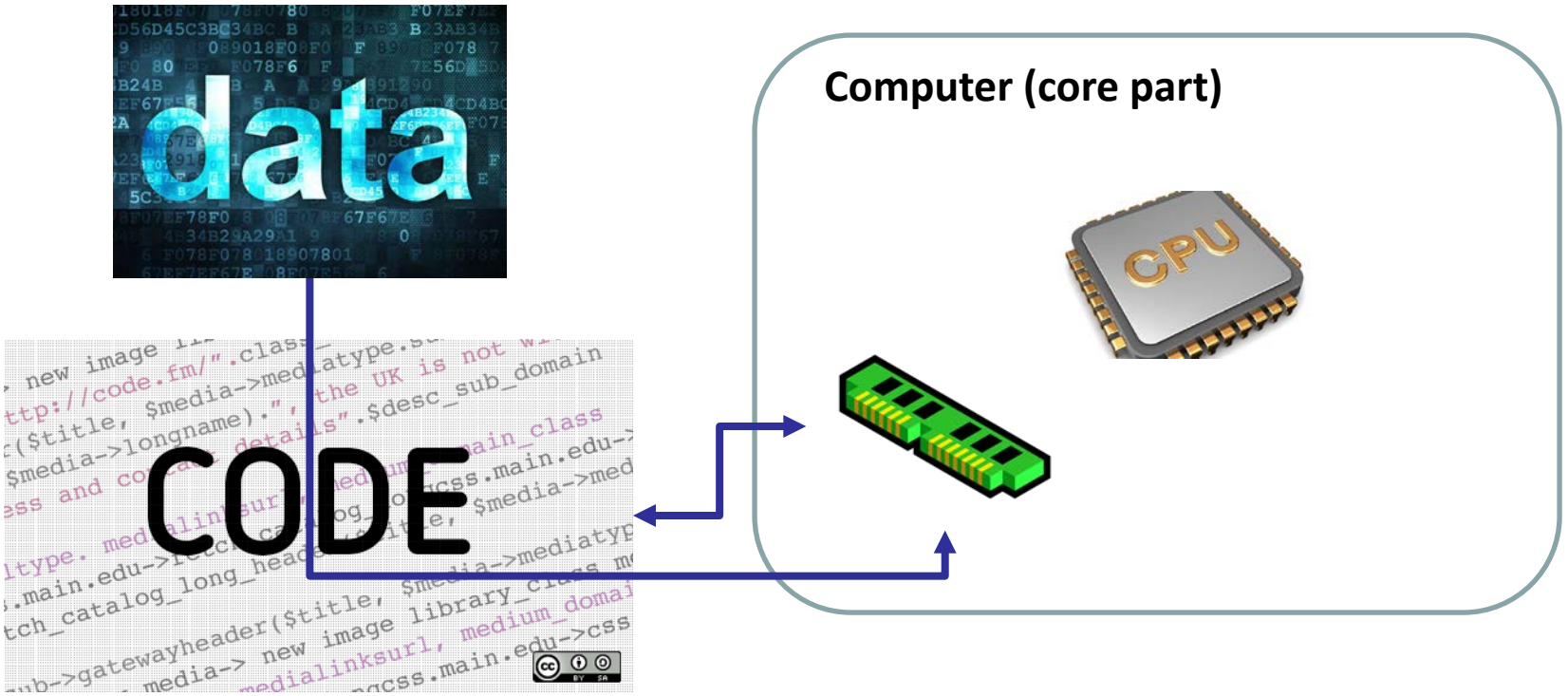
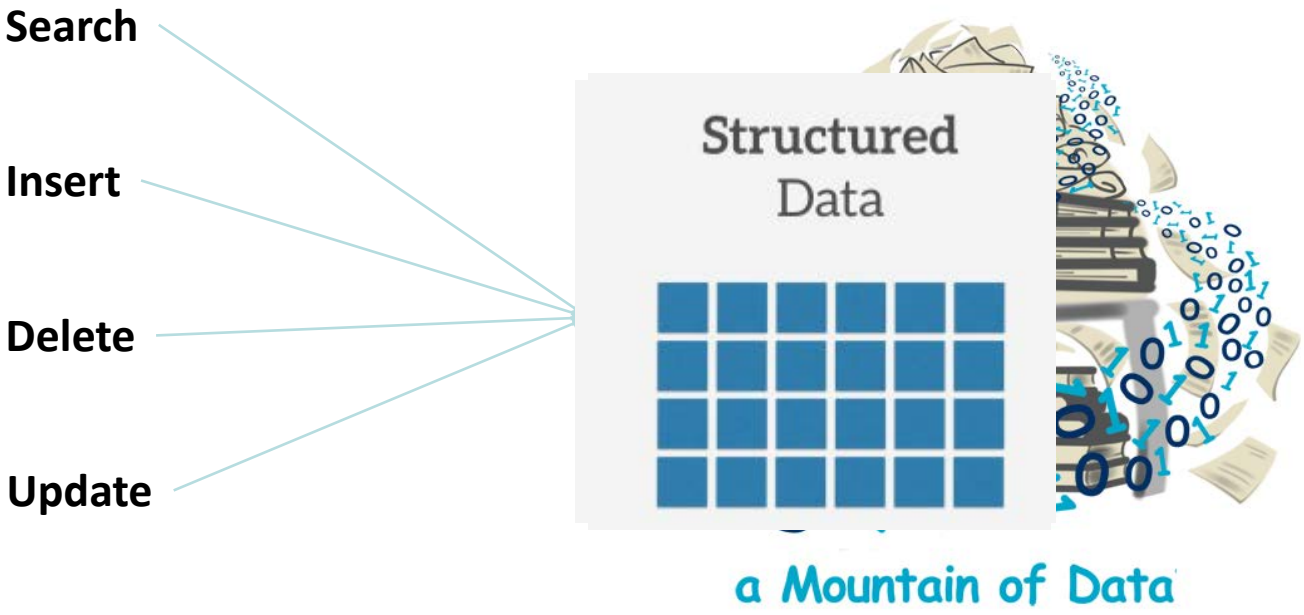
Overview of This Lecture

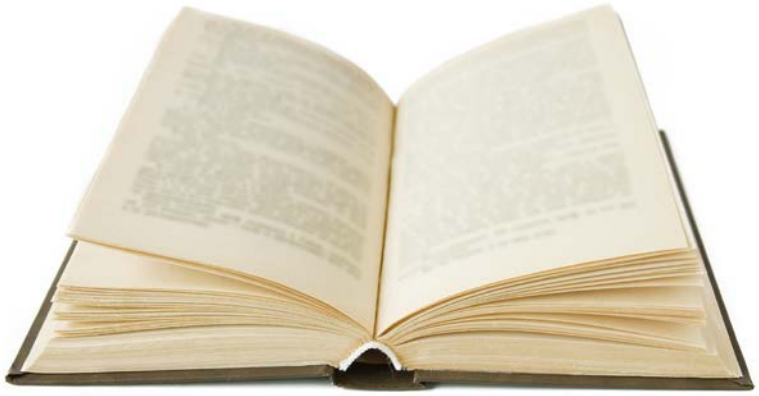
Overview of Programs

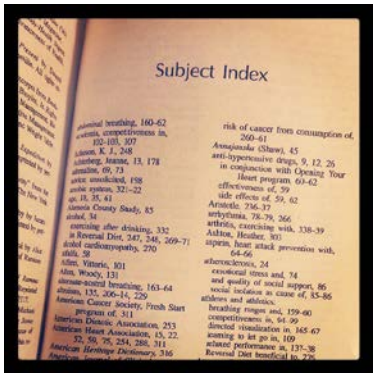


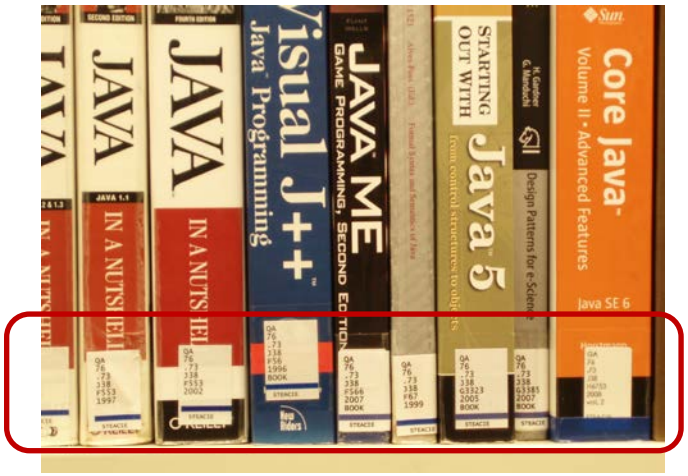
Overview of Data Structures



E.g., In Real Life (Book & Library)

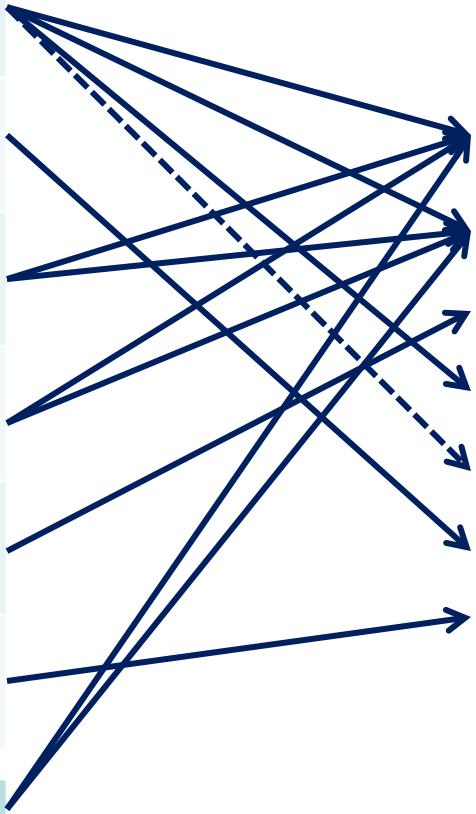






Overview Related to STL in C++

STL in C++	Function
multiset / multimap	Insert, delete & search
set / map	Insert, delete & search
stack	Insert & delete’
queue	Insert & delete’
priority queue	Insert & delete’
hash_map	Insert, delete & search
Graph	



Basic Data Structures	STL in C++
Array (static & dynamic)	vector (dynamic)
List	list
Heap	-
Binary search tree	-
Balanced BST	-
B-tree	-
Hash table	-

Comparison of Time Complexities

Size of n	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$
10^3	10 ns	1 µs	10 µs	1 ms
10^6	20 ns	1 ms	20 ms	16.6 m
10^9	30 ns	1 s	30 s	31.7 year
10^{12}	40 ns	16.6 m	11 hr	31.7 mega-year
10^{15}	50 ns	11.6 day	1.6 year	31.7 tera-year
10^{18}	60 ns	31.7 year	1.9 kilo-year	N.A.
10^{21}	70 ns	31.7 kilo-year	2.2 mega-year	N.A.

We assume that unit operation takes 1 ns (= 10^{-9} s).

Examples of Using Efficient DS

Algorithm	Time complexity	Used DS
Selection Sort	$O(n^2)$	Array
Insertion Sort	$O(n^2)$	Array

Algorithm	Time complexity	Used DS
Heap Sort	$O(n \log n)$	Binary heap
Tree Sort	$O(n \log n)$	Balanced BST

Examples of Using Efficient DS

Algorithm	Time Complexity	Used DS
Prim's algorithm for the minimum spanning tree problem	$O(n^2)$	Array
	$O(m \log n)$	Binary heap & List
	$O(m + n \log n)$	Fibonacci heap & List

Note that: n = # of vertices & m = # of edges & $n \leq m \leq n^2$

Efficiency of Basic DS

Basic Data Structures	Insert	Delete	Search
Array	$O(1)$	$O(n)$	$O(n)$
List	$O(1)$	$O(n)$	$O(n)$
Heap	$O(\log n)$	$O(\log n)$	-
Binary search tree	$O(n)$	$O(n)$	$O(n)$
Balanced BST	$O(\log n)$	$O(\log n)$	$O(\log n)$
B-tree	$O(\log n)$	$O(\log n)$	$O(\log n)$
Hash table (perfect)	$O(1)$	$O(1)$	$O(1)$
Hash table (worst)	$O(n)$	$O(n)$	$O(n)$