CECS 528, LO11 Assessment, 5/8/2026, Dr. Ebert

Directions: Solve each of the following problems. Your solution to each problem should be handwritten on a **single sheet of paper (front and back)** to be turned in before or after the exam on Monday May 11th (MW Students) or Friday May 15th (Friday Students). Turn in **two sheets** one per problem. Friday students may also upload their solutions on or before Saturday, May 17th. Plagiarized papers will result in a failing course grade. **A minimum of 40 points must be earned to qualify for passing LO11**. **Please show all work!**

Problems

1. The analysis of the randomization algorithms for `Find Statistic` and `Quicksort` demonstrate how both the Master Theorem and Substitution Method remain useful as analysis tools. Similarly, we may sometimes use dynamic-programming recurrences to analyze the performance of a randomized algorithm for a problem that would otherwise be solved with dynamic programming. As an example, recall the `Optimal Binary Search Tree (OBST)` problem. Recall that an instance of `OBST` is a sequence of weights $(w_1, \ldots, w_n)$, where w_i is the weight associated with key k_i, $1 \leq i \leq n$. Perhaps the simplest randomized algorithm for this problem is to randomly (via a uniform distribution) select the root k of the BST followed by recursively calling the algorithm on input sequence $(w_1, \ldots, w_{k-1})$, followed by recursively calling the algorithm on input sequence $(w_{k+1}, \ldots, w_n)$.

   (a) Let ewac(i, j) denote the expected weighted access cost of the subtree that holds keys i to j and formed by the randomized algorithm described above. Provide a recurrence for computing ewac(i, j), including all necessary base cases. Hint: your recurrence should look very similar to the `wac` recurrence provided on page 14 of the Dynamic Programming lecture. (15 pts)

   (b) Use the recurrence from part a to compute the expected weighted access cost that results when applying the randomized algorithm to the instance of `OBST` from Example 6 of the Dynamic Programming lecture. Provide the matrix of subproblem expectations. Round all expectations to the nearest integer. Notice how keeping track of "k" values makes no sense for this matrix (why?). (10 pts)

   (c) The advantage of the original DP algorithm is that it gives the optimal solution. The advantage of the randomized algorithm is that it requires O(n) steps. Therefore, to obtain the "best of both worlds" we can combine both algorithms by using the original DP solution for a certain number d of diagonals, where $1 \leq d \leq n$. Then we apply the randomized algorithm to the remaining $n - d$ diagonals. Compute the value of d for such a hybrid algorithm assuming that i) i steps are required by the DP algorithm to compute a single matrix entry in diagonal i, for each $i \in \{1, \ldots, n\}$, and ii) we are limiting the total number of DP steps to no more than $n^{1+\epsilon}$, where $\epsilon \geq 0$ is some constant. Hint: for full credit you must provide an exact (but perhaps unsimplified) algebraic formula for the total number steps used by the DP algorithm up to diagonal d. However, when solving for d, you may use a big-O approximation of the formula to determine the value of d. For example, if q

is a fourth-degree polynomial in r and we need to equate it to n^5, then we may estimate q as r^4 and equate it to n^5 to obtain $r = n^{\frac{5}{4}}$. (15 pts)

2. Keeping with the theme of dynamic programming, consider a graph consisting of two four-cliques, $C_1 = \{a, b, c, d\}$ and $C_2 = \{e, f, g, h\}$ both being connected via the single "bridge" edge (a, e). Determine the exact probability that Karger's algorithm will correctly return $\{(a, e)\}$ as the minimum cut. Do this by creating a 5×5 matrix P, where P_{ij} denotes the probability that Karger's algorithm will return $\{(a, e)\}$ provided the algorithm begins with a graph that has an (i, j)-configuration, $1 \leq i, j \leq 5$. For example,

   (a) A $(1, j)$-configuration occurs when C_1 has been reduced to the compound vertex *abcd*.

   (b) A $(2, j)$-configuration occurs when C_1 has been reduced to two compound nodes with three edges between them.

   (c) A $(3, j)$-configuration occurs when C_1 has been reduced to two compound nodes with four edges between them.

   (d) A $(4, j)$-configuration occurs when C_1 has been reduced to three vertices: two orginal vertices and one compound vertex. In other words, it is a configuration that occurs after two vertices of C_1 become merged.

   (e) A $(5, j)$-configuration occurs when C_1 is in its original form.

   Configuration types $(i, 1), \ldots, (i, 5)$ are defined analogously. Thus, i refers to the current deformed state of C_1, while j referse to the current deformed state of C_2. Round each probability to four decimal places. Hint: please note that P_{ij} is a symmetric matrix (why?) (20 pts)