# Introduction to Property Testing
Lecture 5

Lecture by Dr. Reut Levi
Typeset by Steven Karas

2020-05-20
Last edited 22:25:53 2020-05-20

**Disclaimer** These notes are based on the lectures for the course Introduction to Property Testing, taught by Dr. Reut Levi at IDC Herzliyah in the spring semester of 2019/2020. Sections may be based on the lecture slides prepared by Dr. Reut Levi.

## 1 Agenda

- Monotonicity
- Graph properties

## 2 Testing Monotonicity

Consider $f : [n] \to R_n$, where $R_n$ is an arbitrary totally ordered set. w.l.o.g. we may assume that the values of $f$ are distinct (break ties according to index - i.e. augment $f(i)$ to $(f(i), i)$). Note that we can consider $f$ as an array filled with values.

**Algorithm 2**

1. select $i \in [n]$ u.a.r.
2. find $f(i)$ in the array by performing a binary search.
3. accept iff $f(i)$ is found

This has query complexity $1 + \lceil \log n \rceil$.

**Correctness** If our algorithm accepts w.p. $\geq 1 - \delta$, then $f$ is $\delta$-close to monotone.

Note that the only random choice is $i$. We define that $i$ is good (w.r.t. $f$) if the execution on $i$ accepts. if $i < j$ are both good, then $f(i) < f(j)$. Let $t$ be the first location where the binary search for $i$ and $j$ take different halves. If $f(i) \leq f(t)$ and $f(j) > f(t)$, then it follows that $f(i) < f(j)$.

Observe that the restriction of $f$ to the set of good points is a monotone function. If we correct $f$ on the non-good points we obtain a monotone function. There are $\geq (1 - \delta) \cdot n$ good points thus $f$ is $\delta$-close to monotone.

**A related non-adaptive tester** Observe that after selecting $i \in [n]$, the choice of queries can be determined a priori. If we search for the index $i$ rather than the value, we can make these queries and check that there are no violations. Correctness follows similarly to above.

**2-query algorithm** There exists a 2-query POT for monotonicity with detection probability $\delta / \lceil \log_2 n \rceil$.

1. select $i \in [n]$ u.a.r.
2. select one of the queries of the nonadaptive algorithm u.a.r.
3. reject iff we see a violation.

# 3 Dense Graph Model

In the adjacency matrix model[1](a.k.a. the dense graph model), a $k$-vertex graph $G = ([k], E)$ is repesented by $g : [k] \times [k] \to \{0, 1\}$ such that $g(u, v) = 1$ iff $\{u, v\} \in E$. We define the distance between $G$ and $G'$ as represented by $g$ and $g'$, respectively:

$$\delta(G, G') = \frac{|\{(u, v) : g(u, v) \neq g'(u, v)\}|}{|V|^2}$$

We say that graph properties are sets of graphs that are closed under isomorphism. That is, $\Pi$ is a graph property if $\forall G = ([k], E)$ and every permutation of $[k]$ : $G \in \Pi$ iff $a(G) \in \Pi$ where $a(G) \stackrel{\text{def}}{=} ([k], \{\{a(u), a(v)\} : \{u, v\} \in E\})$. In simple terms, the labels of the vertices do not matter, only the structure of the edges.

## 3.1 Testing bipartiteness

We define a bipartite graph as a graph whose vertices can be partitioned into $A, B$ such that every edge is in $A \times B$. [2]

**Algorithm** Given an input $k, \varepsilon$:

1. u.a.r. select $\tilde{O}(1/\varepsilon^2)$ vertices denoted by $R$.
2. accept iff subgraph induced on $R$ is bipartite.

This algorithm rejects graphs that are $\varepsilon$-far from being bipartite with probability $\geq 2/3$.

**Correctness** Assume $G$ is $\varepsilon$-far from bipartite and we'll see that with probability $\geq 2/3$, $G([R])$ is not bipartite. View $R$ as a union of $U$ and $S$ (disjoint) where $t = |U| = O(\frac{\log \frac{1}{\varepsilon}}{\varepsilon})$ and $m = |S| = O(\frac{t}{\varepsilon})$. Consider all 2-partitions of $U$. For each 2-partition $U_1, U_2$ of $U$ consider the partial 2-partition of $[k]$: One side: all neighbors of $U_1$ are opposite to $U_1$. The other side: all neighbors of $U_2$ are opposite to $U_2$. If $v$ is adjacent to both $U_1$ and $U_2$ place it opposite to $U_2$.

Denote $N(v) \stackrel{\text{def}}{=} \{u : \{u, v\} \in E\}$ and $N(X) \stackrel{\text{def}}{=} \bigcup_{v \in X} N(v)$.

Given a 2-partition $U_1, U_2$ of $U$, define a possibly partial 2-partition of $[k]$: $V_1, V_2$ where $V_1 = N(U_2)$ and $V_2 = N(U_1) \setminus V_1$.

Define a vertex $v$ as being high-degree if its degree $\geq \frac{\varepsilon k}{6}$. $U$ is good if all but at most $\frac{\varepsilon k}{6}$ of high-degree vertices are in $N(U)$. With probability $\geq 5/6$, $U$ is good. Proof follows from any $v$ that has high degree $\Pr[N(v) \cap U = \emptyset] = (1 - (\varepsilon/6))^t < \varepsilon/36$ (due to Markov's inequality).

# References


[1] Oded Goldreich. *Introduction to property testing.* Cambridge University Press, 2017.


[1]Sparse matrices are typically stored as lists of edges per vertex

[2]There is no POT with constant query complexity, and we will likely have this as a question as part of the next homework set (exercise 8.5 in the book).