Structured Programming Language

Lecture 6

Applications of Array

- ✓Searching in Array.
- ✓Sorting of an Array.
- ✓Merging of 2 Arrays.

Searching Techniques

Searching

✓***Searching*** is the process of finding the location of a target value within a list.

✓C supports two basic algorithms for searching arrays:

- ✓The **Sequential Search** (may use on an unsorted list)
- ✓The **Binary Search** (must use only on a sorted list)

Sequential Search

✓We should only use the sequential search on small lists that aren't searched very often.

✓The sequential search algorithm begins searching for the target at the first element in an array and continues until (1) it finds the target or (2) it reaches the end of the list without finding the target

Sequential Search Program

```
found=0;
for(i=0; i<size; i++)
{
if (a[i]==SKey)
{found=1; pos=i;}
}
if(found)
printf("Element is found @ position %d" ,pos+1);
else
printf("Element Not exists in the list");
```

Binary Search

✓We should use the binary search on large lists (>50 elements) that are sorted.

✓The binary search divides the list into two halves.

✓First, it searches for the target at the midpoint of the list. It can then determine if the target is in the first half of the list or the second, thus eliminating half of the values.

✓With each subsequent pass of the data, the algorithm re-calculates the midpoint and searches for the target.

✓With each pass, the algorithm narrows the search scope by half.

✓The search always needs to track three values – the midpoint, the first position in the scope and the last position in the scope.

Binary Search- needs sorted list

```
beg=1;
end=n;
mid=(beg+end)/2;                    // Find Mid Location of Array
while(beg<=end && a[mid]!=item)
{
   if(a[mid]<item)       beg=mid+1;
   else                  end=mid-1;
   mid=(beg+end)/2;
}
if(a[mid]==item)
   { printf("\nData is Found at Location : %d",mid);    }
else  {          printf("Data is Not Found");                  }
```

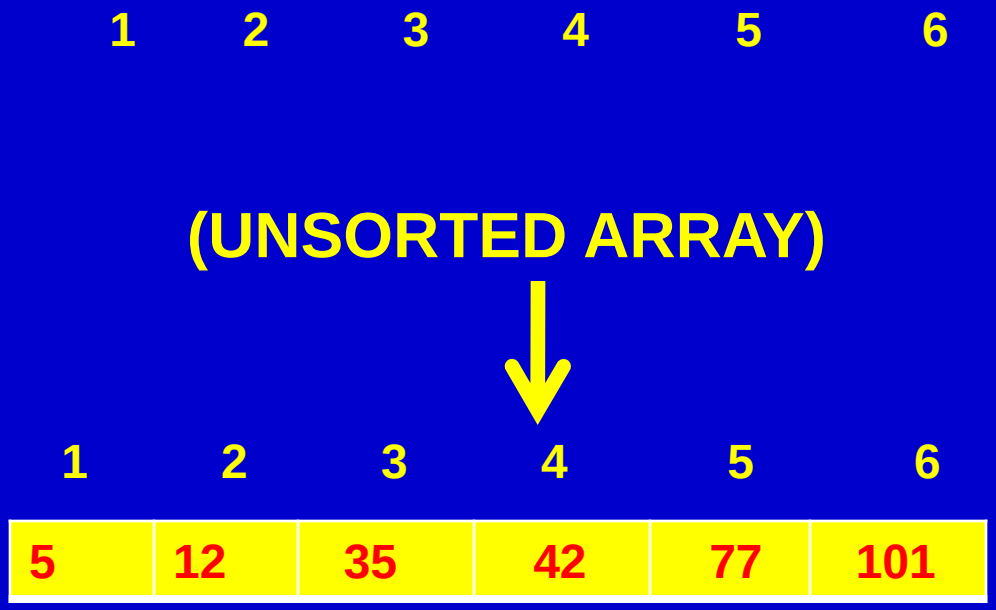
Sorting Techniques

Objectives

1. What is sorting?
2. Various types of sorting technique.
3. Selection sort
4. Bubble sort
5. Insertion sort
6. Comparison of all sorts
7. Questions on sorting
8. Review

Sorting

- ✓ **Sorting takes an unordered collection and makes it an ordered one i.e. either in ascending or in descending order.**



Sorting

✓Sorting is the “process through which data are arranged according to their values.”

✓Sorted lists allow us to search for data more efficiently.

✓Sorting algorithms depend heavily on swapping elements – **remember, to swap elements, we need to use a temporary variable!**

✓We’ll examine three sorting algorithms – the Selection Sort, the Bubble Sort and the Insertion Sort.

Types of Sorting algorithms

There are many types of sorting algorithms, but the primary ones are:

- ✓ Bubble Sort
- ✓ Selection Sort
- ✓ Insertion Sort
- ✓ Merge Sort
- ✓ Shell Sort
- ✓ Heap Sort
- ✓ Quick Sort
- ✓ Radix Sort
- ✓ Swap sort

Selection Sort

- ✓ Divide the list into two sublists: sorted and unsorted, with the sorted sublist preceding the unsorted sublist.
- ✓ In each pass,
- ✓ Find the smallest item in the unsorted sublist
- ✓ Exchange the selected item with the first item in the unsorted sublist.
- ✓ Thus selection sort is known as exchange selection sort that requires single array to work with.

Selection sort program

```
void selectionSort(int a[ ], int N)
{
      int i, j, small,minIndex,tmp;
        for (i = 0; i < (N - 1); i++)
        {
              small = a[i]; minIndex=i;
         // Find the index of the minimum element
          for (j = i + 1; j < N; j++)
          {
               if (a[j] < small)
                  {  small=a[j];
                     minIndex = j;
                  }
          }
       tmp=a[i]; a[i]=a[minIndex]; a[minIndex]=tmp;
}   }
```

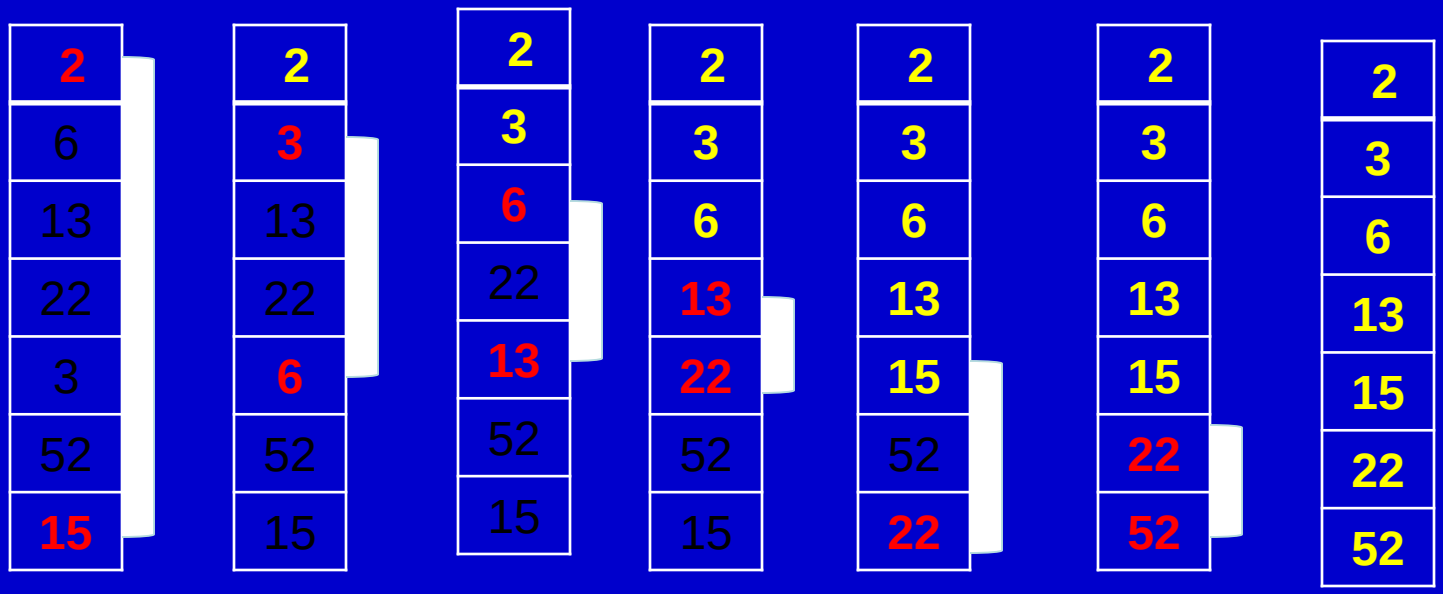
Selection sort example

Array **numlist** contains

15	6	13	22	3	52	2

Smallest element is 2. Exchange 2 with element in 1st array position (*i.e.* element 0) which is 15

Step 1 **Step 2** **Step 3** **Step 4** **Step 5** **Step 6** **Step 7**



Observations about Selection sort

Advantages:

1. Easy to use.
2. The efficiency DOES NOT depend on initial arrangement of data
3. Could be a good choice over other methods when data moves are costly but comparisons are not.

Disadvantages:

1. Performs more transfers and comparison compared to bubble sort.
2. The memory requirement is more compared to insertion & bubble sort.

Bubble Sort

Basic Idea:-

- ✓ Divide list into sorted and unsorted sublists. The unsorted sublist is "bubbled" up into the sorted sublist.
- ✓ Repeat until done:

  1.Compare adjacent pairs of records/nodes.

  2. If the pair of nodes are out of order, exchange them, and continue with the next pair of records/nodes.

  3. If the pair is in order, and continue with the next pair of records/nodes.
- ✓ Thus in each step the largest element will be bubbled up from the unsorted list into sorted list.

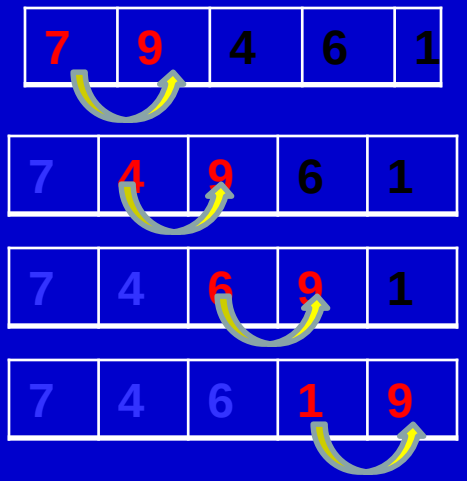
Bubble Sort Program

```
void BubbleSort(int a[],int N)
{
   int  i , j , tmp;
   for (i=0;i<N; i++)
   {
     for(j=0 ; j < (N-1)-i ; j++)
     {
         if (a[j]>a[j+1])/*swap the values if
  previous is greater than next one*/
         {  tmp=a[j];
            a[j]=a[j+1];
            a[j+1]=tmp;
         }
      }
} }
```

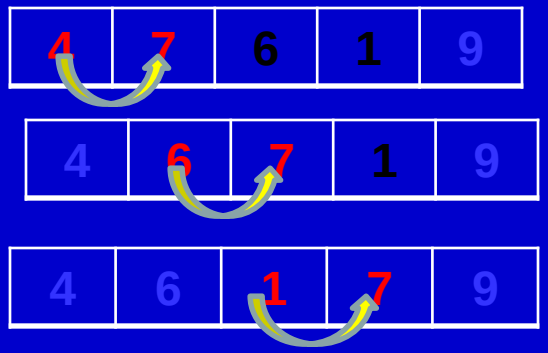
Bubble sort example

Initial array elements

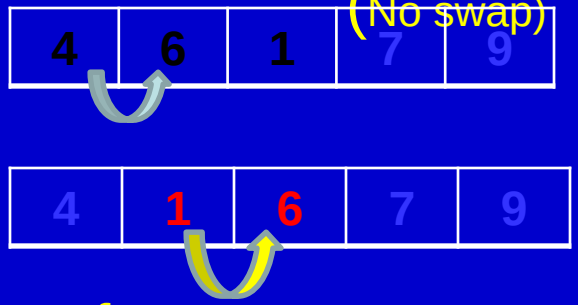
Array after pass-1



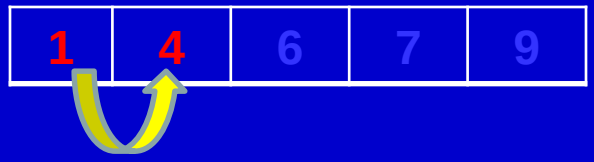
Array after pass-2



Array after pass-3



Array after pass-4



Final sorted array

Observations about Bubble Sort

Advantages:

1. Easy to understand & implement.
2. Requires both comparison and swapping.
3. Lesser memory is required compared to other sorts.

Disadvantages:

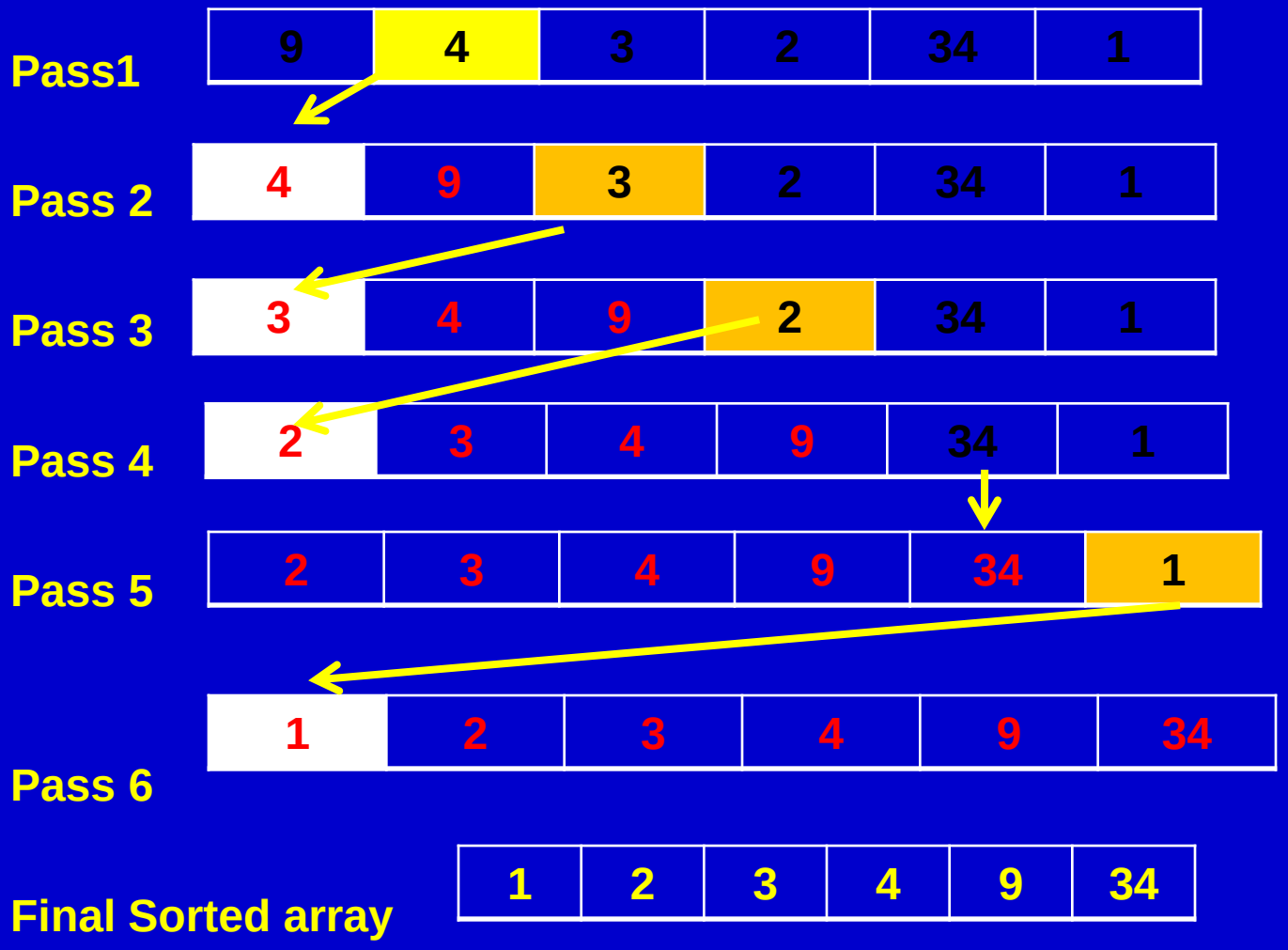
1. As the list grows larger the performance of bubble sort get reduced dramatically.
2. The bubble sort has a higher probability of high movement of data.

Insertion sort

- ✓ Commonly used by card players: As each card is picked up, it is placed into the proper sequence in their hand.
- ✓ Divide the list into a sorted sublist and an unsorted sublist.
- ✓ In each pass, one or more pieces of data are removed from the unsorted sublist and inserted into their correct position in a sorted sublist.

Insertion sort program

Initially array contents are (9,4,3,2,34,1)



Insertion sort program

```
void insertionSort(int a[ ], int N)
{
        int i,j,tmp;
        for(i=0;i<N;i++)
        {
            for(j=i-1;j>=0;j--)
            {
                if(a[j]>a[j+1])
                {
                    tmp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=tmp;
                }
            }
        }
}
```

Observations about insertion sort

Advantages:

1. Simplicity of the insertion sort makes it a reasonable approach.
2. The programming effort in this techniques is trivial.
3. With lesser memory available insertion sort proves useful.

Disadvantage:

1. The sorting does depend upon the distribution of element values.
2. For large arrays -- it can be extremely inefficient!

Merging of two sorted arrays into third array in sorted order

- ✓ Algorithm to merge arrays a[m](sorted in ascending order) and b[n](sorted in descending order) into third array C [n + m] in ascending order.

```
void main()
{
int a[5]={2,4,5,6,7},m=5; //a is in ascending order
int b[6]={15,12,4,3,2,1},n=6; //b is in descending order
int c[11];
int i=0; //    i points to the smallest element of the array a
    which is at index 0
 int j=n-1;// j points to the smallest element of the array b
//which is at the index m-1 since b is sorted in
    //descending order
```

Merging of two sorted arrays into third array in sorted order

```
int k=0;  //k points to the first element of the array c
while(i<m&&j>=0)
     {
         if(a[i]<b[j])
        c[k++]=a[i++]; // copy from array a into array c and
                              then increment i and k
      else

         c[k++]=b[j--]; // copy from array b into array c and
                              then decrement j and increment k
     }
```

Merging of two sorted arrays into third array in sorted order

```
while(i<m)        //copy all remaining elements of array a
    {
   c[k++]=a[i++];
   }
while(j>=0)       //copy all remaining elements of array b
   {
   c[k++]=b[j--];
   }
printf("The merged array is :\n");
for(int i=0; i<m+n; i++)
   printf("%d, ", c[i]);
}  // end of Function.
```

Merging of two sorted arrays into third array in sorted order

Output is

The merged array is:

1, 2, 2, 3, 4, 4, 5, 6, 7, 12, 15

Revision

1. In selection sort, algorithm continuous goes on finding the smallest element and swap that element with appropriate position element.

2. In bubble sort, adjacent elements checked and put in correct order if unsorted.

3. In insertion sort, the exact position of the element is searched and put that element at its particular position.