Introduction to Data Structures and Algorithm

Chapter 1

Contents

- What are Data Structures and Algorithms
- Overview of Data Structures
- Overview of Algorithms
- Java and Java Library Data Structures

What Are Data Structures and Algorithms

- Data Structure :
  - An arrangement of data in a computer’s memory (or sometimes on a disks) so that it can be used efficiently.
  - Include arrays, linked lists, stacks, binary trees, heaps and graphs, among others.

What Are Data Structures and Algorithms

- Algorithm :
  - Manipulate the data in those structures in various ways, such as a searching for a particular data item and sorting the data.
  - A set of rules or a process to solve a problem.
  - For example : multiply two positive integer
    - 124 * 234
    - 87654321 * 46721964

What Are Data Structures and Algorithms

- American (From right to left)
- English (From left to right)
- à la russe
- Divide-and-conquer
- etc.

What Are Data Structures and Algorithms

- American (From right to left)
- English (From left to right)

What Are Data Structures and Algorithms

- à la russe

What Are Data Structures and Algorithms

- Divide-and-conquer

What Are Data Structures and Algorithms

- Divide-and-conquer
- 4.09538203020E15
- 8765|4321 * 4672|1964

1. 4321*1964 = 8486444
2. 4321*4672 = 20187712- - - -
3. 8765*1964 = 17214460- - - -
4. 8765*4672 = 40950080- - - - - - - -

4095382030206444

Overview of Data Structures

Data Structures	Advantages	Disadvantages
Array (Unordered array)	Quick insertion, very fast access if index known	Slow search, slow deletion, fixed sized
Ordered array	Quicker search than unsorted array	Slow insertion and deletion, fixed size
Stack	Provides last-in-first-out access	Slow access to other items
Queue	Provide first-in-first-out access	Slow access to other items
Linked list	Quick insertion, quick deletion, unlimited sized	Slow search
Binary tree	Quick search, insertion, deletion.	Deletion algorithm is complex
Heap	Fast insertion, deletion, access to largest item.	Slow access to other items
Graph	Models real-world situations	Some algorithms are slow and complex

Overview of Algorithms

- There are many algorithms apply to specific data structures.
- For most data structures, you need to know how to
  - ❑ Insert a new data item
  - ❑ Search for a specified item
  - ❑ Delete a specified item
  - ❑ Iterate through all the items
  - ❑ Sort the data
  - ❑ Use recursion

Overview of Algorithms

- When we set out to solve a problem, how to decide which algorithm for its solution should be used?
  - ❑ The size of the instance to be solved
  - ❑ The way / method in which the problem is presented
  - ❑ The speed and the memory size of the computing equipment

Java and Java Library Data Structures

- No Pointers
  - ❑ Java doesn't use pointers.
  - ❑ In Java, pointer is used in the form of memory addresses. (references)
- References

```
Int intVar;  // an int variable called intVar
BookAccount bc1; // reference to a BankAccount object
```

- ❑ A memory location called intVar actually holds a numerical value.
- ❑ The memory location bc1 does not hold the data of a BankAccount object. Instead, it contains the *address* of a BankAccount object that is actually stored.
- ❑ The name bc1 is a *reference* to this object; it's not the object itself.

Java and Java Library Data Structures

- The new Operator
  - ❑ Any object in Java must be created using new.
  - ❑ In Java, new returns a reference.
- Java Library Data Structures
  - ❑ You can find some of the structures useful from Java Library
  - ❑ Before you use object of the class, you must use the line

    Import java.util.*;

Trace a Program

Linear Loops

```
public class Counter1
{
   public static void main (String[] args)
   {
      int count = 1;

      while (count <= 25)
      {
         System.out.println (count);
         count = count + 1;
      }
      System.out.println ("Done");
    }
}
```

Trace a Program

Linear Loops

```
public class counter2
{
   public static void main (String[] args)
   {
      final int LIMIT = 25;
      int count = 0;

      do
      {
         count = count + 1;
         System.out.println (count);
      } while (count < LIMIT);

      System.out.println ("Done");
   }
}
```

Trace a Program

Linear Loop

```
public class Counter3
{
   public static void main (String[] args)
   {
      final int LIMIT = 25;

      for (int count=1; count <= LIMIT; count++)
         System.out.println (count);

      System.out.println ("Done");
   }
}
```

Trace a Program

Nested Loops: Quadratic loop

```
public class Counter4
{
   public static void main (String[] args)
   {
      final int LIMIT1 = 5;
      final int LIMIT2 = 5;

      for (int count=1; count <= LIMIT1; count++)
         for (int i = 1; i <= LIMIT2; i++)
           System.out.println(i + count);

      System.out.println ("Done");
   }
}
```