Practice Quiz 2

Problem 1. It can be shown that in any minimum spanning tree (of a connected, weighted graph), if we remove an edge (u, v), then the two remaining trees are each MSTs on their respective sets of nodes, and the edge (u, v) is a least-weight edge crossing between those two sets.

These facts inspire Professors Indor and Tidyk to suggest the following algorithm for finding an MST on a graph $G = (V, E)$: split the nodes arbitrarily into two (nearly) equal-sized sets, and recursively find MSTs on those sets. Then connect the two trees with a least-cost edge (which is found by iterating over E).

Would you want to use this algorithm? Why or why not?

Problem 2. True or False, and Justify

Circle **T** or **F** for each of the following statements to indicate whether the statement is true or false, respectively. If the statement is correct, briefly state why. If the statement is wrong, explain why. Your justification is worth more points than your true-or-false designation.

T F To determine if two binary search trees are identical trees, one could perform an inorder tree walk on both and compare the output lists.

T F Constructing a binary search tree on n elements takes $\Omega(n \log n)$ time in the worst case (in the comparison model).

T F A greedy algorithm for a problem can never give an optimal solution on all inputs.

T F Suppose we have computed a minimum spanning tree of a graph and its weight. If we make a new graph by doubling the weight of every edge in the original graph, we still need $\Omega(E)$ time to compute the cost of the MST of the new graph.

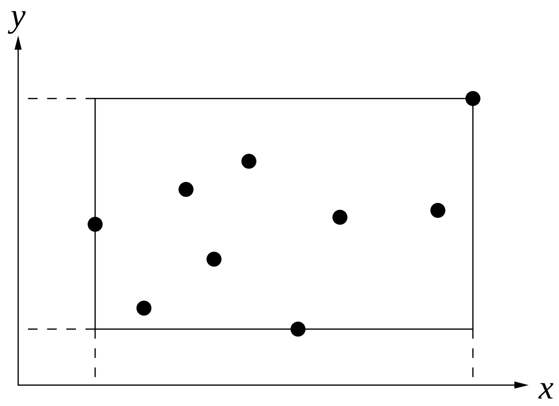
T F 2-3-4 trees are a special case of B-trees.

T F A maximum spanning tree (i.e., a spanning tree that maximizes the sum of the weights) can be constructed in $O(E \log V)$ time.

T F Let $G = (V, E)$ be a connected, undirected graph with edge-weight function $w : E$ to reals. Let $u \in V$ be an arbitrary vertex, and let $(u, v) \in E$ be the least-weight edge incident on u; that is, $w(u, v) = \min \{w(u, v') : (u, v') \in E\}$. (u, v) belongs to some minimum spanning tree of G.

Problem 3. Bounding Boxes

An important notion in computer graphics is the **bounding box** of a set of objects, which is the smallest rectangle (where each side is parallel to the x- or y-axis) such that all the objects are contained in the rectangle (including its border). Note that a bounding box can be described by the two x-coordinates of its two vertical edges, and the two y-coordinates of its two horizontal edges. For example, the figure below shows a collection of points and their bounding box, which can be represented by the indicated pairs of x- and y-coordinates.



In this problem we are interested in designing a dynamic set of points in the plane which supports the standard INSERT and SEARCH operations. It must also support an operation B-BOX(x_0, x_1), which computes the bounding box of *only* those points in the set whose x-coordinates are between x_0 and x_1 (inclusive). That is, it should return a pair (y_0, y_1) containing the y-coordinates of the bounding box's horizontal edges.

(a) What data structure would you augment to implement this data structure, and what auxiliary data would you keep? Explain why the running times of INSERT and SEARCH are $O(\log n)$, where n is the number of points in the set.

(b) Explain how to implement B-BOX(x_0, x_1) so that it runs in $O(\log n)$ time. You may assume that x_0 and x_1 are the x-coordinates of some points in the set.

Problem 4. Test-Taking Strategies

Consider (if you haven't already!) a quiz with n questions. For each $i = 1, \ldots, n$, question i has integral point value $v_i > 0$ and requires $m_i > 0$ minutes to solve. Suppose further that no partial credit is awarded (unlike this quiz).

Your goal is to come up with an algorithm which, given v_1, v_2, $\ldots$, v_n, m_1, m_2, $\ldots$, m_n, and V, computes the minimum number of minutes required to earn at least V points on the quiz. For example, you might use this algorithm to determine how quickly you can get an A on the quiz.

(a) Let $M(i, v)$ denote the minimum number of minutes needed to earn v points when you are restricted to selecting from questions 1 through i. Give a recurrence expression for $M(i, v)$.

We shall do the base cases for you: for all i, and $v \leq 0$, $M(i, v) = 0$; for $v > 0$, $M(0, v) = \infty$.

(b) Give pseudocode for an $O(nV)$-time dynamic programming algorithm to compute the minimum number of minutes required to earn V points on the quiz.

(c) Explain how to extend your solution from the previous part to output a list S of the questions to solve, such that $V \leq \sum_{i \in S} v_i$ and $\sum_{i \in S} m_i$ is minimized.

(d) Suppose partial credit is given, so that the number of points you receive on a question is proportional to the number of minutes you spend working on it. That is, you earn v_i/m_i points per minute on question i (up to a total of v_i points), and you can work for fractions of minutes. Give an $O(n \log n)$-time algorithm to determine which questions to solve (and how much time to devote to them) in order to receive V points the fastest.