6.896 Sublinear Time Algorithms	September 14, 2004
Lecture 2	
Lecturer: Ronitt Rubinfeld	*Scribe: Akshay Patil*

1 Last Time

Types of approximation

- Optimization problems - standard (saw the notion of α-additive and γ-multiplicative relative approximations).
- Decision problems - saw example of monotone sequences

2 Definition of Property Tester

2.1 Decision Problems

Define a language P to be a class of inputs that have a certain property.
examples:

- class of allowable graphs (e.g. connected, bipartite, has large cut)
- class of allowable sequences (e.g. monotonic increasing integers)
- class of allowable functions, represented by input/output tables (e.g. homomorphisms)

2.1.1 Behavior of a decision algorithm for P

$A(x)$:
  $x \in P \Rightarrow$ pass
  $x \notin P \Rightarrow$ fail

2.2 Representation/Model

Representation is **very** important with respect to distance functions. Also important is how we access the input. Most common is the *random access* model. In this model, accessing each of the following would count as 1 step:

- ith bit of a string
- function value $f(i)$
- the i, j^{th} entry of a matrix

2.3 Distance function

In order to define a property tester, it is important to define a notion of distance from having a property. Define some problem specific $\delta(x, y)$ distance function between inputs, with $\delta(x, y) \in [0, 1]$. Define $d(x, P) = \min_{y \in P} \delta(x, y)$. **examples:**

- "Hamming distance" = # bits/entries to change x into y
- Edit distance = minimum # inserts/deletes to change x into y

2.4 Property Tester

A *Property Tester* for P, d is defined as:

Given input x, $0 < \epsilon < 1$

if $x \in P$, then Pr[output $Pass$] $\geq \frac{2}{3}$

if $d(x, P) \geq \epsilon$ then Pr[output $Fail$] $\geq \frac{2}{3}$

Comments:

1. It is important to quantify the runtime and query complexity in terms of $|x|, \epsilon$ and possibly other parameters of P (e.g. degree bound on graphs, dimension of space)
2. If $x \notin P$ but $d(x, P) < \epsilon$, then both $Pass$ and $Fail$ are fine
3. Note that the probabilistic guarantees depend only on the coin tosses of the tester, not on x, P, d
4. The 1/3 probability of error is arbitary; in homework you will use Chernoff bounds to show that one can reduce the error to any given δ by paying only a $O(\log \frac{1}{\delta})$ multiplicative factor in running time.
5. There is also the notion of a *tolerant tester*, one that always outputs $Pass$ when $d(x, P) \leq \epsilon'$
6. To make the point of why the representation of the input is so important, consider the following example: In an adjacency matrix representation of a graph, ϵ-close means that ϵn^2 entries can be changed. This implies that all graphs are ϵ-close to the class of connected graphs. **But** if an adjacency list represenation is used on a degree d bounded graph, where ϵ-close means that ϵdn entries can be changed, connectivity is **not** trivial.

3 Connectivity

Problem statement:

1. input class: degree $\leq d$ graphs (G^d), given as adjacency lists.
2. desired property: connectedness; $P =$ class of degree $\leq d$ connected graphs
3. access to graph is in the *bounded degree graph model* (first proposed by Goldreich and Ron).
   (a) for a degree $\leq d$ graph of n nodes (G_n^d) is represented as a size n array (N) with each element (N_i) pointing to a size d array (D_i).
   (b) each element of D_i is the name of another element in N or is empty ($\perp$).
   (c) the elements of D_i are in no particular order, non-empty elements are not neccessarily before empty elements.

We use the bounded degree graph model to define the function $f_G : V \times [d] \to V \cup \{\perp\}$, where $[d]$ signifies the set $1...d$. So, for example, $f_G(i, l) = j$ if (i, j) is the l^{th} edge in D_i (and $= \perp$ if there the l^{th} element is empty). We'll consider one query (i.e. one step) to be one access to f_G.

Distance definitions:

$dist(G_1, G_2) = \delta(G_1, G_2) = \frac{|\{(i,l): 1 \leq i \leq n, 1 \leq l \leq d, G_1(i,l) \neq G_2(i,l)\}|}{dn}$

$dist(G, P) = d(G, P) = \min_{G_2 \in P} \ \delta(G, G_2)$

3.1 Property Tester for Connectivity

Idea: If a graph is far from connected, there must be many components, which in turn implies that there are many small components.

Algorithm: (also by Goldreich and Ron)

1. Pick $m = O(\frac{1}{\epsilon d})$ random nodes
2. $\forall S \in \{\text{picked nodes}\}$ do BFS and stop if
   (a) see at least $\frac{8}{\epsilon d}$ new nodes
   (b) exhaused the component (i.e. there are $< \frac{8}{\epsilon d}$ nodes)
3. If (b) ever happens, $Fail$ and halt; else $Pass$

Runtime is $\tilde{O}(\frac{1}{\epsilon d} \cdot \frac{8}{\epsilon d} \cdot d) = \tilde{O}(\frac{1}{\epsilon^2 d})$. $\tilde{O}$ signifies that we don't care about logarithmic factors concerning ϵ or d.

Why does it work?

Define G_n^d to be the class of degree d bounded graphs on n nodes, P_n to be the class of connected graphs over n nodes and P_n^d to be the class of degree d bounded connected graphs on n nodes.

Assume $d \geq 2$.

If $G \in P_n^d$, the proof that algorithm outputs $Pass$ is trivial. So what if $G \notin P_n^d$?

Lemma: if $G \in G_n^d$ is ϵ-far from P_n^d then G has $\geq \frac{\epsilon}{4} dn$ connected components.

Proof left as an exercise

Easier Lemma: if $G \in G_n^d$ is ϵ-far from $P_n \Rightarrow \geq \epsilon dn$ components.

Proof: $< \epsilon dn$ components $\Rightarrow$ can add $< \epsilon dn$ edges to connect it (note that this can make the degree much bigger than d).

Corollary to First Lemma: $G \in G_n^d$ is ϵ-far from $P_n^d \Rightarrow G$ has $\geq \frac{\epsilon dn}{8}$ componenents of size $\leq \frac{8}{\epsilon d}$

Proof: let $l = \#$ components of size $\leq \frac{8}{\epsilon d}$ and $l' = \#$ components of size $> \frac{8}{\epsilon d}$. Then $l + l' \geq \frac{\epsilon dn}{8}$. Well, $l' \cdot \frac{8}{\epsilon d} \leq n \Rightarrow l' \leq \frac{\epsilon dn}{8} \Rightarrow l \geq \frac{\epsilon dn}{4} - \frac{\epsilon dn}{8} = \frac{\epsilon dn}{8}$.

Theorem: $G \in G_n^d$

- if G connected, tester passes
- if G ϵ-far from P_n^d, Pr[tester $Fails$]$\geq \frac{2}{3}$

Proof: By the corollary, $\Pr[s \in \text{small component}] = \frac{\#\text{ nodes in small components}}{n} \geq \frac{\#\text{ small components}}{n} \geq \frac{\epsilon d}{8}$ (where the first inequality is because each component is of size at least one and they are disjoint). Since m is chosen to be $\frac{c}{\epsilon d}$, $\Pr[\text{no } s \in \text{small component}] \leq (1 - \frac{\epsilon d}{8})^{\frac{c}{\epsilon d}} \leq e^{-c'} < \frac{1}{3}$ (picking c', c large enough).