Lecture 22: Sublinear Time Algorithms

- Models

- Classical Approximation
  - diameter of a point set (see slides)

- Property Testing: approximation for decision problems
  - "sortedness" of a list
  - connectedness of a graph
Testing "sortedness" of a list

**Definition**
A list of size $n$ is $\epsilon$-close to sorted if you can delete at most $\epsilon n$ elements to get a sorted list.

**Requirements of property testing algorithm**:

- **Input**: $\epsilon$, list $y_1 \ldots y_n$
- **Output**:
  - If $y_1 \leq y_2 \leq \ldots \leq y_n$, output "PASS"
  - If $y_1 \ldots y_n$ is not $\epsilon$-close to sorted, output "FAIL" with prob $\geq \frac{3}{4}$

**Note**: $p \geq q$ is equivalent to $7q \geq 7p$

So this is the same as:

> "If likely to output PASS (with prob $\geq \frac{1}{4}$) on this list, then list must be $\epsilon$-close to sorted."

**Comment**
We didn't specify what should happen if $y_i$'s are not sorted, but not far, either output is reasonable, but this leeway allows major speedup!
- Two tests that don't work well:
  1) Pick random i & test whether y_i < y_{i+1}
  2) Pick random i<j & test whether y_i < y_j

  why not?

- Wlog, assume list distinct (i.e., x_1 < x_2 < x_3 ...)
  (if not, append index to each element)
  \( x_1 ... x_n \Rightarrow (x_1, 1)(x_2, 2) ... (x_n, n) \)
  breaks ties without changing order

- The test:

  Do \( O(V/e) \) times:
  - Pick random i
  - look at y_i's value & do binary search on list of y_i's for this value
  - If don't end up at location i or find an inconsistency along
    search path, output "FAIL" & halt

  (If no problems found) Output "PASS"

- Runtime \( O(V/e \log n) \)
Behavior

index $i$ is "good" if binary search for $y_i$

successful.

e.g. 1 4 2 5 7 11 14 19

bin search for 7-19 won't find any inconsistency

$\Rightarrow$ 5-8 are good

bin search for index 1: (value 1)

4 \(\rightarrow\) 5 \(\Rightarrow\) 1 is good

bin search for index 3: (value 2)

2 not found $\Rightarrow$ index 3 is not good

bin search for index 2: (value 4)

2 found $\Rightarrow$ index 2 is good

bin search for index 4: (value 5)

5 found $\Rightarrow$ index 4 is good
Behavior (cont.)

- if list sorted,
  all is good \( \Rightarrow \) list always passes
    \[
    \uparrow
    \]
  uses distinctness

- if list not \( \varepsilon \)-close,
  to show: test fails with prob \( \geq 3/4 \)

Equivalently will show:
If list passes test with prob \( > 1/4 \) then
must be \( \varepsilon \)-close.

Why?
If list passes test with prob \( > 1/4 \)
then \( \geq (1-\varepsilon)n \) i's are good
(if \( > \varepsilon n \) i's are bad,
then in \( \frac{\varepsilon}{2} \) loops will
choose one with prob \( \geq 1 - (1-2)^{\frac{\varepsilon}{2}} \)
\( \geq 1 - e^{-\varepsilon} \)
\( \geq 3/4 \)
then test fails
which contradicts that test passes with prob \( > 1/4 \).
Note: good elements are in the right order!

Claim: if \( i < j \) are both good then \( y_i < y_j \)

Why? let \( k \) be least common ancestor of \( i, j \)

Search for \( i \) went "left" \( \Rightarrow y_i < y_k \)

Search for \( j \) "right" \( \Rightarrow y_k < y_j \)

So \( y_i < y_j \).

So, delete <en bad elements gives sorted list.
Testing "connectedness" of a graph

Given: graph $G$ with $n$ vertices, $m$ edges, max degree $d$, adjacency list representation

$$\boxed{\text{def } G \text{ is } \varepsilon\text{-close to connected} \text{ if adding } \leq \varepsilon d n \text{ edges transforms it to connected}}$$

(today it is ok if transformation violates max degree $d$ requirement)

Property tester requirements:

Input: $\varepsilon$, $G$

Output:
- if $G$ connected, output "Pass"
- if $G$ not $\varepsilon$-close, output "FAIL" with prob $= 3/4$

Idea for tester:

If $G$ far from connected, then
- must have many (23en) connected components
- many small connected components
- many nodes in small connected components
Algorithm

Do $O(\frac{1}{\varepsilon d})$ times:

Pick random node $s$, run BFS from $s$ until

(a) $\leq \frac{2}{\varepsilon d}$ distinct nodes seen \( \leq \) good case, continue

or

(b) see that $s$ is in component of size $\leq \frac{2}{\varepsilon d}$ nodes

\[ \uparrow \text{bad case, output "FAIL" and halt} \]

(If never entered (b)) Output "PASS"

Runtime

$O(\frac{1}{\varepsilon d})$ loops

$O(\frac{1}{\varepsilon d})$ steps of BFS $\times O(d)$ time per step

$\Rightarrow O(\frac{1}{\varepsilon^2 d})$ time

Behavior

Lemma: If $G$ $\varepsilon$-far from connected

then $\exists n \geq \varepsilon d n$ connected components

Proof:

Assume $\leq \varepsilon d n$ components

\[ \uparrow \text{add $\leq \varepsilon d n$ edges connects up graph} \]

\[ \Rightarrow \text{not } \varepsilon \text{-far!} \]
Lemma 2: If \( \geq \frac{ed}{n} \) components
then \( \geq \frac{ed}{n} \) components of size \( \frac{2}{ed} \)

Proof:

\[ L \equiv \# \text{conn comp} \]
\[ L' \equiv \# \text{conn comp of size} \leq \frac{2}{ed} \]
\[ L \equiv \frac{L' + L}{2} \]

Note \( L \cdot \frac{2}{ed} \leq n \) (else too many nodes!)

So \( L \leq \frac{ed}{2} \)

So \( L' = L - L \geq ed - \frac{ed}{2} = \frac{ed}{2} \)

Observation: If \( \geq \frac{ed}{2} \) components of size \( \leq \frac{2}{ed} \)
then \( (\frac{ed}{2}) \cdot n \) nodes in components of size \( \frac{2}{ed} \)

(since each small component has \( \geq 1 \) node)

Putting it together (Lemma 1.2 + Observation):
If \( G \) \( \varepsilon \)-far from connected,
\( \geq \frac{ed}{5} \) fraction of nodes in small components

These reach (6) + FAIL!

So \( \text{Prob}[\text{fail}] \geq 1 - (1 - \frac{ed}{2})^{\frac{4}{ed}} \geq 1 - e^{-\frac{4}{2}} = \frac{3}{4} \) for \( c \) big enough.