CSC 477/677 Spring 2022 Final Examination 1:00-3:00 May 9, 2022

Topics to Study

1. Asymptotic Classes of Functions
   - (i) O, Ω, and Θ
   - (ii) Logarithmic and Polylogarithmic
   - (iii) Polynomial
   - (iv) Exponential
2. Time Complexity and Asymptotic Time Complexity of Programs
   - (i) Complexity of Algorithm we've Learned
   - (ii) Recurrences
3. Computational Paradigmns
   - (i) What is a greedy algorithm?
   - (ii) What is a dynamic programming algorithm?
     Relationship with directed acyclic graphs
   - (iii) What is a divide and conquer algorithm?
     Relationship with recursion
4. Sorting
   - (i) Selection sort.
   - (ii) Insertion sort.
   - (iii) Margesort.
   - (iv) Quicksort.
   - (v) Heapsort.
   - (vi) Treesort.
   - (vii) Radix, or Bucket, sort.
   - (viii) Theoretical $\Omega(n \log n)$ bound.
5. Searching
   - (i) Linear search.
   - (ii) Binary search.
6. Selection
   - (i) Median of medians.
   - (ii) Randomized selection.

7. Graphs

   (i) BFS and DFS
   (ii) Vertices and edges
   (iii) Neighbor list
   (iv) Subgraphs
   (v) Connected graphs
   (vi) Components
   (vii) Acyclic graphs
   (viii) Planar graphs, Euler's formula: $m \leq 3n - 6$
   (ix) Complete graphs
   (x) Trees

      A graph is a tree if it is acyclic and connected.
   (xi) Weighted graphs, minimum spanning trees

   Directed graphs = digraphs

   (i) Vertices and arcs
   (ii) In-neighbors, out-neighbors
   (iii) Subgraphs
   (iv) BFS and DFS
   (v) Strongly connected digraphs
   (vi) Strong components
   (vii) Acyclic digraph = dag (very different from acyclic graphs)
   (viii) Topological order of a dag
   (ix) Trees (not the same definition as above)
   (x) Transitive closure and transitive reduction of a digraph
   (xi) Weighted digraphs and shortest path problems
      i. Single pair.
      ii. Single source.
      iii. All pairs.
      iv. Bellman-Ford algorithm.
      v. Floyd-Warshall algorithm.
      vi. Dijkstra's algorithm.
      vii. Johnson's algorithm.
      viii. A^* algorithm.

8. Search Structures A search structure holds items, and can be searched for a specific item. Items can be inserted or deleted.

   (i) Unordered Lists.

      i. Move to Front.

   (ii) Trees.

      i. Binary Search Tree.
      ii. B-tree.
      iii. Treap.

   (iii) Hash Tables.

      i. Collisions
      ii. Closed hashing, open addressing.
      iii. Open hashing, separate chaining.
      iv. Cuckoo hashing.
      v. Perfect hashing.

9. Priority Queues A priority queue holds a number of items, one of which is the item of highest priority. Any item can be inserted into a priority queue, but only the highest priority item can be deleted.

   Typically, the items in a priority queue represent unfulfilled obligations, and deletion corresponds to fulfilling that obligation.

   (i) Stack. The item of highest priority is the most recently inserted item. Insertion of an item is usually called *push*, and deletion of an item is usually called *pop*. A stack is usually implemented as a list ordered by priority, that is, the time the item was inserted.

      Stacks are the most important priority queues.

   (ii) Queue. The item of highest priority is the least recently inserted item. Insertion of an item is sometimes called *enqueue*, and deletion of an item is sometimes called *dequeue*. A queue is usually implemented as a list ordered by priority, that is, the time the item was inserted.

      My favorite two implementations of a queue are

      i. An array of fixed length where the front and rear are moving indices in the array. In C++ you could use vectors.
      ii. A circular linked list, with a pointer to a dummy node.

   (iii) Heap. The item of highest priority is the item which has the largest or smallest value of some key, such as size, weight, or whatever. The heap is said to be a maxheap or minheap, if priority item has the largest or smallest key, respectively. In a pure heap, the key cannot be altered while it is in the heap, but there are variations where the key can be changed, causing the heap to be reorganized. For example. in Dijkstra's algorithm, the heap is a minheap, and the key of an item in the heap can be decreased.

      There is more than one way to implement a heap, but I only showed one of those in class. The heap is implemented as an almost complete binary tree, in turn implemented as an array, where the tree nodes are stored in level order.

10. Arrays

    (i) Fetch and store

    (ii) Row-major and column-major order

    (iii) Storing a multi-dimensional array in main memory

    (iv) Triangular and ragged arrays

    (v) Sparse arrays

        How can you implement a sparse array using a search structure?

11. Dynamic Programming

    (i) Identify subproblems.

    (ii) Work the subproblems in topological order.

    (iii) Memoization.

    If I give you a dynamic programming problem to solve (other than Levenstein distance) I will tell you what the subproblems are: the rest is up to you.

    Example: Given a sequence $x_1, \ldots x_n$, find a monotone increasing subsequence of maximum length.

    There are n subproblems. Subroblem i is to find the maximal length monotone subsequence of $x_1, \ldots, x_i$ which ends at x_i.

12. Huffman's Algorithm

13. Loop Invariants