Department of Computer Science & Engineering

Varendra University

CSE-211: Data Structures

Books

1. “Introduction to Algorithms” by Thomas H. Coremen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Prentice-Hall ,2nd Edition, ISBN - 81 - 203-2141-3.

2. “Data Structures” by Schaum’s Series.

2. “Data Structures and Algorithm Analysis in C by Mark Allen Weiss, Addison Wesly, ISBN 81-7808-167-9.

3. “Fundamentals of Computer Algorithms” by Ellis Horowitz, Sartaj Sahni and Sanguthevar Rajasekaran, Galgotia, ISBN -81-7515-257-5.

4. “Theory & Problems of Data Structures” by Seymour Lipschutz, McGraw-Hill, ISBN 0-07-038001-5.

5. “Data Structures & Algorithms in JAVA by Robert Lafore, Techmedia, ISBN 81-7635-186-5.

Introduction to Data Structures

- **Data Structures**

The logical or mathematical model of a particular organization of data is called a data structure.

- **Types of Data Structure**

1. Linear Data Structure

   Example: Arrays, Linked Lists, Stacks, Queues

2. Nonlinear Data Structure

   Example: Trees, Graphs

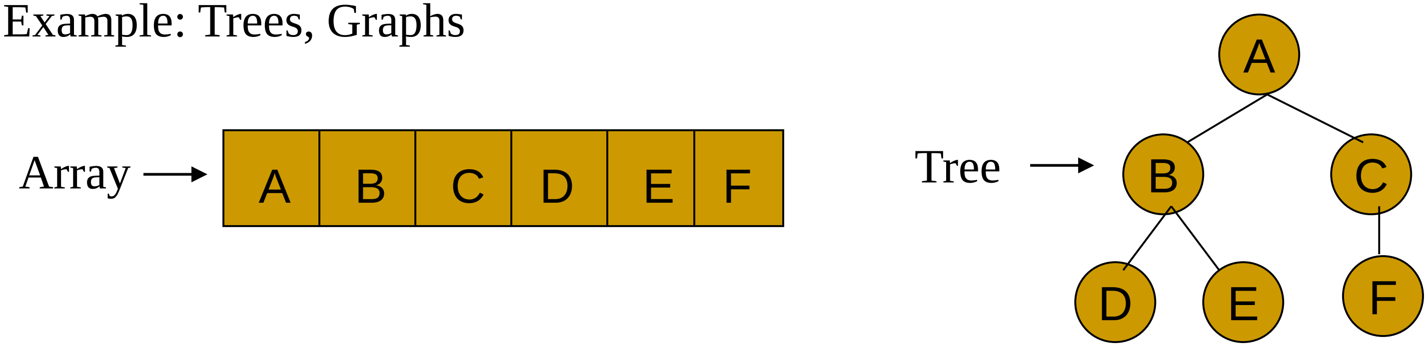

Figure: Linear and nonlinear structures

Choice of Data Structures

The choice of data structures depends on two considerations:

1. It must be rich enough in structure to mirror the actual relationships of data in the real world.

2. The structure should be simple enough that one can effectively process data when necessary.

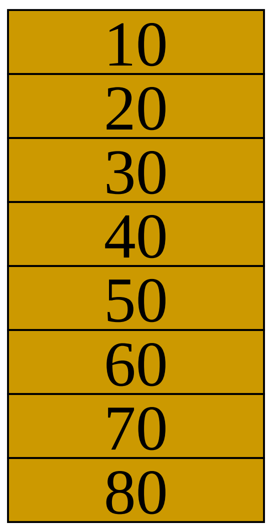

Figure 2: Array with 8 items

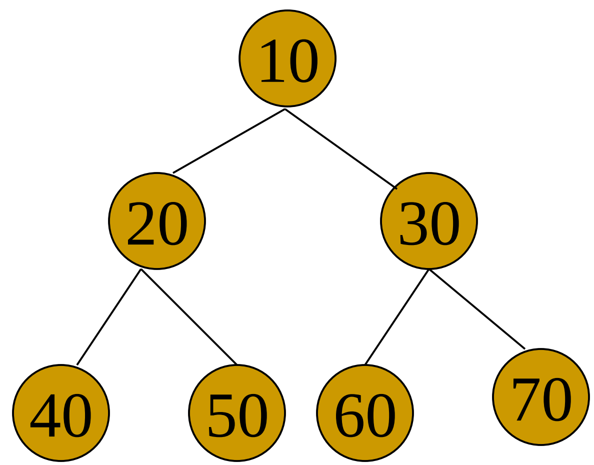

Figure 3: Tree with 8 nodes

❑ Data Structure Operations

1. **Traversing:** Accessing each record exactly once so that certain items in the record may be processed.
2. **Searching:** Finding the location of the record with a given key value.
3. **Inserting:** Adding a new record to the structure.
4. **Deleting:** Removing a record from the structure.
5. **Sorting:** Arranging the records in some logical order.
6. **Merging:** Combing the records in two different sorted files into a single sorted file.

❑ Algorithms

It is a well-defined set of instructions used to solve a particular problem.

Example:

Write an algorithm for finding the location of the largest element of an array Data.

Largest-Item (Data, N, Loc)

1. Input N elements in Data
2. set k:=1, Loc:=1 and Max:=Data[1]
3. while k<=N repeat steps 4, 5
4. If Max < Data[k] then Set Loc:=k and Max:=Data[k]
5. Set k:=k+1
6. write: Max and Loc
7. exit

❑ **Complexity of Algorithms**

- The complexity of an algorithm M is the function f(n) which gives the running time and/or storage space requirement of the algorithm in terms of the size n of the input data.

- Two types of complexity

  1. Time Complexity

  2. Space Complexity

❑ Asymptotic Notation

These notations are used to describe the running time or space requirement of an algorithm.

1. 0-notation
2. Ω-notation
3. Θ-notation
4. o-notation
5. ω-notation

1. 0-notation

- A function f(n)=0(g(n)) if there are positive constants c and n_0 such that

  $0 <= f(n) <= c.g(n)$ for all $n>= n_0$.

- When f(n)=0(g(n)), it is guaranteed that f(n) grows at a rate no faster than g(n).

  So g(n) is an upper bound on f(n).

Example:

(a) f(n) = 3n+2

Here f(n) <= 5n for n>=1

So, f(n) = 0(n).

(b) f(n) = $3n^2$-2

Here f(n) < $3n^2$ for n>=1

So, f(n) = $0(n^2)$.

2. Ω-notation

- A function $f(n) = \Omega(g(n))$ if there are positive constant c and n_0 such that $c.g(n) <= f(n)$ for all $n >= n_0$.

- When $f(n) = \Omega(g(n))$, it is guaranteed that f(n) grows at a rate faster than g(n). So g(n) is a lower bound on f(n).

Example:

(a) $f(n) = 3n+2$

Here $f(n) > n$ for $n>=1$

So, $f(n) = \Omega(n)$.

(b) $f(n) = 3n^2+2$

Here $f(n) > 3n^2$ for $n>=1$

So, $f(n) = \Omega(n^2)$.

3. **Θ-notation**

- A function f(n)= Θ(g(n)) if there are positive constants c_1, c_2 and n_0 such that

  $0<=c_1.g(n)<=f(n)<=c_2.g(n)$ for all $n>= n_0$.

- The Θ-notation asymptotically bounds a function from above and below.

Example:

(a) f(n) = 3n+2

Here, f(n) = 0(n) and f(n)= Ω(n).

So, f(n) = Θ(n)

(b) f(n) = $3n^2$+2

Here, f(n) = 0(n) and f(n) = Ω(n).

So, f(n) = Θ(n^2)

4. o-notation

The function f(n) = o(g(n)) if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Example: Suppose f(n) = 3n+2

Let $g(n)=n^2$.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{3n+2}{n^2} = 0.$$

So, $3n+2 = o(n^2)$.

5. ω-notation

The function f(n) = ω(g(n)) if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

Example: Suppose $f(n) = 3n^2+2$

Let g(n)=n.

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n}{3n^2+2} = 0.$$

So, $3n^2+2 = \omega(n)$.

Some rules related to asymptotic notation

Rule-1

If $f_a(n) = 0(g_a(n))$ and $f_b(n) = 0((g_b(n))$ then

(a) $f_a(n)+f_b(n) = \max(0(g_a(n)),0(g_b(n))$

(b) $f_a(n) * f_b(n) = 0(g_a(n) * g_b(n))$

Rule-2

If f(n) is a polynomial of degree k, then $f(n) = \Theta(n^k)$.

Rule-3

$\mathrm{Log}^k n = 0(n)$ for any constant.

Typical Growth Rates

Function	Name
c	Constant
$\log n$	Logarithmic
$\log^2 n$	Log-squared
n	Linear
$n \log n$	
n^2	Quadratic
n^3	Cubic
2^n	Exponential

Thank You!