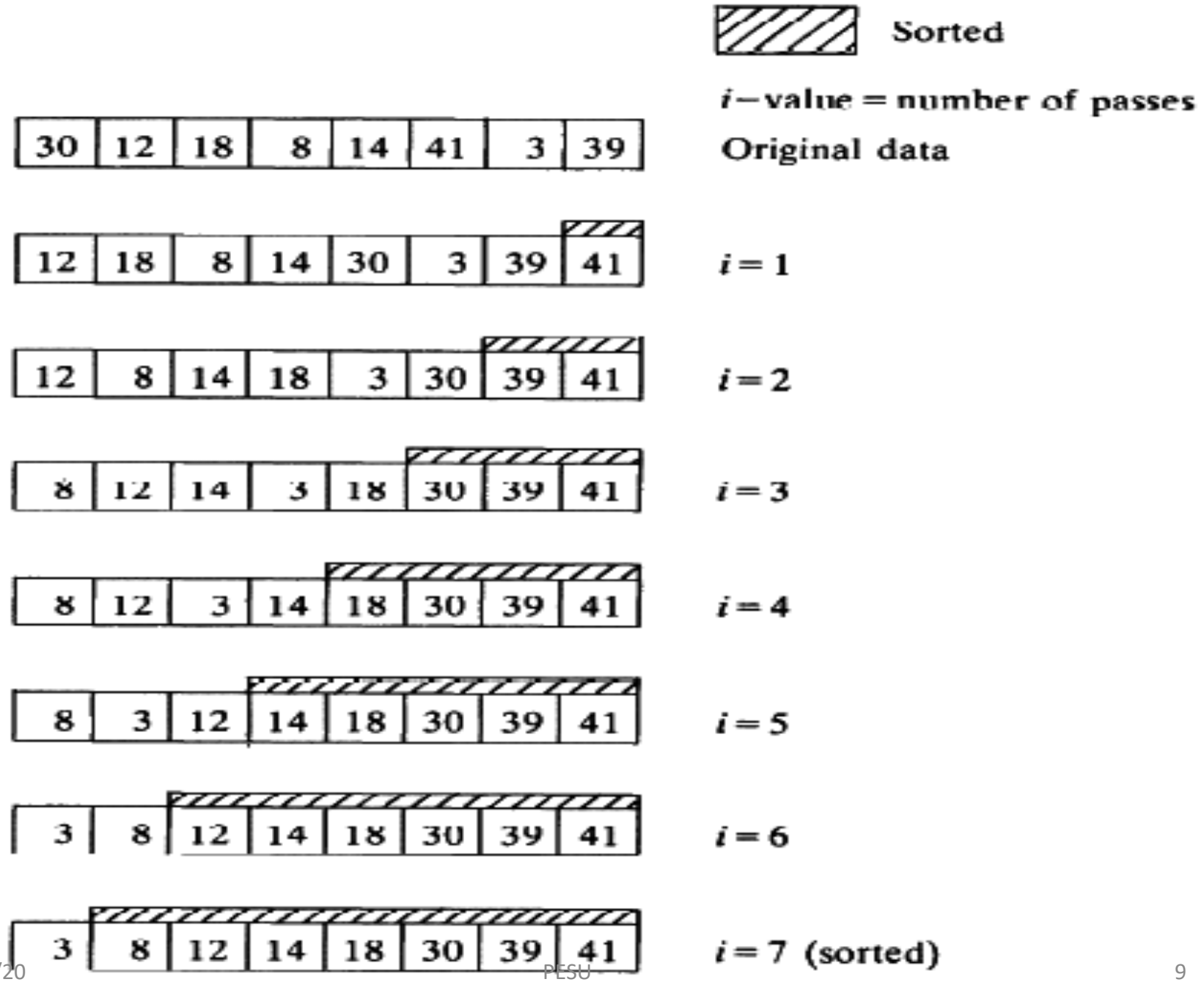
Algorithm description

1. Establish the array $a[1..n]$ of n elements.
2. While the array is still not *sorted* do
   (a) set the order indicator *sorted* to *true*;
   (b) for all adjacent pairs of elements in the unsorted part of the array do
      (b.1) if current adjacent pair not in non-descending order then
         (1.a) exchange the elements of the pair,
         (1.b) set *sorted* to *false*.
3. Return the sorted array.

```
/* The Bubble Sort. */
void bubble(int *a, int count)
{
 int i, b, t;

for(i=1; i < count; ++i)
for(b=count-1; b >= a; --b) {
if(a[b-1] > a[b]) {
/* exchange elements */
t = a[b-1];
a[b-1] =a[b];
a[b] = t;
}
}
}
```

```
/* Sort Driver */
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
void bubble(int *a, int count);
int main (void)
{
int  a[10],count,i;
printf(“enter the no. of elements\n”);
scanf(“%d”,&count);
printf(“enter the elements\n”);
for(i=0;i<count;i++)
scanf(“%d”,&a[i]);
bubble(a, count);
printf(“ the sorted elements\n”);
for(i=0;i<count;i++)
printf(“%d\t”,a[i]);
return 0;
}
```


Sorted
i−value = number of passes
30 12 18 8 14 41 3 39
Original data
12 18 8 14 30 3 39 41
i = 1
12 8 14 18 3 30 39 41
i = 2
8 12 14 3 18 30 39 41
i = 3
8 12 3 14 18 30 39 41
i = 4
8 3 12 14 18 30 39 41
i = 5
3 8 12 14 18 30 39 41
i = 6
3 8 12 14 18 30 39 41
i = 7 (sorted)

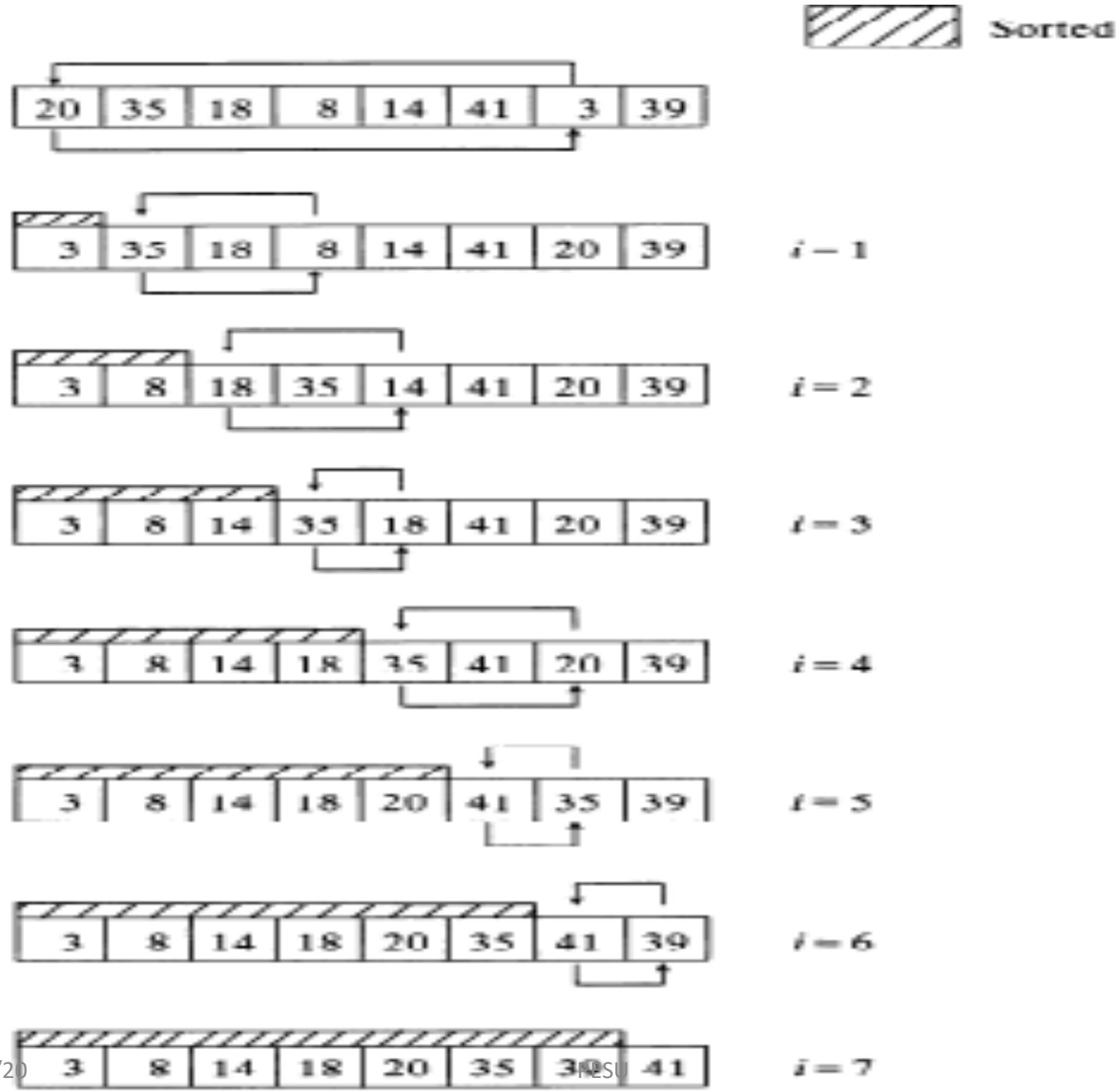
Sorting by Selection

- *A selection sort selects the element with the lowest value and exchanges it with the first element.*
- Then, from the remaining *n–1 elements, the element with the smallest key is found and exchanged* with the second element, and so forth

Algorithm description

1. Establish the array $a[1..n]$ of n elements.
2. While there are still elements in the unsorted part of the array do
   (a) find the minimum *min* and its location p in the unsorted part of the array $a[i..n]$;
   (b) exchange the minimum *min* in the unsorted part of the array with the first element $a[i]$ in the unsorted array.

```
/* The Selection Sort. */
void select(int *a, int count)
{
 int i, b, c;
int exchange;
int t;
For(i=0; i < count-1; ++i) {
exchange = 0;
c = i;
t = a[i];
for(b=i+1; b < count; ++b) {
if(a[b] < t) {
c = b;
t = a[b];
exchange = 1;
}
}
if(exchange) {
a[c] = a[i];
a[i] = t;
}}}
```


Sorted
20 35 18 8 14 41 3 39
3 35 18 8 14 41 20 39
i − 1
3 8 18 35 14 41 20 39
i = 2
3 8 14 35 18 41 20 39
i = 3
3 8 14 18 35 41 20 39
i = 4
3 8 14 18 20 41 35 39
i = 5
3 8 14 18 20 35 41 39
i = 6
3 8 14 18 20 35 39 41
i = 7