Final Exam


1. (20 points) A spreadsheet may have a large number of simple arithmetic statements. For each such statement, we have a dependency list consisting of all the variables on the right side of the statement. For example, here we have some assignment statements and the corresponding dependency lists:

   \[
   \begin{align*}
   x_1 &= 2 & \rightarrow & x_1: \text{NULL} \\
   x_2 &= 2x_1 & \rightarrow & x_2: (x_1) \\
   x_3 &= x_1 + x_2 & \rightarrow & x_3: (x_1, x_2) \\
   x_4 &= 21 & \rightarrow & x_4: \text{NULL} \\
   x_5 &= x_3x_4 & \rightarrow & x_5: (x_3, x_4) \\
   x_6 &= x_4x_5 & \rightarrow & x_6: (x_4, x_5) \\
   x_7 &= x_1+x_2x_3 & \rightarrow & x_7: (x_1, x_2, x_3)
   \end{align*}
   \]

For any variable, \( v \), the spreadsheet creates a dependency list, \( v.dl \). We can loop over the variables in the dependency list using the code: \( \text{for ( w in v.dl) } \). The update set for \( v \) is the set of other variables whose value might change if the definition of \( v \) changes. For example, the update list for \( x_2 \) is \( (x_3, x_5, x_6, x_7) \). A circular dependence is a situation in which a variable appears in its own update list. Give pseudocode for an algorithm to determine whether variables \( u \) and \( v \) have disjoint update lists. You can write a method \( \text{upDis}(u, v) \) that has access to all the dependency list information and returns true if the update lists for \( u \) and \( v \) are disjoint. The work for your algorithm should be \( O(V + D) \) where \( V \) is the number of variables in all and \( D \) is the number of dependencies. Assume there are no circular dependencies.

2. (10 points) We have an array of \( n \) numbers.

   A. Give pseudocode for an algorithm to determine whether the numbers are in min heap order.

   B. Give pseudocode for an algorithm that deletes the value \( a[k] \) and restores the remaining \( n - 1 \) numbers in the array to min heap order.

3. (20 points) We have an array of \( n \) numbers not in order. We are given a number, \( W \), and we want the largest \( k \) so that the smallest \( k \) numbers in the array add up to less than \( W \). Give pseudocode to do this using a variant of randomized quick select. Show that the expected time for your algorithm is \( O(n) \).
4. (15 points) For each of the tasks listed below, name the algorithm and/or data structure you recommend and give your reasoning. You need not describe the algorithm in detail. Just explain why it is the best choice.

A. We have a connected undirected graph. We want to build a subgraph that is a spanning tree so that we may rapidly reach any given vertex starting from a “root” vertex r.

B. We have a million numbers with 50 digits each. We want to find whether the list has pair of numbers within 512 of each other.

C. We want to maintain a list of records indexed by social security number (9 digits). We need to find a record if we know the number, add a new record, or delete a record. Currently there are about 100,000 records.

5. (10 points) A B-tree with parameter \( t = 64 \) has 256 million elements in it. Give the minimum and maximum possible height of the tree. The parameter \( t \) is the minimum branching factor for a node. You may use the approximation 1 million \( \approx 2^{20} \). If you do rough calculations you may be off by one either way. This is fine.

6. (10 points) In each case, give a simple order of magnitude expression for the quantity and explain your answer. For example \( a(n) = 1 + 2 + \cdots + n = \Theta(n^2) \) because \( \int_1^n xdx = n^2/2 \), or because \( a(n) = n(n-1)/2 = \Theta(n) \), or because \( a(n) < n \cdot n \) and \( a(n) > n/2 + (1 + n/2) + \cdots + n > n/2 \cdot n/2 \). (You need only give one reason, but there may be different correct answers).

A. \( W(n) = (5 \cdot n + \lg(n)) \cdot \lg(n^2 + n^4) \).

B. \( T(n) = 3 \cdot T\left(\left\lceil \frac{n}{4} \right\rceil \right) + n^2 \), \( T(1) = 1 \). Assume \( T(n) \) is an increasing function of \( n \).

7. (20 points) In each case, say whether the statement is true or false and briefly explain your answer.

A. If the work for an algorithm is \( W(n) = O(n^2) \), then for all sufficiently large \( n \) we have \( W(2n) > W(n) \).

B. If we put \( n > 100,000 \) items into a hash table of size \( 200 \cdot n \), then there probably will be no collisions.

C. Performing a single rotation in a binary search tree or a B-tree or a red black tree takes \( O(1) \) work.

D. We have a connected directed graph with edge weights. Computer scientists know a simple polynomial time algorithm to find a path of minimum total through the graph that visits every vertex.