# Searching

Search algorithms form an important branch of computer science. They are concerned with searching a data structure to see if some required data is there. Usually a key value (e.g. a person's name or PPS number) is used to locate the data and the data (e.g. personal information) once located is then returned.

Different types of data structure can be used to speed up the search for the required data. Examples are:

- sequential search on an unsorted array
- binary search on a sorted array
- a hash table
- a binary search tree

In our examples below, we keep things simple with an int for the key value and no data. We then want to see if the key value is in the array.

## *Sequential Search*

Sequential search takes on average N/2 steps and at most N steps to locate a key in an array a[ ] of size N.

```
int sequentialSearch( int x, int a[ ], int n)
Begin
    for j = 0 to n-1
        if a[j] == x
            return j  // found at position j

    // key not found
    return –1
End
```

## *Binary Search*

Binary search is a simple and efficient algorithm for searching a sorted array for a value. If the array a[ ] is of size N, binary search will return a result in at most $\log_2 N$ steps.

```
int binarySearch( int x, int a[ ], int l, int r)
Begin
    // l > r indicates x is not in a[ ]
    if l > r
        return  –1

    j = (l + r) / 2

    if x == a[j]
        return j   // return position where x found
    else if x < a[j]
        return binarySearch(x, a, l, j-1)
    else
        return binarySearch(x, a, j+1, r)
End
```

### *binarySearch.c*

```
#include <stdio.h>

//function to search for x in array a[] of size n

int binarySearch( int x, int a[], int n)
{

      // do yourself, see pseudocode

}


void main()  // this is to test the algorithm
{
     //declare and initialise a sorted array
     int myarray[10] = {2, 4, 7, 8, 9, 12, 23, 32, 36, 39 };

     int i, position, x;

     printf("\nArray is ");
     for(i=0; i<10; ++i)
          printf("%2d ", myarray[i]);

     printf("\nInput value to search for: ");
     scanf("%d", &x);

     //call the binary search algorithm

     position = binarySearch(x, myarray, 10);

     if( position == -1)
          printf("\nValue not found\n\n");
     else
          printf("\nValue found at array position %d\n\n", position);

}
```

### *Recursive Version*

```
int binarySearch( int x, int a[], int l, int r)
{
     int j;

    if (l > r)
        return -1;

     j = (l + r) / 2;

     if( x == a[j])
          return j;      //return position of x in a[]
     else if ( x < a[j])
          return binarySearch( x, a, l, j-1);
     else
          return binarySearch(x, a, j+1, r);
}
```

### *Iterative Version*

```
int binarySearch( int x, int a[], int n)
{
     int j, upper, lower;

     upper = n-1;
     lower = 0;

     while( lower <= upper)
     {
          j = (lower + upper) / 2;

          if( x == a[j])
               return j;      //return position of x in a[]
          else if ( x < a[j])
               upper = j-1;
          else if (x > a[j])
               lower = j+1;
     }
     return -1;  // indicates x not found
}
```

```
//search for x in array a[] of size n

int sequentialSearch( int x, int a[], int n)
{
     int j;

     for( j = 0; j < n; ++j)  //possibly search whole array
     {
          if( x == a[j])
               return j;      //return position of x in a[]
     }

     return -1;  // indicates x not found
}
```