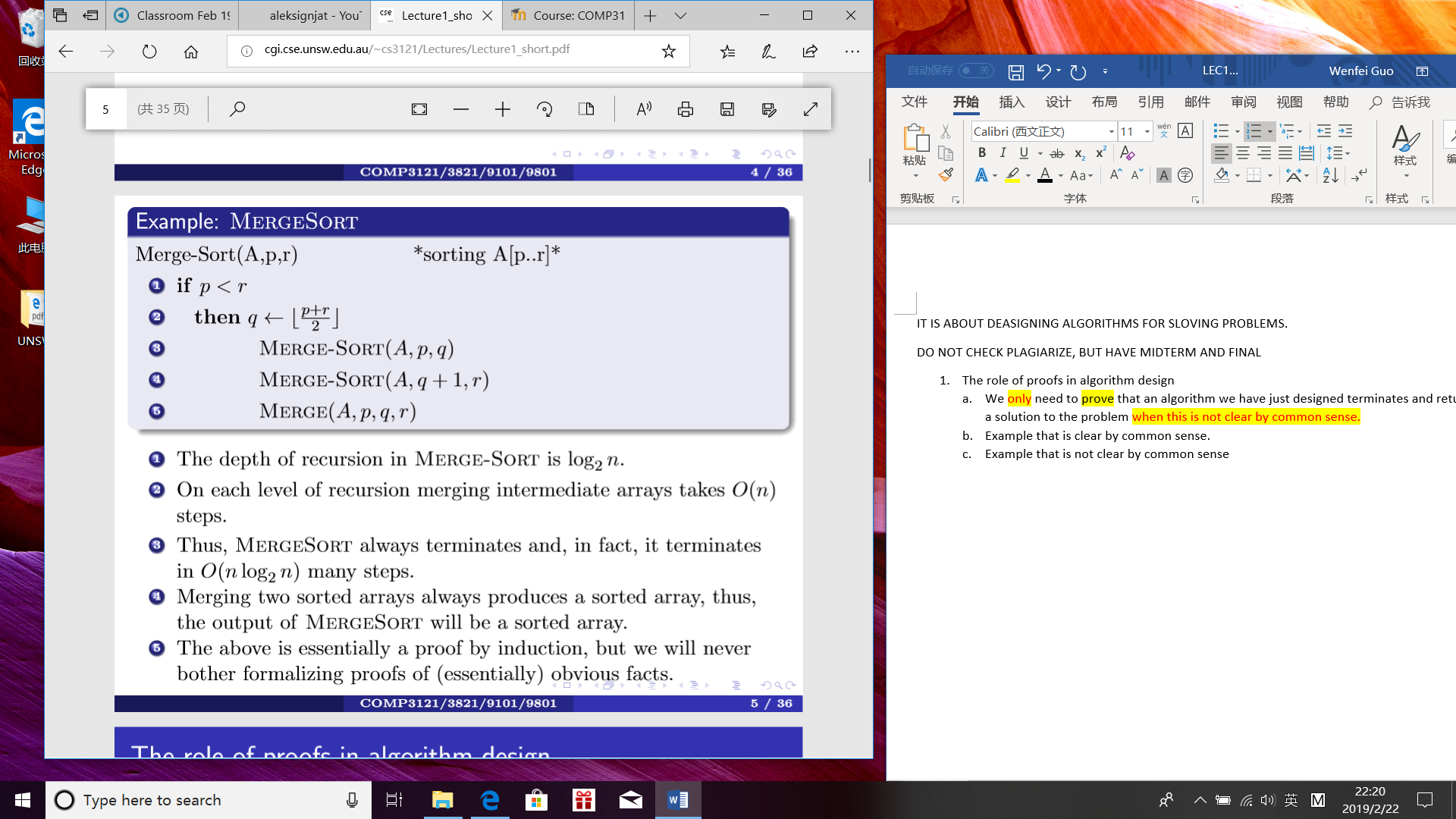
IT IS ABOUT DEASIGNING ALGORITHMS FOR SLOVING PROBLEMS.

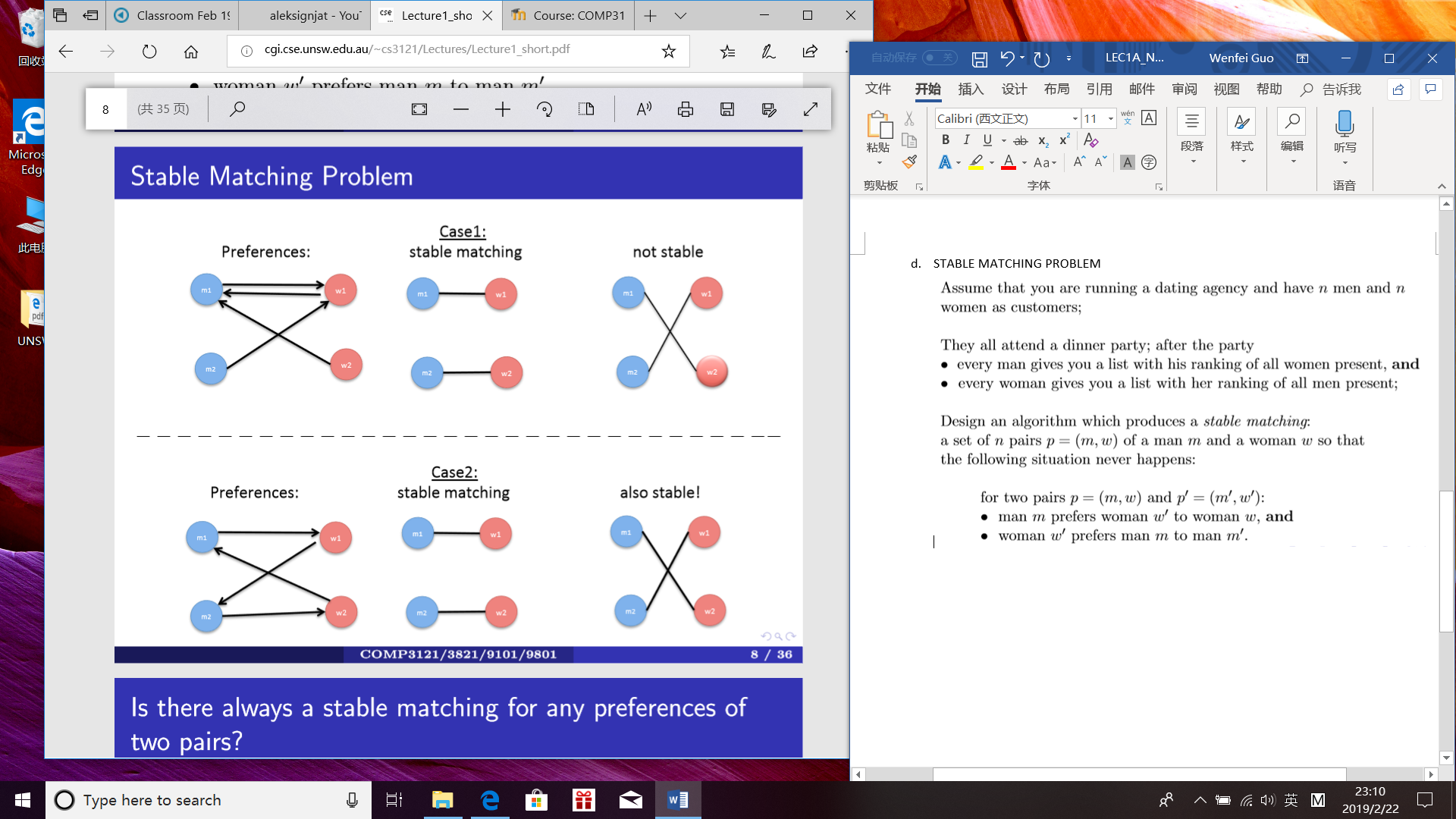
DO NOT CHECK PLAGIARIZE, BUT HAVE MIDTERM AND FINAL

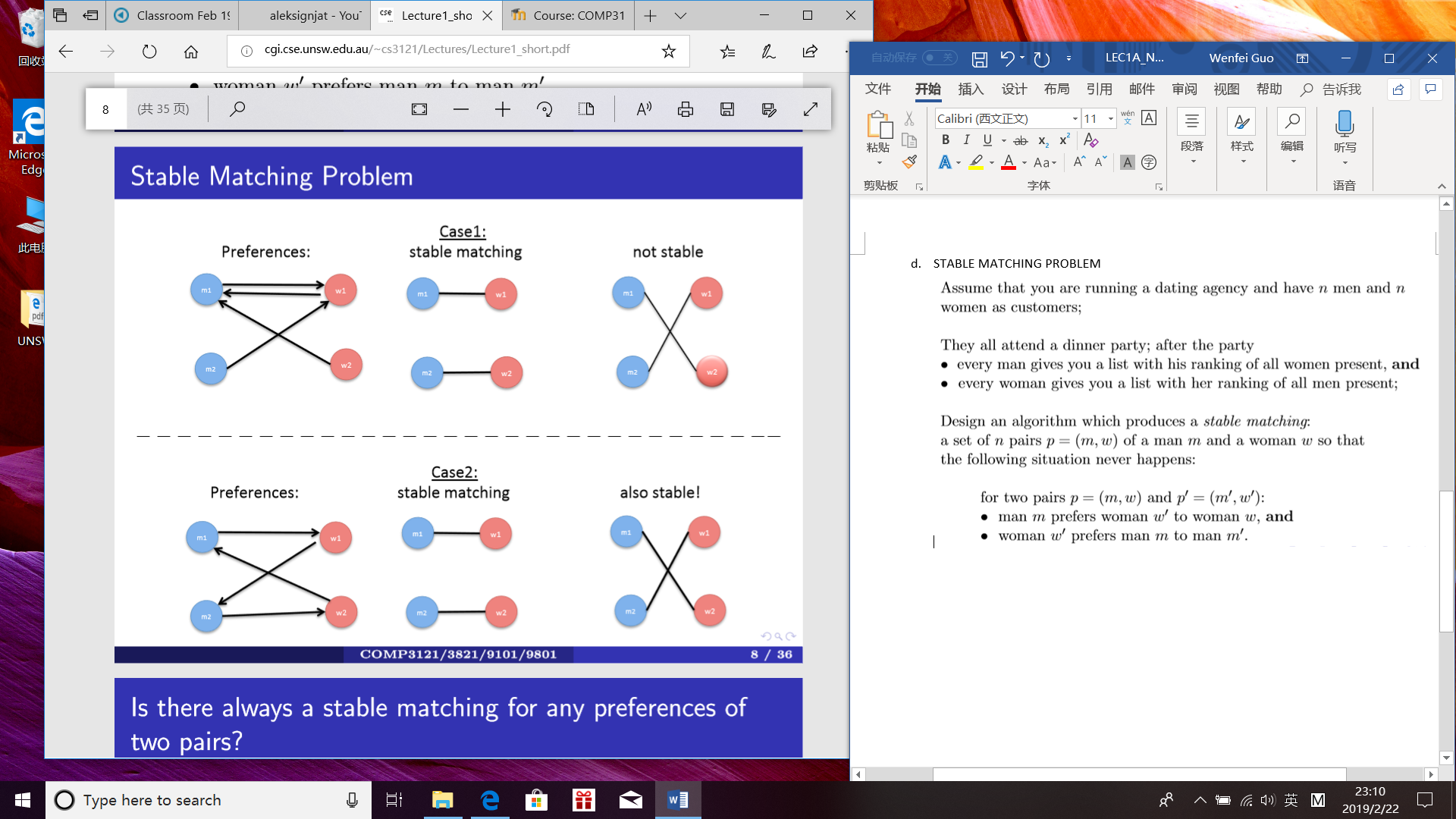
1. The role of proofs in algorithm design
2. We only need to prove that an algorithm we have just designed terminates and returns a solution to the problem when this is not clear by common sense.
3. Example that is clear by common sense.



1. Example that is not clear by common sense
2. However, sometimes it is NOT clear from a description of an algorithm that such an algorithm will not enter an inﬁnite loop and fail to terminate;
3. Sometimes it is not clear that an algorithm will not run in exponentially many steps (in the size of the input), which is essentially as bad as never terminating;
4. Sometimes it is not clear from a description of an algorithm why such an algorithm, after it terminates, produces a desired solution.
5. Proofs are needed for such circumstances; thus, proofs are NOT academic embellishments - in lots of cases they are the only way to know that the algorithm will always work!
6. For that reason we will NEVER prove the obvious (your CLRS textbook sometimes does just that, being too pedantic!) and will prove only what is genuinely nontrivial.
7. **However be very careful what you call trivial!!**
8. STABLE MATCHING PROBLEM







1. Stable Matching Problem: **Gale - Shapley algorithm**

(invented to pair newly graduated physicians with US hospitals for residency training)

1. Question 1: Is it true that for every possible collection of n lists of preferences provided by all men, and n lists of preferences provided by all women, a stable matching exists?

Answer: YES, but this is NOT obvious!

Question 2: Given n men and n women, how many ways are there to match them, i.e., just to form n couples? Answer: n! ≈ (n/e)n - more than exponentially many in n (e ≈ 2.71); Can we ﬁnd a stable matching in a reasonable amount of time??

Answer: YES, using the Gale - Shapley algorithm.

1. • Produces pairs in stages, with possible revisions;

• A man who has not been paired with a woman will be called free.

• Men will be proposing to women. Women will decide if they accept a proposal or not.

• Start with all men free;

While there exists a free man who has not proposed to all women pick such a free man m and have him propose to the highest ranking woman w on his list to whom he has not proposed yet;

If no one has proposed to w yet

she always accepts and a pair p = (m,w) is formed;

Else she is already in a pair p0 = (m0,w);

If m is higher on her preference list than m0 the pair p0 = (m0,w) is deleted;

m0 becomes a free man;

a new pair p = (m,w) is formed;

Else m is lower on her preference list than m0;

The proposal is rejected and m remains free.

1. Claim 1: **Algorithm terminates after ≤ n2 rounds of the While loop**

Proof:

• In every round of the While loop one man proposes to one woman;

• every man can propose to a woman at most once;

• thus, every man can make at most n proposals;

• there are n men, so in total they can make ≤ n2 proposals

Thus the While loop can be executed no more than n2 many times.

1. Claim 2: **Algorithm produces a matching,**

i.e., every man is eventually paired with a woman (and thus also every woman is paired to a man)

Proof:

• Assume that the while While loop has terminated, but m is still free.

• This means that m has already proposed to every woman.

• Thus, every woman is paired with a man, because a woman is not paired with anyone only if no one has made a proposal to her.

• But this would mean that n women are paired with all of n men so m cannot be free. Contradiction!

1. **Claim 3: The matching produced by the algorithm is stable**.

Proof: Note that during the While loop:

• a woman is paired with men of increasing ranks on her list;

• a man is paired with women of decreasing ranks on his list.

Assume now the opposite, that the matching is not stable;

thus, there are two pairs p = (m,w) and p0 = (m0,w0) such that:

m prefers w0 over w;

w0 prefers m over m0.

Since m prefers w0 over w, he must have proposed to w0 before proposing to w;

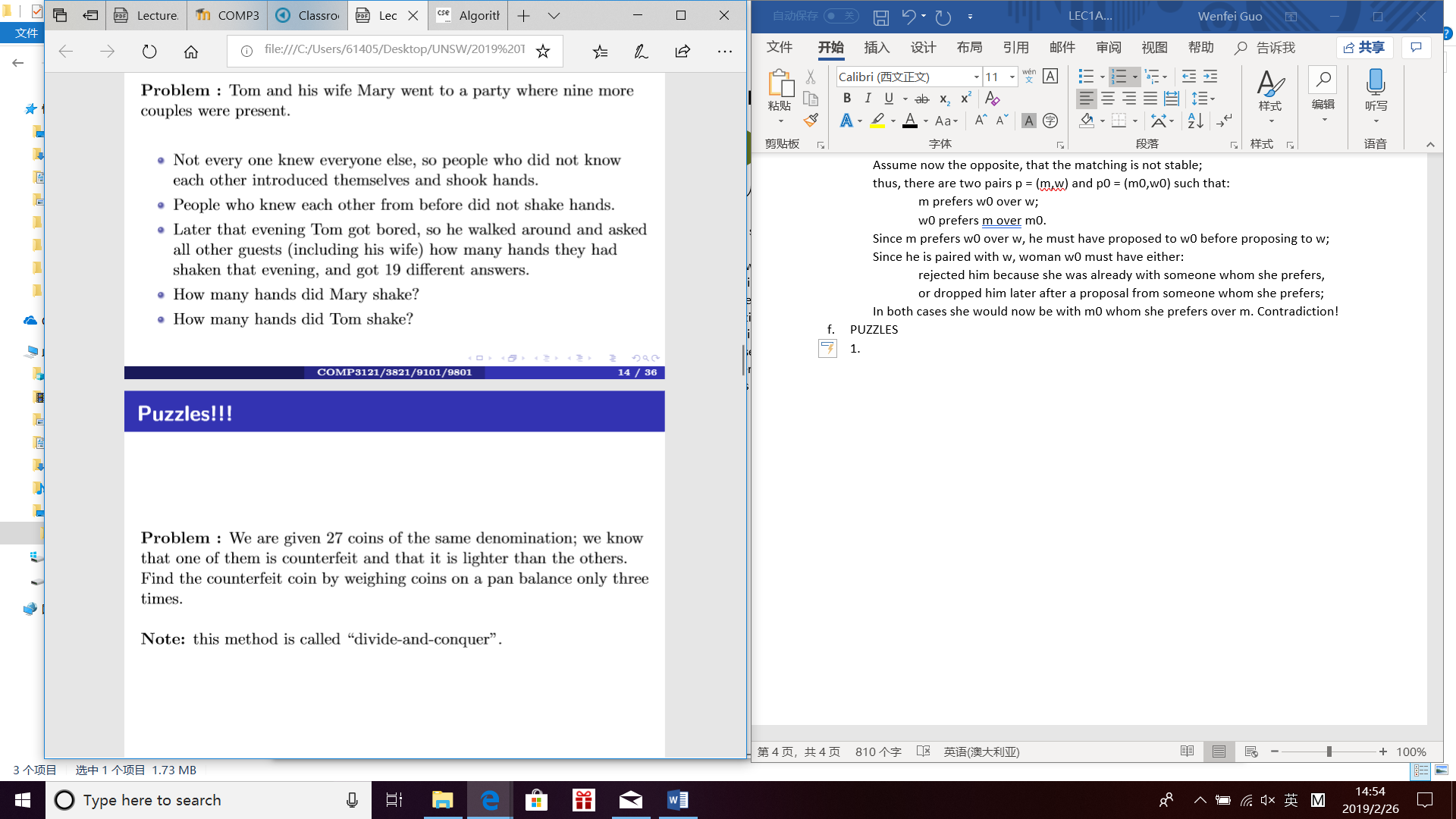
Since he is paired with w, woman w0 must have either:

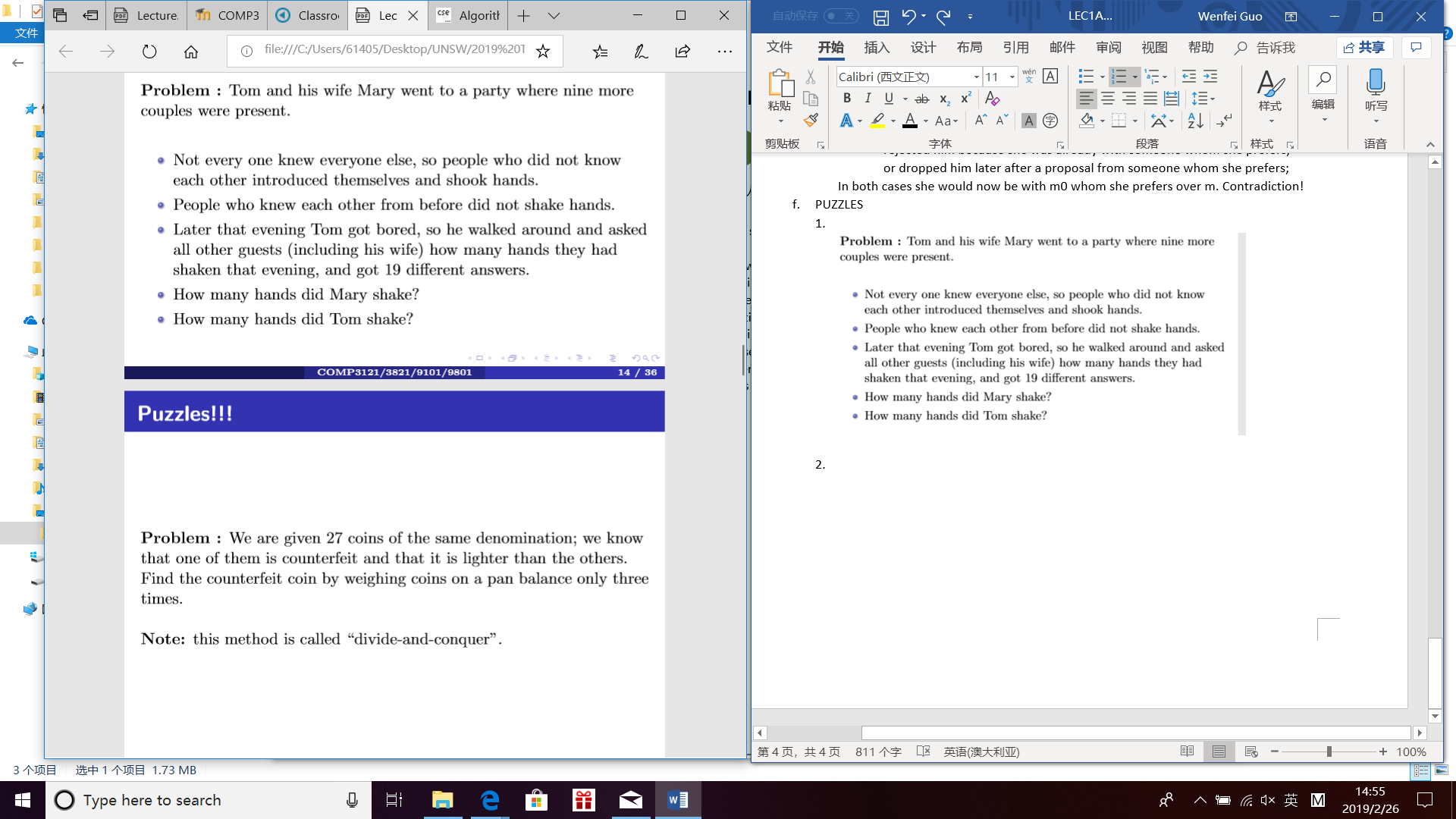
rejected him because she was already with someone whom she prefers,

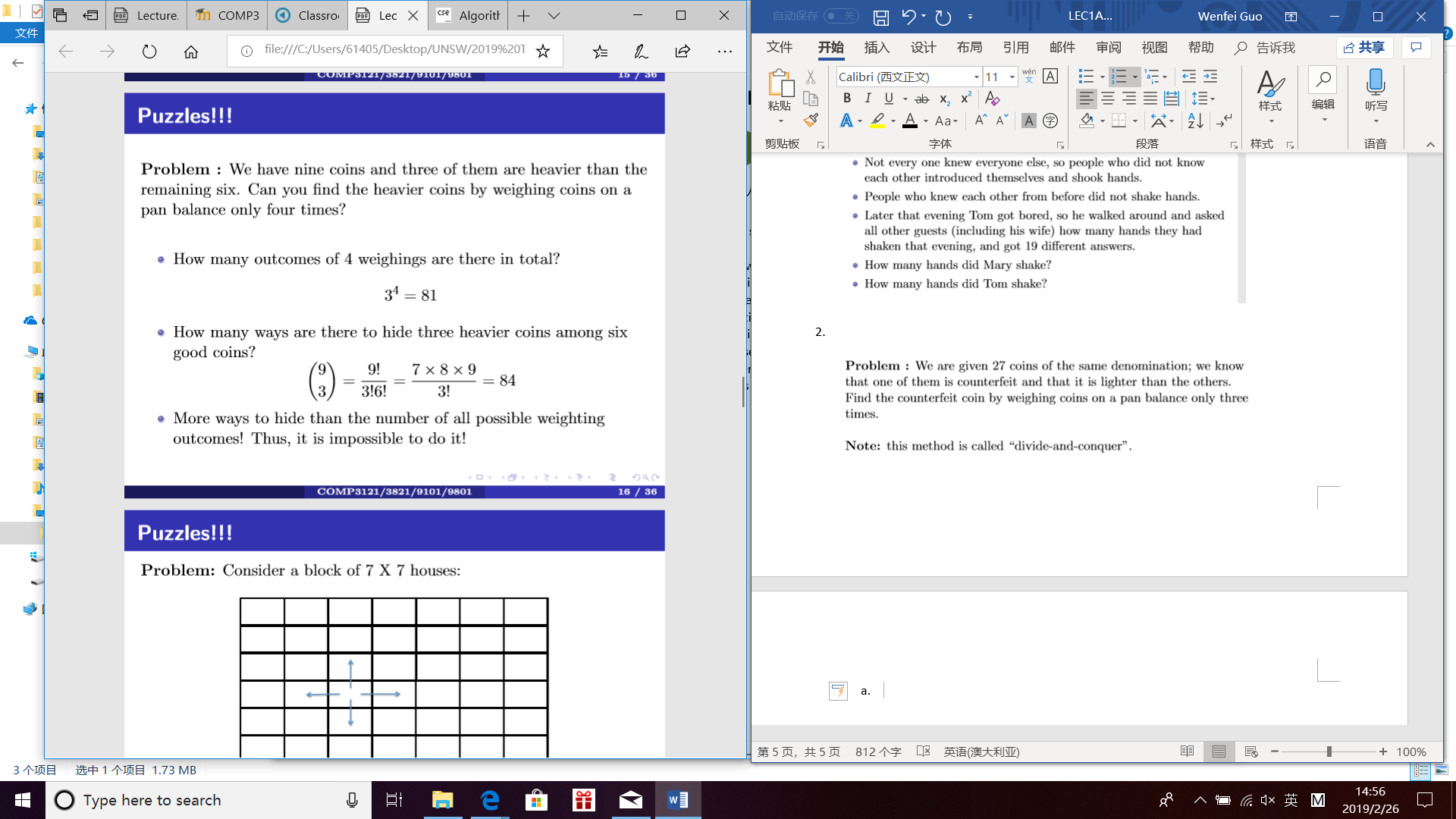
or dropped him later after a proposal from someone whom she prefers;

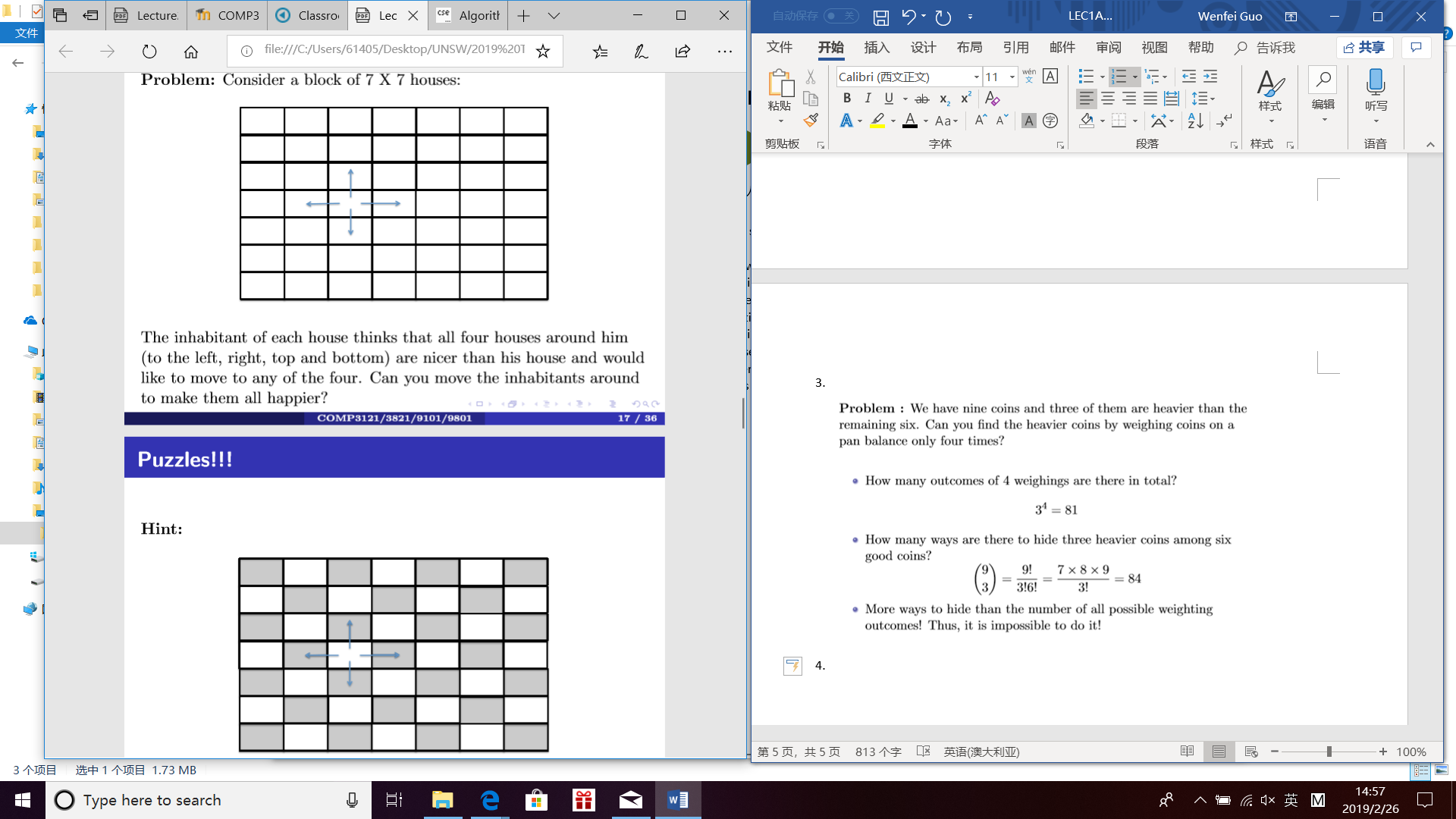
In both cases she would now be with m0 whom she prefers over m. Contradiction!

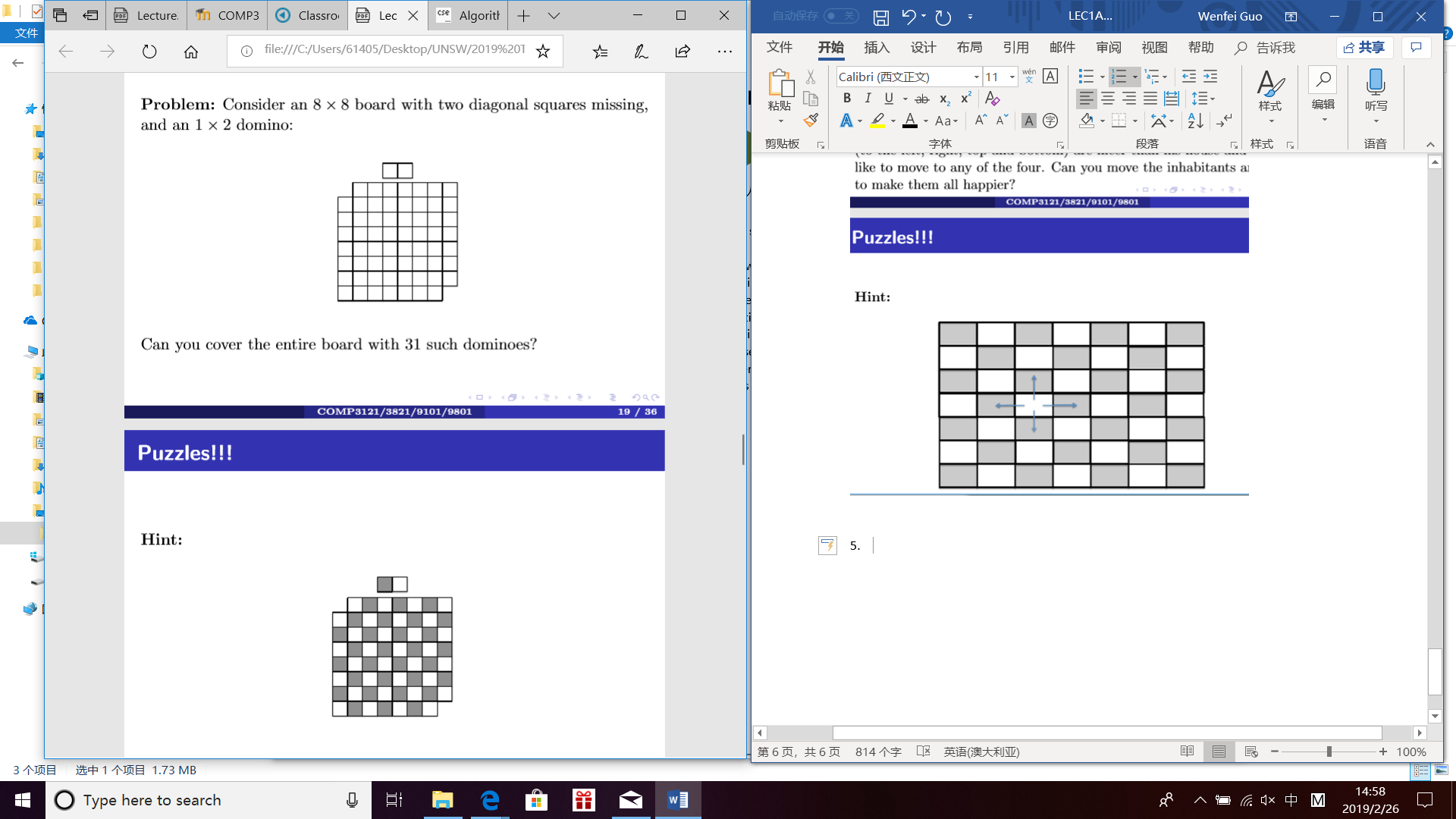
1. PUZZLES

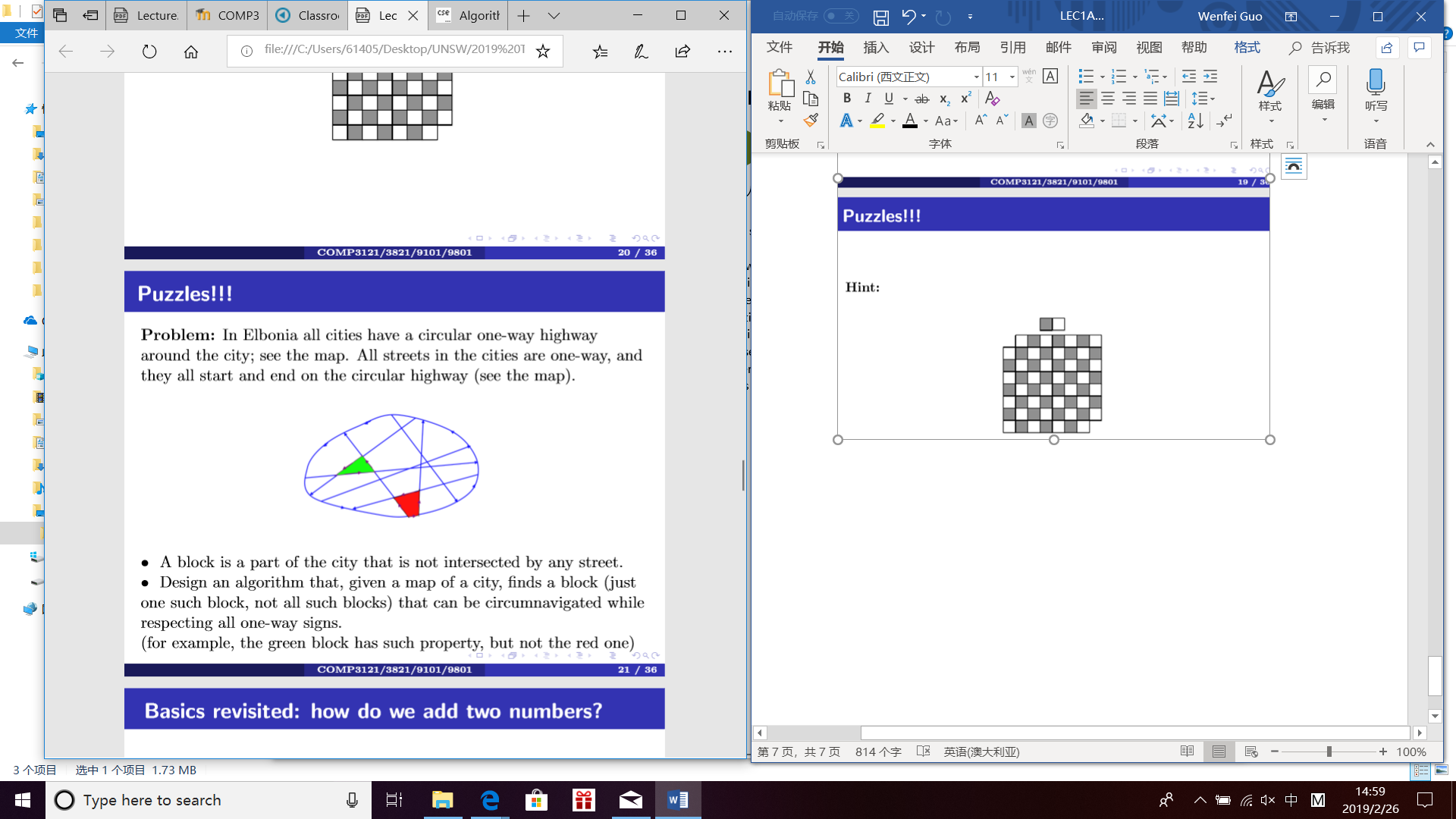












1. Basics revisited
2. how do we add two numbers?

O(n) many steps.

Can we do it faster than in linear time?

no, because we have to read every bit of the input

no asymptotically faster algorithm

1. how do we multiply two numbers?

O(n2).

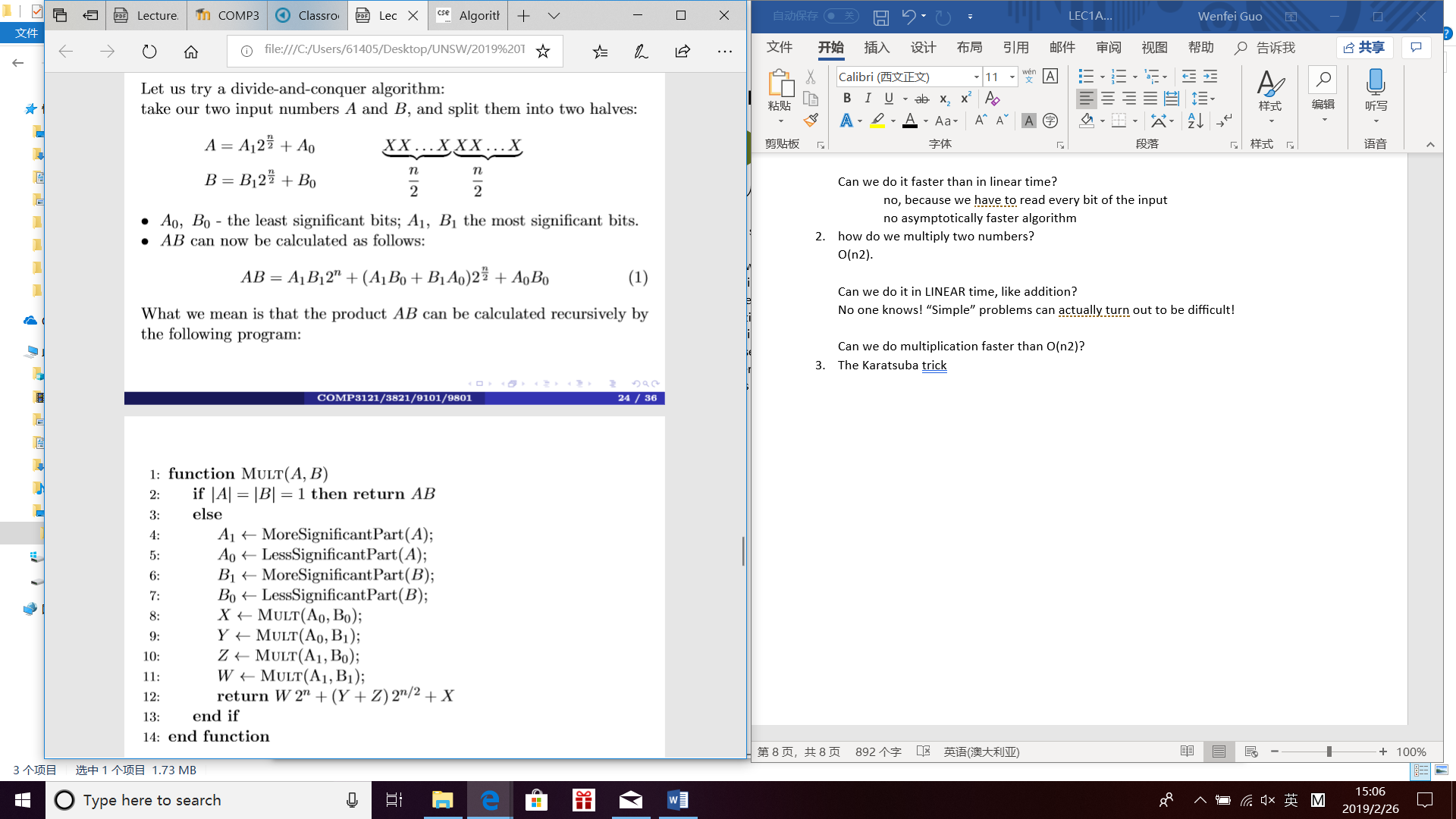
Can we do it in LINEAR time, like addition?

No one knows! “Simple” problems can actually turn out to be diﬃcult!

Can we do multiplication faster than O(n2)?

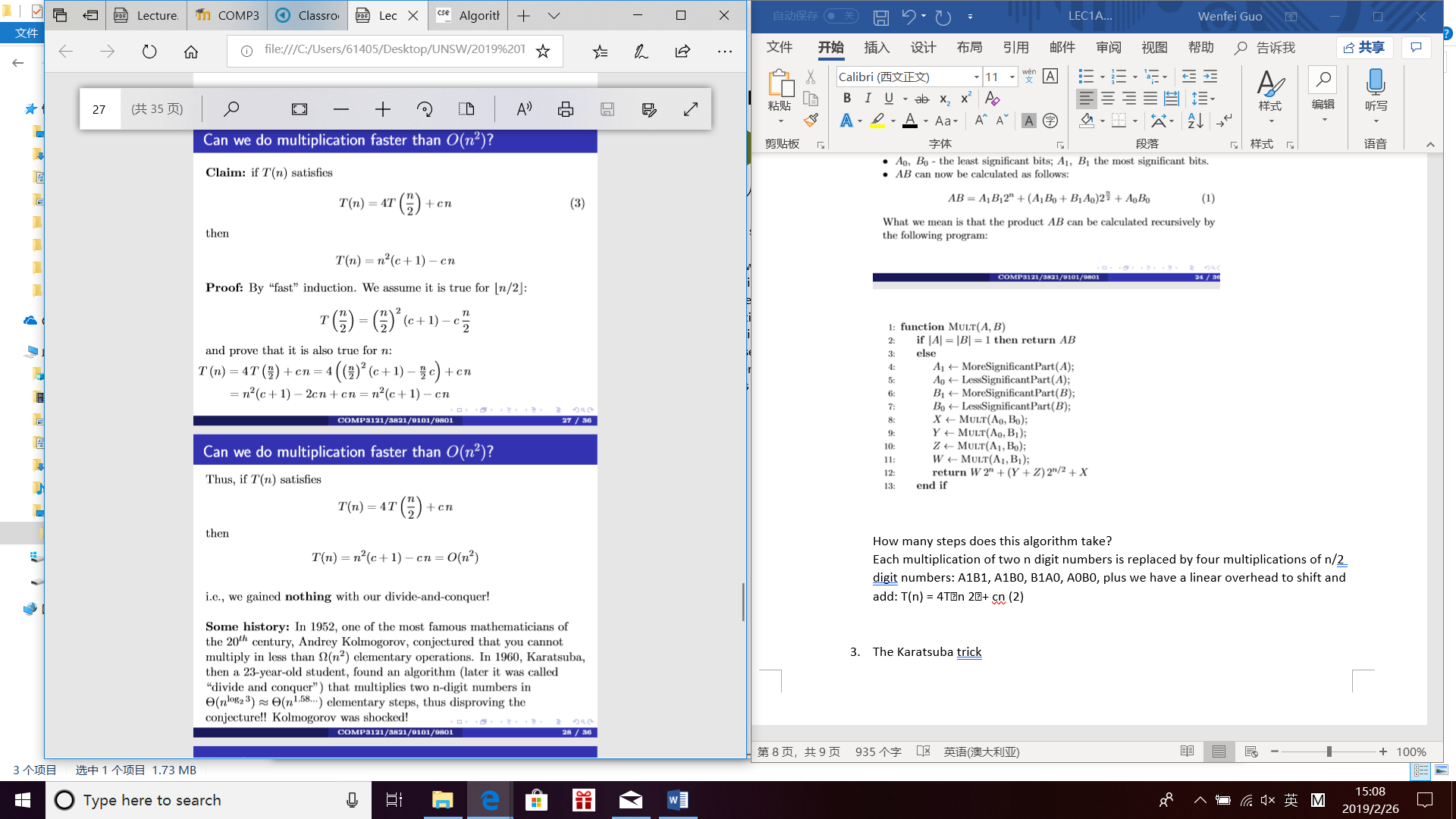
YES

METHOD: DEVIDE AND CONCURE



How many steps does this algorithm take?

Each multiplication of two n digit numbers is replaced by four multiplications of n/2 digit numbers: A1B1, A1B0, B1A0, A0B0, plus we have a linear overhead to shift and add: T(n) = 4Tn 2+ cn (2)



1. The Karatsuba trick

