**Searching Arrays**

Searching an array means we are looking to see if a specific value exists within the elements of an array.

There are two searches we will examine: Linear Search and Binary Search

**Linear Search**

Beginning with the first element in the array, we compare its value to the value we are searching for. If it doesn’t match, we proceed to the second element in the array. If the value does match the value we are searching for, it returns the index of that element. If the value is not found, this process continues until either the value is found or the entire array has been searched. If no match is found, the linear search will return a value of -1.

**Linear Search Code**

Assume we are sending this array: int [ ] numbers = {4,7,3,9,10,2, 5}; and a user provided integer, *key*

public static int LinearSearch(int [ ] nums, int k) {

for(int x = 0; x < nums.length; x++){

if (key == nums[x]) return x;}

return -1; }

**Results**

int a = LinearSearch(numbers, 9); //returns 3 int c = LinearSearch(numbers, 1); //returns -1

**Binary Search**

The binary search method requires the array to be sorted (we’ll assume ascending order) before the search begins. Like in the linear search, we have a number that we are searching for called a *key*. Here’s the process:

1. The binary search compares the key with the element in the middle of the array. One of three results occurs:
   1. If the key is less than the middle element, continue the search within the first half of the array.
   2. If the key is greater than the middle element, continue the search with the second half of the array.
   3. If the key is equal to the middle element, the search ends with a match.
2. After each comparison, the binary search eliminates half of the remaining array.
3. Given *n* elements in an array, the binary search will find an element in *x+ 1* searches or fewer, where 2x > n. For example, if you have 100 elements, it would take at most 8 searches. (27 = 128) x = 7 there, so x + 1 = 8

**Binary Search Code**

In the following code, *low* and *high* represent the first and last index of the array being searched. *mid* represents the middle element being searched.

public static int BinarySearch( int [] nums, int k){

int low = 0;

int high = list.length – 1;

while(high >=low){

int mid = (low + high) /2;

if(k < nums[mid]) high = mid - 1;

else if (k==nums[mid]) return mid;

else low = mid + 1;

) //end while

return -1; //not found}