ISM 001 Introduction to Programming

Searching

Florence Fok

Lingnan University CC

Searching

- Searching → determine of a value (search key) present in the data → find value's location
- 1) Linear search
  - Searches each element sequentially → inefficient
  - If search key is not present
    - Tests each element
    - When algorithm reaches end of array, informs user search key is not present
  - If search key is present
    - Return the location

```
import javax.swing.*;
public class LinearSearch {
  public static void main(String argv[]) {

   int test[]=new int[5];
   System.out.println("Please enter " + test.length + " marks");

   for(int i=0; i<test.length; i++) {
        String strNum = JOptionPane.showInputDialog("Enter the
   number ");
        test[i]=Integer.parseInt(strNum);

   }
   int total = 0;
   // add each element's value to total
   for ( int i = 0; i < test.length; i++ )
    total += test[ i ];

     System.out.println("The total mark is " +total );
      System.out.println( "Average mark is: " +total/test.length );
```

```
        String strKey = JOptionPane.showInputDialog("Enter the key ");
        int sKey=Integer.parseInt(strKey);
        int position=linearSearch(test, sKey);

        if(position == -1)
                System.out.println("No key found");
        else
                System.out.println("We find the key in position "+ position);

    }//end main

    public static int linearSearch ( int a[], int key ) {
        for (int n=0; n < a.length; n++) {
                if ( a[n] == key )
                        return n;

       }//end for
         return -1;

    }//end method


}
```

Searching

- 1) Binary search
- More efficient
- Require the array to be sorted
- Tests the middle element in an array
  - If it is the search key, algorithm returns
  - Otherwise, if the search key is smaller, eliminates larger half of array
  - If the search key is larger, eliminates smaller half of array
- Each iteration eliminates half of the remaining elements

```
 13 23 24 34 35 36 38 42 47 51 68 74 75 85 97
Please enter an integer value (-1 to quit): 23
13 23 24 34 35 36 38 42 47 51 68 74 75 85 97
                      *
13 23 24 34 35 36 38
          *
13 23 24
    *
The integer 23 was found in position 1.
Please enter an integer value (-1 to quit): 75
13 23 24 34 35 36 38 42 47 51 68 74 75 85 97
                      *
                        47 51 68 74 75 85 97
                                  *
                                    75 85 97
                                        *
                                    75
                                     *
The integer 75 was found in position 12.
Please enter an integer value (-1 to quit): 52
13 23 24 34 35 36 38 42 47 51 68 74 75 85 97
                      *
                        47 51 68 74 75 85 97
                                  *
                        47 51 68
                            *
                              68
                               *
The integer 52 was not found.
Please enter an integer value (-1 to quit): -1
```

```
public static int binarySearch( int array[], int key ){

        int low = 0; // low subscript
        int high = array.length - 1; // high subscript
        int middle; // middle subscript
        while ( low <= high ) {

                middle = ( low + high ) / 2;
                if ( key == array[ middle ] ) // match
                return middle;

                else if ( key < array[ middle ] )
                        high = middle - 1; // search low end of array
                else
                        low = middle + 1; // search high end of array
                }
                return -1;

   }//end binary search
```

Try

- Try to use binary search method to search a key with an array of data with 15 elements (such as the one in Slide 6)