6.895 Randomness and Computation April 4, 2008

Lecture 18

Lecturer: Ronitt Rubinfeld, lecture given by Krzysztof Onak *Scribe: Curtis Fonger*

1 Review of Current Knowledge

Let USTCONN be the problem of checking if two nodes s and t in an undirected graph G are connected. STCONN is the same problem for all graphs, including directed graphs.

1. USTCONN $\in$ RL (we've already seen this result in class)
2. STCONN $\in$ NL (non-deterministic log-space)
3. NL $\subset$ L^2 (by Savitch's Theorem)
4. RL $\subset$ L$^{3/2}$ (Saks & Zhou 1999)
5. USTCONN $\in$ L (Reingold in STOC 2005, we will see this result today)
6. There is strong evidence that L = RL (see Reingold, Trevisan, Vadhan 2006), but this is still an open problem.

Definition 1 *We say that a graph is an* (N, D, λ)-graph *if it is a D-regular graph on N nodes such that the absolute values of all eigenvalues but one are bounded by λ.*

Fact 2 *For all $\lambda < 1$, there exists an $\epsilon > 0$ such that for all (N, D, λ)-graphs G and $S \subset G$ such that $|S| < N/2$, $|S \cup \text{Neighborhood}(S)| \geq (1 + \epsilon)|S|$.*

The above fact implies the following.

Fact 3 *Let $\lambda < 1$ be a constant. Each (N, D, λ)-graph has $O(\log N)$ diameter.*

2 Proof Idea

We apply several graph operations to the input graph G and turn every component of the graph into a (N_i, D, λ)-graph, where N_i is the final number of nodes in the i-th component after the transformation. Let G' be the transform ed graph. It holds that s and t are connected in G if and only if some nodes s' and t' in G' that correspond to s and t, respectively, are connected in G'. G' has a number of nodes which is polynomial in the number of nodes of G, and hence, by Fact 3, it suffices to explore all paths of logarithmic length that start at s' to check if s' and t' lie in the same connected component. Obviously, we cannot compute and store G' explicitly. Nevertheless, it turns out that in small space we can compute edges of G' whenever we need them to explore G'.

3 Graph Operations

3.1 Operation 1: Squaring

If we have a D-regular graph $G = (V, E)$, then we define the D^2-regular graph G^2 as follows: $G^2 = (V, E')$, where $E' = \{(v, w) : \text{ there exists } t \text{ where } (v, t), (t, w) \in E\}$. Every edge in G^2 is a length-2 path in G (and we drop all length-1 paths in G).

Lemma 4 *If we have an (N, D, λ)-graph G, then G^2 is a (N, D^2, λ^2)-graph.*

Proof If M is a random walk matrix of G, then for all eigenvectors v perpendicular to $[1, 1, ..., 1] = u$:

$$\|vM^2\| \leq \lambda\|vM\| \leq \lambda^2\|v\|.$$

■

3.2 Operation 2: Zig-zag Product

We define the zig-zag product $G \textcircled{z} H$, a D^2-regular graph, as the following: if we have $H = (V_2, E_2)$ (a D-regular graph) and $G = (V_1, E_1)$ (a $|V_2|$-regular graph), then:

- Replace each node of G with a copy of H (thin edges).
- Associate each node of H with one edge of G (thick edges). That is, for each edge $e = (v, w)$ in G, there is a thick edge that connects a node in the copy of H that replaces v to a node in the copy of H that replaces w, and moreover, each node in the new graph graph is incident to at most one thick edge.
- $G \textcircled{z} H = (V_1 \times V_2, E_3)$, where we define each edge of E_3 as the result of 3 steps:
    1. Start at a node, then make one step in a copy of H along a thin edge.
    2. Make one step along a thick edge.
    3. Repeat step 1: make one step in a copy of H along a thin edge.

The following lemma states the properties of the zig-zag product (we omit its proof).

Lemma 5 *If we have a $(N_1, D_1, \lambda_1 = 1 - \gamma_1)$-graph G and a $(D_1, D_2, \lambda_2 = 1 - \gamma_2)$-graph H, then we have that $G \textcircled{z} H$ is a $(N_1 D_1, D_2^2, \lambda_3 = 1 - \gamma_1\gamma_2^2)$-graph.*

3.3 Algorithm

The following fact holds.

Fact 6 *There exists $D > 1$ and a graph H such that H is a $(D^4, D, 1/4)$-graph.*

From now on, we assume that H is an arbitrary fixed graph of the above properties. We now describe the algorithm.

1. Reduce the input (G, s, t) to (G_0, s_0, t_0) where G_0 is a D^2-regular, non-bipartite and s_0 and t_0 are connected if and only if s and t are connected in G.
2. For each $k = 1, ..., L$, where $L = O(\log N)$:
    (a) $G_k = G_{k-1}^2 \textcircled{z} H$
    (b) Let s_k (and t_k) be any vertex in the copy of H that replaces s_{k-1} (t_{k-1}, respectively) in G_k.
3. Try all paths in G_L of length $O(\log N)$ that start from s_L, and accept the input if any visits t_L.

To see how the graph operations change the degree and the number of nodes, note that:

- G_0 is a D^2-regular graph on N' nodes,
- G_0^2 is a D^4-regular graph on N' nodes,
- $G_1 = G_0^2 \textcircled{z} H$ is a D^2-regular graph on $D^4 \cdot N'$ nodes.

Hence in general, G_i is a D^2-regular graph on $D^{4i} \cdot N'$ nodes. The following fact is relatively easy to prove.

Fact 7 *For any non-bipartite, undirected graph G with N nodes, we have that*

$$\lambda(G) \leq 1 - \frac{1}{\mathrm{poly}(N)}.$$

It says that the initial spectral gap of each component of the graph is large enough to be amplified in $O(\log N)$ graph operations to at least a constant. Let C_0 be any connected component in G_0, and then let C_{i+1} be the component corresponding to C_i in G_{i+1}. We have

$$\gamma(C_0) \geq \frac{1}{\mathrm{poly}(N)}.$$

By the properties of squaring,

$$1 - \gamma(C_{k-1}^2) \leq (1 - \gamma(C_{k-1}))^2,$$

and hence,

$$\gamma(C_{k-1}^2) \geq 2\gamma(C_{k-1}) \left(1 - \frac{1}{2}\gamma(C_{k-1})\right).$$

Finally,

$$\begin{aligned}\gamma(C_k) = \gamma(C_{k-1}^2 \text{ⓩ} H) \geq \gamma(H)^2 \cdot \gamma(C_{k-1}^2) \\ &\geq \left(\frac{3}{4}\right)^2 \cdot 2 \cdot \gamma(C_{k-1}) \cdot \left(1 - \frac{1}{2}\gamma(C_{k-1})\right) \\ &\geq \min\left\{\frac{17}{16}\gamma(C_{k-1}), \frac{1}{18}\right\}.\end{aligned}$$

We have the last inequality because we have two cases: if $\gamma(C_{k-1}) \leq \frac{1}{18}$, then $\gamma(C_k) \geq \frac{17}{16}\gamma(C_{k-1})$, and if $\gamma(C_{k-1}) \geq \frac{1}{18}$, then $\gamma(C_k) \geq \frac{1}{18}$. This implies that after $L = O(\log N)$ iterations we must have $\gamma(C_L) \geq \frac{1}{18}$.

4 Implementation

Why can this algorithm be implemented in logarithmic space? Because:

- We keep the current path throughout the algorithm. We only need $O(1)$ bits per each step, since G' has bounded degree. Besides, the length of the path is $O(\log N)$.
- When going back, we can compute the location from scratch.
- We can compute G_0 in logarithmic space and use it as a subroutine.
- We can compute each G_k recursively with $O(1)$ space per recursion level. We omit the details here.

5 Open Problem

The following challenging question remains open: RL = L?