CS 367 - Introduction to Data Structures
Thursday, December 15th, 2016

Final Exam
- Monday, December 19th, 5:05 PM to 7:05 PM
  - Lecture 1 (all students): Room 105 of Psychology Building
  - Lecture 2 (Last names A thru V): Room 2103 of Chamberlin Hall
  - Lecture 2 (Last names W thru Z): Room 103 of Psychology Building
  - Lecture 3 (all students): Room 113 of Psychology Building
- UW ID required
- Makeup exam emails sent
- See posted exam information
- Solution to sample questions will be posted on LEARN@UW on Saturday

Program 5 due 10 pm TONIGHT Thursday, Dec 15th

Verify that your scores are correctly entered on Learn@UW.
Send your instructor an email if there is an inconsistency.

Last Time
Better Sorts
- heap sort
- merge sort
- quick sort
Stable Sorts
Sorting in Java

Today
Radix Sort
Sorting out Sorting
Course Overview Sheets
Final Exam Info
Evaluations (if possible, bring web enabled device to lecture – to complete evals online)
Radix Sort

Assumptions

number of items (N):

range of unique digits (RANGE):

length of item’s sequence of digits (LEN):

Idea

Sort the following integers:

121 367 354 873 777 333 123 222 411 262 897

→ What is N?  RANGE?  LEN?

Pass 1:
0  1  2  3  4  5  6  7  8  9

Pass 2:
0  1  2  3  4  5  6  7  8  9

Pass 3:
0  1  2  3  4  5  6  7  8  9
Radix Sort

Algorithm

List[] digitQ = new List[RANGE]
for (i = 0; i < RANGE; i++)
    digitQ[i] = new Queue()
for (pos = LEN-1; pos >= 0; pos--)
    for (j = 0; j < N; j++)
        let x = digit in pos position of the item in A[j]
        digitQ[x].enqueue(A[j])

index = 0
for (j = 0; j < RANGE; j++)
    while (!digitQ[j].isEmpty())
        A[index] = digitQ[j].dequeue()
        index++

Complexity

O(N + R) * LEN vs O(N log N)
Abstract Data Types (ADTs) and Data Structures (DS)

ADT
DS

Layout of Collection

- **Linear**
  
  List
  
  array, SimpleArrayList, shadow array
  
  chain of nodes, ListNode, SimpleLinkedList
  
  tail, header, doubly linked, circularly

- **Linked**
  
  Stack
  
  Queue
  
  Deque
  
  circular array

- **Hierarchical**
  
  general tree, Treenode
  
  binary tree, BinaryTreenode
  
  binary search tree, BSTnode
  
  balanced search tree
  
  red-black tree

- **Graphical**
  
  Graph
  
  Graphnode
  
  adjacency matrix
  
  adjacency list

Orientation of Operations

- **Position oriented** - operations occur at a specified position
  
  list, stack (top), queue (front/rear), deque ("double ended")

- **Value oriented** - operations occur at position determined by item’s key value
  
  sorted list
  
  search trees
  
  Map
  
  hash table

- **Hybrid?**
  
  PriorityQueue
  
  heap
  
  hash table

Use these references pages as you wish for Exam Preparation
Algorithms

Operations on ADTs/data structures

insert, lookup, delete

Recursion

vs. iteration
rules, guiding questions
call stack trace
execution tree trace

Traversing

list
tree      level      pre/in/post
graph     DFS (stack)  BFS (queue)  spanning trees

Searching

linear O(N)
binary O(logN)

Hashing

hash function: hash code (extracting, weighting, folding) → hash index (compressing)
table size: prime size, load factor, rehashing
collisions: open addressing, buckets

Graphs

topological ordering
Dijkstra’s (priority queue)

Sorting

basic O(N^2): bubble, insertion, selection
better O(NlogN): heap, merge, quick
stable sorts
Complexity

Complexity

1, logN, N, NlogN, N\(^2\), N\(^3\), 2\(^N\), N!

time: abstract, dominant ops

space: memory

worst/average/best-case

big-O

Determining Complexity

informal
constant
linear
quadratic
code
loops
method calls
time equation
simplify

recurrence equations
base \( T( ) = \)
recursive \( T(N) = + T( ) \)
equations \( \rightarrow \) table, guess solution \( \rightarrow \) verify \( \rightarrow \) complexity

Caveats

small problem size
same complexity
Java Concepts

Primitives vs. References

Command-line Arguments

Exceptions

throw
try/catch/finally
throws (checked vs unchecked)
defining

Programming for Generality

Object
generics

Interfaces

Comparable, compareTo
ADTs

Iterators

Iterable: iterator()
Iterator: hasNext(), next()

indirect
direct

Package Visibility

Java Collections Framework

Iterable<T>, Iterator<E>
List<T>: ArrayList<T>, LinkedList<T>
Vector<E>, Stack<E>
Hashtable<K,V>
Map<K,V>: TreeMap<K,V>, HashMap<K,V>
Set<E>: TreeSet<E>, HashSet<E>
Course Evals

Online Course Evaluations
www.aefis.wisc.edu
Log In with your Net ID, find course, take survey.

Course evaluations are the primary way that we learn what students feel about the courses we teach.

Comments help us know what is and is not working for students.

Comments work best when hear the good and the bad (to make sure that we don’t accidently change something that was working in our effort to change what was not working)

(complete cs367 Course Evaluation at home or in-class on Thursday)

Thank you for taking your time to share your thoughts on cs367!

Questions to Consider

What was effective in helping you learn?

What wasn’t effective in helping you learn?

What would you use to evaluate the extent of your learning?

If you were instructor what would you do differently?