CSci 231 Final Review

Here is a list of topics for the final. Generally you are responsible for anything discussed in class and anything appearing on the homeworks.

1. Asymptotic growth of functions (O, Ω, Θ)
2. Summations
   - Basic summations: arithmetic, geometric, harmonic
3. Recurrences
   - Iteration
   - Substitution (induction)
4. (Comparison-based) Sorting

   stable sort, in-place sort
   - Insertion sort
   - Mergesort
   - Quicksort (Partition)
   - Randomized quicksort
   - Heapsort
   - Comparison-based sorting lower bound
5. Linear-time sorting
   - Counting sort
   - Radix sort
   - Bucket sort
6. Selection
7. (Abstract) Data structures
   - Priority queue (FIND-MIN, DELETE-MIN, INSERT, DELETE, CHANGE-KEY)
     - Max priority queue, Min priority queue
   - Dictionary (INSERT, DELETE, SEARCH)
   - Union-Find (MAKE-SET, FIND-SET, UNION-SET)

8. Data structure implementations
    - Priority queue
        - Heap
            * heap property
            * HEAPIFY, BUILDHEAP
    - Dictionary
        - Binary search trees (INSERT, DELETE, SEARCH, MIN, MAX, PRED, SUCC)
            * binary search tree property
            * tree walks
        - Red-black trees
            * red-black tree invariants
    - Union-Find
        - Linked list (and pointers to head of list) with weighted-union heuristic
9. Augmented search trees
    - augment every node x with some additional information $f(x)$; maintain $f(x)$ under INSERT and DELETE in the same asymptotic bounds.
    - Sufficient if $f(x)$ can be computed using only $f(left(x))$ and $f(right(x))$.
    - implement SELECT(i), RANK(x) by augmenting every node x with $size(x)$
    - Interval tree
10. Dynamic programming
    - optimal substructure, overlapping subproblems
    - recursive formulation,
    - running time without storing table
    - running time with dynamic programming (storing table)
    - examples: weighted interval scheduling, subset sum, weighted subset sum (0-1 knapsack), sequence alignment, longest common subsequence, shortest paths: Bellman Ford, APSP
11. Greedy algorithms
    - Correctness proof
    - examples: interval scheduling, fractional knapsack, Dijkstra, Prim, Kruskal
12. Graph algorithms
    - Basic definitions, graph representation (adjacency list, adjacency matrix)
    - Graph traversal: BFS
        - G directed or undirected
        - find connected components (G undirected), check bipartiteness (G undirected), compute shortest paths (G un-weighted, all edges have weight 1)

- Graph traversal: DFS
  - G directed or undirected
  - find cycles (back edges), topological sort (G directed, acyclic)
- Minimum spanning tree (MST)
  - G connected, undirected, weighted
  - Prim's MST algorithm
  - Kruskal's MST algorithm
- Shortest paths: SP
  - Dijkstra's SSSP algorithm
    - G directed or undirected, weighted, non-negative weights
  - SP on DAGs
  - SSSP on graphs with negative edge weights
  - SP with dynamic programming
    - APSP and matrix multiplication
    - APSP Floyd-Warsall's algorithm

Review Questions

1. Is it true that $\sum_{i=0}^{i=n}(3/4)^i = O(1)$?

2. Give a formula for the arithmetic sum: $0 + 1 + 2 + ... + n = \sum_{i=0}^{i=n} i =$

3. Is it true that $7^{\lg n} = O(n^3)$?

4. Is it true that $7^{\lg n} = \Omega(n^2)$?

5. Is it true that $\log \sqrt{n} = \Theta(\sqrt{\log n})$?

6. Is Mergesort stable?

7. Is Quicksort in place?

8. Is is true that the running time of Quicksort is $\Theta(n^2)$?

9. Is it true that the worst-case running time of Quicksort is $\Theta(n^2)$?

10. What is the running time of Mergesort when input is sorted in reverse order?

11. What is the running time of Heapsort when input is sorted?

12. Which of the following algorithms (as presented in class) is stable: Quicksort, Heapsort, Counting Sort?

13. Recall that a sorting algorithm is *in place* is it requires $O(1)$ additional storage. Which of the following algorithms (as presented in class) is in place: Quicksort, Heapsort, Counting Sort?

14. You have a heap containing n keys. As a function of n, how long does it take to change a key? To delete a key?

15. Is it true that any sorting algorithm must take $\Omega(n \log n)$ in the worst-case?

16. Counting Sort sorts n integers in time $O(n + k)$. What is the assumption?

17. How fast can one find the minimum element in an array of n elements in the best case? In the worst case?

18. Is it true that $SELECT(A, n)$, where A is an array of n elements, returns the largest element in the array?

19. Given a node x in a min heap, with children nodes l and r, what does the heap property tell us about $x.key, l.key, r.key$?

20. Is it true that building a heap of n elements takes $O(n)$?

21. Given a heap with n keys, is it true that you can search for a key in $O(\log n)$ time?

22. Given a binary search tree with n elements, its height can be as small as $\Omega($ $)$ and as large as $O($ $)$.

23. Is it true that the height of a red-black tree with n elements is $\Theta(\log n)$?

24. How long does it take, in the worst case, to build a red-black tree of n elements?

25. You have a binary search tree T with n elements. Is it true that the predecessor of an element in T can be found in $O(1)$ time?

26. Is it true that some dynamic programming problems can be solved faster using greedy algorithms?

27. As a function of $|V|$, what is the minimum number of edges in a connected undirected graph with $|V|$ vertices?

28. How many edges are in a complete graph with $|V|$ vertices?

29. You have an undirected, connected, weighted graph G and a source vertex s. Is it true that BFS computes shortest paths from s to every other vertex?

30. Is it true that Prim's and Kruskal's algorithms are greedy?

31. Is it true that Dijkstra's SSSP algorithm works only on graphs with non-negative edge weights?

32. Let $p = u \to v_1 \to v_2 ... \to v_k \to v$ be the shortest path from u to v in a graph G. Let v_i and v_j be two vertices on p such that $1 \leq i < j \leq k$. Is it true that the subpath $v_i \to v_{i+1} ... \to v_j$ in p is the shortest path in G from v_i to v_j?

33. Can a shortest path contain cycles?

34. Is it true that running Dijkstra's algorithm on a graph $G = (V, E)$ takes $O(|E| \cdot \log |V|)$?

35. You have a complete graph with $|V|$ vertices. How long does it take to run Dijkstra's algorithm on this graph?

36. Is it true that you can compute a MST using a Union-Find data structure?

37. How long does it take to run Prim's algorithm on a graph $G = (V, E)$?

38. Is it true that Prim's algorithm only works on undirected graphs?

39. How fast can you compute shortest paths between two arbitrary nodes in a graph with negative edge weights?

40. How fast can one identify negative cycles in a graph?

Practice problems

1. Rank the following functions in increasing order of asymptotic growth.

$$500n^3, 17n + (2/n^2), 7n \log\log n, 4\log_3 n, 20\log(n^2), 2^{3\log n}, 2^{\sqrt{n}}$$

2. In this problem we consider a data structure $\mathcal{D}$ for maintaining a set of integers under the normal INIT, INSERT, DELETE, and FIND operations, as well as a COUNT operation, defined as follows:

   - INIT($\mathcal{D}$): Create an empty structure $\mathcal{D}$.
   - INSERT($\mathcal{D}$,x): Insert x in $\mathcal{D}$.
   - DELETE($\mathcal{D}$,x): Delete x from $\mathcal{D}$.
   - FIND($\mathcal{D}$,x): Return pointer to x in $\mathcal{D}$.
   - COUNT($\mathcal{D}$,x): Return number of elements larger than x in $\mathcal{D}$.

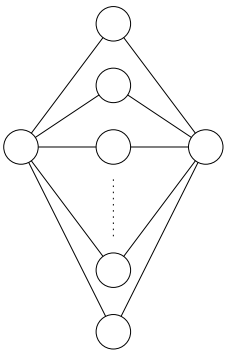
   Describe how to modify a standard red-black tree in order to implement $\mathcal{D}$ such that INIT is supported in $O(1)$ time and INSERT, DELETE, FIND, and COUNT are supported in $O(\log n)$ time.

   b) Given an array $S[1..n]$ of integers, an *inversion* is a pair of elements $S[i]$ and $S[j]$, $i < j$, such that $S[i] > S[j]$. How many inversions does the array $[5, 2, 7, 1, 9, 4, 6]$ have?

   c) Using the data structure $\mathcal{D}$ designed in problem a), describe an $O(n \log n)$ algorithm for computing the number of inversions in an array $S[1..n]$.

3. You are given an array $A[1..n]$ of real numbers, some positive, some negative. Design an $O(n \log n)$ algorithm which determines whether A contains two elements $A[i]$ and $A[j]$ such that $A[i] = -A[j]$. (If A contains the element 0, then the answer is always YES.)

4. Consider a *pole-graph* which is an undirected graph with positive edge weights, consisting of two poles connected through a layer of nodes as follows:



Let n be the number of vertices in a pole-graph and assume that the graph is given in normal edge-list representation.

(a) How long would it take Dijkstra's algorithm to find the single-source-shortest-paths from one of the poles in a pole-graph to all other nodes?

(b) Describe and analyze a more efficient algorithm for solving the single-source-shortest-paths problem on a pole-graph. Remember to prove that the algorithm is correct.

5. The binomial coefficient $C(n,k)$ counts the number of ways of choosing k distinct items from a set of n items. It can be defined as follows:

$$\begin{array}{ll} C(n,k) = C(n-1,k-1) + C(n-1,k) & \text{for } n > 0 \text{ and } k > 0 \\ C(n,0) = 1 & \text{for } n \geq 0 \\ C(0,k) = 0 & \text{for } k > 0 \end{array}$$

The procedure $\textsc{Bino}(n,k)$ below computes $C(n,k)$ using the recursive formulation above.

```
Bino(n,k)
    if(k=0)
       return 1
    else if (n=0)
       return 0
    else
       return Bino(n-1,k-1) + Bino(n-1,k)
end
```

(a) Write the recurrence for the running time $T(n,k)$ of $\textsc{Bino}(n,k)$ and show that it is exponential. (*Hint: Show that* $T_{ComputecBinCoeff}(n,k) = \Omega(2^k)$.)

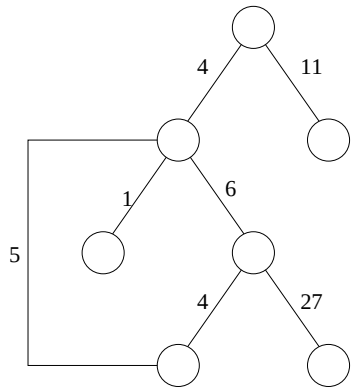
(b) Describe a dynamic programming algorithm for calculating $C(n,k)$ and analyze its running time.

6. Suppose the degree requirements for a computer science major are organized as a *dag* (directed acyclic graph), where vertices are required courses and an edge (x, y) means course x must be completed prior to beginning course y. Make the following assumptions:

   - All prerequisites must be obeyed.
   - There is a course, CPS1, that must be taken before any other course.
   - Every course is offered every semester.
   - There is no limit to the number of courses you can take in one semester.

   Describe an efficient algorithm to compute the minimum number of semesters required to complete the degree and analyze its running time.

7. Consider an undirected weighted graph which is formed by taking a binary tree and adding an edge from *exactly one* of the leaves to another node in the tree. We call such a graph a *loop-tree*. An example of a loop-tree could be the following:



   Let n be the number of vertices in a loop-tree and assume that the graph is given in the normal edge-list representation without any extra information. In particular, the representation does not contain information about which vertex is the root.

   (a) How long time would it take Prim's or Kruskal's algorithms to find the minimal spanning tree of a loop-tree? Make your bound as tight as possible.

   (b) Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a loop-tree. Remember to prove that the algorithm is correct.