Introduction to Algorithms	**Solution Set 1**
CS 4820, Spring 2018	

(1) *(5 points)*
For any positive integer n, let L_n denote an L-shaped region in the plane obtained by starting with a square of side length 2^n and deleting its upper right quadrant. For example, L_1 is the "L-shaped tromino tile" discussed in class on Wednesday. See Figure 1 for additional examples.

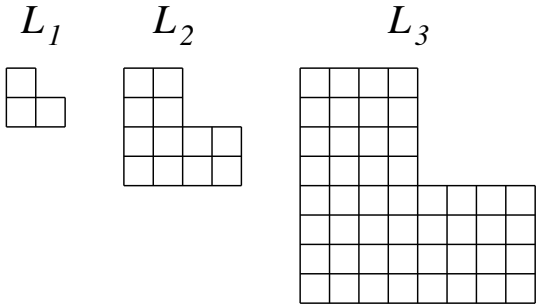

Figure 1: The regions L_1, L_2, L_3

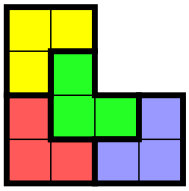

Figure 2: Tiling L_2 with copies of L_1

*Prove, **using mathematical induction,** that for every positive integer n it is possible to tile L_n using copies of L_1. In other words, you should prove that L_n can be partitioned into regions, each congruent to L_1. See Figure 2 for an example when $n = 2$.*

Try to make your proof as clear and precise as possible. You do not need to describe an algorithm to compute the tiling, nor to analyze its running time. You only need to prove that such a tiling exists.

Solution. The proof uses the following three lemmas.

Lemma 1. *If R, S are two regions in the plane such that R can be tiled using congruent copies of S, then any plane region congruent to R can also be tiled using congruent copies of S.*

Proof. If R' is a plane region congruent to R, then there exists a rigid motion of the plane, T, such that $T(R) = R'$. If $S_1, S_2, \ldots, S_m$ are congruent copies of S that constitute a tiling of R, then $T(S_1), T(S_2), \ldots, T(S_m)$ are congruent copies of S that constitute a tiling of R'. □

Lemma 2. *Let D denote the "doubling" transformation of the plane, i.e. the function which maps each point (x, y) to the point $(2x, 2y)$. If R is a plane region that can be tiled using congruent copies of S, then $D(R)$ can be tiled using congruent copies of $D(S)$.*

Proof. If $S_1, \ldots, S_m$ are congruent copies of S that constitute a tiling of R, then $D(S_1), \ldots, D(S_m)$ are congruent copies of $D(S)$ that constitute a tiling of $D(R)$. □

Lemma 3. *If Q, R, S are three plane regions such that Q can be tiled using congruent copies of R, and R can be tiled using congruent copies of S, then Q can be tiled using congruent copies of S.*

Proof. Let $R_1, \ldots, R_m$ denote congruent copies of R that constitute a tiling of Q. Since each R_i is congruent to R, Lemma 1 implies that each R_i can be tiled using congruent copies of S. Let $S_{i1}, \ldots, S_{in}$ denote the copies of S in this tiling of R_i. Then the tiles $\{S_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ constitute a tiling of Q by congruent copies of S. □

To prove that L_n can be tiled using copies of L_1, we use induction on n. In the base case $n = 1$, the tiling is trivial: L_1 itself is a copy of L_1. For the induction step, assume that L_{n-1} can be tiled by copies of L_1. Recall the doubling transformation D from Lemma 2, and note that $L_n = D(L_{n-1})$ and $L_2 = D(L_1)$. The lemma implies that L_n can be tiled using copies of L_2. Figure 2 on the problem set shows that L_2 can be tiled using copies of L_1. An application of Lemma 3 with $Q = L_n$, $R = L_2$, $S = L_1$ now implies that L_n can be tiled using copies of L_1, which completes the induction step.

Remarks. The purpose of this problem was to test that you know how to write proofs using induction. CS 2800 (or an equivalent course) is a prerequisite for CS 4820, so everyone in the class should know how to do this. If you struggled with this problem because you were having a hard time grasping how to properly state a proof by mathematical induction, you will need to work on strengthening that skill.

After reading the solution above, you may wonder whether we expect students' proofs to be as detailed as this. The answer is no. I've tried, above, to present a proof that gives the maximum reasonable amount of detail. A typical full-credit-worthy solution to this problem looks like the one-paragraph inductive proof that constitutes the final paragraph of the solution above. In other words, rather than stating and proving Lemmas 1-3, it is permissible to simply assume those facts without proof and treat them as obvious properties of the "R can be tiled by copies of S" relation.

A more problematic case arises when you attempt to use a "proof by picture" to substitute for words. It's great to use pictures to *reinforce the language* of the proof, but in my experience, when students in 4820 use pictures to *substitute for language*, their pictures leave room for ambiguity in interpretation, and ambiguity is to be avoided.

(2) *(10 points)*
Consider the following scenario: n students get flown out to the Bay Area for a day of interviews at a large technology company. The interviews are organized as follows. There are m time slots during the day, and n interviewers, where $m \geq n$. Each student s has a fixed schedule *which gives, for each of the n interviewers, the time slot in which s meets with that interviewer. This, in turn, defines a schedule for each interviewer i, giving the time slots in which i meets each student. The schedules have the property that*

- *each student sees each interviewer exactly once,*
- *no two students see the same interviewer in the same time slot, and*
- *no two interviewers see the same student in the same time slot.*

Now, the interviewers decide that a full day of interviews like this seems pretty tedious, so they come up with the following scheme. Each interviewer i will pick a distinct *student s. At the end of i's scheduled meeting with s, i will take s to one of the company's many in-house eateries, and they'll both blow off the entire rest of the day sipping espresso and rubbing elbows with celebrity chefs.*

Specifically, the plan is for each interviewer i, and his or her chosen student s, to truncate *their*

schedules at the time of their meeting; in other words, they will follow their original schedules up to the time slot of this meeting, and then they will cancel all their meetings for the entire rest of the day.

The crucial thing is, the interviewers want to plan this cooperatively so as to avoid the following bad situation*: some student s whose schedule has not yet been truncated (and so is still following his/her original schedule) shows up for an interview with an interviewer who's already left for the day.*

Give an efficient algorithm to arrange the coordinated departures of the interviewers and students so that this scheme works out and the bad situation *described above does not happen.*

Solution. The key to solving this problem is to reduce to the stable matching problem. Assume that the input data is in the following format: we are given a list of n^2 (student,interviewer) pairs arranged according to the time slots when those pairs are scheduled to meet, in increasing chronological order. By iterating forwards through this list in a single pass, we can construct n lists, one for each student, listing the interviewers the student is to meet with ordered according to their meeting times from earliest to latest. By again iterating through the input list, this time backwards, we can construct n lists, one for each interviewer, listing the students the interviewer is to meet with ordered according to their meeting times from latest to earliest. We treat these lists as the preference lists of students ("applicants") and interviewers ("employers") in an instance of the stable matching problem, and we solve this stable matching instance using the Gale-Shapley Algorithm. Then we schedule a coordinated departure of student s and interviewer i if and only if (s, i) is in the stable matching.

Constructing the input instance to the stable matching problem takes $O(n^2)$ time — two passes through a list of size n^2, performing $O(1)$ operations per list element in each pass. Running the Gale-Shapley Algorithm also taks $O(n^2)$ time, so the entire algorithm runs in $O(n^2)$.

Proving the correctness of the algorithm amounts to proving that it outputs a set of coordinated departures satisfying two properties. In proving these properties, we will use M to denote the set of (s, i) pairs in the stable matching that the algorithm computes.

1. **Each interviewer picks a distinct student.** Since M is a perfect matching, every interviewer belongs to exactly one element of M.

2. **The "bad situation" described in the problem does not happen.** When s and i are scheduled to meet in time slot t, either (s, i) belongs to M in which case they depart together (and i has not departed with another student earlier) or (s, i) does not belong to M. In the latter case, since M is a stable matching, either s is matched to a interviewer i' is strictly preferred to i in his/her ordering — implying that i' has already departed with s — or i is matched to a student s' that is strictly preferred to s in his/her ordering, implying that i has not yet departed with s'. Either way, it cannot be the case that s arrives for an interview with i to find that i has already left for the day.

Remarks. This solution exemplifies how to design and analyze a *reduction* from one problem to another. Reductions allow you to encapsulate the properties of one algorithm (in this case, the *correctness guarantee* and *worst-case running time guarantee* of the Gale-Shapley Proposal Algorithm) in a form that can be re-used, without proof, when analyzing another algorithm. This is much more convenient for the writer and reader than repeating all the steps from the proof of that algorithm.