This is the usual open-everything, but no outside help take-home test. Check "Class News", where I will post “frequently asked questions” about the test. Please submit your solutions typed, with this page on top and each problem on a separate sheet.

1. Amortized Complexity of Union – Find
Here we assume that we have a partition of \( n \) elements into disjoint sets, where each set is represented by a tree in which non-root nodes have pointers to their parent nodes.

[5] (i) Prove by induction that subtrees rooted in the children of the root of a binomial tree of order \( k \) are binomial trees of all smaller orders.

[5] (ii) Characterize the structure of subtrees rooted in nodes on the longest path from a root to a leaf of a binomial tree of order \( k \).

[10] (iii) Assume Find with path compression and Union with an arbitrary link of roots. A series of consecutive Union operations involving \( n/2 = 2^k \) elements may result in a binomial tree \( T \) of order \( k \). Set \( i = n/2 \) and repeat the following \( n/2 \) times:

\[
\{ \text{perform Union of } T \text{ and the set } \{i+1\} \text{ resulting in tree } T' \text{ rooted at } i+1; \\
\text{perform Find on the deepest leaf of } T'; \text{ this results in a new tree } T; \text{ } i++ \}
\]

What is the complexity of the above algorithm and what does it tell you about lower bounds on the amortized complexity of this particular implementation of Union-Find?

2. Greedy Algorithms vs. DP
[10] Both the prefix-free code problem (“Huffman’s code”) and the OBST problem (with data in the leaves) strive to minimize the weighted external path length in a tree. Why does a greedy algorithm work for the former but not for the latter?
3. Dynamic Programming

[10] (i) D.E. Knuth observed that the optimal root \( r(1, n) \) of an OBST that includes keys 1, \ldots, \( n \) lies between the optimal roots of the OBST for keys 1, \ldots, \( n-1 \) and the OBST for keys 2, \ldots, \( n \): \( r(1, n-1) \leq r(1, n) \leq r(2, n) \). What consequence, if any, does this observation have for the complexity of the DP construction algorithm?

[10] (ii) Consider an \( n \)-gon on the plane (each of the \( n \) vertices is given by a pair of coordinates) that is convex (the straight line joining any two interior points does not intersect its sides). A triangulation of the \( n \)-gon includes \( n-3 \) diagonals that divide its interior into \( n-2 \) triangular regions. Design an efficient algorithm finding a triangulation of the minimum total weight, where the weight of a triangle is the total length of its sides. (As a warm-up, solve the problem when the weight of a triangle is defined as its area.)

4. Polynomial-time reductions and NP-completeness

[5] (i) Prove that the relation of polynomial-time reduction between decision problems \( \leq_p \) is transitive and reflexive. Is it symmetric?

[5] (ii) Is a pseudopolynomial-time reduction (polynomial as a function of the maximum value of a number in a problem instance) sufficient to prove \( NP \)-hardness? Why?

[5] (iii) Is a polynomial-time non-deterministic reduction sufficient to prove \( NP \)-hardness?

[7] (iv) Assume that there is a polynomial time algorithm CLQ to solve the Clique decision problem:

**Instance:** A graph \( G \) and an integer \( K \)

**Question:** Does \( G \) have a completely connected set of \( K \) vertices?

Show how to use CLQ to find a maximum size clique of a given graph in polynomial time.

[8] (v) Show that Clique is NP-complete by polynomial time reduction from VertexCover: Is there a set of \( W \) vertices covering all edges of the given graph \( H \)?