Lecture 25

Last time
- Local Computation Algorithms (LCAs)
- Distributed LOCAL model
- Maximal Independent Set (MIS)

Today
- Finish LCA for MIS

Project Reports are due today, presentations next week
**Local Computation Algorithms (LCAs)**

**Motivation:** to have sublinear-time algorithms for problems with long output
- User should be able to ``probe” bits of the output.
- If there are multiple possible outputs, LCA should be giving answers consistent with one.
- The order of the probes should not affect the answers (instantiations of LCA should be able to consistently answer probes in parallel)
- They can have access to the same random string.
- [Rubinfeld, Tamir, Vardi, Xie 11]
Maximal Independent Set (MIS)

For a graph $G = (V, E)$, a set $M \subseteq V$ is a maximal independent set if

- $M$ is independent: $\forall u, v \in M$, the pair $(u, v) \notin E$
- $M$ is maximal: no larger independent set contains $M$ as a subset.

Example:

- MIS can be found in poly time by greedily adding vertices to $M$ and removing them and their neighbors from consideration.
- It is NP-hard to compute a maximum independent set.

Goal: An LCA for MIS

- Given query access to a graph $G$ of maximum degree $\Delta$, provide probe access to an MIS $M$:
  
  \[
  \text{in-MIS}(v): \text{Is } v \text{ in } M?
  \]

Main idea: modify an existing distributed algorithm for MIS.

Based on Ronitt Rubinfeld’s and Sepehr Assadi’s lecture notes
Distributed LOCAL Model

- The input graph is a communication network; each node is a processor.
- In each round:
  - Communication: each vertex can send any message to each neighbor (possibly different messages to different neighbors).
  - Computation: each vertex can decide on its actions for the next round, based on received messages.
- At the end of the last round, each vertex decides on its final status (e.g., whether it is in the MIS $M$).
- Goal: to minimize the number of rounds.

(A variant of) Luby’s MIS Algorithm for the LOCAL Model

1. Initialize $Active(v) = True; M(v) = False$ for all $v \in V$.
2. For each (out of $R$) rounds, all vertices $v$ run the following in parallel:
   a. Vertex $v$ selects itself with probability $\frac{1}{2\Delta}$
   b. Vertex $v$ wins if $v$ is selected, and no neighbor of $v$ is selected
   c. If $v$ won and $Active(v) = True$, then set $M(v) = True$ and $Active(u) = False \ \forall u \in \{v\} \cup N(v)$
## Analyzing the Number of Rounds (New)

### Termination Theorem

Fix \( v \in V \) and round \( R \geq 1 \). Let \( L(v) \) be the event that \( v \) lost in all \( R \) rounds. Then

\[
\Pr[Active(v) = True \text{ after } R \text{ rounds of Luby's algorithm}] 
\leq \Pr[L(v)] 
\leq \exp\left(-\frac{R}{4\Delta}\right).
\]

### Proof:
For each \( v \in V \) and round \( r \geq 1 \), define the following events.

- \( S_r(v) \): the event that \( v \) is selected in round \( r \)
- \( W_r(v) \): the event that \( v \) wins round \( r \), i.e., \( v \) is the only selected vertex in \( \{v\} \cup N(v) \)

\[
\Pr[W_r(v)] = \Pr\left[S_r(v) \land \forall u \in N(v): \overline{S_r(u)}\right]
= \Pr[S_r(v)] \cdot \Pr\left[\forall u \in N(v): \overline{S_r(u)}\right]
\geq \Pr[S_r(v)] \cdot \left(1 - \sum_{u \in N(v)} \Pr[S_r(u)]\right)
\geq \frac{1}{2\Delta} \cdot \left(1 - \Delta \cdot \frac{1}{2\Delta}\right) = \frac{1}{4\Delta}
\]

Events \( S_r(v) \) are independent

By a union bound
### Analyzing the Number of Rounds (New)

**Termination Theorem**

Fix $v \in V$ and round $R \geq 1$. Let $L(v)$ be the event that $v$ lost in all $R$ rounds. Then

$$\Pr[Active(v) = \text{True} \text{ after } R \text{ rounds of Luby's algorithm}] \leq \Pr[L(v)] \leq \exp\left(-\frac{R}{4\Delta}\right).$$

**Proof:**

- $W_r(v)$: the event that $v$ wins round $r$

  - $\Pr[W_r(v)] \geq \frac{1}{4\Delta}$
  - Events $W_r(v)$ are independent for different rounds
  - The probability that $v$ is active after $R$ rounds is at most

    $$\Pr[L(v)] \leq \prod_{r=1}^{R} \Pr[W_r(v)] \leq \left(1 - \frac{1}{4\Delta}\right)^R \leq \exp\left(-\frac{R}{4\Delta}\right).$$

  - If $v$ wins, it is no longer active

**Conclusion:** Set $R = 8\Delta \cdot \ln n$.

- Then a specific vertex remains active after $R$ rounds w.p. at most $1/n^2$
- By a union bound, no vertex remains active w.p. at least $1 - 1/n$
Converting Luby’s MIS Algorithm to LCA

- **Key observation:** What happens to vertex $v$ in $R$ rounds depends only on $R$-hop neighborhood of $v$

- If we simulate Luby’s algorithm for $R = \Theta(\Delta \log n)$ rounds, we need to consider $R$-hop neighborhood of $v$, which takes $\Delta^{\Theta(\Delta \log n)} = \Omega(n)$ time.

- **Idea 1:** Simulate it for $R = \Theta(\Delta \log \Delta)$ rounds instead (no dependence on $n$)

- **Idea 2:** Prove that, at the end, active vertices form small connected components. (We say that the graph is shattered.)

- For each probe $v$, if its MIS status has not been decided (i.e., $v$ is still active) after $R$ rounds, we will find MIS for its connected component deterministically.
**LCA for MIS**

### LubyStatus($(v, R)$)

1. Simulate Luby's algorithm on vertex $v$ for $R$ rounds
2. If $Active(v) = False$ then
3.   if $M(v) = True$, return IN-MIS; otherwise, return NOT-IN-MIS
4.   else return ACTIVE

### Answer Probe in-MIS($(v)$)

1. Set $R = 12\Delta \cdot \ln(2\Delta)$
2. Compute $status \leftarrow \text{LubyStatus}(v, R)$
3. If $status$ is IN-MIS or NOT-IN-MIS, return $status$
4. Otherwise, find the connected component $C_v$ of $v$ as follows:
5.   Run DFS on $v$
6.   For every visited node $u$, compute $\text{LubyStatus}(u, R)$
7.   Continue DFS only on active nodes
8.   Compute lexicographically first MIS of $C_v$ greedily, ordering vertices according to their ID.
9. Return whether $v$ belongs to MIS of $C_v$
**Correctness**

The output is an independent set

- Luby's algorithm maintains an independent set.
- Active vertices are not adjacent to vertices already in MIS.
Correctness

The output is an independent set

- Luby’s algorithm maintains an independent set.
- Active vertices are not adjacent to vertices already in MIS.
- If \( C_u \neq C_v \) then \((u, v) \notin E\), so when we add independent sets for connected components, the resulting set is independent.

The output is a maximal independent set

- Each deactivated vertex that is not in the output \( M \) is adjacent to a vertex in \( M \), so it cannot be added.
- If \( v \) was in a connected component \( C_v \), but is not in \( M \), it cannot be added because \( M \) includes an MIS for \( C_v \).
# Running Time

## Runtime Theorem

W.p. $\geq 2/3$ over random strings, each probe $\text{in-MIS}(v)$ is answered in $\Delta^O(\Delta \cdot \log \Delta) \cdot \log n$ time when the algorithm uses the chosen random string.

## Lemma

For each $v$, it take time $\Delta^O(\Delta \cdot \log \Delta) \cdot |C_v|$ to answer probe $\text{in-MIS}(v)$.

**Proof:** Consider running $\text{LubyStatus}(u, R)$ for some $u \in V$.

- There are at most $\Delta^R$ vertices in the $R$-hop neighborhood of $u$.
- Since $R = O(\Delta \log \Delta)$, the running time is $\Delta^O(\Delta \cdot \log \Delta)$.

To answer probe $\text{in-MIS}(v)$, we might run $\text{LubyStatus}(u, R)$ on nodes in $C_v$ and their neighbors, resulting in time at most

$$\Delta^O(\Delta \cdot \log \Delta) \cdot O(\Delta) \cdot |C_v| = \Delta^O(\Delta \cdot \log \Delta) \cdot |C_v|.$$  

It remains to analyze $|C_v|$. 

Analyzing the Sizes of Connected Components

For each \( v \in V \), define

\[ A(v) : \text{the event that } Active(v) = True \text{ after round } R \]

- By Termination Theorem, for each \( v \in V \),
  \[
  \Pr[A(v)] \leq \exp\left(-\frac{R}{4\Delta}\right) = \exp\left(-\frac{12\Delta \cdot \ln(2\Delta)}{4\Delta}\right) = \frac{1}{8\Delta^3}
  \]

- One difficulty is that events \( A(v) \) are not independent.

For each \( v \in V \), define

\[ L(v) : \text{the event that } v \text{ is a loser (in all } R \text{ rounds)} \]

\[
\Pr[L(v)] \leq \frac{1}{8\Delta^3}, \text{ as before.}
\]

Claim. Events \( L(v) \) are independent for all vertices \( u, v \) at distance at least 3.

- \( L(v) \) is only a function of randomness at \( \{v\} \cup N(v) \)
- Sets \( \{u\} \cup N(u) \) and \( \{v\} \cup N(v) \) are disjoint

Idea: Let \( H \) be the subgraph of \( G \) induced by losers.

We will show: if \( H \) has a large CC then it also has many ``independent” nodes
**Graph** $G^{(3)}$

- Let $d_G(u, v)$ denote the distance from $u$ to $v$ in $G$
- Let $G^{(3)}$ be a graph on nodes $V(G)$ with $(u, v) \in E(G^{(3)})$ iff $d_G(u, v) \geq 3$
- Max degree in $G^{(3)}$ is at most $\Delta^3$
- For $S \subseteq V$, let $G[S]$ denote the induced subgraph of $G$ on $S$

<table>
<thead>
<tr>
<th>Big-Tree Claim</th>
</tr>
</thead>
</table>
If $H[S]$ is connected then $H^{(3)}[S]$ contains a tree with a vertex set $T$ as a subgraph, where $|T| \geq \frac{|S|}{\Delta^2 + 1}$ and $d_H(u, v) \geq 3$ for all nodes $u, v \in T$.

**Proof**: We construct $T$ greedily:

1. Pick an arbitrary $v \in S$
2. Repeat until no node remains in $S$:
3. Move $v$ from $S$ to $T$; remove all $u$ with $d_H(u, v) < 3$ from $S$
4. Pick a new node $v \in S$ such that $d_H(u, v) = 3$ for some $u \in T$

For each node added to $T$, we exclude $\leq \Delta^2$ nodes from its 2-hop neighborhood, so $T$ has the desired size.
Counting Trees in $G^{(3)}$

Tree-Counting Claim

For $s \geq 1$, let $T_s$ denote the set of all $s$-node trees that are subgraphs of $G^{(3)}$.

Then $|T_s| \leq n \cdot (4\Delta^3)^s$.

**Proof:** We enumerate trees in $T_s$ using the following steps.

1. Chose the root. $n$ choices

2. Choose an unlabeled $s$-node rooted tree by choosing its DFS sequence represented as $2(s - 1)$-bit string. $\leq 2^{2(s-1)} \leq 4^s$ choices

3. Label the tree starting from the root in the order given by the DFS sequence. To go from a parent to a child, pick one of $\leq \Delta^3$ neighbors of the parent in $G^{(3)}$ as its child.
The Size of Connected Components

- Let $s = \log \frac{n}{3}$

- Let $\mathcal{T}_s^* = \{ T \subseteq V : |T| = s, G^{(3)}[T] \text{ contains a tree, } d_H(u, v) \geq 3 \ \forall u, v \in T \}$

- The probability that there is a set $T \in \mathcal{T}_s^*$ where all nodes are losers is

$$\leq \sum_{T \in \mathcal{T}_s^*} \Pr[L(T)] \leq |\mathcal{T}_s^*| \cdot \left( \frac{1}{(8\Delta)^3} \right)^s \leq n \cdot (4\Delta^3)^s \cdot \left( \frac{1}{8\Delta^3} \right)^s = n \cdot \frac{1}{2^s} = \frac{1}{3}$$

- But if there are no such trees, all CCs in $H$ have size

$$\leq (\Delta^2 + 1) \log \frac{n}{3} = O(\Delta^2 \log n)$$

- That is, with probability at least $2/3$, each probe takes

$$\Delta^O(\Delta \log \Delta) \cdot O(\Delta^2 \log n) = \Delta^O(\Delta \log \Delta) \cdot \log n$$

- Currently best run time of LCA for MIS is $\Delta^O(\log \log \Delta) \cdot \log n$ [Ghaffari Uitto 19]