# Sequential and Binary Search

## Binary Search

The idea of binary search is to cut a *sorted array* into two parts and discard the part which could not contain our searching target. How do we decide which half might not contain the search key? Since the array is sorted (in ascending order), if we look at the mid-point of the array and see that it is less than the search key, then we can discard the lower half - the search item, if it exists, will be in the upper half. If the middle item is greater than the search key, we can discard the upper half, the search key must be in the lower half. If the above two fails, the middle value is equal to the search key, and we have a successful search.

```
/* binarysearch.cpp */
/* Searches sortedArray[first]..sortedArray[last] for key.
   returns index of the matching element if it finds the key.
   Otherwise   returns -1 to indicate the search was a failure
*/
int binarysearch(int sortedArray[], int first, int last, int key) {
   while (first <= last) {
      int mid = (first + last) / 2;  // find mid-point.
      if (key > sortedArray[mid])
         first = mid + 1;  // repeat search in top half.
      else if (key < sortedArray[mid])
         last = mid - 1; // repeat search in bottom half.
      else
         return mid;     // found it. return position /////
   }
   return - 1;    // failed to find key
   /* You could return -first-1 to indicate not only the failure but also the
position where it should be inserted, or should have been if it were present
*/
}
```

## Sequential Search

Sequential search is the intuitive approach of searching an array for a particular element, starting from either the beginning or end of the array and traversing the array until the target is located or until all the elements in the array have been visited.

```
#include<iostream>

int sequentialsearch(int array, int count, int key)
{
  for (int k = 0; k < count; k++){
    if (array[k] == key)
      return k;
  }
   return -1;   // not found
}
```