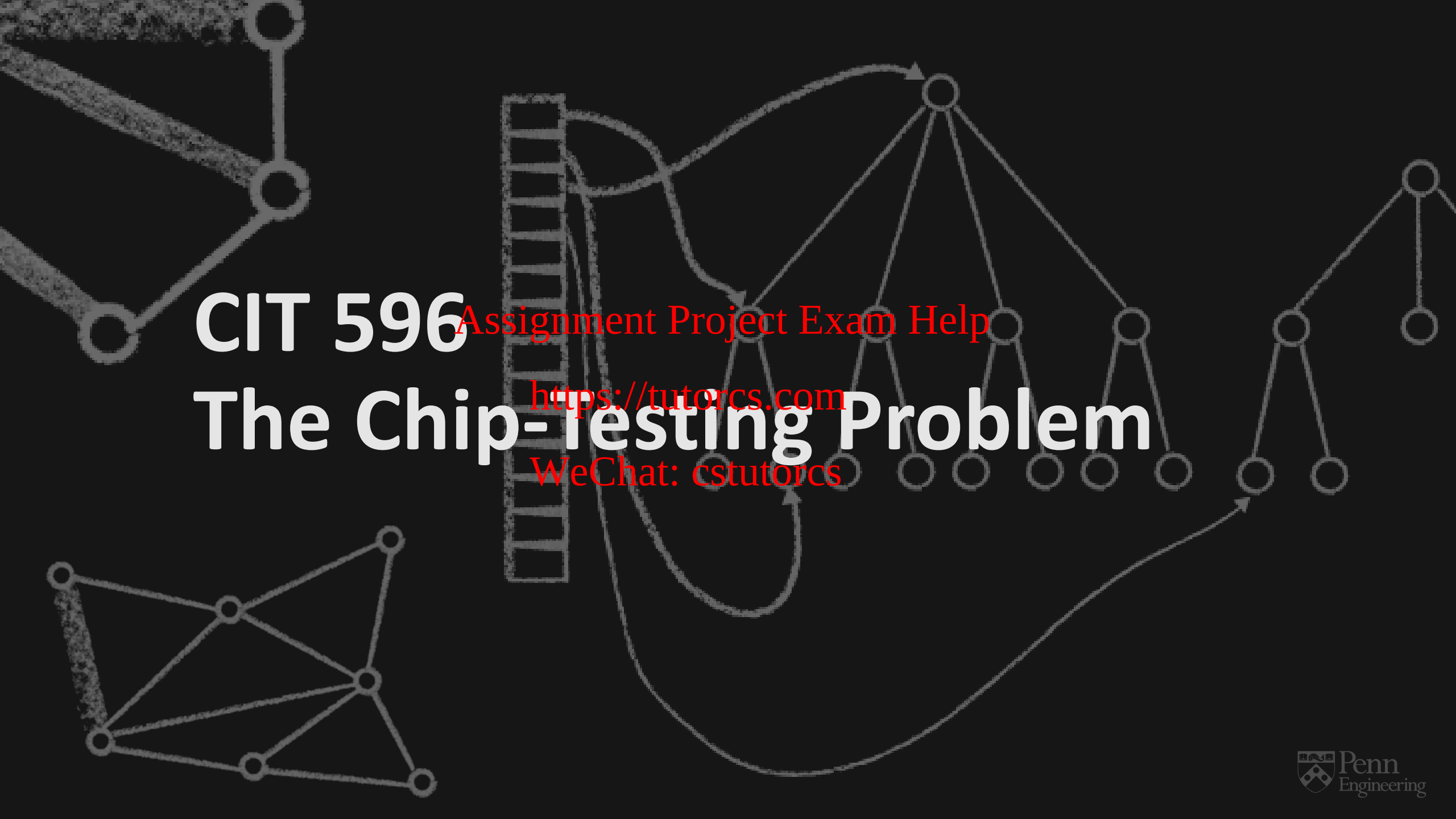

CIT 596
The Chip-Testing Problem

Penn Engineering

The Chip Testing Problem

- This problem is motivated by VLSI chip design.
- In each batch of chips produced, some chips are *good* and some are *bad* (faulty).
- Given a set of n chips, we want to correctly determine the status of each chip!

What tool do we have at our disposal?

- We have a tester where you can plug in a pair of chips and have them test each other.

Goal: Design an algorithm to correctly identify the status of each chip as good/bad using a small number of pairwise tests.

The Behavior of Good vs Bad Chips

- In any pairwise test, a good chip always correctly declares the status of the other chip.
- But a bad chip can declare the other chip as good or bad, regardless of its actual status.
- This is precisely what makes this a difficult problem!

In fact, we can only hope to solve this problem if the good chips are in a *strict majority*.

- Suppose there were an equal number of good and bad chips. Let G be the set of good chips and let B be the set of bad chips.
- Let us do all possible pairwise tests. Any 2 chips in G will declare each other good correctly. But any 2 chips in B can also declare each other good.
- Moreover, whenever we test a chip from G against a chip in B, they can both declare each other bad.
- A perfect symmetry between the behavior of chips in G and B. No way to identify who is good and who is bad!

Some Useful Observations

Observation 1: We will assume that the good chips are in a strict majority in our input.

Observation 2: Any algorithm for our problem requires $\Omega(n)$ pairwise tests.

- Each chip needs to be tested at least once.

Observation 3: It suffices to find a single good chip.

- Once a good chip has been identified, we can use it to test all other chips.
- From now on, we will focus on this task.

A First Attempt

- Pick an arbitrary chip, say C.
- Let all other $(n-1)$ chips test C, one by one.
- If at least $(n-1)/2$ chips say C is good, then we identify C as a good chip -- we are done in this case.
- Otherwise, discard C, and repeat this process on the remaining chips.

A First Attempt

- Pick an arbitrary chip, say C.
- Let all other $(n-1)$ chips test C, one by one.
- If at least $(n-1)/2$ chips say C is good, then identify C as a good chip. We are done in this case.
- Otherwise, discard C, and repeat this process on the remaining chips.

Correctness

- We are deciding the status of C by taking a majority vote.
- The majority vote correctly determines the status of C since the good chips are in a strictly majority.
- A subtle point: convince yourself that if the number of chips declaring C to be good is same as the number of chips declaring C to be bad, then C must be good!

A First Attempt

Complexity Analysis

- Since the good chips are in a strict majority, the algorithm can discard at most $\lfloor \frac{n}{2} \rfloor$ chips before it finds a good chip.
- So the maximum number of tests that the algorithm performs is bounded by $(n-1) + (n-2) + \ldots + \left\lfloor \frac{n}{2} \right\rfloor = O(n^2)$.
- There is no slack in the analysis above – it can actually happen that the algorithm first exhausts testing all the bad chips, and only then finds a good chip.
- So in the worst-case, this algorithm can actually require $\Omega(n^2)$ tests.

Can we do better?

A Second Attempt

- A weakness of the previous approach is that in each round of testing, all tests focus on a single chip.
- If the tested chip turns out to be bad, we make very little progress – we get to discard a single chip.
- We will next develop an alternate algorithmic approach where:
  - In each round of testing, we either find a good chip or discard at least half the remaining chips.
  - We then recursively search for a good chip among the remaining chips.
  - This will lead to an algorithm that uses far fewer tests overall to identify a good chip.

A Second Attempt: A Recursive Algorithm

- If $n = 1$ or $n = 2$, then return any chip as good.
- If n is odd, then
  - Pick an arbitrary chip, say C, and let all other $(n-1)$ chips test C, one by one.
  - If at least $(n-1)/2$ chips say C is good, then identify C as a good chip (we are done in this case).
  - Otherwise discard C, and continue.
- At this point, we are left with an even number of chips. We arbitrarily pair them to create $\left\lfloor \frac{n}{2} \right\rfloor$ pairs where each pair of chips tests one another.
- Based on the test results, we can classify each pair as either G-G (both declare each other good), G-B (one declares good, other declares bad), or B-B (both declare each other bad).
- We now discard all pairs of chips that are either G-B or B-B.
- For the G-G pairs of chips, we arbitrarily discard one chip in each pair.
- We now recursively execute the algorithm on the remaining chips.

Analyzing the Second Algorithm

Correctness

- If $n = 1$ or $n = 2$, then returning any chip as good is correct because the good chips are in a strict majority.
- If n is odd, then our tests correctly determine the status of the chip C by the earlier argument.
- If we are not done at this point, we are now preparing to eliminate many chips and recurse.

Key Invariant: The good chips must remain in a strict majority as we recurse.

- Any G-B or B-B pair is guaranteed to contain at least one bad chip. So discarding all such pairs maintains the strict majority invariant among the remaining chips.
- But why is it ok to arbitrarily discard one chip in each G-G pair?

Analyzing the Second Algorithm

Correctness

- Any G-G pair has the property that either both chips in the pair are good or both chips in the pair are bad.
- Let us denote by g the number of G-G pairs where both chips are good and by b the number of G-G pairs where both chips are bad.
- Then since the strict majority invariant holds thus far, it must be that $g > b$.
- But now if we arbitrarily discard a chip in each G-G pair, then we are left with a total of $g + b$ chips.
- Since $g > b$, the good chips indeed remain in a strict majority!

Analyzing the Second Algorithm

Complexity Analysis

Let us denote by $T(n)$ the worst-case number of tests used by the algorithm in locating a good chip under the promise that the good chips are in a strict majority. We then get the recurrence below.

$$T(n) \leq \begin{cases} 0 & \text{if } n \leq 2 \\ (n-1) + \left\lfloor \frac{n}{2} \right\rfloor + T\left(\left\lceil \frac{n}{2} \right\rceil\right) & \text{if } n > 2 \end{cases}$$

- The base case is straightforward.
- For the recursive case, we note that
  - The algorithm performs $(n-1)$ tests in the beginning whenever n is odd.
  - If we are not done, then we next perform $n/2$ pairwise tests.
  - Finally, when we recurse, we have discarded at least half the chips that we had started with.

Analyzing the Second Algorithm

Complexity Analysis

We can now solve this recurrence by simply expanding it out.

$$
\begin{aligned}
T(n) &\leq (n-1) + \left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) \\
&\leq \left(\frac{3n}{2}\right) + T\left(\frac{n}{2}\right) \\
&\leq \left(\frac{3n}{2}\right) + \left(\frac{3n}{4}\right) + T\left(\frac{n}{4}\right) \\
&\leq \left(\frac{3n}{2}\right) + \left(\frac{3n}{4}\right) + \left(\frac{3n}{8}\right) + \cdots + \left(\frac{3n}{2^i}\right) + T\left(\frac{n}{2^i}\right) \\
&\leq \left(\frac{3n}{2}\right)\left[1 + \frac{1}{2} + \frac{1}{4} + \ldots\right] \\
&\leq 3n
\end{aligned}
$$

Summarizing …

- We have designed an algorithm that solves the chip testing problem using $O(n)$ tests.
- By Observation 2, this is an asymptotically optimal algorithm.
- Our second algorithm illustrates a variation of the divide and conquer paradigm that is referred to as decrease and conquer:
  - In order to solve a given problem instance, we perform some amount of work and create a single new instance of the problem that is much smaller in size.
  - We then recursively solve the problem on this smaller instance.
- Next week we will see another classic example of the decrease and conquer paradigm, namely, binary search.