Lecture 10: Basics of Data Structures

BT 3051 – Data Structures and Algorithms for Biology

Karthik Raman
Department of Biotechnology
Indian Institute of Technology Madras

September 1, 2014

Selecting a Data Structure

Important Questions

- Are all data inserted into the data structure at the beginning, or are insertions interspersed with other operations?
    - i.e. are the data static or dynamic?
- Can data be deleted?
    - This may often demand a more complex representation
- Are all data processed in some well-defined order, or is random access allowed?

Skiena on Data Structures

"Changing a data structure in a slow program can work the same way an organ transplant does in a sick patient. Important classes of abstract data types such as containers, dictionaries, and priority queues, have many different but functionally equivalent data structures that implement them. Changing the data structure does not change the correctness of the program, since we presumably replace a correct implementation with a different correct implementation. However, the new implementation of the data type realizes different trade-offs in the time to execute various operations, so the total performance can improve dramatically."

Properties of a Data Structure

- Efficient utilization of memory and disk space
- Efficient algorithms for:
    - manipulation (e.g. insertion / deletion)
    - data retrieval (e.g. find)
    - creation
- A well-designed data structure uses less resources
    - computational: execution time
    - spatial: memory space

Abstract Data Type: Linear List

- A list of items of a finite length *n*

Operations

- `create()`
- `delete()`
- `isEmpty()`
- `length()`
- `find()` *k*-th element
- `search()` for a given element
- `insert()` an element into the list
- `append()`, `join()`, `copy()`, ...

Think about the cost of each operation ...

A Linear List can be implemented using a contiguous or linked data structure …

Contiguous vs. Linked Data Structures

Skiena

Data structures can be neatly classified as either contiguous or linked, depending upon whether they are based on arrays or pointers:

- Contiguously-allocated structures are composed of single slabs of memory, and include arrays, matrices, heaps, and hash tables
- Linked data structures are composed of distinct chunks of memory bound together by pointers, and include lists, trees, and graph adjacency lists

Contiguous vs. Linked Data Structures

Linked

- Extra storage required
- Better use of fragmented memory
- Insertion/deletion at middle is easier
- Joining lists easier
- ‘Next’ operation requires pointer dereference

Contiguous

- Next and Previous are implicit (less storage)
- Can take advantage of locality
- Random access
- ‘Next’ operation probably faster

Important Data Structures/ADTs

Table 1.1 from Data Structures & Algorithms in Java, by Robert Lafore

Data Structure	Advantages	Disadvantages
Array	Quick insertion, very fast access if index known	Slow search, slow deletion, fixed size
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	Provides first-in, first-out access	Slow access to other items
Linked list	Quick insertion, quick deletion	Slow search
Binary Tree	Quick search, insertion, deletion (if tree remains balanced)	Deletion algorithm is complex
Red-black trees	Quick search, insertion, deletion; tree always balanced	Complex
2-3-4 trees	Quick search, insertion, deletion. Tree always balanced. Similar trees good for disk storage	Complex
Hash table	Very fast access if key known; fast insertion	Slow deletion, access slow if key not known, inefficient memory usage
Heap	Fast insertion, deletion, access to largest item	Slow access to other items
Graph	Models real-world situations	Some algorithms are slow and complex