6.5240 Sublinear Time Algorithms October 27, 2026

Lecture 13

Lecturer: Sijin (Ronitt Rubinfeld) *Scribe: Carmel Baharav*

In this class.

- Property testing for strings
- Lower bound via Yao's method (general technique)

More specifically, we study the problem of property testing for *Concatenations of Palindromes*, and prove a lower bound for the number of queries required by any randomized property tester for L_n by using Yao's method.

1 Intro to Property Testing of Strings

Definition 1 (ε-far) *$w \in \{0,1\}^n$ is ε-far from $P_n \subseteq \{0,1\}^n$ if $\forall y \in P_n$ we have that Hamming$(w,y) > \varepsilon n$. Recall that Hamming distance is the number of bits on which two strings differ.*

Definition 2 (Property tester) *A property tester $\mathcal{A}$ of $P_n \subseteq \{0,1\}^n$ satisfies the following guarantees:*

- $w \in P_n \implies \Pr(\mathcal{A}(w) = \text{"PASS"}) \geq \frac{2}{3}$
- *w is ε-far* $\implies \Pr(\mathcal{A}(w) = \text{"FAIL"}) \geq \frac{2}{3}$

1.1 Warm-Up: Palindrome

English has numerous palindromes that we know and love: racecar, kayak, level, etc. The set of palindromes over binary strings is less fun, but is defined as follows.

Definition 3 (Palindrome) *For even n,*

$$P_n = \{vv^R | v \in \{0,1\}^{n/2}\}$$

where v^R corresponds to the reversal of v.

Consider the property testing algorithm for P_n below:

Algorithm 1 Palindrome Property Tester

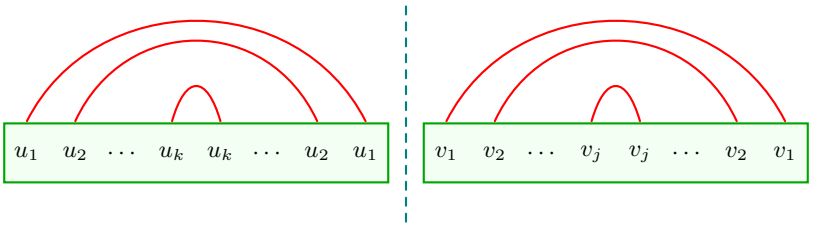
```
for Θ(1/ε) times do
    Pick random i ∈ {1, ..., n}
    if w_i ≠ w_{n−i+1} then
        return FAIL
return PASS
```

Observe that for $w \in P_n$, this algorithm will always return PASS. If w is ε-far from P_n, this means there are at least εn pairs of (w_i, w_{n-i+1}) that don't match. Then in each iteration, there is constant probability of detecting a mismatching pair.

2 Concatenation of Palindromes

Definition 4 (Concatenation of Palindromes) *For even n,*

$$L_n = \{uu^R vv^R \mid |u| + |v| = n/2\}$$



If the property tester knew the length of the first palindrome, we could just apply the same technique from the warm-up (Algorithm 1) to test whether both components of the input string are palindromes. However, the property tester *does not know* the length of the first palindrome, which turns out to introduce a substantial increase in difficulty.

Theorem 5 *Any property tester for L_n requires $\Omega(\sqrt{n})$ probes.*

Note that this theorem encompasses both randomized and adaptive algorithms. We have already seen examples of lower bounds in the course for a variety of problems: average degree, number of connected components, and weight of an MST. These have all followed a *needle in a haystack* approach: the two different families of graphs are extremely similar and the differences are in a very concentrated subset of the input.

For all nontrivial property testing algorithms, you have an immediate lower bound of $\Omega(1/\varepsilon)$. This is because ε-far inputs only differ on an ε fraction of the total bits, so in order to observe any of these bits with a constant probability, you need $\Omega(1/\varepsilon)$ queries. To prove Theorem 5, we will introduce a general tool for lower bounding the query-complexity of randomized algorithms: Yao's principle.

2.1 Yao's Principle

Lemma 6 (Yao's principle) *If there exists a distribution $\mathcal{D}$ over "positive" and "negative" instances such that any deterministic algorithm tester $\mathcal{A}$ with $\leq t$ queries fails on $\mathcal{D}$ with probability $\geq 1/3$, then any randomized tester $\mathcal{A}$ requires $> t$ queries.*

We will not prove Yao's principle in this class, but the intuition is as follows: every randomized algorithm can be thought of as a mixture over deterministic algorithms. If there is a distribution that tricks *every* deterministic algorithm, then all of the deterministic algorithms comprising a randomized algorithm get tricked by this distribution. Yao's principle tells us that:

Average-case LB against deterministic algorithm $\implies$ Worst-case LB against randomized algorithm

Therefore, to show a worst-cased lower bound for a randomized algorithm, all we have to do is (1) Define a distribution $\mathcal{D}$ over instances and (2) Prove a deterministic average-case lower bound. We instantiate this approach in proving Theorem 5.

Step 1: Define $\mathcal{D}$. We define our distribution by defining the distribution over negative instances, $\mathcal{N}$, and positive instances, $\mathcal{P}$, and then setting

$$\mathcal{D} = \frac{1}{2}\mathcal{N} + \frac{1}{2}\mathcal{P}$$

- Negative instance distribution $\mathcal{N} :=$ uniform distribution over all ε-far strings.
- The positive instance distribution $\mathcal{P}$ is described by the following random process.
  1. Sample $k \in_R \left[\frac{n}{6} + 1, \frac{n}{3}\right]$. Without loss of generality, assume $6|n$.
  2. $u \in_R \{0,1\}^k, v \in_R \{0,1\}^{n/2-k}$
  3. Output $uu^R vv^R$

Step 2: Lower Bound. Assume for sake of contradiction that there exists a deterministic algorithm $\mathcal{A}$ using $\leq t = o(\sqrt{n})$ queries. We can visualize $\mathcal{A}$ as a query tree with depth t:

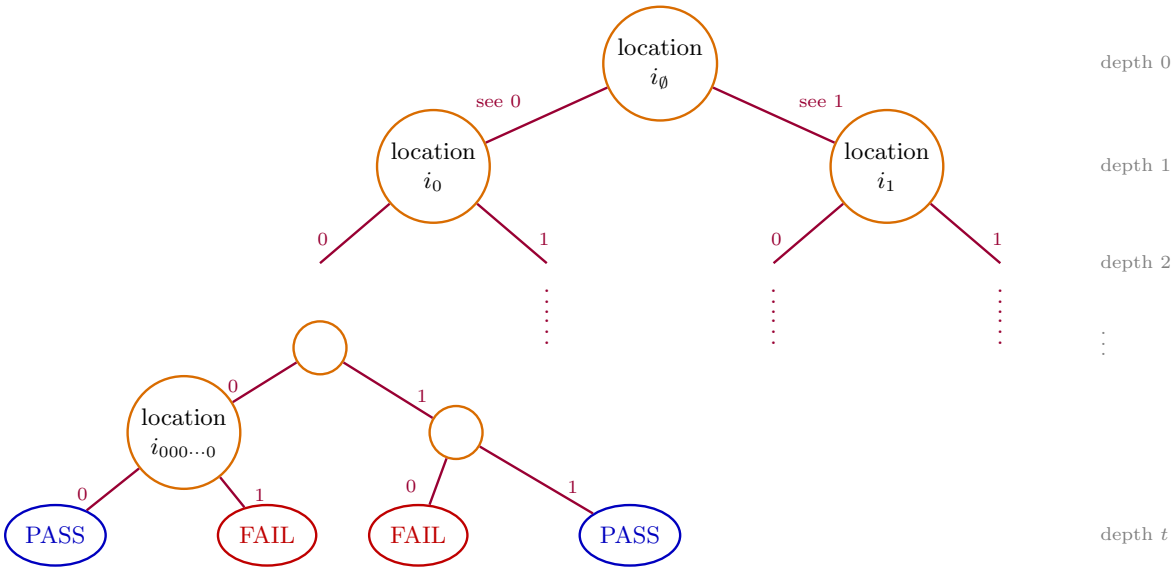


Figure 1: The query tree of $\mathcal{A}$, a deterministic property tester. Each internal node specifies the next location to query according to $\mathcal{A}$, indexed by the bit string observed so far. The edges show which bit is seen when that location is queried. Finally, the leaves specify the algorithm's final decision on the input string.

We make two assumptions on this query tree, both of which are without loss of generality:

1. There are no redundant location queries on any path from the root to a leaf. If this were the case, then there would be one branch of the redundant query that the algorithm would never go down, and all of the leaves down that route would never be reached by $\mathcal{A}$ – this branch could just be excised from the tree entirely.
2. Every root to leaf path has depth exactly t. If this is not the case, you can extend the paths that are too short without changing the behavior of $\mathcal{A}$ by replacing the leaves with query nodes whose two children have the same output.

To determine the total error of $\mathcal{A}$ on our distribution $\mathcal{D}$, we consider the error incurred when $\mathcal{A}$ reaches a specific leaf of the query tree, ℓ. First, we separate out the true yes/no words that reach each leaf:

- Let $E^-(\ell) := \{w \in \{0,1\}^n | w \text{ is } \varepsilon - \text{far and } \mathcal{A}(w) \text{ reaches } \ell\}$
- Let $E^+(\ell) := \{w \in \{0,1\}^n | w \in L_n \text{ and } \mathcal{A}(w) \text{ reaches } \ell\}$

If ℓ is labeled "PASS" the inputs that $\mathcal{A}$ misclassifies are $E^-(\ell)$. If ℓ is labeled "FAIL" then the inputs that $\mathcal{A}$ misclassifies are $E^+(\ell)$. So, by the law of total probability, we have that

$$\Pr_{w\sim\mathcal{D}}(\mathcal{A}(w) \text{ is wrong}) = \underbrace{\sum_{\ell \text{ "PASS"}} \Pr_{w\sim\mathcal{D}}(w \in E^-(\ell))}_{\mathcal{A} \text{ should fail, but reaches passing leaf}} + \underbrace{\sum_{\ell \text{ "FAIL"}} \Pr_{w\sim\mathcal{D}}(w \in E^+(\ell))}_{\mathcal{A} \text{ should pass, but reaches a failing leaf}}$$

Now we lower bound this probability by showing the following two claims:

Claim 7 *If $t = o(n)$, then $\forall \ell$ at depth t, we have*

$$\Pr_{w\sim\mathcal{D}}(w \in E^-(\ell)) \geq \left(\frac{1}{2} - o(1)\right) 2^{-t}$$

Claim 8 *If $t = o(\sqrt{n})$, then $\forall \ell$ at depth t, we have*

$$\Pr_{w\sim\mathcal{D}}(w \in E^+(\ell)) \geq \left(\frac{1}{2} - o(1)\right) 2^{-t}$$

If we recall that overall there are 2^t leaves in this query tree, then with these two claims, we have that if $t = o(\sqrt{n})$, then:

$$\Pr_{w\sim\mathcal{D}}(\mathcal{A}(w) \text{ is wrong}) \geq \sum_{\ell \text{ "PASS"}} \left(\frac{1}{2} - o(1)\right) 2^{-t} + \sum_{\ell \text{ "FAIL"}} \left(\frac{1}{2} - o(1)\right) 2^{-t} = \frac{1}{2}(1 - o(1)) \geq 1/3$$

Proof of Claim 7: Fix a leaf ℓ. First observe that w can only be in $E^-(\ell)$ if w is drawn from $\mathcal{N}$ and not $\mathcal{P}$, which happens with probability 1/2. Notice that if we were sampling from the uniform distribution over all binary strings, instead of just ε-far ones, then we would end up at each leaf with probability 2^{-t} (there are 2^{n-t} distinct words that would end up at leaf ℓ because they only have to match at the t locations queried along the path from the root, and 2^n total words – hence the probability that w ends up at ℓ is $2^{n-t}/2^n$). To show Claim 7, we argue that $\mathcal{N}$ is very close to the uniform distribution over all binary strings.

Let N be the support of $\mathcal{N}$, i.e. all binary strings that are ε-far from L_n. Then define $F_n := \{0,1\}^n \setminus N$ to be the remaining strings. We have that

$$|F_n| \leq |L_n| \times \sum_{i=0}^{\varepsilon n} \binom{n}{i}$$

This is because any string in F_n is not ε-far from L_n, so it can be constructed by starting with a string in L_n and flipping i bits, where $i \in \{0, \ldots, \varepsilon n\}$. Now we can upper bound $|L_n|$ by summing up over all possible lengths of the first palindrome, and counting the different strings that could form the first and second palindromes. The upper bound for the second term in the product is just a convenient identity.

$$\begin{aligned} &\leq \left(\sum_{k=0}^{n/2} 2^k 2^{n/2-k}\right) \times \left(2^{2\varepsilon \log(1/\varepsilon) n}\right) \\ &\leq (n/2+1) \times 2^{n/2} \times 2^{2\varepsilon \log(1/\varepsilon) n} \end{aligned}$$

Now observe that we can compute $|E^-(\ell)|$ by taking the total number of strings that can reach ℓ (2^{n-t}) and subtracting off the number of strings that reach ℓ that *should* pass. We lower bound $|E^-(\ell)|$ by subtracting off the *total* number of strings that should pass ($|F_n|$), instead of just those that reach ℓ.

$$|E^-(\ell)| \geq 2^{n-t} - |F_n| \geq (1 - o(1))2^{n-t}$$
$$|N| = 2^n - |F_n| \geq (1 - o(1))2^n$$

To conclude Claim 7, observe that

$$\Pr_{w\sim\mathcal{D}}(w \in E^-(\ell)) = \frac{1}{2}\Pr_{w\sim\mathcal{N}}(w \in E^-(\ell)) = \frac{1}{2}\frac{|E^-(\ell)|}{|N|} \geq \left(\frac{1}{2} - o(1)\right)2^{-t}$$

■

Proof of Claim 8: Fix a leaf ℓ – we want to show that many strings in L_n end up at this leaf. The main difficulty in showing this is that strings in L_n have some correlated entries. For example, consider the figure below. If a path contains queries to indices both 3 and 8, then these queries are not independent for strings in L_n with $k = 5$.

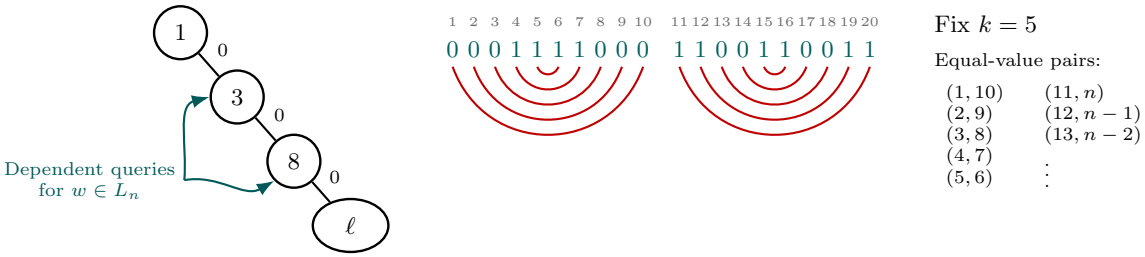


Figure 2: For a fixed k, queried positions that are mirror images in either palindrome must have the same value.

Consider the path to leaf ℓ. There are $\binom{t}{2}$ distinct pairs of queries along the path. Using that $t = o(\sqrt{n})$, we have $o(n)$ pairs. For every pair (i_a, i_b) there are at most two values of k where w_{i_a} and w_{i_b} should be equal for w to be in L_n (if they are either centered in the first palindrome or in the second). We call these values of k "bad". Then this means that there are $o(n)$ bad k overall. Recall in our construction of $\mathcal{P}$ that we sample k uniformly at random from $[n/6+1, n/3]$ — so there are at least $\frac{n}{6} - o(n) = (1-o(1))\frac{n}{6}$ good k.

Putting it all together, and recalling that w can only be in $E^+(\ell)$ if it is drawn from $\mathcal{P}$, we have that

$$\begin{aligned}\Pr_{w\sim\mathcal{D}}(w \in E^+(\ell)) &= \frac{1}{2}\Pr_{w\sim\mathcal{P}}(w \in E^+(\ell)) \\ &= \frac{1}{2}\sum_{w\in L_n}\sum_{k}\underbrace{\Pr_{\mathcal{P}}(w|k)}_{2^{-n/2}}\cdot\underbrace{\Pr(k \text{ chosen})}_{6/n}\cdot\mathbf{1}(w \in E^+(\ell))\end{aligned}$$

Finally, we lower bound the sums by only considering good k for our fixed path to ℓ. For every good k, we know that there are $2^{n/2-t}$ words in L_n that can be generated by this choice of k and that follow the path down to ℓ (by having the right bits for the t queries along the path).

$$\begin{aligned}&\geq \frac{1}{(n/6)\cdot 2^{n/2+1}}\cdot(1-o(1))\cdot\frac{n}{6}\cdot 2^{n/2-t} \\ &= \left(\frac{1}{2} - o(1)\right)2^{-t}\end{aligned}$$

■