Today

- Quick reminder of STUCONN.
- History of the complexity of STUCONN.
- Savitch’s algorithm.
- Zig-zag product reminder.
- Reingold’s log-space algorithm for STUCONN.
ST-UCONN

- Undirected $s, t$, connectivity ST-UCONN:
  - We are given undirected graph $G(V, E)$ two vertices $s, t \in V$.
  - The question is whether $s$ and $t$ are connected in $G$. I.e. does there exist a path from source $s$ to the target $t$?
ST-UCONN

- STUCONN $\in NL$.
- Not known to be complete for NL, probably not, but complete for class SL (symmetric, non-deterministic TM with $O(\log n)$ space).
- Non-deterministic TM is symmetric if whenever transition $s-s'$ possible, so is $s'-s$. 
Savitch’s theorem

- **Theorem.** If A is a problem that can be solved non-deterministically in space \( s(n) \geq \log n \), then it can be solved deterministically in space \( O(s^2(n)) \).
History of space complexity of STUCONN

- STUCONN $\in NL \Rightarrow$ STUCONN $\in SPACE(\log^2 n)$
- Exponentially better space than depth-first search, no longer poly time.
- Time required by Savitch’s algorithm is super-poly.
- A few lectures ago, we saw STUCONN $\in RL$. 
Theorem. If $A$ is a problem that can be solved by a randomized TM in space $s(n) \geq \log n$, then it can be solved deterministically in space $O(s^{3/2}(n))$.

STUCONN $\in RL \Rightarrow$ STUCONN $\in SPACE(\log^{3/2} n)$.

Later improved to $SPACE(\log^{4/3} n)$. 

Saks and Zhou theorem
Savitch’s deterministic simulation

- Squaring improves connectivity.
- Define $G^{sq}$ to be the graph that has an edge $(u,v)$ if $u$ and $v$ are connected in $G$ by a path of length at most 2.
- Not the same as $G^2$ (paths of length 2 between $u$ and $v$).
- If $u$ and $v$ are connected by some path in $G$, then $(u,v)$ is an edge in $G^{sq\log n}$. Savitch gave an $O(\log^2 n)$ algorithm for computing $G^{sq\log n}$ from $G$. 
Let’s do Savitch’s algorithm with $G^2$. Will also solve STUCONN but NOT in $O(\log^2 n)$ space.

Complexity of computing $G^{2 \log n}$ is huge, but the algorithm will illuminate what follows next.

We use rotation map representation of $G$, for d-regular $G$. 
Savitch’s deterministic simulation

- We use rotation map representation of G, for d-regular G.
- $ROT_G: (u, i) \in V \times [d] \rightarrow (v, j) \in V \times [d]$, if the ith edge from u leads to v and the label of this edge from v’s side is j.
- $ROT_G$ permutation.
- $ROT_G(ROT_G(u, i)) = (u, i)$. Can be computed in logn space.
What is the space complexity of computing $ROT_{H^2}$ from $ROT_H$?

Claim: $SPACE(H^2) = SPACE(H) + O(\log \deg(H))$. Where $SPACE(H)$ is the space needed to compute $ROT_H$. 
Savitch’s deterministic simulation

- From claim, we can see that computing the rotation map for $G^n = G^{2\log n}$ has space complexity

  \[ SPACE(G^n) = SPACE\left(G^{2^{\log n - 1}}\right) + O\left(\log \deg\left(G^{2^{\log n - 1}}\right)\right) = \ldots = \sum_{i=1}^{\log n - 1} O\left(\log \deg\left(G^2^i\right)\right). \]

- Degree grows exponentially. If it stayed constant with squaring, we would have a SPACE($\log n$) algorithm.
Zig-Zag product

- Advantage of squaring is that it increases expansion, thus connectivity.
- Can we obtain the same effect without increasing the degree?
- YES! Last time: alternating between squaring and zig-zag product achieves that.
Zig-Zag product recap.

Say that $G$ is a $(N, d, \lambda)$-graph if $G$ is $d$-regular on $N$ vertices and the spectral expansion (normalized second eigenvalue of adjacency matrix) is $\lambda$. 
Zig-Zag product recap.

- Zig-Zag product from last time: The zig-zag product between an \((N, D_1, \lambda_1)\)-graph \(G\) and an \((D_1, D_2, \lambda_2)\)-graph \(H\), is a graph denoted \(G(z)H\), with the following properties
  1. \(G(z)H\) has \(ND_1\) vertices.
  2. \(G(z)H\) is a \(D_2^2\)-regular graph.
  3. \(\text{ROT}_{G(z)H}((u, i), (a_1, a_2)) = ((v, j), (b_1, b_2))\) if there exist \(i', j' \in [D_2]\) such that \(\text{ROT}_H(i, a_1) = (i', b_1), \text{ROT}_G(u, i') = (v, j')\) and \(\text{ROT}_H(j', a_2) = (j, b_2)\)
Zig-Zag product recap.

- We showed that if $G$ is $(N_1, d_1, \lambda_1)$ — expander and $H$ is $(d_1, d_2, \lambda_2)$ — expander, then $G(z)H$ is an $(d_1 N_1, d_2^2, \lambda_1 + \lambda_2 + \lambda_2^2)$ — expander.
- We used that result when $G$ and $H$ are good expanders.

**Lemma 1.** If $G$ is connected and non-bipartite it has expansion $\lambda \leq 1 - \frac{1}{dn^2}$. 
Zig-Zag product recap.

- Zig zag product still useful if H is good expander.
- **Lemma 2.** If \( \lambda(H) \leq \frac{1}{2} \) then \( 1 - \lambda(G(z)H) \geq \left( 1 - \frac{1}{3} \lambda(G) \right) \).

- Lemma 2 says that if H is good expander then the zig zag product only reduces the spectral gap by a factor of 3.
Reingold’s algorithm

- Idea is to alternate squaring with zig zag. Squaring improves spectral gap from $1 - \lambda$ to $1 - \lambda^2$ while zig zag deteriorates the gap by at most $1 - \frac{\lambda}{3}$.
- We start with a constant size graph $H$, which is a $\left(d^{16}, d, \frac{1}{2}\right)$—expander.
- Assume the graph $G$, which we want to solve STUCONN on is $d^{16}$-regular non-bipartite. Remove assumptions later.
Reingold’s algorithm

- Algorithm will work independently for each connected component and checks if \( t \) is in the same component as \( s \). Since every component is \( d^{16} \)-regular, connected and non-bipartite, we have from Lemma 1 that 
  \[
  1 - \frac{1}{d^{16} n^2} \geq \lambda(C),
  \]
  for all components \( C \) of \( G \).
Reingold’s algorithm

- Checking connectivity on an expander can be done in log space.
- Follows from the fact that the distance between any two vertices is $O(\log n)$.
- Suffices to enumerate all the $O(\log n)$ paths in $G$ starting at $s$ and check if any leads to $t$.
- Will transform $G$ into $G'$ s.t. each connected component is an expander (spectral expansion less than $1/2$) and if two vertices are connected in $G$ they are also connected in $G'$. 