COMP 250: Review (2)

Jérôme Waldispühl
School of Computer Science
McGill University
Data Structures

• Array:
  running time for insert, delete, find...

• Single-linked list
  Better than arrays:
    Easier to insert and delete
    No need to know size in advance
  Worse than arrays:
    finding the n-th element is slow (so binarySearch is hard)
    Require more memory (for the "next" member)

• Doubly-linked list
  Allow to move backward
  Makes deleting elements easier

• Stacks and queues
  You should understand all applications we saw
ADT (Abstract Data Structure)

What it is?
  Description of the interface of a data structure. It specifies:
  • What type of data can be stored
  • What kind of operations can be performed
  • Hides the details of implementation

Why it is important?
  Simplifies the way we think of large programs
Trees

• treeNode representation

• Vocabulary: node, leaf, root, parent, sibling, descendants, ancestors, subtree rooted at x, internal and external nodes, ordered, binary, proper binary

• Depth and height
  • Definition
  • Computation of...

• Tree traversal:
  • Pre-order, In-order, Post-order
Dictionary ADTs

• Stores pairs (key, info)
• Operations: find(key), insert(key, info), remove(key)
• Cases where array implementation is bad
• Cases where linked-list implementation is bad
Dictionary ADTs with Binary Search trees (BST)

- **Property:** for any node \( x \),
  - keys in the left subtree of \( x \) have keys smaller or equal to \( \text{key}(x) \) and
  - keys in the right subtree of \( x \) have keys larger or equal to \( \text{key}(x) \)

- **Algorithm to find a key and its running time** \( O(h) = O(\log n) \) if the tree is balanced

- **Inserting a new key.** Running time \( O(h) \). Sequence of insertion that can lead to bad running times.

- **Removing a key.**

- **You need to be able to execute these algorithms by hand on examples**
Dictionary ADTs with Hash Tables

• Implements a dictionary
• Idea:
  • map keys to buckets
  • Each bucket is itself a dictionary
• Hash functions:
  • Goal: minimize collisions
  • Easy to compute
• Best case:
  • keys are distributed uniformly among the buckets. Each bucket contains few keys
• Worst case:
  • All keys end-up in the same bucket
Priority queues

• Heap property:
  • key(x) is smaller or equal to the keys of children of x.
  • All h-1 first levels are full, and in the last level, nodes are packed to the left

• Operations:
  • findMin(). Algorithm. O(1)
  • insert(key). Bubbling-up. O(log n)
  • removeMin(). Bubbling-down. O(log n)

• Array representation of heaps

• HeapSort
  • insert keys one by one
  • removeMin() one by one
Graphs

• All the terminology

• Data structures for representing graphs:
  • Adjacency-list
  • Adjacency-matrix
  • Running time of basic operations with each data structure

• Graph traversal
  • Depth-first search
    • Recursive
    • Iterative using a stack
  • Breadth-first search
    • Iterative using a queue

• IMPORTANT:
  Applications of DFS and BFS
Graph problems

• Shortest path (with variations)
• Eulerian and Hamiltonian cycles
• Graph coloring
• Cliques
• Matching
• Google algorithm
Game strategy

• Single-player
  • Bactracking algorithms
    • 8-queens problem
    • Build solution by adding queens one by one
    • Abandon search as soon as invalid position is reached, and backtrack

• Multi-player
  • Game tree
  • Winning and losing positions
  • Minimax principle
  • evaluation functions for non-terminal boards
Heuristics

• Useful when we don’t have fast exact algorithms
• Greedy algorithms
• Fastest descent algorithms
  • With randomization
  • Genetic algorithms
Sample Question 1

Given a queue implemented by rotating array. The length of this array is 6. The index of the head is 3, the index of the tail is 0. When we remove one element, then add two elements into this queue, what is the head and tail index?

A. 1, 5
B. 2, 4
C. 4, 2
D. 5, 1
E. 0, 3
Sample question 2

What does the following method return?
doSomething( Queue q){
    #initialize new empty Stack s
    while !( q.isEmpty() ) {
        s.push( q.dequeue() )
    }
    while !( s.isEmpty() ) {
        q.enqueue( s.pop() )
    }
    return q
}

A. An empty queue
B. A queue with a single element
C. A queue with elements in same order as input q
D. A queue with two copies of each element
E. A queue with elements in the reverse order as input q
Sample question 3

Consider the following DoublyLinked List with three nodes.

A. Which of the following evaluates to true?
B. head.next == tail.prev.prev.next
C. head.next.prev == tail.prev
D. head.next.next == tail.prev
E. head.next.prev == tail.prev.prev.next
202 Intro Program

206 Software Systems
- 310 Oper. Sys.

273 Computer Systems
- 302 Program Lang

250 Intro Comp Sci
- 303 Software Design

MATHEMATICS
(prereqs for many upper level COMP courses)
- 240 Disc. Str. 1
- 223 Linear Alg.
- 222 Cal III
- 323 Prob.

251/2 Alg & Data Str
- 350 Num. Meth
- 360/2 Alg. Design
- 330 Theory Comp.

SYSTEMS & SOFTWARE
(compilers, concurrency, databases, distributed sys, networks, ..)

APPLICATIONS
(AI, bioinf, graphics, vision, games, NLP, machine learning, ...)

THEORY
(crypto, optimization, game theory, logic, correctness, computability..)