Data Structure

David Phillip Wagner

Chapter 1

A Data Structure is a way to store data in the Memory of computer.

E.g. Array, Linked list, Stack, Queue

- Search for sth.
- Insert sth.
- Delete sth.

Array

		Array with no extra memory	Array with extra memory	Sorted Array
Search	Best	Fast	Fast	Fast
	Worst	Slow	Slow	Fast
	Average	Slow	Slow	Fast
Insert	Best	Slow	Fast	Fast
	Worst	Slow	Fast	Slow
	Average	Slow	Fast	Slow

Sorting an Array

- Bubble Sort
- Insertion Sort
- Merge Sort

Bubble Sort $\theta(n^2)$

Sorting an N size Array with Bubble Sort takes $n^2/250,000,000$ seconds

The running time of Bubble Sort is related to n^2

Merge Sort $\theta(nlogn)$

1. Split in half **(until size = 1)**
2. Sort each half
3. Merge the two halves

$n * logn = \theta(nlogn)$

Bubble Sort & Merge Sort

Algorithms	Running Time
Bubble Sort	$\theta(n^2)$
Merge Sort	$\theta(nlogn)$

Algorithm Speed & Operation Speed

Algorithm Speed		Operation Speed	
$\theta(1)$	Very Fast	$O(1)$	Very Fast
$\theta(logn)$	Very Fast		
$\theta(n)$	Fast	$O(logn)$	Fast
$\theta(nlogn)$	Fast		
$\theta(n^2)$	Slow	$O(n)$	Slow
$\theta(2^n)$	Very Slow	$O(n^2)$	Very Slow
$\theta(n^n)$	Very Slow		

Linked List

A linked list is a Data Structure which stores each element in a different part of memory.

Each element is stored together with the location of the next element.

1. Create a space in memory for the new number.
2. Write the number into the space.
3. Write a pointer to start into this space.
4. Change start to point to the new number.

		Linked List	Array with no extra memory	Array with extra memory	Sorted Array
Search	Best	$\theta(1)$	$\theta(1)$	$\theta(1)$	$\theta(1)$
	Worst	$\theta(n)$	$\theta(n)$	$\theta(n)$	$\theta(logn)$
	Average	$\theta(n)$	$\theta(n)$	$\theta(n)$	$\theta(logn)$
Insert	Best	$\theta(1)$	$\theta(n)$	$\theta(1)$	$\theta(1)$
	Worst	$\theta(1)$	$\theta(n)$	$\theta(1)$	$\theta(n)$
	Average	$\theta(1)$	$\theta(n)$	$\theta(1)$	$\theta(n)$

Array V.S. Linked List

Search Speed -> **Sorted Array**

Insert Speed -> **Linked List**

Stack

A stack is a data structure with two operations **Push** & **Pop**.

- **Push:** Add a number on the top of the stack.
- **Pop:** Romove the topmost number from the stack.

Push & **Pop** both take $\theta(1)$ time.

Queue

A stack is a data structure with two operations **Enqueue** & **Dequeue**.

- **Enqueue** takes $\theta(1)$ time.
- **Dequeue** takes $\theta(n)$ time.