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
   - Bounding summations
3. Recurrences
   - Iteration
   - Substitution (induction)
4. (Comparison-based) Sorting
   - Insertion sort
   - Mergesort
   - Quicksort (Partition)
   - Randomized quicksort
   - Heapsort
   - Comparison-based sorting lower bound
5. Linear-time sorting
   - Counting sort
   - Radix sort
   - Bucket sort

   (remember stable sort, in-place 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
     - Skip lists
   - Union-Find
     - Linked list (and pointers to head of list)
       - 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
   - Matrix-chain multiplication, 0-1 Knapsack
     - recursive formulation,
     - running time without storing table
     - running time with dynamic programming (storing table)

11. Greedy algorithms
   - Correctness proof
   - Activity selection, make change, pharmacist

12. Amortized analysis
   - Accounting method
     - conditions for charged costs to give an upper bound on total running time

- Potential method
  - conditions for potential to give an upper bound on total running time
- Stack with MULTIPOP, binary counter, dynamic table

13. 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 $\Omega(\qquad)$ and as large as $O(\qquad)$.

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 Kruskal's algorithm uses 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. (**Duke Midterm 2002**) Consider the following recurrence

$$T(n) = \begin{cases} T(\sqrt{n}) + \log\log n & \text{if } n > 4 \\ 1 & \text{otherwise} \end{cases}$$

   (a) Using the iteration method find an asymptotic tight bound for $T(n)$.

   (b) Using the substitution method prove that the recurrence has solution $T(n) = O((\log\log n)^2)$.

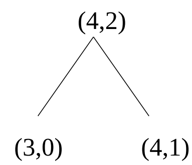
2. Rank the following functions in increasing order of asymptotic growth.

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

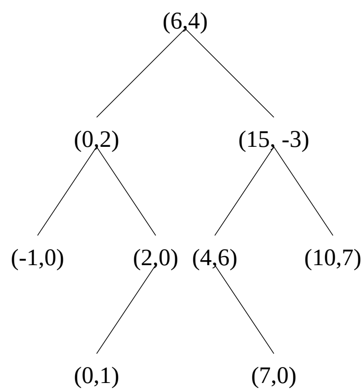
3. You are given an array $A[1..n]$ of real numbers, some positive, some negative. Design an $O(n \log n)$ algorithm which determines whether A contains two elements $A[i]$ and $A[j]$ such that $A[i] = -A[j]$. (If A contains the element 0, then the answer is always YES.)

4. Consider a binary search tree in which each node v contains a key as well as an additional value called *addend*. The addend of node v is implicitly added to all keys in the subtree rooted at v (including v). Let (*key, addend*) denote the contents of any node v.

   For example, the following tree contains the elements $\{5, 6, 7\}$:



   (a) Which elements does the following tree contain?



   (b) Let h be the height of a tree as defined above. Describe how to perform the following operations in $O(h)$ time:

   - FIND(x, T): return YES if element x is stored in tree T
   - INSERT(x, T): inserts element x in tree T
   - PUSH(x, k, T): add k to all elements $\geq x$

   (c) Describe how it can be ensured that $h = O(\log n)$ during the above operations. (*Hint:* show how to perform rotations.)

5. (**Duke Final 2002**) Let $x = x_1 x_2 \ldots x_n$ and $y = y_1 y_2 \ldots y_m$ and $z = z_1 z_2 \ldots z_{n+m}$ be three strings of length n, m, and $n + m$, respectively. We say that z is a *merge* of x and y if x and y can be found as two disjoint subsequences in z.

   Example: *algo*datastruc*ri*tu*th*res*ms* is a merge of *algorithms* and datastructures.

   For $0 \leq i \leq n$ and $0 \leq j \leq m$, $Merge(i, j)$ is TRUE if $z = z_1 z_2 \ldots z_{i+j}$ is a merge of $x = x_1 x_2 \ldots x_i$ and $y = y_1 y_2 \ldots y_j$ ($x = x_1 x_2 \ldots x_i$ is the empty string if $i = 0$. Similarly for y and z.)

We can compute $Merge(i,j)$ using the following formula

$$Merge(i,j) = \begin{cases} X_{ij} \vee Y_{ij} & \text{if } i,j \geq 1 \\ X_{ij} & \text{if } i \geq 1, j = 0 \\ Y_{ij} & \text{if } i = 0, j \geq 1 \\ \text{TRUE} & \text{if } i = 0, j = 0 \end{cases}$$

where X_{ij} is defined as

$$(z_{i+j} = x_i) \wedge Merge(i-1, j)$$

and Y_{ij} is defined as

$$(z_{i+j} = y_j) \wedge Merge(i, j-1)$$

This can be implemented as follows

```
Merge(i,j)
  IF i=0 AND j=0 THEN RETURN True
  IF i>0 THEN X = (z[i+j]==x[i] AND Merge(i-1,j))
  IF j>0 THEN Y = (z[i+j]==y[j] AND Merge(i,j-1))
  IF i>0 and j>0 THEN RETURN X OR Y
  IF j=0 THEN RETURN X
  IF i=0 THEN RETURN Y
END
```

a) Show that the running time of $Merge(n,m)$ is exponential in n and m.

b) Describe an $O(nm)$ algorithm for solving the problem. Remember to argue for both running time and correctness.

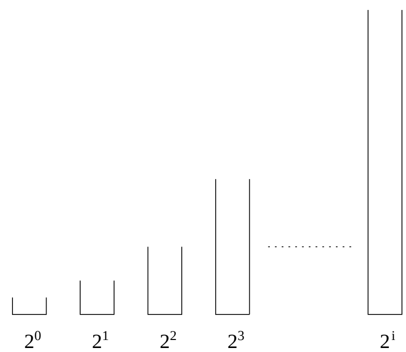
If $Merge(n,m) = \text{TRUE}$ we are interested in knowing which subsequence of z corresponds to x and which corresponds to y in a possible merge. We can characterize such a merge by the indexes of z where a new subsequence starts.

> Example: The merge of *algorithms* and datastructures into *algo*datastruc*ritu*th*res*ms* is described by the indices $1, 5, 14, 16, 18, 20, 23$.

c) Describe how your $O(nm)$ algorithm can be extended such that if $Merge(n,m) = \text{TRUE}$, the algorithm also returns the list of indexes defining a possible merge.

6. (**Duke Final 2002**) Consider a *meta-stack* consisting of an infinite series of stacks $S_0, S_1, S_2, \ldots$ where the ith stack S_i can hold at most 2^i elements. An element x is PUSHed onto a meta-stack by PUSHing x onto S_0. If S_0 is full, its elements are PUSHed onto stack S_1. In general,

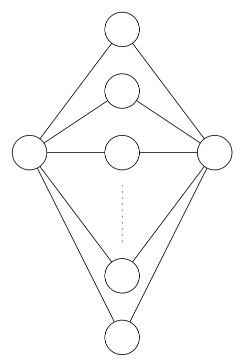
if S_i runs full all its elements are PUSHed onto stack S_{i+1}.



a) What is the worst-case running time of a meta-stack PUSH operation in a sequence of n such operations?

b) Argue that the amortized cost of a meta-stack PUSH operation in a sequence of n such operations is $O(\log n)$.

7. (**Duke Final 2000**) 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.