CS 130A Su19 Practice Final
Please print your name and perm number
True or False

1. All trees are graphs.
2. All undirected, connected graphs are trees.
3. All undirected, acyclic graphs are trees.
4. Kruskal's algorithm is more efficient than Prim's for computing the minimum spanning tree on a sparse graph.
5. A Bloom filter is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. False negative matches are possible, but false positives are not.
6. Dijkstra's shortest path algorithm runs in $O(V^3)$ time.
7. The Bellman–Ford shortest path algorithm runs in $\Theta(V^3)$ time in the worst case.
8. The Floyd–Warshall all-pairs shortest path algorithm runs in $\Theta(V^3)$ time in the worst case.
9. The set of linear modular functions forms a universal hash function family.
10. On a connected, undirected graph with only positive edge costs, adding a positive cost c to every edge will increase the shortest path cost between any two nodes by an integer multiple of c.
11. On a connected, undirected graph with only positive edge costs, adding a positive cost c to every edge will increase the minimum spanning tree cost by an integer multiple of c.

Algorithm Complexity Categorization

For each operation, provide the worst-case Θ runtime for the appropriate n-item data structure.

12. Perfect hash insert
13. Hash table (with separate chaining collision resolution) lookup
14. Perfect hash resize
15. Binary heap build_heap
16. Binary heap build_heap using sort and the median method
17. Binary heap find_min
18. Binary heap delete_min
19. Binary heap increment_key
20. Binary heap insert
21. AVL tree insert
22. AVL tree find
23. Red-Black tree insert
24. Red-Black tree remove
25. 2-3-4 tree insert
26. Leftist heap merge
27. Amortized cost per operation on the union-find data structure
28. Quicksort
29. Find-in-range for a k-d tree of 3 dimensions (returns y items)
30. Find-in-range for a k-d tree of 2 dimensions (returns y items)

31. Recurrence Relation

Solve the following recurrence relations, and provide an example algorithm that can be modelled by the relation (the common name will do). Assume T(1) is a constant

a. $T(n) = 2*T(n/2) + n$

b. $T(n) = n + T(n-1)$

c. $T(n) = 1 + T(n/2)$

32. AVL Problem

a. Insert the following nodes in order into an AVL tree and draw the final result: 15, 20, 10, 5, 12, 25.

b. Insert 22 and draw the final result.

c. Delete 15 and draw the final result.

33. You are implementing a B-Tree. The block size on a disk is 4096 bytes.

a. For a 4-byte key, what is the maximum number of keys that can fit in a single internal node?

b. How many keys can be uniquely stored in the tree?

c. What is the minimum height of a tree that can store all of the unique keys?

34. For a 2-3-4 Tree, start from scratch and insert the numbers 1,2,3,4,5 in sequence. Draw the tree at the conclusion of each insertion.

35. Amortized Analysis

In Programming Assignment 2, you implemented a modified queue that supported constant time push(), pop(), and contains(). To do this, you used a linked-list queue and a hash table. It turns out, you can use an array implementation of a queue, called a circular array. You need to maintain two index values for the head and the tail. The array wraps around by using % length (modulo, not percent). This allows constant time push() and pop(), except for when the array fills up. Then you'll need to double the size of the array and copy everything over. That takes time linear in the number of elements copied. You should probably double the size of the hash table at this point as well. Fortunately, that also just take time linear in the number of elements copied. For this problem, assume we start with a queue of size 1.

a. What is the worst case sequence of m operations?

b. What is the cost of those m operations?

c. What is the amortized cost of one operation?

36. Graph Problem

Given an undirected graph G={E,V} with positive edge costs C_{ij} ((i,j) ∈ E, i ∈ V, j ∈ V) and given a vertex v', v' ∈ V, describe an efficient algorithm to determine if v' is NOT on any shortest path for which it is not an endpoint (obviously v' is on the shortest path if it is the destination). That is, $\forall_{ij}$ i,j ∈ V and i≠v' and j≠v', determine v' ∉ shortest_path(i, j). Note, if your algorithm is not the most efficient, but is still correct, you will still get partial credit. What is the worst-case runtime of your algorithm in terms of |E| and |V|?

37. Splay Trees

a. Insert the nodes 1, 2, 3, 4, 5 into a splay tree. Draw the resulting tree.

b. Find node 3. Draw the resulting tree.

c. Delete node 2. Draw the resulting tree.