Introduction to Data Structure

Chapter 1: Introduction

Abstract Data Types:
ADT

- A set of data values and associated operations that are precisely specified independent of any particular implementation.
- Example: stack, queue, priority queue.

DataStructures

- The term data structure refers to a scheme for organizing related pieces of Information.
- The basic types of data structures include: files, lists, arrays,records, trees, tables.
- A data structure is the concrete implementation of that type, specifying how much memory is required and, crucially, how fast the execution of each operation will be

Timing

- Every time we run the program we need to estimate how long a program will run since we are going to have different input values so the running time will vary.
- The worst case running time represents the maximum running time possible for all input values.
- We call the worst case timing big Oh written O(n).The n represents the worst case execution time units.

Complexities

- Simple programming statement
- Example:

```
k++;
```

- Complexity : O(1)
- Simple programming statements are considered 1 time unit.

Complexities

- Linear ***for*** loops
- Example:

```
k = 0;
for(i = 0; i < n; i++)
   k++;
```

- Complexity : O(n)
- ***for*** loops are considered n time units because they will repeat a programming statement n times.
- The term linear means the for loop increments or decrements by 1

Complexities

- Non Linear loops
- Example:

```
k=0;
for(i=n; i>0; i=i/2)
    k++;
```

- Complexity : O(log n)
- For every iteration of the loop counter i will divide by 2.
- If i starts is at 16 then then successive i's would be 16, 8, 4, 2, 1. The final value of k would be 4. Non linear loops are logarithmic.
- The timing here is definitely $\log_2 n$ because $2^4 = 16$. Can also works for multiplication.

Complexities

- Nested ***for*** loops
- Example:

```
k=0;
for(i=0; i<n; i++)
   for(j=0; j<n; j++)
     k++;
```

- Complexity : $O(n^2)$
- Nested for loops are considered n^2 time units because they represent a loop executing inside another loop.

Complexities

- Sequential ***for*** loops
- Example:

```
k=0;
for(i=0; i<n; i++)
   k++;
k=0;
for(j=0; j<n; j++)
   k++;
```

- Complexity : O(n)
- Sequential for loops are not related and loop independently of each other.

Complexities

- Loops with non-linear inner loops
- Example:

```
k=0;
for(i=0;i<n;i++)
   for(j=i; j>0; j=j/2)
      k++;
```

- Complexity : O(n log n)

- The outer loop is O(n) since it increments linear.
- The inner loop is O(n log n) and is non-linear because decrements by dividing by 2.
- The final worst case timing is: O(n) * O(log n) = O(n log n)

Complexities

- Inner loop incrementer initialized to outer loop incrementer
- Example:

```
k=0;
for(i=0;i<n;i++)
   for(j=i;j<n;j++)
      k++;
```

- Complexity : $O(n^2)$
- In this situation we calculate the worst case timing using both loops.
- For every i loop and for start of the inner loop j will be n-1 , n-2, n-3.

Complexities

- Power Loops
- Example:

```
k=0;
for(i=1;i<=n;i=i*2)
   for(j=1;j<=i;j++)
      k++;
```

- Complexity : $O(2^n)$
- To calculate worst case timing we need to combine the results of both loops.
- For every iteration of the loop counter i will multiply by 2.
- The values for j will be 1, 2, 4, 8, 16 and k will be the sum of these numbers 31 which is $2^n - 1$.

Complexities

- If-else constructs
- Example:

```
/* O(n) */
if (x == 5)
{
     k=0;
     for(i=0; i<n; i++)
       k++;
 }
/* O(n2) */
else
{
    k=0;
    for(i=0;i<n;i++)
        for(j=i; j>0; j=j/2)
            k++;
}
```

- Complexity : $O(n^2)$

Complexities

- Recursive
- Example:

```
int f(int n)
{
      if(n == 0)
            return 0;
       else
            return f(n-1) + n
 }
```

- Complexity : O(n)

Stages:Program Design

- Identify the data structures.
- Operations: Algorithms
- Efficiency(Complexity)
- Implementation
  - What goes into header file?
  - What goes into C program?
  - What are libraries? Why do we need them?
  - How to create libraries.