Final Exam Review

True-false questions

(1) **T F** The **best case** running time for INSERTION SORT to sort an n element array is $O(n)$.

(2) **T F** By the master theorem, the solution to the recurrence $T(n) = 3T(n/3) + \log n$ is $T(n) = \Theta(n \log n)$.

(3) **T F** Given *any* binary tree, we can print its elements in sorted order in $O(n)$ time by performing an inorder tree walk.

(4) **T F** Computing the median of n elements takes $\Omega(n \log n)$ time for any algorithm working in the comparison-based model.

(5) **T F** Every binary search tree on n nodes has height $O(\log n)$.

(6) **T F** Given a graph $G = (V, E)$ with cost on edges and a set $S \subseteq V$, let (u, v) be an edge such that (u, v) is the minimum cost edge between any vertex in S and any vertex in $V - S$. Then, the minimum spanning tree of G must include the edge (u, v). (You may assume the costs on all edges are distinct, if needed.)

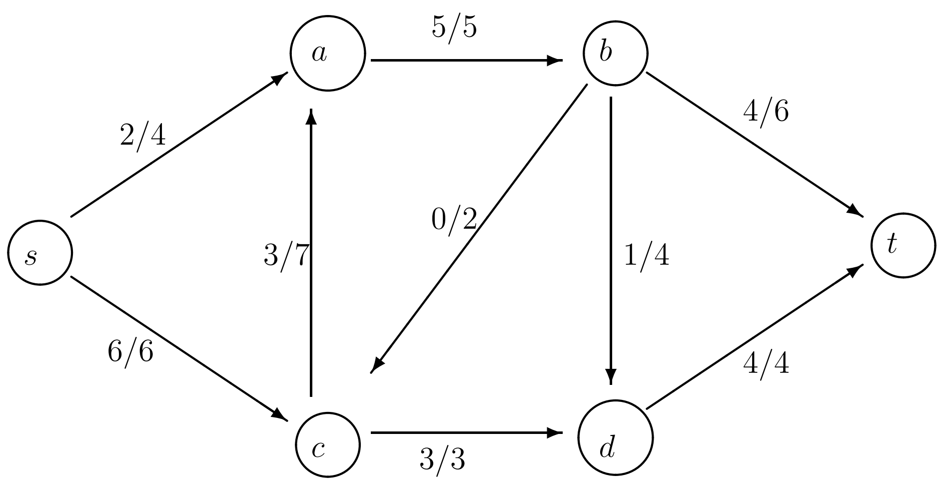
(7) **T F** Computing a^b takes exponential time in n, for n-bit integers a and b.

(8) **T F** There exists a data structure to maintain a dynamic set with operations Insert(x,S), Delete(x,S), and Member?(x,S) that has an expected running time of $O(1)$ per operation.

(9) **T F** The total amortized cost of a sequence of n operations (i.e., the sum over all operations, of the amortized cost per operation) gives a lower bound on the total actual cost of the sequence.

(10) **T F** The figure below describes a flow assignment in a flow network. The notation a/b describes a units of flow in an edge of capacity b.

True or False: The following flow is a maximal flow.



(11) **T F** Let $G = (V, E)$ be a weighted graph and let M be a minimum spanning tree of G. The path in M between any pair of vertices v_1 and v_2 must be a shortest path in G.

(12) **T F** $n \lg n = O(n^2)$

(13) **T F** Let P be a shortest path from some vertex s to some other vertex t in a graph. If the weight of each edge in the graph is increased by one, P remains a shortest path from s to t.

(14) **T F** Suppose we are given n intervals (l_i, u_i) for $i = 1, \cdots, n$ and we would like to find a set S of non-overlapping intervals maximizing $\sum_{i \in S} w_i$, where w_i represents the weight of interval (l_i, u_i). Consider the following greedy algorithm. Select (in the set S) the interval, say (l_i, u_i) of maximum weight w_i, remove all intervals that overlap with (l_i, u_i) and repeat. This algorithm always provides an optimum solution to this interval selection problem.

(15) **T F** Given a set of n elements, one can output in sorted order the k elements following the median in sorted order in time $O(n + k \log k)$.

(16) **T F** Consider a graph $G = (V, E)$ with a weight $w_e > 0$ defined for every edge $e \in E$. If a spanning tree T minimizes $\sum_{e \in T} w_e$ then it also minimizes $\sum_{e \in E} w_e^2$, and vice versa.

(17) **T F** The breadth first search algorithm makes use of a stack.

(18) **T F** In the worst case, merge sort runs in $O(n^2)$ time.

(19) **T F** A heap can be constructed from an unordered array of numbers in linear worst-case time.

(20) **T F** No adversary can elicit the $\Theta(n^2)$ worst-case running time of randomized quicksort.

(21) **T F** Radix sort requires an “in place” auxiliary sort in order to work properly.

(22) **T F** A longest path in a dag $G = (V, E)$ can be found in $O(V + E)$ time.

(23) **T F** The Bellman-Ford algorithm is not suitable if the input graph has negative-weight edges.

(24) **T F** Memoization is the basis for a top-down alternative to the usual bottom-up version of dynamic programming.

(25) **T F** Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, the shortest path between every pair of vertices $u, v \in V$ can be determined in $O(V^3)$ worst-case time.

(26) **T F** For hashing an item into a hash table in which collisions are resolved by chaining, the worst-case time is proportional to the load factor of the table.

(27) **T F** A red-black tree on 128 keys must have at least 1 red node.

(28) **T F** The move-to-front heuristic for self-organizing lists runs no more than a constant factor slower than any other reorganization strategy.

(29) **T F** Depth-first search of a graph is asymptotically faster than breadth-first search.

(30) **T F** Dijkstra's algorithm is an example of a greedy algorithm.

(31) **T F** Fibonacci heaps can be used to make Dijkstra's algorithm run in $O(E + V \lg V)$ time on a graph $G = (V, E)$.

(32) **T F** The Floyd-Warshall algorithm solves the all-pairs shortest-paths problem using dynamic programming.

(33) **T F** A maximum matching in a bipartite graph can be found using a maximum-flow algorithm.

(34) **T F** For any directed acyclic graph, there is only one topological ordering of the vertices.

(35) **T F** If some of the edge weights in a graph are negative, the shortest path from s to t can be obtained using Dijkstra's algorithm by first adding a large constant C to each edge weight, where C is chosen large enough that every resulting edge weight will be nonnegative.

(36) **T F** If all edge capacities in a graph are integer multiples of 5 then the maximum flow value is a multiple of 5.

(37) **T F** For any graph G with edge capacities and vertices s and t, there always exists an edge such that increasing the capacity on that edge will increase the maximum flow from s to t in G. (Assume that there is at least one path in the graph from s to t.)

(38) **T F** Heapsort, quicksort, and mergesort are all asymmptotically optimal, stable comparison-based sort algorithms.

(39) **T F** If each operation on a data structure runs in O(1) amortized time, then n consecutive operations run in $O(n)$ time in the worst case.

(40) **T F** A graph algorithm with $\Theta(E \log V)$ running time is asymptotically better than an algorithm with a $\Theta(E \log E)$ running time for a connected, undirected graph $G(V, E)$.

(41) **T F** In $O(V + E)$ time a matching in a bipartite graph $G = (V, E)$ can be tested to determine if it is maximum.

(42) **T F** n integers each of value less than n^{100} can be sorted in linear time.

(43) **T F** For any network and any maximal flow on this network there always exists an edge such that increasing the capacity on that edge will increase the network's maximal flow.

(44) **T F** If the depth-first search of a graph G yields no back edges, then the graph G is acyclic.

(45) **T F** Insertion in a binary search tree is "commutative". That is, inserting x and then y into a binary search tree leaves the same tree as inserting y and then x.

(46) **T F** A heap with n elements can be converted into a binary search tree in $O(n)$ time.