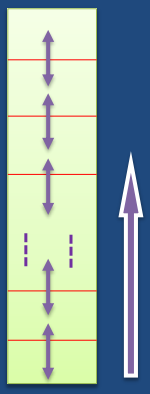
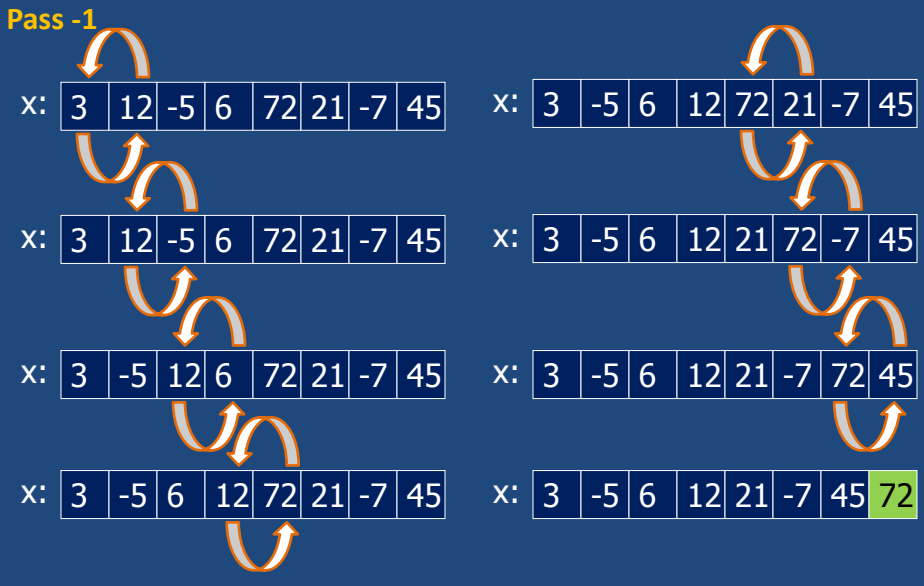
Bubble Sort



In every iteration heaviest element drops at the bottom.

The bottom moves upward.

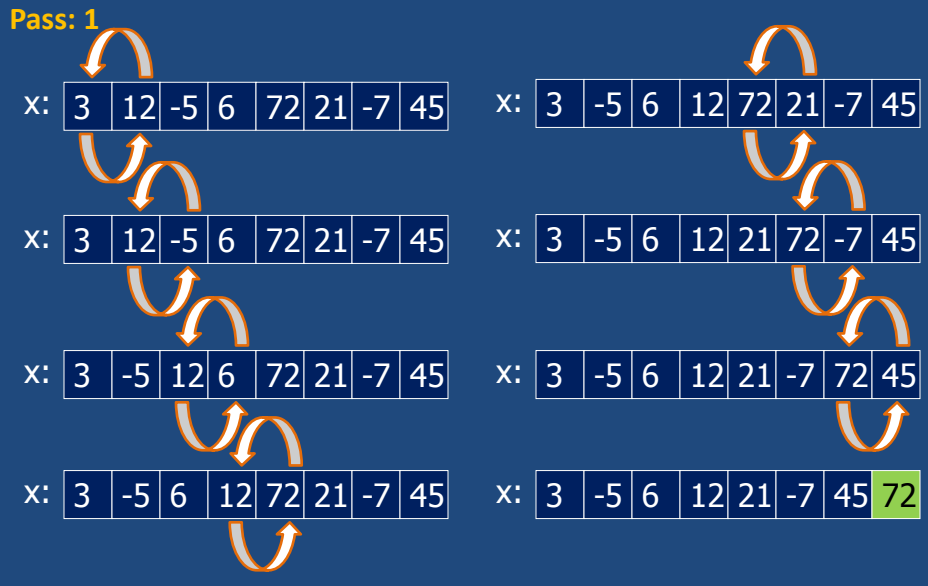
Example



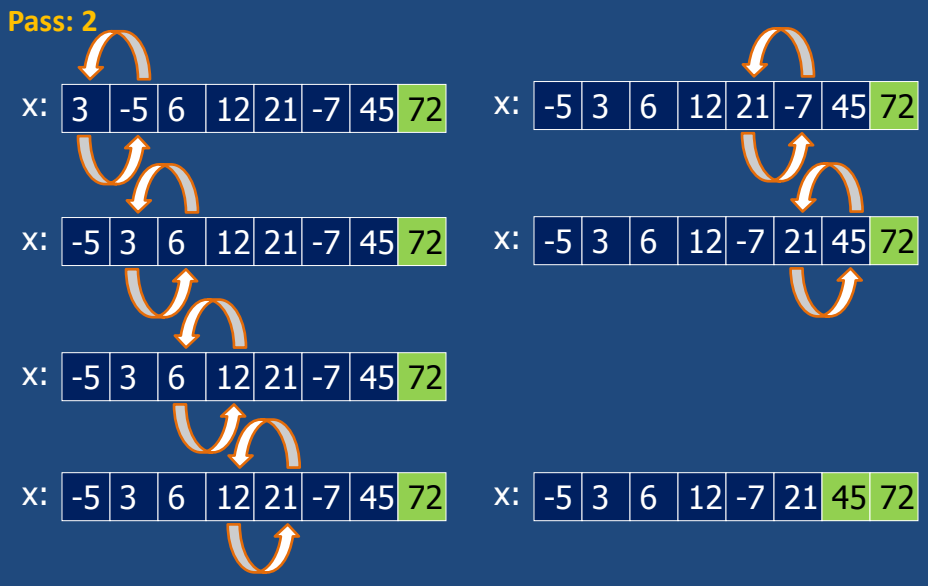
Bubble Sort

```
int pass,j,a[100],temp;
/* read number of elements as  n and elements as a[] */
for(pass=0; pass<n; pass++) {
        for(j=0; j<n-1; j++) {
                if(a[j]>a[j+1]) {
                        temp = a[j+1];
                        a[j+1] = a[j];
                        a[j] = temp;
                }
        }
}
/* print sorted list of elements */
```

Example



Example



Time Complexity

- Number of comparisons :
  - Worst Case?

$1+2+3+ \ldots\ldots +(n-1) = n(n-1)/2$

  - Best Case?

Same

How do you make best case with (n-1) comparisons only?

Bubble Sort

```
int pass,j,a[100],temp,swapflag;
/* read number of elements as  n and elements as a[] */
for(pass=0; pass<n; pass++) {
        swapflag=0;
        for(j=0; j<n-1; j++) {
                if(a[j]>a[j+1]) {
                        temp = a[j+1];
                        a[j+1] = a[j];
                        a[j] = temp;
                        swapflag=1;
                }
        }
        if(swapflag==0)
                break;
}
/* print sorted list of elements */
```

Can we improve the sorting time?

Basis of efficient sorting algorithms

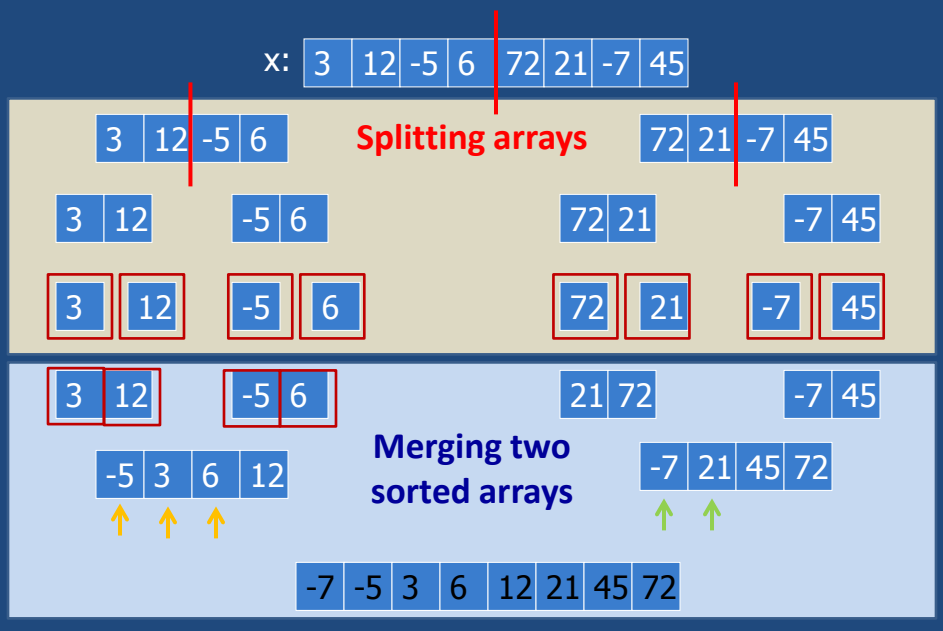
- Two of the most popular sorting algorithms are based on divide-and-conquer approach.
  - Quick sort
  - Merge sort

- Basic concept:

```
sort (list)
{
   if the list has length greater than 1
   {
      Partition the list into lowlist and highlist;
      sort (lowlist);
      sort (highlist);
      combine (lowlist, highlist);
   }
}
```

Merge Sort

Example



Merge Sort C program

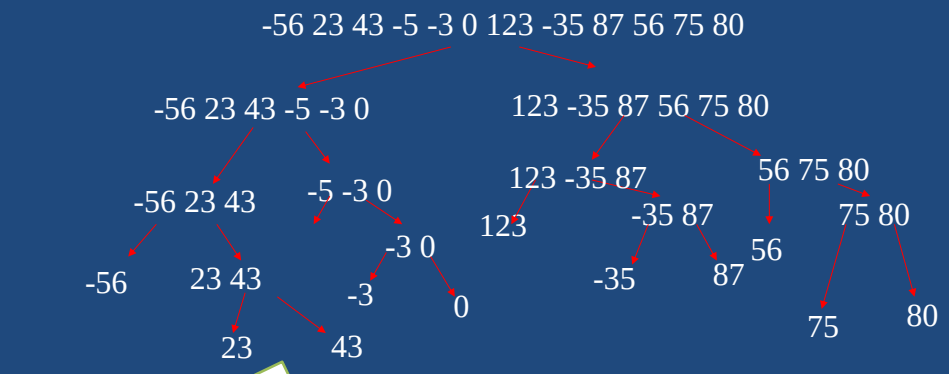
```
#include<stdio.h>
void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);

int main()
{
   int a[30],n,i;
   printf("Enter no of elements:");
   scanf("%d",&n);
   printf("Enter array elements:");

   for(i=0;i<n;i++)
      scanf("%d",&a[i]);

   mergesort(a,0,n-1);

   printf("\nSorted array is :");
   for(i=0;i<n;i++)
      printf("%d ",a[i]);

   return 0;
}
```

```
void mergesort(int a[],int i,int j)
{
   int mid;

   if(i<j)    {
      mid=(i+j)/2;
            /* left recursion  */
      mergesort(a,i,mid);
            /* right recursion */
      mergesort(a,mid+1,j);
/* merging of two sorted sub-arrays */
      merge(a,i,mid,mid+1,j);
    }
}
```

Merge Sort C program

```
void merge(int a[],int i1,int j1,int i2,int j2)
{
   int temp[50];    //array used for merging
   int i=i1,j=i2,k=0;

   while(i<=j1 && j<=j2)    //while elements in both lists
   {
      if(a[i]<a[j])
         temp[k++]=a[i++];
      else
         temp[k++]=a[j++];
   }

   while(i<=j1)    //copy remaining elements of the first list
      temp[k++]=a[i++];

   while(j<=j2)    //copy remaining elements of the second list
      temp[k++]=a[j++];

      for(i=i1,j=0;i<=j2;i++,j++)
      a[i]=temp[j];                //Transfer elements from temp[] back to a[]
}
```

Splitting Trace



Space Complexity??

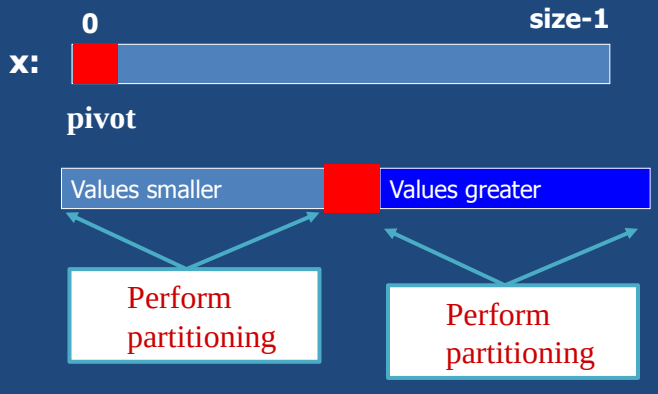
-56 -35 -5 -3 0 23 43 56 75 80 87 123

Worst Case: O(n.log(n))

Quicksort

- At every step, we select a ***pivot element*** in the list (usually the first element).
  - We put the pivot element in the final position of the sorted list.
  - All the elements less than or equal to the pivot element are to the left.
  - All the elements greater than the pivot element are to the right.

Partitioning



Quick Sort program

```
#include <stdio.h>
void quickSort( int[], int, int);
int partition( int[], int, int);
void main()
{
        int i,a[] = { 7, 12, 1, -2, 0, 15, 4, 11, 9};
        printf("\n\nUnsorted array is: ");
        for(i = 0; i < 9; ++i)
                printf(" %d ", a[i]);
        quickSort( a, 0, 8);
        printf("\n\nSorted array is: ");
        for(i = 0; i < 9; ++i)
                printf(" %d ", a[i]);
}
void quickSort( int a[], int l, int r)
{
        int j;
        if( l < r ) { // divide and conquer
                j = partition( a, l, r);
                quickSort( a, l, j-1);
                quickSort( a, j+1, r);
        }
}
```

```
int partition( int a[], int l, int r)
{
        int pivot, i, j, t;
        pivot = a[l];
        i = l;
        j = r+1;
        while( 1) {
            do {
                ++i;
            } while(a[i]<=pivot && i<=r);
            do {
                --j;
            } while( a[j] > pivot );
            if( i >= j ) break;
            t = a[i];
            a[i] = a[j];
            a[j] = t;
        }
        t = a[l];
        a[l] = a[j];
        a[j] = t;
        return j;
}
```

Trace of Partitioning

Input: 45 -56 78 90 -3 -6 123 0 -3 45 69 68

45 -56 78 90 -3 -6 123 0 -3 45 69 68

-6 -56 -3 0 -3 45 123 90 78 45 69 68

-56 -6 -3 0 -3 68 90 78 45 69 123

-3 0 -3 45 68 78 90 69

-3 0 69 78 90

Output: -56 -6 -3 -3 0 45 45 68 69 78 90 123

Time Complexity

- Partitioning with n elements.
  - No. of comparisons:
    n-1

Choice of pivot element affects the time complexity.

- Worst Case Performance:
  (n-1)+(n-2)+(n-3)+……. +1= n(n-1)/2

- Best Case performance:
  (n-1)+2((n-1)/2-1)+4(((n-1)/2-1)-1)/2-1) .. k steps
  = O(n. log(n))

$2^k=n$

Exercise

- Implement non-recursive version of quick sort and merge sort.

Multi-file programs

Example 1

- Determine the value of x which causes the function

$$y = x \cos(x)$$

to be maximized within a specified interval.

```c
/* find the maximum of a function within a specified interval */
#include <stdio.h>
double a,b,xl,yl,xr,yr,cnst=0.0001;    /* external variable declaration */
extern void reduce(void);              /* external function prototype */
extern double curve(double xl);        /* external function prototype */
int main()
{
      double xmax, ymax;
      printf("\na= ");
      scanf("%lf",&a);
      printf("\nb= ");
      scanf("%lf",&b);
      do
        reduce();
      while((yl!=yr) && ((b-a)>3*cnst));
      xmax=0.5*(xl+xr);
      ymax=curve(xmax);
      printf("\nxmax= %8.6lf ymax= %8.6lf\n",xmax,ymax);
}
```

mainFile.c

```c
extern double a,b,xl,yl,xr,yr,cnst;
extern double curve(double xl);

extern void reduce(void)
{
      xl=a+0.5*(b-a-cnst);
      xr=xl+cnst;
      yl=curve(xl);
      yr=curve(xr);
      if(yl>yr) {
            b=xr;
            return;
      }
      if(yl<yr)
            a=xl;
      return;
}
```

intReduce.c

```c
#include <math.h>

extern double curve(double x)
{
      return (x*cos(x));
}
```

evalFunc.c

Example 1 – Compile and Execute

```
$ cc  -c  -Wall  mainFile.c  intReduce.c  evalFunc.c
$ cc  mainFile.o  intReduce.o  evalFunc.o  -lm
$ ./a.out

a= 0

b= 3.141593

xmax= 0.860394    ymax= 0.561096
```

Example -2

Write a complete C program that will read N (to be read from the user) number of integers and sort them. User will input his/her choice of sorting technique from a pool of Selection/Insertion/Bubble sort. Each sorting technique will be implemented as separate function. User choice will pick right sorting function using switch-case statement.

Example -2

mainfile.c

```
#include <stdio.h>
#define SIZE    100

void selSort(int a[],int n);
void insSort(int a[],int n);
void bubbleSort(int a[],int n);

int main()
{
      char choice;
      int k,x[SIZE],size;

      do {
            printf("Enter the number of elements: ");
            scanf("%d",&size);
            printf("Enter the elements: ");
            for(k=0;k<size;k++)
                  scanf("%d",&x[k]);
            printf("Sorting method [I/S/B]: ");
            getchar();
            choice=getchar();
            switch(choice) {
                        case 'I':
                        case 'i':   insSort(x,size);
                                    break;
                        case 'S':
                        case 's':   selSort(x,size);
                                    break;
                        case 'B':
                        case 'b':    bubbleSort(x,size);
                                    break;
                        default:   printf("No Sorting");
                  }

            for(k=0;k<size;k++)
                  printf("%d ",x[k]);
            printf("\Continue [Y/N]: ");
            getchar();
            choice=getchar();
      } while(choice=='Y' || choice=='y');
      return 0;
}
```

bubble.c

```
void bubbleSort(int a[],int n)
{
      int pass,j,temp,swapflag;

      for(pass=0; pass<n; pass++) {
            swapflag=0;
            for(j=0; j<n-1; j++) {
                  if(a[j]>a[j+1]) {
                        temp = a[j+1];
                        a[j+1] = a[j];
                        a[j] = temp;
                        swapflag=1;
                  }
            }
            if(swapflag==0)
                  break;
      }
}
```

selection.c

```
void selSort(int x[],int size)
{
      int k,j,pos,temp;

      for (k=0; k<size-1; k++)  {
            pos = k;
            for (j=k+1; j<size; j++) {
                  if (x[j] < x[pos]) {
                        pos = j;
                  }
            }
            temp = x[k];
            x[k] = x[pos];
            x[pos] = temp;
      }
}
```

insertion.c

```
void insSort(int x[],int size)
{
      int i,j,temp;
      for (i=1; i<size; i++) {
            temp = x[i] ;
            for (j=i-1; (j>=0)&& (x[j] > temp); j--)
                  x[j+1] = x[j];
            x[j+1] = temp ;
      }
}
```

EXAMPLE PROBLEMS

Problem 1

Write a C program to print a Geometric Progression (GP) series and its sum till N terms.

Expression for a GP Series: $\sum_{k=0}^{n-1}(ar^k) = a\left(\frac{1-r^n}{1-r}\right)$

where, a is the first term
r is the common ratio
n is the number of terms

Problem 2

Write a C program to generate a FLOYD's triangle.

```
1
2 3
4 5 6
7 8 9 10
11 13 14 15
16 17 18 19 20 21
```

Problem 3

Write a C program to check whether a number is a strong number or not.

What is a strong number ?

A number whose sum of the factorial of its digit is equal to the number itself.

Example: 145 = 1! +4! +5! = 1 + 24 + 120 = 145 is a strong number.

Problem 4

Write a C program to arrange elements in an array in ascending order.

Problem 5

Write a C program to reverse a string (array of characters) without using any extra array.

Problem 6

Write a C program to concatenate two strings without using library function.