Data Structure

In computer science, a **data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently. **Efficiency** is used to describe properties of an algorithm relating to how much of various types of resources it consumes, the goal is to reduce resource consumption, including time to completion.

Data structures provide a means to manage huge amounts of data efficiently, such as large databases and internet indexing services. Usually, efficient data structures are a key to designing efficient algorithms.

Types of Data Types

1. **Primitive data types**
    - Boolean (for boolean values True/False)
    - Char (for character values)
    - Float (for storing real number values)
    - Double (a larger size of type float)
    - int (for integral or fixed-precision values)
    - String (for string of chars)
    - Enumerated type
2. **Composite data types**
    - Array
    - Record (also called struct)
    - Union
3. **Abstract data types**
    - Queue
    - Stack
    - Tree
    - Gaph
    - Hash

Searching Algorithoms

Linear Search

Linear search is a search algorithm, also known as **sequential search**,that is suitable for searching a list of data for a particular value.

It operates by checking every element of a list one at a time in sequence until a match is found.

The Linear Search, or sequential search, is simply examining each element in a list one by one until the desired element is found. The Linear Search is not very efficient. If the item of data to be found is at the end of the list, then all previous items must be read and checked before the item that matches the search criteria is found.This is a very straightforward loop comparing every element in the array with the key. As soon as an equal value is found, it returns. If the loop finishes without finding a match, the search failed and -1 is returned.For small arrays, linear search is a good solution because it's so straightforward. In an array of a million elements linear search on average will take 500,000 comparisons to find the key.

Linear Search Algorithm:

1. **Begin**
2. **Get Number of Array elements (Size)**
3. **Get Array elements (List)**
4. **Get search key (searchkey – Target -)**
5. **Let Pos = 0**
6. **If List [Pos] = searchkey Then**
7. **Go to step 12**
8. **Pos = Pos +1**
9. **If Pos < N Then**
10. **Go to step 6**

    Else
11. **Go to step 14**
12. **Display Message ‘ Search key Found in Position ’ Pos .**
13. **Goto step 15**
14. **Display Message ‘ Search key Not Found ‘ .**
15. **End**

Linear Search Function:

```
int linearSearch (int Size , int List[ ], int searchkey )
{
    // function:
    //   Searches List [0].. List [Size] for searchkey.
    // returns: index of the matching element if it finds search key,
    //  otherwise  -1.
    // parameters:
    //   List          in array of (possibly unsorted) values.
    //   first, Size -1  in lower and upper subscript bounds
    //   searchkey        in value to search for.
    // returns:
    //   index of searchkey, or -1 if key is not in the array.
 for (int Pos=0; Pos < Size; Pos ++)
{
    if (searchkey == List [Pos]) {
       return Pos;
    }
 }
 return -1;    // failed to find search key
}
```