CS 367 - Introduction to Data Structures  
Thursday, May 5, 2016  

Final Exam  
- Sunday, May 8th, 2:45 to 4:45 pm  
- Lec 1: room 272 of Bascom Hall  
- Lec 2: room 2650 of Mosse Humanities Building (note new room)  
- Lec 3: room 1351 of Chemistry Building (note new room)  
- 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 Tomorrow, May 6th  

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 (bring a pencil)
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
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

- hierarchical

  Stack
  Queue
  Deque
  circular array

- 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
  hash table

- hybrid?

  PriorityQueue
  heap
  hash table
Algorithms

Operations on ADTs/data structures

insert, lookup, delete

Recursion

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

Traversing

list
tree
graph
level
pre/in/post
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

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 \[ T(N) = \]
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>