We consider the undirected connectivity problem. Given an undirected graph $G$ represented by an adjacency matrix and two vertices $u$ and $v$, the undirected connectivity problem is to decide whether there is a path from $u$ to $v$. Formally we define the language $USTCON$ as follows.

**Definition 9.1.** $USTCON$ is defined as a set of triples $(G, s, t)$ where $G = (V, E)$ is an undirected graph, $s, t$ are two vertices in $G$ so that there is a path from $t$ to $t$ in $G$.

This problem has received a lot of attention in the past few decades and the complexity of $USTCON$ has been well studied. The first randomized log-space algorithm for $USTCON$ was shown in 1979 by Aleliunas, Karp, Lipton, Lovász and Rackoff. In 1970, Savitch demonstrated a simulation of a non-deterministic space $S$ machine by a deterministic space $S^2$ machine. Thus $USTCON \in \text{SPACE}(\log^2 n)$. Nisan, Szemerdi and Wigderson in 1989 showed that $USTCON \in \text{SPACE}(\log^{3/2} n)$. Armoni, Ta-Shma, Wigderson and Zhou in 2000 proved that $USTCON \in \text{SPACE}(\log^{4/3} n)$. In 2005, Reingold presented a log-space algorithm for solving $USTCON$. Since $USTCON$ is complete for the class $\text{SL}$ of problems solvable by symmetric, non-deterministic, log-space computation, this result implies $\text{SL} = \text{L}$.

It is easy to see that $USTCON$ can be solved in linear-time using breadth-first or depth-first search. Moreover, the theorem below shows that we can solve $USTCON$ in $O(\log^2 n)$ space.

**Theorem 9.2.** There is an algorithm deciding $USTCON$ using $O(\log^2 n)$ space.

**Proof.** We design the recursive procedure $\text{IsPath}(G, u, v, k)$ which decides if there is a path between $u$ and $v$ of length at most $k$. The algorithm description is as follows:

- If $k = 0$, accept if $u = v$;
- If $k = 1$, accept if $u = v$ or $(u, v)$ is an edge in $G$;
- Otherwise, loop through all vertices $w$ of $G$ and accept if both $\text{IsPath}(G, u, w, \lceil k/2 \rceil)$ and $\text{IsPath}(G, v, w, \lfloor k/2 \rfloor)$ accept for some $w$.

Hence we can solve the $USTCON$ problem by running $\text{IsPath}(G, s, t, n)$. The algorithm uses $\log n$ levels and $O(\log n)$ bits in every level to store the vertex $w$. Therefore the space complexity is $O(\log^2 n)$.

1 Algorithm

We first give the intuitions behind the algorithms. Two main insights are: (1) $USTCON$ can be solved in log-space on constant-degree graphs in which every connected-component is an expander. Since every expander graph has logarithmic diameter, it is enough to enumerate all logarithmical paths starting from $s$ and to see if one of these paths visits $t$. (2) Any graph can be reduced to constant-degree expanders in logarithmic space.

More precisely, the algorithm reduces the input $G$ to an expander $G_\ell$ such that

- The size of $G_\ell$ does not increase too much, i.e. $|V[G_\ell]| = \text{poly}(|V[G]|)$.
• \( G_\ell \) is regular and the degree of \( G_\ell \) is constant.
• For any two vertices \( u \) and \( v \) in \( G \), \( u \) and \( v \) are connected if and only if the vertices in \( G_\ell \) that correspond to \( u \) and \( v \) are also connected.
• Each connected component of \( G_\ell \) is an expander. (The spectral expansion is at most 1/2.)

Therefore for any two vertices \( u \) and \( v \) in \( G \), \( u \) and \( v \) are connected if and only if there is a path of length \( O(\log |V(G)|) = O(\log |V|) \) to connect the vertices in \( G_\ell \) that correspond to \( u \) and \( v \).

In the preprocessing step, we would like to transform the input graph \( G \) into a \( D^{16} \)-regular graph \( G_1 \) and transform \( s, t \in V[G] \) into vertices \( s_1, t_1 \in V[G_1] \) such that \( s, t \) are connected if and only if \( s_1, t_1 \) are connected in \( G_1 \). Now let \( G_1 \) be a \( D^{16} \)-regular graph on \([n]\) and \( H \) is a \((D^{16}, D, 1/2)\)-graph. The existence of such graphs is proven by probabilistic methods and for a constant \( D \), we can find \( H \) by exhaustive search in constant time (since \( D \) is constant). Moreover, we can express \( H \) by the rotation map in constant time.

Let \( \ell \) be the smallest integer such that \((1 - \frac{1}{D^n})^{2^\ell} \leq 1/2 \). The algorithm is as follows.

• For \( \ell = 1 \) to \( \ell = O(\log |V(G)|) \) do \( G_{i+1} = (G_i \otimes H)^8 \)
• Check if \( s \) and \( t \) are connected in \( G_\ell \) by enumerating over all paths of length \( O(\log n) \) originating at \( s \).

Note that each \( G_i \) is a \( D^{16} \)-regular graph over \([n] \times ([D^{16}])^i \). Since \( D \) is constant and \( \ell = O(\log n) \), \( G_\ell \) has \( \text{poly}(n) \) vertices.

## 2 Analysis

The working space of the algorithm depends on two things: The space for calculating \( G_i \) iteratively and the space for deciding the connectivity between \( s \) and \( t \) in \( G_\ell \).

Now assume that the input graph \( G \) is connected and we prove that \( G_\ell \) is an expander.

**Lemma 9.3.** Let \( G \) be a \( d \)-regular, connected, non-bipartite graph with \( n \) vertices. Then \( \lambda(G) \leq 1 - 1/D \cdot n^2 \).

**Theorem 9.4.** If \( \lambda(H) \leq 1/2 \), then \( 1 - \lambda(G \otimes H) \geq 1/3 \cdot (1 - \lambda(G)) \).

**Theorem 9.5.** For \( i = 2, \ldots, \ell \), we have \( \lambda(G_i) \leq \max \{ \lambda^2(G_{i-1}), 1/2 \} \).

**Proof.** Since \( G_i = (G_{i-1} \otimes H)^8 \), by Theorem 9.4 we have

\[
\lambda(G_i) = \lambda^8(G_{i-1} \otimes H) \leq \left( 1 - \frac{1}{3} \cdot (1 - \lambda(G_{i-1})) \right)^8.
\]

We consider the following two cases.

1. \( \lambda(G_i) \leq 1/2 \). Then
   \[
   \lambda(G_i) = \lambda^8(G_{i-1} \otimes H) \leq \left( 1 - \frac{1}{3} \cdot (1 - \frac{1}{2}) \right)^8 \leq \left( \frac{5}{6} \right)^8 \leq \frac{1}{2}.
   \]

2. \( \lambda(G_i) > 1/2 \). Because for any \( x \in [1/2, 1] \) it holds that
   \[
   \left( 1 - \frac{1}{3} \cdot (1 - x) \right)^4 \leq x,
   \]
   we have
   \[
   \lambda(G_i) = \lambda^8(G_{i-1} \otimes H) \leq \left( 1 - \frac{1}{3} \cdot (1 - \lambda(G_{i-1})) \right)^8 \leq \lambda^2(G_{i-1}).
   \]

Therefore for any \( i \in \{2, \ldots, \ell\} \), \( \lambda(G_i) \leq \max \{ \lambda^2(G_{i-1}), 1/2 \} \). \( \square \)
Corollary 9.6. The spectral expansion of each connected component of $G_\ell$ is at most $1/2$.

Proof. By Lemma 9.3 and Theorem 9.5. \qed

Lemma 9.7. For every constant $D$, the transformation of $G_i$ can be computed in space $O(\log n)$ on inputs $G$ and $H$, where $G$ is a $D^{16}$-regular graphs on $[n]$ and $H$ is a $D$-regular graph on $[D^{16}]$.

Note that we cannot generate the whole graph $G_\ell$ off-line because of the memory restriction. Instead of that, we require the expander graphs constructed by the Zig-Zag product to be very explicit. We will skip this in our course.

Theorem 9.8. $\textsc{USTCON} \in \text{L}$.

Since $\textsc{USTCON}$ is complete of $\text{SL}$, an logarithmic-space algorithm for $\textsc{USTCON}$ implies $\text{SL} = \text{L}$. Given this result, the current view of log-space complexity classes is

$$L = \text{SL} \subseteq \text{RL} \subseteq \text{NL} \subseteq L^2.$$ 

As mentioned in Reingold’s paper on $\text{SL} = \text{L}$, a very natural question is whether the technique of proving $\text{SL} = \text{L}$ can be used towards a proof of $\text{RL} = \text{L}$. So far, the best deterministic simulation known for $\text{RL}$ is $\text{DSPACE}(\log^{3/2} n)$, which is based on the pseudorandom generators for log-space computation.

Appendix

Definition 9.9. The complexity class $\text{L}$ consists of the language decidable within deterministic logarithmic space.

Definition 9.10. $\text{SL}$ is the class of problems solvable by a nondeterministic Turing machine in logarithmic space, such that:

1. If the answer is ‘yes’, one or more computation paths accept.
2. If the answer is ‘no’, all paths reject.
3. If the machine can make a nondeterministic transition from configuration $A$ to configuration $B$, then it can also transition from $B$ to $A$. (This is what ‘symmetric’ means.)