Last lecture showed the parallel LDD algorithm of MPX'83:
Computes a \((\beta, O\left(\frac{\log n}{\beta}\right))\) - decomposition for \(0 < \beta < 1\) in \(O(n+m)\) expected work, \(O(\log^2 n/\beta)\) depth whp

\[\Rightarrow \beta m \text{ edges cut in expectation}\]
\[\Rightarrow \text{ Strong diameter of } O\left(\frac{\log n}{\beta}\right) \text{ whp.}\]

This lecture: use LDD to solve other graph problems.

**Graph Connectivity**

Input: Undirected graph \(G(V, E)\)
Output: Connectivity labeling, \(L\).

\[\text{i.e. } L[u] = L[v] \text{ iff } u, v \text{ in the same component in a path between } u \text{ and } v\]

- Sequentially, easy to solve in \(O(n+m)\) work using graph search (BFS/DFS).

\[\text{Can we get a work-efficient, poly-log depth connectivity alg?}\]

- Lot of work on this problem in early days of parallel algorithms.
  - Culminated in a randomized work-efficient algorithm with \(O(\log n)\) depth on PRAM using \(O(m+n)/\log(n)\) processors, i.e. optimal PRAM alg.
  - But the algorithm is extremely complicated and impractical. \(\Rightarrow\) not suitable to implement.

  Several other work-efficient poly-log depth algorithms since then:
  - Halperin/Zwick, Poon-Ramachandran, Cole-Klein-Tarjan
  - Some based on linear-work MST (will see this later).

- Usually use sampling/fitting of edges for work-efficiency and need pretty complex arguments to prove work-efficiency.

This lecture: very simple LDD-based alg that is almost optimal.
LDD-Based Connectivity:

\[ \text{find } CC((V, E), \beta) : \]
\[ L = \text{LDD}((V, E), \beta) \]
\[ G'(V', E') = \text{Contract}((V, L)) \quad \text{// contract all vertices in an LDD cluster into a single vertex; only keep inter-piece edges and remove any duplicates} \]
\[ \text{if } |E'| = 0 \text{ then return } L \]
\[ \text{else} \]
\[ L' = \text{CC}(G', \beta) \quad \text{// recursive call} \]
\[ L'' = \{ L'([u]) | u \in V \} \quad \text{// propagate label information} \]

\[ \text{Claim that the algorithm is correct.} \]

\[ \text{Set } \beta \text{ to a constant } > 0. \]

- Each edge decreases from \( m \) to \( \beta m \) in expectation and the rate of reduction is independent across iterations \( \Rightarrow O(\log_{1/\beta} m) \text{ whp} \)
- Each recursive call is \( O(n^2/\beta) \) depth, \( O(m) \) work where \( m \) is the component edges
  - Depth = \( O(\log_{1/\beta} m \log^2 n/\beta) = O(\log^3 n) \text{ whp} \)
  - Work = \( \sum_{i=1}^{m} \beta^i m = O(m) \text{ in expectation} \)

Details:

- Consider one level of the adjacency list \( m', n' \) be current \( m, n \)

  - LDD: \( O(m' + n') \text{ work, } O(\log^2 n/\beta) \text{ depth whp} \)

  - Contract:
    - filtering out intra-cluster: \( O(m') \text{ work, } O(\log n') \text{ depth} \)
    - removing duplicates: \( O(m') \text{ expected work, } O(\log n') \text{ depth whp using semi-sort} \)

  - Relabeling: \( O(n) \text{ work, } O(\log n') \text{ depth} \)

  - Overall: \( O(m' + n') \text{ expected work, } O(\log^2 n/\beta) \text{ depth whp (in a single level).} \)
Is this algorithm practical? Yes!

⇒ Show/analyze code in ABBS.

Spanner: sparse subgraphs that approximate distances in the original graph.

Let \( G(V, E) \) be undirected, unweighted.

A subgraph \( H \) of \( G \) is a \( k \)-spanner if \( \forall u, v \in V \)

\[ \text{dist}_H(u, v) \leq k \cdot \text{dist}_G(u, v) \]

\( k \) is called the "stretch" factor.

Example: the mesh graph \( M_4 \), e.g. \( M_4 \) below:

\[ \begin{array}{c}
\text{G} = \begin{array}{c}
\text{u} \\
\text{y} \\
\text{v}
\end{array}
\end{array}
\begin{array}{c}
\text{1} \\
\text{1} \\
\text{1}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{u} \\
\text{y} \\
\text{v}
\end{array}
\begin{array}{c}
\text{1} \\
\text{1} \\
\text{1}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\end{array} \]

\[ \text{G} = \begin{array}{c}
\text{u} \\
\text{y} \\
\text{v}
\end{array}
\begin{array}{c}
\text{1} \\
\text{1} \\
\text{1}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{u} \\
\text{y} \\
\text{v}
\end{array}
\begin{array}{c}
\text{1} \\
\text{1} \\
\text{1}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\begin{array}{c}
\text{w}
\end{array}
\end{array} \]

\[ \text{stretch}(H) = 3 \] since \( \text{dist}_H(u, y) = 3 \leq 3 \cdot \text{dist}_G(u, y) = 5 \)

To show a subgraph \( H(V, E') \) is a spanner, sufficient to show stretch bounds for every edge \( e \in E \setminus E' \).

Thm-2: For any undirected graph on \( n \) vertices admits \((2k-1)\)-spanners with \( O\left(n^{1+1/k}\right) \) edges. This is essentially as good as possible (conditioned on EAC, below):

- \( \gamma(n, k) = \max \# \text{edges in } n \text{-vertex graph with girth} \geq k \)

- Unweighted, undirected graphs of girth \( \geq 2k+1 \) do not have subgraphs that are \( k \)-spanners.

If one removes an edge in such a graph, the distance goes from \( 1 \to 2 \)

⇒ Spanner prop. not satisfied
There is a \((4k+1)\)-spanner with \(O(n^{1/k})\) edges that we can construct in \(\omega(n)\) depth.

Spanner \((G_0(V,E), k)\)

\[
\beta \leftarrow \log \left( \frac{n}{2k} \right)
\]

\[
L \leftarrow \text{LDD}(G, \beta)
\]

\[
H \leftarrow \emptyset
\]

- Add all BFS tree edges used in the LDD algorithm to \(H\).
- For each vertex \(v \in V\) where \(v\) is a boundary vertex, add one edge from \(v\) to each adjacent cluster, to \(H\).

return \(H\).

Recall our lemma from last class:

Lemma: For any vertex or midpoint of an edge, the probability that the smallest and \(k\)-th smallest value from \(\{T_u + d(v, u) : v \in V\} \) differ by \(\beta\) is \(\leq (2\beta)^{-k+1}\).

Lemma: In \(\text{LDD}(L, \beta = \log \frac{n}{2k})\), any vertex intersects \(O(n^{1/k})\) other clusters in expectation.

Consider \(u \in V\) with \(u \rightarrow v\)

\[
(a = 2)
\]

Need to set \(\beta = \frac{\log n}{2k}\).
Let $L = \#$ intersecting clusters. We have

$$E[L] = \sum_{l=1}^{\infty} \left( \frac{e^{-2\beta}}{1-e^{-2\beta}} \right)^{l-1} \leq \frac{1}{1-(1-e^{-2\beta})} = e^{2\beta}$$

Plugging in $\beta = \frac{\log(n)}{2k}$, we have $e^{2\beta} = e^{\frac{\log(n)}{k}} = n^{1/k}$.

The number of overall intersecting clusters is $O(n^{1+1/k})$ in expectation and $O(n^{1+1/k})$ boundary edges added in expectation.

---

**Stretch?**

Consider an edge $e$. There are a few cases:

1. $e$ is internal to a cluster.

\[ \gamma \text{ path through center of this cluster that has length } \leq 2\gamma \text{ where } \gamma \text{ is radius of the cluster.} \]

What is $\gamma$? $\gamma = O\left(\frac{\log(n)}{\beta}\right) = O(k) \text{ w.h.p.}$

$$\frac{\log(n)}{\beta} \text{ in expectation } \Rightarrow 2k \text{ in expectation.}$$

$\Rightarrow$ stretch = $4k$ in expectation.

\[ X_i = E \chi(p) \]
\[ E(X_i) = \frac{1}{\beta} \]
\[ \text{Expected value of max } = \ln(n) \]
(2) \( e \) is inter-cluster

- if \((u, v)\) added as the single inter-cluster edge between \(c_1\) and \(c_2\), stretch is 1.
- Otherwise, \(\exists\) some other \((u, v')\) edge that's added.

\[ u \rightarrow v' \rightarrow c_2 \rightarrow v \]  
\[ l \]  
\[ r \]  
\[ r \]  
\[ r \]  
\[ (r = 2k \text{ exp.}) \]

Overall stretch is \(4k+1\) in expectation.

Overall algorithm runs in \(O(m+n)\) work, \(O\left(\log^2 \frac{n}{\beta}\right)\) depth w.h.p.

\[ O(k \log(n)) \]  
\[ \text{depth w.h.p.} \]