Arrays and Searching

- A very common programming process.

- *Sequential Search*: the simplest method. Begin search at one end of the array and scan down it until the desired value is found or the other end is reached.

- Algorithm:

```
int find(int big_array[], int size, int value)
{
   int found=0,
       loc = 0;

   while(!found && loc < size){
      if (big_array[loc] == value)
          found = 1;
      else
          loc = loc + 1;

   if (found)
      return loc;
   else
      return –1;
}
```


Sorted Arrays

- What happens if the array is sorted?
  ⇨we can terminate search as soon as a value which is greater than or equal to the target value is found.

- Algorithm:

```
while (!found && loc < size){
  if (big_array[loc] >= value)
     found = 1;
  else
     loc = loc + 1;
}

if (found && big_array[loc] != value)
  found = 0;
```

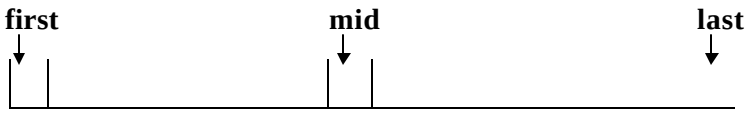
Binary Search

- Sequential search is easy to implement and efficient for short arrays, but it is a disaster for long arrays. (e.g. trying to find "Smith" in a telephone directory with sequential search)

- To find an item in a long list, there are far more efficient methods.

⇨ *Binary search*

- Array must be sorted.
- Compare the value with the one in the center of the list and then restrict attention to only the first or second half of the array depending on whether the value being searched comes before or after the central element's value.



- at each step we reduce the length of the array to be searched by half.
- (e.g. array size = 1 million ⇒ 20 comparisons needed to locate any item)

Algorithm

- Several slightly different algorithms for binary search can be written.
- Termination condition: either the target value is found or no items left in the array to be searched. Initially low index = 0 and high index = Size – 1. The loop should terminate when high < low, provided that we have not terminated the loop earlier by finding the target.

In C:

```
int binary Search(int L[], int size, int value)
{
   int high, low, mid;  //mid will be the index of
   int found = 0;       //target when it's found.

   low = 0;
   high = size –1;

   while ((low <= high) && !found) {
    mid = (high + low)/2;
    if (L[mid] == value)
        found = 1;
    else if (value < L[mid])
        high  = mid – 1;// reduce to the lower
                        // half of the array
    else
        low = mid + 1;  //reduce to the top half
                        //of the array
   }
   return found;
}
```

- When **`high == low`**, the loop iterates one more time.
- Trace the algorithm for:

0	1	2	3	4	5	6	7	8	9
3	5	7	9	10	13	15	17	19	25

– value = 19 $\Rightarrow$ 3 comparisons.
– value = 14 $\Rightarrow$ 4 comparisons.

Recursive Binary Search Function

```
int Find(int numbers[], int val,
         int low, int high)
{
   int mid;

   mid = (low + high)/2;

   if (low > high)
      return 0;
   else if (numbers[mid] == val)
      return 1;
   else if (val < numbers[mid])
      return Find(numbers, val, low, mid-1);
   else
      return Find(numbers, val, mid+1, high);
}
```

- We not need to pass in the length of the array to this function for it to work properly.

Merging Two Sorted Arrays

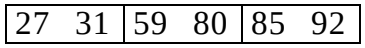
The problem: Given two sorted arrays, combine the values in each of the arrays into a larger array so that the larger array contains all values in a sorted way.

The idea:

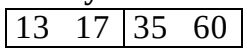
1. For each of the given arrays, keep track of the index of the smallest value that hasn't been placed in the larger array.
2. Compare these two smallest values. Place the smaller in the next available location in the large array.
3. Adjust the index for the appropriate array and also for the large array.
4. Continue this process (steps 1 -3) until all values in one of the shorter arrays are placed in the large array.
5. Copy the remaining values in the other short array to the remaining locations of the large array.

Illustration of merging two sorted arrays:

Array 1



Array 2



Sorted Big Array

Implementation

```
void merge(int a[], int b[], int c[],
           int sizeA, int sizeB)
{
   int index1 = 0,  // index for array a
       index2 = 0,  // index for array b
       index3 = 0;  // index for array c

   while (index1 < sizeA && index2 < sizeB)
   {
      if (a[index1] < b[index2]) {
         c[index3] = a[index1];
         index1 ++;
      }
      else {
         c[index3] = b[index2];
         index2 ++;
      }
      index3++;
   }

   while (index1 < sizeA){
      c[index3] = a[index1];
      index1++;
      index3++;
   }

   while (index2 < sizeB){
      c[index3] = b[index2];
      index2++;
      index3++;
   }
}
```