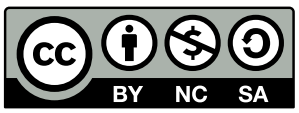
CS289a: Great Theory Hits of 21st Century

Tingfeng Xia

Winter 2023

Notes reorganized from https://hackmd.io/@raghum/greathits.



Contents

1 Undirected s-t Connectedness **5**
1.1 Computing Resources 5
1.2 Problem Statement 5
1.3 Randomized Algorithm for Connectivity 6
1.4 Log Space USTCON 6
1.5 Spectral Graph Theory 6
1.6 Path Enumeration 9
1.6.1 Reingold's Idea 9
1.6.2 Reducing Degree 10
1.6.3 Improving Spectral Gap 10

Chapter 1

Undirected s-t Connectedness

1.1 Computing Resources

Four main computing resources that we consider as limited (and measure the performance of our algorithms against)

- Time
- Memory
- Randomness
- Communication

1.2 Problem Statement

- **Input**: Graph $G = (V, E)$; with source and target marked as s, t
- **Output**: YES iff s and t are connected, NO otw.

Above is the "traditional" definition of $s-t$ connectivity which we can solve with a vanilla BFS or DFS. This will take $O(|V| + |E|)$ and $O(|V|)$ extra bits of space / memory. The question is then, can we solve the same problem with sub-linear extra memory usage.

Proposition 1.1 *There is a randomized algorithm with* $5 \log |V|$ *bits of additional memory (directed and undirected graphs).*

Proposition 1.2 (Omer Reingold, 2005) *There is a log space* ($O(\log |V|)$) *algorithm* ***(deterministic)*** *for undirected graphs.* [1.2.1]

It is yet unknown if we can achieve log space for directed graphs (with deterministic algorithm). The best known algorithms runs with $O(\log |V|)^{3/2}$ bits of memory. Why is this so challenging?

Proposition 1.3 *If divided* $s-t$ *connectivity can be solved with* $O(\log |V|)$ *extra bits of memory (without randomness), then any randomized algorithm can be made deterministic at the expenses of a constant factor increase in memory.*

[1.2.1] first great hit ...

1.3 Randomized Algorithm for Connectivity

Algorithm 1.1 (Random Walk Algorithm for Connectivity) *Here is the algorithm*

- *steps* $\leftarrow 0$
- *current* $\leftarrow s$; *target* $\leftarrow t$
- *while steps* $< T$
  - *current* $\leftarrow$ *random neighbor of current*
  - *if current* $==$ *target return YES*
- *return NO*

The total memory for this algorithm is

$$2\log N + \log T \leq 5\log N \tag{1.3.1}$$

extra bits, assuming we can get random neighbor.

Proposition 1.4 (Alenilaus, 80s) *If* $T = 100N^3$ *steps, then Pr[Algorithm wrong]* $< \frac{1}{3}$

which can improved to arbitrary accuracy by repeating the algorithm. Algorithms of this nature can perform bad on graphs known as "Lollipop Graphs" and even worse a "Dumbell Graph"

1.4 Log Space USTCON

Here we highlight the progression in space complexity in various papers

- **Nisan, 92**: Space $O(\log^2 N)$, time $N^{O(1)}$ algorithm... improved to $O(\log^{4/3} N)$ in space.
- **Reingold, 05**: Space $O(\log N)$, time $N^{O(1)}$ algorithm.
- **Trifornov, 05**: Space $O((\log N)(\log\log N))$ algorithm.

1.5 Spectral Graph Theory

Consider an undirected graph $G = (V, E)$,

Definition 1.1 (Degree) *Degree of a vertex v is the number f edges v is connected to.*

Definition 1.2 (Regular) *Graphs is "regular" if all vertices have same degree.*

Definition 1.3 (Adjacency Matrix) *$A(G)$ is a symmetric matrix where $A(G)_{ij} = 1$ if $\{i, j\}$ is an edge, 0 otw.*

Definition 1.4 (Normalized Adj Matrix) *If G is regular and has degree D, then the normalized adjacency matrix is defined as*

$$M(G) \equiv \frac{A(G)}{D} \tag{1.5.1}$$

Lemma 1.1 *If G is regular, then* 1 *is an eigenvalue of $M(G)$. And $\mathbf{v}_1 = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^\top$ is an eigenvector with eigenvalue 1.*

Proposition 1.5 (Eigenvalues of Regular Graphs) *If G is regular, then all eigenvalues of $M(G)$ have magnitude ≤ 1.*

Proof: WLOG assume x_3 is the largest entry in the vector $\mathbf{x}$, then

$$\begin{aligned} \lambda|x_3| &= |M_{31}x_1 + M_{32}x_2 + ... + M_{3N}x_N| && (1.5.2)\\ &\leq M_{31}|x_3| + M_{32}|x_3| + ... + M_{3N}|x_3| && (1.5.3)\\ &= (M_{31} + ... + M_{3N})|x_3| && (1.5.4)\\ &= 1|x_3| && (1.5.5) \end{aligned}$$

Thus, $\lambda \leq 1$. ■

Proposition 1.6 (Connectedness and Matrices) *Regular $G = (V, E)$ is connected if and only if the only eigenvector with eigenvalue 1 for $M(G)$ is the all 1 vector.* Might appear on exam 1

Proposition 1.7 (Eigenvalues of a Regular Graph) *If G is regular, then the eigenvalues of $M(G)$ are* Might appear on exam 1

$$1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \quad (1.5.6)$$

Proposition 1.8 *G is connected and regular if and only if* Might appear on exam 1

$$\max(|\lambda_2|, |\lambda_3|, \dots, |\lambda_N|) \leq 1 \quad (1.5.7)$$

Proposition 1.9 (Eigenvalues of D-Regular Graphs) *If G is a D-regular graph, then*

- *1 is an eigenvalue of $M(G)$, and*
- *all eigenvalues of $M(G)$ are at most 1 in absolute value*

Definition 1.5 (Self Loops) *We add connections from each node in the graph to themselves. In the matrix representation, we set $G_{ii} = 1, \forall i$.*

Definition 1.6 (Second Largest Eigenvalue) *... denoted as $\lambda(G)$ or $\lambda_2(G)$.*

Lemma 1.2 *If G is D-regular and **has self loops**, then G is connected if and only if $\lambda(G) < 1$.*

Proof: ***We first show G is disconnected implies $\lambda(G) = 1$ (via contrapositive).*** Consider a graph G such that it is comprised of two clouds of disjoint graphs G_1 and G_2. Then the adjacency matrix of G will take a block matrix form

$$M_G = \begin{bmatrix} M_{G_1} & [\mathbf{0}] \\ [\mathbf{0}] & M_{G_2} \end{bmatrix} \quad (1.5.8)$$

From linear algebra, we know that the eigenvalues of M_G will be the union of eigenvalues of M_{G_1} and M_{G_2}. Now, consider

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x}^{(2)} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \quad (1.5.9)$$

are both eigenvectors of M_G with eigenvalues of 1. Hence, there are two orthogonal eigenvectors with eigenvalue 1, and $\lambda(G) = 1$ as wanted. ■

We now show that if G is connected, then $\lambda(G) < 1$. We already know that 1 is an eigenvalue of M_G with the $\mathbf{1}$ vector as eigenvector. Suppose λ is also an eigenvalue with $\mathbf{v}$ as an eigenvector and $\mathbf{v}$ is perpendicular to $\mathbf{1}$. Now,

$$\mathbf{1} \perp \mathbf{v} \implies \langle \mathbf{v}, \mathbf{1} \rangle = v_1 + v_2 + \cdots + v_N = 0 \tag{1.5.10}$$

The vector $\mathbf{v}$ must contain some positive entries and some negative entries, we separate them into two sets

$$P = \{i : v_i \geq 0\} \quad \text{and} \quad N = \{i : v_i < 0\} \tag{1.5.11}$$

where both sets are non-empty by Eq. 1.5.10. Taking a step back and reorganize the goal into matrix form

$$M_G \begin{bmatrix} + \\ + \\ \vdots \\ - \\ - \end{bmatrix} = \lambda \begin{bmatrix} + \\ + \\ \vdots \\ - \\ - \end{bmatrix} \qquad \text{where} \quad \begin{bmatrix} + \\ + \\ \vdots \\ - \\ - \end{bmatrix} = \begin{bmatrix} \mathbf{P} \\ - \\ \mathbf{N} \end{bmatrix} = \mathbf{v} \tag{1.5.12}$$

Per element,

$$\sum_{j=1}^{N} M_G[i,j] \cdot v_j = \lambda \cdot v_i, \quad \forall i \tag{1.5.13}$$

By the connectedness assumption, there must always be some edge connecting P and N the two sets, so

$$\lambda \left(\sum_{i \in P} v_i \right) = \sum_{i \in P} \left(\sum_{j=1}^{N} M_G[i,j] \cdot v_j \right) \tag{1.5.14}$$

$$= \sum_{j=1}^{N} v_j \sum_{i \in P} M_G[i,j] \tag{1.5.15}$$

$$= \sum_{j \in P} v_j \left(\sum_{i \in P} M_G[i,j] \right) + \sum_{j \in N} v_j \left(\sum_{i \in P} M_G[i,j] \right) \tag{1.5.16}$$

$$\leq \sum_{j \in P} v_j (1) + \sum_{j \in N} v_j \left(\sum_{i \in p} M_G[i,j] \right) \tag{1.5.17}$$

$$< \sum_{j \in P} v_j \tag{1.5.18}$$

where in the last two steps we utilized the facts that M_G's columns add up to 1 and we have at least 1 non-zero entry in each row and col of M_G.

In summary, we obtained

$$\lambda \left(\sum_{i \in P} v_i \right) < \left(\sum_{j \in P} v_j \right) \quad \implies \quad \lambda < 1 \tag{1.5.19}$$

■

Definition 1.7 (Spectral Gap) *Spectral Gap of a D-regular graph G is defined as*

$$\text{Spectral Graph} \equiv 1 - \lambda(G) \tag{1.5.20}$$

Lemma 1.3 *If G is a D-regular connected graph with self-loops, then*

$$\lambda(G) \leq 1 - \frac{1}{2D^2 \cdot N^2} \tag{1.5.21}$$

Definition 1.8 *We say a graph G is (N, D, λ) if it has N vertices, D regular and $\lambda(G) \leq \lambda$.*

1.6 Path Enumeration

The simplest case is wen the shortest path between s, t is short. Then, we can enumerate all paths of some length and see if t is reached.

The algorithm goes as follows

Algorithm 1.2 *1, Explore all paths of length less than or equal to T from s. 2, If you reach t in these explorations, output YES. If not, output NO.*

This takes $O(\log D) \cdot T$ extra space, where D is the degree of the graph and T is the loop times.

Definition 1.9 (Graph Diameter) *Diameter of a graph is defined as the length of the longest shortest path for any pair of vertices. By convention,*

- *G disconnected, diameter = ∞, and*
- *G connected, diameter = $\max_{i \neq j} (ShortestPath(i, j))$*

Proposition 1.10 (Extra Space for Path Enumeration) *Path enumeration will solve the $s - t$ connectivity in with max extra space*

$$(\log D) \cdot \Delta(G) \tag{1.6.1}$$

bits, where $\Delta(G)$ is the max diameter of connected components of G.

Proposition 1.11 *If G is connected, D-regular, has self-loops, then*[1.6.1]

$$Diameter(G) \leq \lceil \log_{\frac{1}{\lambda}} N \rceil + 1 \tag{1.6.2}$$

1.6.1 Reingold's Idea

We see from the proposition above that the bigger the spectral gap, the smaller the number of extra bits we need in space for the algorithm. The problem then is how we can transform the graph enlarging the spectral gap while not hurting the degree too much. Formally, we want to transform (G, s, t) to $(\bar{G}, \bar{s}, \bar{t})$ such that

- s, t connected in G if and only if $\bar{s}, \bar{t}$ connected in $\bar{G}$, and
- $\lambda(\bar{G}) < \lambda(G)$, and
- $Degree(\bar{G})$ is not much worse then $Degree(G)$

[1.6.1] λ is the second largest eigenvalue, N is the matrix size (number of nodes).

1.6.2 Reducing Degree

For the first part of Eq. 1.6.1, we can reduce the degree of any graph with

Algorithm 1.3 (Degree Reduce Procedure) *The procedure,*

- *Break each edge into two vertices, and*
- *Add local edges at each "old" vertices, and*
- *Add self loops to make graph*

Proposition 1.12 *The procedure outlined above generates a degree 4 graph.*

1.6.3 Improving Spectral Gap

Definition 1.10 (Multi-graphs) *A multi-graph is a superset of our old definition of a graph, except we allow repeated edges between nodes. This is represented as values larger than 1 in the adjacency matrix. All definitions are carried over without change: degree, normalized adjacency matrix, and $\lambda(G)$.*

With the degree reducing algorithm in Sec. 1.6.1, we can reduce any graph to a degree of 4. This means Eq. 1.6.1 is now transformed into

$$(\log 4) \cdot \Delta(G) = 2 \cdot \Delta(G) \tag{1.6.3}$$

extra bits of storage. How should we improve

$$\Delta(G) \leq \log_{\frac{1}{\lambda}} N \tag{1.6.4}$$

which is the largest diameter of any connected component?

Idea: Input G, s, t where G has self-loops and transform that into G', s', t' where

$$\lambda(G') \ll \lambda(G) \tag{1.6.5}$$

Goal: Operations to improve (decrease) the second largest eigenvalue.

Definition 1.11 (Squaring the Graph) *Add new edges: if (u, v) and (v, w) are edges, then add an edge (u, w).*

Proposition 1.13 (Adjacency of Squared Graph)

$$A_{G^2} = (A_G)^2 \tag{1.6.6}$$

in matrix representation, and we allow multi-graph in this setting.

Proof:

$$(A_G)^2_{[i,j]} = \sum_{k=1}^{N} (A_G)_{[i,k]} (A_G)_{[k,j]} \tag{1.6.7}$$

■

Proposition 1.14 (Squared Graph Spectral Gap) *If G is a (N, D, λ) graph with self loops, then G^2 is a (N, D^2, λ^2) graph with self loop. Since connected $\implies \lambda < 0, \lambda^2 < \lambda$.*

Theorem 1.4 (Squared Matrix Spectral Decomposition) *M is a symmetric matrix with eigenvalues*

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_N \tag{1.6.8}$$

then, M^2 is a symmetric matrix with the same eigenvectors but with eigenvalues

$$\lambda_1^2, \lambda_2^2, \ldots, \lambda_N^2 \tag{1.6.9}$$

Proof: For M, we have

$$M\mathbf{x} = \lambda\mathbf{x} \tag{1.6.10}$$

Then,

$$M^2\mathbf{x} = \lambda M\mathbf{x} = \lambda^2\mathbf{x} \tag{1.6.11}$$

This concludes the proof. ■

Corollary 1.4.1 (Squared Graph Eigenvalues) *It follows from Thm. 1.4 directly that if $\lambda_1, \ldots, \lambda_N$ are eigenvalues for the original graph matrix M, then the new squared M^2 matrix has the same eigenvectors but with eigenvalues $\lambda_1^2, \lambda_2^2, \ldots, \lambda_N^2$ instead.*

Proposition 1.15 (Normalized Adjacency of Squared Graph) *The normalized graph matrix of G^2, is such that*

$$M_{G^2} = (M_G)^2 \tag{1.6.12}$$

Proof: Recall that

$$M_G = \frac{A_G}{D} \tag{1.6.13}$$

Then,

$$M_{G^2} = \frac{A_{G^2}}{D^2} = \frac{(A_G)^2}{D^2} = \left(\frac{A_G}{D}\right)^2 = (M^G)^2 \tag{1.6.14}$$

This concludes the proof. ■

Proposition 1.16 (Square Graph Does Not Save Memory) *Recall that our initial goal was to save extra memory used. Here with squaring, though we enlarged the spectral gap as desired ($(1-\lambda) \to (1-\lambda^2)$), the degree got larger ($D \to D^2$). In total, extra bits is*

$$(\log D) \cdot \log_{\frac{1}{\lambda}} N \rightsquigarrow \left(\log D^2\right) \log_{\frac{1}{\lambda^2}} N \tag{1.6.15}$$

$$= 2 \cdot \log D \cdot \frac{1}{2} \cdot \log_{\frac{1}{\lambda}} N \tag{1.6.16}$$

$$= (\log D) \cdot \log_{\frac{1}{\lambda}} N \tag{1.6.17}$$

which is exactly what we had before. This suffices as a proof for squaring matrices alone does not bring any memory savings. ■

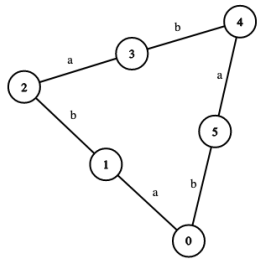


Figure 1.1: Illustration of consistent labelling.

Goal Taking a step back, we can see that we need to find a powering operation that improves the second largest eigenvalue **while not increasing degree too much**. This leads to the following algorithm:

Algorithm 1.4 (Reingold, 2005) *For a graph specified as (G, s, t) where G is 4-regular and has self-loops, define a recursive relationship*

$$G_{i+1} = G_i^2 ⓩ H \tag{1.6.18}$$

where H is a special graph. This recursion covers the transformation

$$(G, s, t) \rightsquigarrow (G_1 = G^2 ⓩ H, \bar{s}, \bar{t}) \rightsquigarrow (G_2 = G_1^2 ⓩ H, \bar{\bar{s}}, \bar{\bar{t}}) \rightsquigarrow \ldots \tag{1.6.19}$$

Remark G_i^2 part decreases the second largest eigenvalue, and the $ⓩ H$ part brings down the degree while not hurting second largest eigenvalue.

Definition 1.12 (Consistent Labelling) *G is a D-regular graph. A consistent labelling is a mapping*

$$L : \mathbb{E} \to [D] \tag{1.6.20}$$

such that at each vertex all edges of the vertex have distinct labels.

Example Figure 1.1 depicts a consistent edge labelling of the graph.

Definition 1.13 (Zig Zag Product) ***Input & Output***

$$\left.\begin{aligned} G &: (N, D, -) \\ H &: (D, D_1, -) \end{aligned}\right\} \to G ⓩ H : (ND, D_1^2, -) \tag{1.6.21}$$

Rotations

$$\left.\begin{aligned} Rot_G &: [N] \times [D] \to [N] \times [D] \\ Rot_H &: [D] \times [D_1] \to [D] \times [D_1] \end{aligned}\right\} \tag{1.6.22}$$

$$\to Rot_{G\textcircled{z}H} : [N \cdot D] \times \left([D_1^2]\right) \to [N \cdot D] \times \left([D_1^2]\right) \tag{1.6.23}$$

$$\equiv Rot_{G\textcircled{z}H} : [N \cdot D] \times ([D_1] \times [D_1]) \to [N \cdot D] \times ([D_1] \times [D_1]) \tag{1.6.24}$$

and

$$Rot_{G\textcircled{z}H}\left((v,a),(k_1,k_2)\right) : \tag{1.6.25}$$

$$\to (a', i') \leftarrow Rot_H(a, k_1) \tag{1.6.26}$$

$$\to (w, b') \leftarrow Rot_G(v, a') \tag{1.6.27}$$

$$\to (b, i'') \leftarrow Rot_H(b', k_2) \tag{1.6.28}$$

$$\to output\left((w,b),(k_2,k_1)\right) \tag{1.6.29}$$

English Explanation *The Zig-Zag product $G\textcircled{z}H$ replaces each vertex of G with a copy (cloud) of H, and connects the vertices by moving a small step (zig) inside the cloud, followed by a big step between two clouds, and finally performs another small step (zag) inside the destination cloud.*