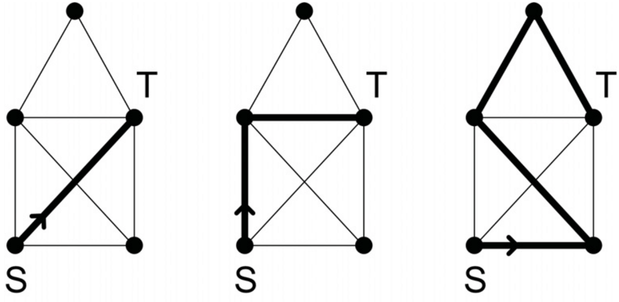
UPATH is in L

Anish Hebbar, Shravan Mehra, Prashant Gokhale

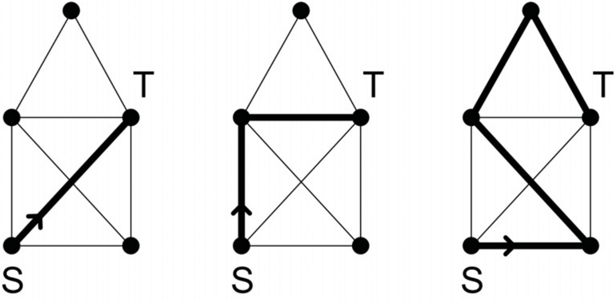
November 25, 2022

- Consists of (G, s, t) , where $G = (V, E)$ is a simple graph with a path joining the vertices s, t.

- Consists of (G, s, t) , where $G = (V, E)$ is a simple graph with a path joining the vertices s, t.



- Consists of (G, s, t) , where $G = (V, E)$ is a simple graph with a path joining the vertices s, t.



- In class, we saw that *PATH* was in *NL*

Some History

- UPATH can be decided in linear time, using basic search algorithms like BFS or DFS. However, these algorithms have $O(n)$ space requirements.

Some History

- UPATH can be decided in linear time, using basic search algorithms like BFS or DFS. However, these algorithms have $O(n)$ space requirements.
- A more space efficient algorithm was given by Savitch [Sav70], which showed that *UPATH* can be decided using $O(\log^2(n))$ space (and super polynomial time).

Some History

- UPATH can be decided in linear time, using basic search algorithms like BFS or DFS. However, these algorithms have $O(n)$ space requirements.
- A more space efficient algorithm was given by Savitch [Sav70], which showed that *UPATH* can be decided using $O(\log^2(n))$ space (and super polynomial time).
- Aleliunas, Karp, Lipton, Lovasz, and Rackoff [AKLLR79] gave a *randomized* logspace algorithm for UPATH, showing $UPATH \in RL$.

Some History

- UPATH can be decided in linear time, using basic search algorithms like BFS or DFS. However, these algorithms have $O(n)$ space requirements.
- A more space efficient algorithm was given by Savitch [Sav70], which showed that $UPATH$ can be decided using $O(\log^2(n))$ space (and super polynomial time).
- Aleliunas, Karp, Lipton, Lovasz, and Rackoff [AKLLR79] gave a *randomized* logspace algorithm for UPATH, showing $UPATH \in RL$.
- Armoni, Ta-Shma, Wigderson, Zhou [ATSWZ00] later showed $UPATH \in L^{4/3}$, using the technique of derandomization.

Some History

- UPATH can be decided in linear time, using basic search algorithms like BFS or DFS. However, these algorithms have $O(n)$ space requirements.
- A more space efficient algorithm was given by Savitch [Sav70], which showed that *UPATH* can be decided using $O(\log^2(n))$ space (and super polynomial time).
- Aleliunas, Karp, Lipton, Lovasz, and Rackoff [AKLLR79] gave a *randomized* logspace algorithm for UPATH, showing $UPATH \in RL$.
- Armoni, Ta-Shma, Wigderson, Zhou [ATSWZ00] later showed $UPATH \in L^{4/3}$, using the technique of derandomization.
- Trifonov[Tri05] proved that *UPATH* can be decided deterministically. using $O(\log n \log\log n)$ space.

UPATH is in RL

$$\texttt{UPATH} \in \mathbf{RL}$$

Random Walks and Markov Chains

- Consider the following random walk on the (simple) graph G. For any vertex $u \in V$, we move to vertex $v \in V$ with probability

$$P_{uv} = \begin{cases} \frac{1}{d(u)}, & \text{if } \{u, v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

Random Walks and Markov Chains

- Consider the following random walk on the (simple) graph G. For any vertex $u \in V$, we move to vertex $v \in V$ with probability

$$P_{uv} = \begin{cases} \frac{1}{d(u)}, & \text{if } \{u, v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

- In this random walk, the probability distribution of the next state ONLY depends on the previous state (i.e., it has short-term memory)

Random Walks and Markov Chains

- Consider the following random walk on the (simple) graph G. For any vertex $u \in V$, we move to vertex $v \in V$ with probability

$$P_{uv} = \begin{cases} \frac{1}{d(u)}, & \text{if } \{u, v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

- In this random walk, the probability distribution of the next state ONLY depends on the previous state (i.e., it has short-term memory)
- This corresponds to a Markov Chain on the graph G, with state space $|V|$ and transition matrix P with transition probabilities P_{uv}

Random Walks and Markov Chains

- $$P = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{pmatrix}$$

Random Walks and Markov Chains

- $$P = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{pmatrix}$$
- A finite markov chain is an infinite sequence $X_0, X_1, \cdots$ of random variables, on a statespace Ω such that for all $i, j, a_0, a_1 \dots, a_{k-2} \in \Omega$, we have

$$P(X_k = j | X_0 = a_0, \dots X_{k-2} = a_{k-2}, X_{k-1} = i)$$
$$= P(X_k = j | X_{k-1} = i) = P_{ij}$$

Random Walks and Markov Chains

- For a Markov Chain M on a finite statespace Ω, $(|\Omega| = n)$ with transition matrix P, we say a $1 \times n$ row vector π (transpose is a column vector in $\mathbb{R}^n$) is a stationary distribution for M if $\pi P = \pi$

$$\Big(\pi(1) \cdots \pi(n)\Big) \cdot \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{pmatrix} = \Big(\pi(1) \cdots \pi(n)\Big)$$

Random Walks and Markov Chains

- $\pi(i) = d(i)$ is a stationary distribution for any random walk on a graph G

$$\Big(d(1) \cdots d(n)\Big) \cdot \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \ldots & P_{nn} \end{pmatrix} = \Big(d(1) \cdots d(n)\Big)$$

Random Walks and Markov Chains

- $\pi(i) = d(i)$ is a stationary distribution for any random walk on a graph G

$$\left(d(1) \cdots d(n)\right) \cdot \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{pmatrix} = \left(d(1) \cdots d(n)\right)$$

- $$\sum_{j \in V} d(j) P_{ji} = \sum_{j \in N(i)} d(j) \cdot \frac{1}{d(j)} = |N(i)| = d(i)$$

Random Walks and Markov Chains

- $\pi(i) = d(i)$ is a stationary distribution for any random walk on a graph G

$$\begin{pmatrix} d(1) \cdots d(n) \end{pmatrix} \cdot \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{pmatrix} = \begin{pmatrix} d(1) \cdots d(n) \end{pmatrix}$$

- $$\sum_{j \in V} d(j) P_{ji} = \sum_{j \in N(i)} d(j) \cdot \frac{1}{d(j)} = |N(i)| = d(i)$$

- $$Y^0 = Y, Y^1 = Y \cdot P = YP, Y^2 = Y^1 \cdot P = YP^2, \cdots Y^k = YP^k$$

Random Walks and Markov Chains

- A markov chain is said to be irreducible if and only for any $i, j \in \Omega$, we have that there exists $t > 0$ (possibly depending on i, j) such that

$$P(X_t = j | X_0 = i) > 0$$

Random Walks and Markov Chains

- A markov chain is said to be irreducible if and only for any $i, j \in \Omega$, we have that there exists $t > 0$ (possibly depending on i, j) such that

$$P(X_t = j | X_0 = i) > 0$$

- So, if we start a random walk at some $i \in \Omega$, we will be at j after t steps with some *positive probability*, which means there is a path between i and j.

Random Walks and Markov Chains

- A markov chain is said to be irreducible if and only for any $i, j \in \Omega$, we have that there exists $t > 0$ (possibly depending on i, j) such that

$$P(X_t = j | X_0 = i) > 0$$

- So, if we start a random walk at some $i \in \Omega$, we will be at j after t steps with some *positive probability*, which means there is a path between i and j.
- The markov chain corresponding to a random walk on a connected graph G is irreducible, as there is a path between any 2 vertices.

Random Walks and Markov Chains

- A markov chain is said to be irreducible if and only for any $i, j \in \Omega$, we have that there exists $t > 0$ (possibly depending on i, j) such that

$$P(X_t = j | X_0 = i) > 0$$

- So, if we start a random walk at some $i \in \Omega$, we will be at j after t steps with some *positive probability*, which means there is a path between i and j.
- The markov chain corresponding to a random walk on a connected graph G is irreducible, as there is a path between any 2 vertices.
- For a finite irreducible markov chain M, there is a **unique** stationary distribution for M, up to a multiplicative factor.

Random Walks and Markov Chains

- Suppose there are 2 stationary distributions, a and b. Note that $a(i), b(i) \neq 0 \ \forall i \in \Omega$, due to irreducibility and stationarity.

Random Walks and Markov Chains

- Suppose there are 2 stationary distributions, a and b. Note that $a(i), b(i) \neq 0 \ \forall i \in \Omega$, due to irreducibility and stationarity.
- Let $j \in \Omega$ be chosen to minimize $\frac{a(i)}{b(i)}$, i.e., $j = argmin_{i \in \Omega} \frac{a(i)}{b(i)}$, and define $r = \frac{a(j)}{b(j)}$

Random Walks and Markov Chains

- Suppose there are 2 stationary distributions, a and b. Note that $a(i), b(i) \neq 0 \ \forall i \in \Omega$, due to irreducibility and stationarity.
- Let $j \in \Omega$ be chosen to minimize $\frac{a(i)}{b(i)}$, i.e., $j = argmin_{i\in\Omega}\frac{a(i)}{b(i)}$, and define $r = \frac{a(j)}{b(j)}$

$$\begin{aligned}
a(j) = \sum_{i\in\Omega} a(i)P_{ij} &= \sum_{i\in\Omega} \frac{a(i)}{b(i)} b(i)P_{ij} \\
&\geq \sum_{i\in\Omega} r \cdot b(i)P_{ij} \\
&= r\sum_{i\in\Omega} b(i)P_{ij} \\
&= rb(j) \\
&= a(j)
\end{aligned}$$

Random Walks and Markov Chains

- $$\frac{a(i)}{b(i)}P_{ij} = rP_{ij} \implies a(i) = rb(i) \text{ IF } P_{ij} > 0$$

Random Walks and Markov Chains

- $$\frac{a(i)}{b(i)} P_{ij} = rP_{ij} \implies a(i) = rb(i) \text{ IF } P_{ij} > 0$$

- By irreducibility, for any i, j there exists t such that $P_{ij}^t > 0$

$$\begin{aligned} a(j) = \sum_{i \in \Omega} a(i) P_{ij}^t &= \sum_{i \in \Omega} \frac{a(i)}{b(i)} b(i) P_{ij}^t \\ &\geq \sum_{i \in \Omega} r \cdot b(i) P_{ij}^t \\ &= r \sum_{i \in \Omega} b(i) P_{ij}^t \\ &= rb(j) \\ &= a(j) \end{aligned}$$

Random Walks and Markov Chains

- It turns out that any finite irreducible markov chain actually has a stationary distribution, given by

$$\pi(i) = \frac{1}{T^+(i)}$$

where $T^+(i)$ is the expected first return time of a random walk starting at i.

Random Walks and Markov Chains

- It turns out that any finite irreducible markov chain actually has a stationary distribution, given by

$$\pi(i) = \frac{1}{T^+(i)}$$

where $T^+(i)$ is the expected first return time of a random walk starting at i.

- A random commute from i to j is a random walk starting at i that ends the first time it visits i, **provided that it has travelled through j**. Define T_{ij} to be the expected number of steps in such a random commute.

Random Walks and Markov Chains

- It turns out that any finite irreducible markov chain actually has a stationary distribution, given by

$$\pi(i) = \frac{1}{T^+(i)}$$

where $T^+(i)$ is the expected first return time of a random walk starting at i.

- A random commute from i to j is a random walk starting at i that ends the first time it visits i, **provided that it has travelled through j**. Define T_{ij} to be the expected number of steps in such a random commute.
- For any $i, j, u, v \in V$ with $\{u, v\} \in E$, define θ_{ijuv} to be the expected number of times the edge $\{u, v\}$ (in that order) is visited in a random commute from i to j

Random Walks and Markov Chains

- $\theta_{ijuv} = \theta_{ijuv'}, \quad \forall v' \in N(u)$. This is because in any random commute from i to j, once u is visited, any of its neighbours are visited with equal probability. So we write $\theta_{ijuv} = \theta_{iju}$

Random Walks and Markov Chains

- $\theta_{ijuv} = \theta_{ijuv'}, \quad \forall v' \in N(u)$. This is because in any random commute from i to j, once u is visited, any of its neighbours are visited with equal probability. So we write $\theta_{ijuv} = \theta_{iju}$
- θ_{iju} is actually independent of u.

$$\sum_{v \in N(u)} \theta_{ijuv} = d(u)\theta_{iju} = \sum_{v \in N(u)} \theta_{ijvu}$$

$$= \sum_{v \in N(u)} d(v)\theta_{ijv} \frac{1}{d(v)} = \sum_{v \in V} d(v)\theta_{ijv} P_{vu}$$

Think of the LHS as the number of times the random commute leaves u, which must be equal to the number of times it enters u in any random commute from i to j.

Random Walks and Markov Chains

- $\theta_{ijuv} = \theta_{ijuv'}, \quad \forall v' \in N(u)$. This is because in any random commute from i to j, once u is visited, any of its neighbours are visited with equal probability. So we write $\theta_{ijuv} = \theta_{iju}$
- θ_{iju} is actually independent of u.

$$\sum_{v \in N(u)} \theta_{ijuv} = d(u)\theta_{iju} = \sum_{v \in N(u)} \theta_{ijvu}$$
$$= \sum_{v \in N(u)} d(v)\theta_{ijv}\frac{1}{d(v)} = \sum_{v \in V} d(v)\theta_{ijv}P_{vu}$$

Think of the LHS as the number of times the random commute leaves u, which must be equal to the number of times it enters u in any random commute from i to j.
- Let $\pi'(u) = d(u)\theta_{iju}$ Then $\pi' = \pi' P$

Random Walks and Markov Chains

- $T_{ij} \leq 2|E|$, for any edge $\{i, j\} \in E$

Random Walks and Markov Chains

- $T_{ij} \leq 2|E|$, for any edge $\{i,j\} \in E$
- $\theta_{ijij} \leq 1$ for any random commute from i to j.

$$T_{ij} = \sum_{\{u,v\} \in E} (\theta_{ijuv} + \theta_{ijvu}) = \sum_{\{u,v\} \in E} 2\theta_{ijij} = \theta_{ijij}(\sum_{\{u,v\} \in E} 2) \leq 2|E|$$

Random Walks and Markov Chains

- Random commute times satisfy the triangle inequality, i.e.,

$$T_{ij} \leq T_{ik} + T_{kj}$$

Random Walks and Markov Chains

- Random commute times satisfy the triangle inequality, i.e.,

$$T_{ij} \leq T_{ik} + T_{kj}$$

- So for any s, t that are connected by a path of length $\leq n$, we have

$$T_{st} \leq n \cdot 2|E| \leq 2n^3$$

Random Walks and Markov Chains

- Random commute times satisfy the triangle inequality, i.e.,

$$T_{ij} \leq T_{ik} + T_{kj}$$

- So for any s, t that are connected by a path of length $\leq n$, we have

$$T_{st} \leq n \cdot 2|E| \leq 2n^3$$

Note that even if the graph is not connected, the above bound holds as long as s, t are in the same component, as we can simply take our original graph to be the connected component containing s, t in the analysis.

Algorithm for UPATH $\in$ RL

- Algorithm: Consider a random walk starting at s that runs for $6n^3$ steps. Output YES if you hit t, no otherwise.

Algorithm for UPATH $\in$ RL

- Algorithm: Consider a random walk starting at s that runs for $6n^3$ steps. Output YES if you hit t, no otherwise.
- Let X_{st} be the time taken for a random walk starting at s to hit t. Note that $E[X_{st}] \leq E[T_{st}] \leq 2n^3$.

Algorithm for UPATH $\in$ RL

- Algorithm: Consider a random walk starting at s that runs for $6n^3$ steps. Output YES if you hit t, no otherwise.
- Let X_{st} be the time taken for a random walk starting at s to hit t. Note that $E[X_{st}] \leq E[T_{st}] \leq 2n^3$.
- $P(Failure) = P(X_{st} > 6n^3) \leq \frac{E[X_{st}]}{6n^3} \leq \frac{2n^3}{6n^3} \leq 1/3$

Algorithm for UPATH $\in$ RL

- Algorithm: Consider a random walk starting at s that runs for $6n^3$ steps. Output YES if you hit t, no otherwise.
- Let X_{st} be the time taken for a random walk starting at s to hit t. Note that $E[X_{st}] \leq E[T_{st}] \leq 2n^3$.
- $P(Failure) = P(X_{st} > 6n^3) \leq \frac{E[X_{st}]}{6n^3} \leq \frac{2n^3}{6n^3} \leq 1/3$
- This is in logspace as it takes $O(\log n)$ bits to index any vertex, and we don't need to keep track of previously visited vertices.

Random Walk Matrix (for d-regular graphs)

- Let G be a d-regular n-vertex graph.
- M is the adjacency matrix of G, where

$$M_{i,j} = \text{number of edges between } i \text{ and } j$$

Random Walk Matrix (for d-regular graphs)

- Let G be a d-regular n-vertex graph.
- M is the adjacency matrix of G, where

$$M_{i,j} = \text{number of edges between } i \text{ and } j$$

- Define $A = \frac{1}{d}M$. A is the random walk matrix of G.

Random Walk Matrix (for d-regular graphs)

- Let G be a d-regular n-vertex graph.
- M is the adjacency matrix of G, where

$$M_{i,j} = \text{number of edges between } i \text{ and } j$$

- Define $A = \frac{1}{d}M$. A is the random walk matrix of G.
- Observe that rows and colums of A sum to 1, and that A is symmetric.

Parameter $\lambda(G)$

- Denote by $\mathbf{1}$ the vector $\left(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right)$. Denote by $\mathbf{1}^{\perp}$ the set of vectors perpendicular to $\mathbf{1}$.

Parameter $\lambda(G)$

- Denote by $\mathbf{1}$ the vector $\left(\frac{1}{n}, \frac{1}{n}, \ldots., \frac{1}{n}\right)$. Denote by $\mathbf{1}^{\perp}$ the set of vectors perpendicular to $\mathbf{1}$.
- The parameter $\lambda(A)$, also denoted as $\lambda(G)$, is the maximum value of $\|A\mathbf{v}\|_2$ over all vectors $\mathbf{v} \in \mathbf{1}^{\perp}$ with $\|\mathbf{v}\|_2 = 1$

$$\lambda(G) = \max_{\mathbf{v} \in \mathbf{1}^{\perp}:\ \|\mathbf{v}\|_2 = 1} \|A\mathbf{v}\|_2$$

Claim

$\lambda(G)$ is the absolute value of the second largest eigenvalue of A (the random walk matrix of G).
In particular, for connected graphs

$$1 = \lambda_1 > \lambda(G) \geq |\lambda_3| \cdots \geq |\lambda_n|$$

Proof

- Since A is a symmetric matrix, we can find an orthogonal basis of eigenvectors $v_1, \ldots, v_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ which we can sort to ensure $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.

Proof

- Since A is a symmetric matrix, we can find an orthogonal basis of eigenvectors $v_1, \ldots, v_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ which we can sort to ensure $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.
- Note that $A\mathbf{1} = \mathbf{1}$. This is because $(A\mathbf{1})_i$ is the dot product i^{th} row of A and the vector $\mathbf{1}$. Because row sum for any row is equal to 1, we get $(A\mathbf{1})_i = 1/n$

Proof

- Since A is a symmetric matrix, we can find an orthogonal basis of eigenvectors $v_1, \ldots, v_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ which we can sort to ensure $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.
- Note that $A\mathbf{1} = \mathbf{1}$. This is because $(A\mathbf{1})_i$ is the dot product i^{th} row of A and the vector $\mathbf{1}$. Because row sum for any row is equal to 1, we get $(A\mathbf{1})_i = 1/n$
- Therefore, $\mathbf{1}$ is an *eigenvector* of A and the corresponding eigenvalue is equal to 1.

Proof

- Since A is a symmetric matrix, we can find an orthogonal basis of eigenvectors $v_1, \ldots, v_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ which we can sort to ensure $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.
- Note that $A\mathbf{1} = \mathbf{1}$. This is because $(A\mathbf{1})_i$ is the dot product i^{th} row of A and the vector $\mathbf{1}$. Because row sum for any row is equal to 1, we get $(A\mathbf{1})_i = 1/n$
- Therefore, $\mathbf{1}$ is an *eigenvector* of A and the corresponding eigenvalue is equal to 1.
- Now suppose $Ax = \lambda x$. Then, $A^n x = \lambda^n x$. Since any row in A^n has sum 1, we are taking positive weight combinations of entries in x.

Proof

- Since A is a symmetric matrix, we can find an orthogonal basis of eigenvectors $v_1, \ldots, v_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ which we can sort to ensure $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.
- Note that $A\mathbf{1} = \mathbf{1}$. This is because $(A\mathbf{1})_i$ is the dot product i^{th} row of A and the vector $\mathbf{1}$. Because row sum for any row is equal to 1, we get $(A\mathbf{1})_i = 1/n$
- Therefore, $\mathbf{1}$ is an *eigenvector* of A and the corresponding eigenvalue is equal to 1.
- Now suppose $Ax = \lambda x$. Then, $A^n x = \lambda^n x$. Since any row in A^n has sum 1, we are taking positive weight combinations of entries in x.
- So, the absolute value of any entry in $A^n x$ is at most the maximum absolute value in the vector x. Thus, $|\lambda_i| \leq 1 \, \forall i \in [n]$, and $\lambda_1 = 1$ and $v_1 = \mathbf{1}$.

Proof

- Since algebraic multiplicity = geometric multiplicity for symmetric matrices, the eigenvalue 1 has multiplicity 1.

Proof

- Since algebraic multiplicity = geometric multiplicity for symmetric matrices, the eigenvalue 1 has multiplicity 1.
- Moreover, this inequality is strict if not all entries of x are equal. So, all the other eigenvalues have eigenvalue strictly less than 1.

Proof

- Since algebraic multiplicity = geometric multiplicity for symmetric matrices, the eigenvalue 1 has multiplicity 1.
- Moreover, this inequality is strict if not all entries of x are equal. So, all the other eigenvalues have eigenvalue strictly less than 1.
- Also $\mathbf{1}^\perp = \mathrm{Span}\{v_2, \ldots, v_n\}$ and the value of $\|Av\|_2$ for $v \in \mathbf{1}^\perp$ is maximized when $v = v_2$. So, if $v = \sum_{i=2}^n c_i v_i$

$$\|Av\|_2 = \sqrt{\sum_{i=2}^{n} c_i^2 \lambda_i^2} \leq |\lambda_2|$$

Therefore, $|\lambda_2| = G(\lambda)$

Lemma 1

Let G be an n-vertex regular graph and $\mathbf{p}$ a probability distribution over G's vertices, then

$$\left\| A^t \mathbf{p} - \mathbf{1} \right\|_2 \leq \lambda^t$$

where $\lambda = \lambda(G)$

Proof

- We know, $\|A\mathbf{v}\|_2 \leq \lambda \|\mathbf{v}\|_2$ for all $\mathbf{v} \perp \mathbf{1}$.

Proof

- We know, $\|A\mathbf{v}\|_2 \leq \lambda \|\mathbf{v}\|_2$ for all $\mathbf{v} \perp \mathbf{1}$.
- Observe that if $\mathbf{v} \perp \mathbf{1}$, then $A\mathbf{v} \perp \mathbf{1}$ since $\langle\, \mathbf{1}, A\mathbf{v} \,\rangle = \left\langle\, A^T\mathbf{1}, \mathbf{v} \,\right\rangle = \langle\, 1, \mathbf{v} \,\rangle = 0.$ Thus A maps the subspace $\mathbf{1}^{\perp}$ to itself.

Proof

- We know, $\|A\mathbf{v}\|_2 \leq \lambda \|\mathbf{v}\|_2$ for all $\mathbf{v} \perp \mathbf{1}$.
- Observe that if $\mathbf{v} \perp \mathbf{1}$, then $A\mathbf{v} \perp \mathbf{1}$ since $\langle \mathbf{1}, A\mathbf{v} \rangle = \left\langle A^T\mathbf{1}, \mathbf{v} \right\rangle = \langle 1, \mathbf{v} \rangle = 0$. Thus A maps the subspace $\mathbf{1}^\perp$ to itself.
- Also the eigen vectors that are different from $\mathbf{1}$ span this subspace, therefore A must shrink every vector in $\mathbf{1}^\perp$ by atleast λ.

Proof

- We know, $\|A\mathbf{v}\|_2 \leq \lambda \|\mathbf{v}\|_2$ for all $\mathbf{v} \perp \mathbf{1}$.
- Observe that if $\mathbf{v} \perp \mathbf{1}$, then $A\mathbf{v} \perp \mathbf{1}$ since $\langle \mathbf{1}, A\mathbf{v} \rangle = \left\langle A^T\mathbf{1}, \mathbf{v} \right\rangle = \langle 1, \mathbf{v} \rangle = 0$. Thus A maps the subspace $\mathbf{1}^\perp$ to itself.
- Also the eigen vectors that are different from $\mathbf{1}$ span this subspace, therefore A must shrink every vector in $\mathbf{1}^\perp$ by atleast λ.
- A^t shrinks every vector in $\mathbf{1}^\perp$ by a factor of atleast λ^t. Therefore, we can say that $\lambda(A^t) \leq \lambda(A)^t$. (In fact, using diagonalization we can show $\lambda(A^t) = \lambda(A)^t$)

- Let $\mathbf{p}$ be some vector. We can break $\mathbf{p}$ into component parallel and orthogonal to $\mathbf{1}$, i.e, $\mathbf{p} = \alpha\mathbf{1} + \mathbf{p}'$. As $\mathbf{p}$ is a probability distribution, we must have $\alpha = 1$ since sum of coordinates in $\mathbf{p}'$ is zero.

- Let $\mathbf{p}$ be some vector. We can break $\mathbf{p}$ into component parallel and orthogonal to $\mathbf{1}$, i.e, $\mathbf{p} = \alpha\mathbf{1} + \mathbf{p}'$. As $\mathbf{p}$ is a probability distribution, we must have $\alpha = 1$ since sum of coordinates in $\mathbf{p}'$ is zero.
- Therefore,
$$A^t\mathbf{p} = A^t(\mathbf{1} + \mathbf{p}') = \mathbf{1} + A^t\mathbf{p}'$$
and we get
$$\left\|A^t\mathbf{p} - \mathbf{1}\right\|_2 = \left\|A^t\mathbf{p}'\right\|_2 \leq \lambda^t\left\|\mathbf{p}'\right\|_2 \leq \lambda^t$$
because $\left\|p'\right\|_2 \leq \|p\|_2 \leq \|p\|_1 = 1$

Lemma 2

If G is a *regular connected graph* with self-loops at each vertex, then $\lambda(G) \leq 1 - \frac{1}{4dn^2}$

Proof

- Let $\epsilon = \frac{1}{2dn^2}$, let $\mathbf{u} \perp \mathbf{1}$ be a unit vector and let $\mathbf{v} = A\mathbf{u}$. We need to prove that $\|\mathbf{v}\|_2 \leq 1 - \epsilon/2$ and for this it suffices to prove that $1 - \|\mathbf{v}\|_2^2 \geq \epsilon$.

Proof

- Let $\epsilon = \frac{1}{2dn^2}$, let $\mathbf{u} \perp \mathbf{1}$ be a unit vector and let $\mathbf{v} = A\mathbf{u}$. We need to prove that $\|\mathbf{v}\|_2 \leq 1 - \epsilon/2$ and for this it suffices to prove that $1 - \|\mathbf{v}\|_2^2 \geq \epsilon$.
  This is because if $\|\mathbf{v}\|_2 > 1 - \epsilon/2$, then
  $\|\mathbf{v}\|_2^2 > 1 - \epsilon \implies 1 - \|\mathbf{v}\|_2^2 < \epsilon$

Proof

- Let $\epsilon = \frac{1}{2dn^2}$, let $\mathbf{u} \perp \mathbf{1}$ be a unit vector and let $\mathbf{v} = A\mathbf{u}$. We need to prove that $\|\mathbf{v}\|_2 \leq 1 - \epsilon/2$ and for this it suffices to prove that $1 - \|\mathbf{v}\|_2^2 \geq \epsilon$.
  This is because if $\|\mathbf{v}\|_2 > 1 - \epsilon/2$, then
  $\|\mathbf{v}\|_2^2 > 1 - \epsilon \implies 1 - \|\mathbf{v}\|_2^2 < \epsilon$
- Since $\mathbf{u}$ is a unit vector, we get $1 - \|\mathbf{v}\|_2^2 = \|\mathbf{u}\|_2^2 - \|\mathbf{v}\|_2^2$. We claim that this is equal to $\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2$ where i, j ranges from 1 to n.

- This is because

$$\begin{aligned}\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 &= \sum_{i,j} A_{i,j}\mathbf{u}_i^2 - 2\sum_{i,j} A_{i,j}\mathbf{u}_i\mathbf{v}_j + \sum_{i,j} A_{i,j}\mathbf{v}_j^2 \\ &= \|\mathbf{u}\|_2^2 - 2\langle A\mathbf{u}, \mathbf{v}\rangle + \|\mathbf{v}\|_2^2 \\ &= \|\mathbf{u}\|_2^2 - 2\|\mathbf{v}\|_2^2 + \|\mathbf{v}\|_2^2 \\ &= \|\mathbf{u}\|_2^2 - \|\mathbf{v}\|_2^2\end{aligned}$$

$$\|\mathbf{v}\|_2^2 = \langle \mathbf{v}, \mathbf{v}\rangle = \langle A\mathbf{u}, \mathbf{v}\rangle = \sum_{i,j} A_{i,j}\mathbf{u}_i\mathbf{v}_j$$

- Therefore, now we want to show that $\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \epsilon$. Since $\mathbf{u}$ is a unit vector with coordinates summing to zero, there must exist vertices i, j such that $\mathbf{u}_i > 0$ and $\mathbf{u}_j < 0$ and atleast one of these coordinates has absolute value $\geq \frac{1}{\sqrt{n}}$, which implies that $\mathbf{u}_i - \mathbf{u}_j \geq \frac{1}{\sqrt{n}}$.

- Therefore, now we want to show that $\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \epsilon$. Since $\mathbf{u}$ is a unit vector with coordinates summing to zero, there must exist vertices i, j such that $\mathbf{u}_i > 0$ and $\mathbf{u}_j < 0$ and atleast one of these coordinates has absolute value $\geq \frac{1}{\sqrt{n}}$, which implies that $\mathbf{u}_i - \mathbf{u}_j \geq \frac{1}{\sqrt{n}}$.
- Also because G is connected, there is a path between i and j containing atmost $D + 1$ vertices (D is the diameter of the graph G). Let us rename the vertices, and assume that $i = 1$ and $j = D + 1$, and the coordinates $2, 3, \ldots, D$ correspond to the vertices on this path in order.

Then, we have

$$\begin{aligned}\frac{1}{\sqrt{n}} &\leq \mathbf{u}_1 - \mathbf{u}_{D+1}\\ &= (\mathbf{u}_1 - \mathbf{v}_1) + (\mathbf{v}_1 - \mathbf{u}_2) + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})\\ &\leq |\mathbf{u}_1 - \mathbf{v}_1| + |\mathbf{v}_1 - \mathbf{u}_2| + \ldots + |\mathbf{v}_D - \mathbf{u}_{D+1}|\\ &\leq \sqrt{(\mathbf{u}_1 - \mathbf{v}_1)^2 + (\mathbf{v}_1 - \mathbf{u}_2)^2 + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})^2}\sqrt{2D+1}\end{aligned}$$

Then, we have

$$\begin{aligned}\frac{1}{\sqrt{n}} &\leq \mathbf{u}_1 - \mathbf{u}_{D+1} \\ &= (\mathbf{u}_1 - \mathbf{v}_1) + (\mathbf{v}_1 - \mathbf{u}_2) + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1}) \\ &\leq |\mathbf{u}_1 - \mathbf{v}_1| + |\mathbf{v}_1 - \mathbf{u}_2| + \ldots + |\mathbf{v}_D - \mathbf{u}_{D+1}| \\ &\leq \sqrt{(\mathbf{u}_1 - \mathbf{v}_1)^2 + (\mathbf{v}_1 - \mathbf{u}_2)^2 + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})^2}\sqrt{2D+1}\end{aligned}$$

Therefore, we get

$$(\mathbf{u}_1 - \mathbf{v}_1)^2 + (\mathbf{v}_1 - \mathbf{u}_2)^2 + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})^2 \geq \frac{1}{n(2D+1)}$$

- Observe that

$$\begin{aligned}\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 &\geq \sum_i A_{i,i}(\mathbf{u}_i - \mathbf{v}_i)^2 + A_{i,i+1}(\mathbf{v}_i - \mathbf{u}_{i+1})^2 \\ &\geq \frac{1}{d}(\mathbf{u}_1 - \mathbf{v}_1)^2 + (\mathbf{v}_1 - \mathbf{u}_2)^2 + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})^2 \\ &\geq \frac{1}{nd(2D+1)}\end{aligned}$$

- Observe that

$$\begin{aligned}\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 &\geq \sum_i A_{i,i}(\mathbf{u}_i - \mathbf{v}_i)^2 + A_{i,i+1}(\mathbf{v}_i - \mathbf{u}_{i+1})^2 \\ &\geq \frac{1}{d}(\mathbf{u}_1 - \mathbf{v}_1)^2 + (\mathbf{v}_1 - \mathbf{u}_2)^2 + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})^2 \\ &\geq \frac{1}{nd(2D+1)}\end{aligned}$$

- Using this bound and substituting $D \leq n - 1$ we get

$$1 - \|\mathbf{v}\|_2^2 \geq \frac{1}{2dn^2}$$

which implies that $\lambda(G) \leq 1 - \frac{1}{4dn^2}$

- Observe that

$$\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \sum_i A_{i,i}(\mathbf{u}_i - \mathbf{v}_i)^2 + A_{i,i+1}(\mathbf{v}_i - \mathbf{u}_{i+1})^2$$
$$\geq \frac{1}{d}(\mathbf{u}_1 - \mathbf{v}_1)^2 + (\mathbf{v}_1 - \mathbf{u}_2)^2 + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})^2$$
$$\geq \frac{1}{nd(2D+1)}$$

- Using this bound and substituting $D \leq n - 1$ we get

$$1 - \|\mathbf{v}\|_2^2 \geq \frac{1}{2dn^2}$$

which implies that $\lambda(G) \leq 1 - \frac{1}{4dn^2}$

- In fact, we can show $D \leq 3n/(d+1)$

Expander Graphs

There are two ways to define Expander graphs

- Algebraic definition
- Combinatorial definition

Algrebraic Definition

- (n, d, λ)-**expander graphs:** If G is an n-vertex d-regular graph with $\lambda(G) \leq \lambda$ for some number $\lambda < 1$, then we say that G is an (n, d, λ)-graph.

Algrebraic Definition

- (n, d, λ)-**expander graphs:** If G is an n-vertex d-regular graph with $\lambda(G) \leq \lambda$ for some number $\lambda < 1$, then we say that G is an (n, d, λ)-graph.
- A family of graphs $\{G_n\}_{n \in N}$ is an *expander graph family* if there are some constants $d \in \mathbb{N}$ and $\lambda < 1$ such that for every n, G_n is an (n, d, λ)-graph.

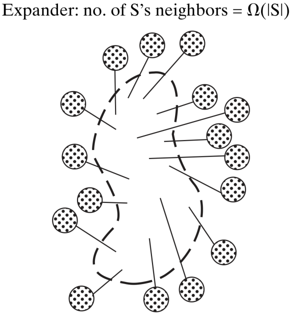
Combinatorial (edge) Definition

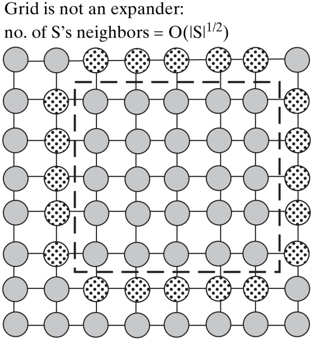
- An n-vertex d-regular graph $G = (V, E)$ is called an (n, d, ρ)-combinatorial edge expander if for every subset S of vertices satisfying $|S| \leq n/2$,

$$|E(S, \overline{S})| \geq \rho d|S|$$

where $\overline{S}$ denotes the complement of S and for any 2 subsets S, T of vertices, $E(S, T)$ denotes the set of edges between S and T

Edge expander graphs





Relation between the two definitions

- If G is an (n, d, λ)-expander graph, then it is an $(n, d, (1 - \lambda)/2)$ edge expander.

Relation between the two definitions

- If G is an (n, d, λ)-expander graph, then it is an $(n, d, (1-\lambda)/2)$ edge expander.
- If G is an (n, d, ρ) edge expander, then its second largest eigenvalue (without taking absolute values) is at most $1 - \frac{\rho^2}{2}$. If furthermore G has all self loops, then it is an $(n, d, 1-\epsilon)$-expander where $\epsilon = \min\left\{\frac{2}{d}, \frac{\rho^2}{2}\right\}$

Rotation Maps

- For a d-regular graph, we can assign a permutation of $[d]$ to label the outgoing edges of a vertex.

Rotation Maps

- For a d-regular graph, we can assign a permutation of $[d]$ to label the outgoing edges of a vertex.
- Let G be a d-regular graph on n vertices. Go to each vertex, and label it's outgoing edges with a permutation of $[d]$. Capture this by the function $\hat{G} : [n] \times [d] \mapsto [n] \times [d]$ which maps $\langle v, i \rangle$ to $\langle u, j \rangle$ where u is the i^{th} neighbour of v and v is the j^{th} neighbour of u.

Rotation Maps

- For a d-regular graph, we can assign a permutation of $[d]$ to label the outgoing edges of a vertex.
- Let G be a d-regular graph on n vertices. Go to each vertex, and label it's outgoing edges with a permutation of $[d]$. Capture this by the function $\hat{G} : [n] \times [d] \mapsto [n] \times [d]$ which maps $\langle v, i \rangle$ to $\langle u, j \rangle$ where u is the i^{th} neighbour of v and v is the j^{th} neighbour of u.
- Observe that $\hat{G}$ is a permutation (in fact, it is an involution).

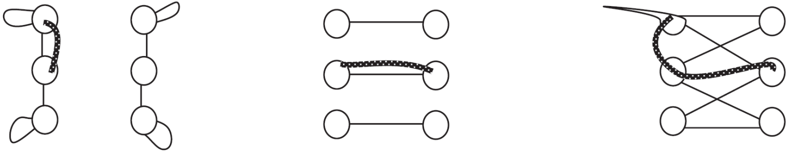
Path Product

- Let G and G' be two n-vertex graphs with degrees d, d' and random-walk matrices A, A' respectively. Then we describe the graph $G'G$ as the graph described by the random-walk matrix $A'A$.

Path Product

- Let G and G' be two n-vertex graphs with degrees d, d' and random-walk matrices A, A' respectively. Then we describe the graph $G'G$ as the graph described by the random-walk matrix $A'A$.
- That is, $G'G$ has an edge (u, v) for every length two-path from u to v where the first step in the path is taken on an edge of G and the second is on an edge of G'.

Path Product



Path Product

- Consider a n vertex graph G with degree d. Let A be it's random walk matrix. Then G^k is a d^k regular graph with random walk matrix A^k.

Path product makes expansion better

- As we have seen earlier (Lemma 1) ,

$$\lambda(G^k) \leq \big(\lambda(G)\big)^k$$

Computing rotation map of G^k

- Relabel the outgoing edges by a k tuple where the tuple represents the walk due to which this edge is present.

Computing rotation map of G^k

- Relabel the outgoing edges by a k tuple where the tuple represents the walk due to which this edge is present.
- That is, the rotation map of G^k is a permutation on $[n] \times [d^k]$.

Replacement Product

Let G and G' be two graphs where G is (n, D)-graph, G' is (D, d)-graph. The replacement product of G and G' is defined as $G \circledR G'$ is $(nD, 2d)$-graph is defined as:

Replacement Product

Let G and $G^{'}$ be two graphs where G is (n, D)-graph, $G^{'}$ is (D, d)-graph. The replacement product of G and $G^{'}$ is defined as $G Ⓡ G^{'}$ is $(nD, 2d)$-graph is defined as:

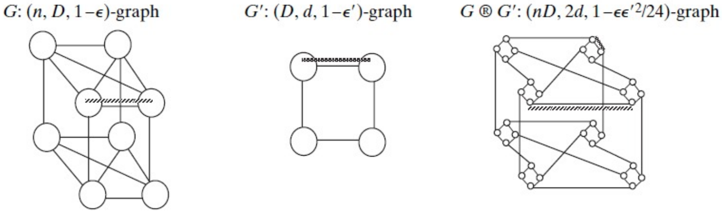
- For each vertex u of G, the graph $G Ⓡ G^{'}$ has a copy of $G^{'}$ (including edges and vertices).

Replacement Product

Let G and $G^{'}$ be two graphs where G is (n, D)-graph, $G^{'}$ is (D, d)-graph. The replacement product of G and $G^{'}$ is defined as $G \textcircled{R} G^{'}$ is $(nD, 2d)$-graph is defined as:

- For each vertex u of G, the graph $G \textcircled{R} G^{'}$ has a copy of $G^{'}$ (including edges and vertices).
- If u, v are two neighbouring vertices in G where $\langle\, u, i \,\rangle$ is mapped to $\langle\, v, j \,\rangle$, then we place d parallel edges between the i^{th} vertex in the copy of $G^{'}$ corresponding to u and the j^{th} vertex in the copy of $G^{'}$ corresponding to v.

Replacement Product



Replacement Product

- Observe that replacement product 'preserves' the connected components in G. (assuming $G^{'}$ is connected)

Replacement Product

- Observe that replacement product 'preserves' the connected components in G. (assuming $G^{'}$ is connected)
- Two copies of $G^{'}$ are connected if and only if they are connected in G.

Rotation map of replacement product

- First, observe that the rotation map will be a permutation over $\big([n] \times [D]\big) \times \big([d] \times \{0,1\}\big)$ as $G \circledR G'$. For some input $\big((u,v),(i,b)\big)$, the rotation map function first checks if $b = 0$ or $b = 1$.

Rotation map of replacement product

- First, observe that the rotation map will be a permutation over $\big([n] \times [D]\big) \times \big([d] \times \{0,1\}\big)$ as $G \circledR G'$. For some input $\big((u,v),(i,b)\big)$, the rotation map function first checks if $b=0$ or $b=1$.
- If $b=0$, then it treats v as a vertex of G'. Thus, it outputs $\Big(u, \hat{G}'(v,i), b\Big)$.

Rotation map of replacement product

- First, observe that the rotation map will be a permutation over $\big([n] \times [D]\big) \times \big([d] \times \{0,1\}\big)$ as $G \circledR G^{'}$. For some input $\big((u,v),(i,b)\big)$, the rotation map function first checks if $b = 0$ or $b = 1$.
- If $b = 0$, then it treats v as a vertex of $G^{'}$. Thus, it outputs $\Big(u, \hat{G}'(v,i), b\Big)$.
- If $b = 1$, then it treats v as an edge label of G. Thus, it outputs $\Big(\hat{G}(u,v), i, b\Big)$.

Rotation Map of replacement product

- In other words, $b = 0$ indicates an edge inside a cluster of G', while $b = 1$ indicates a cross edge between clusters.

Expansion of a replacement product

- Claim: If $\lambda(G) \leq 1 - \epsilon$ and $\lambda(H) \leq 1 - \delta$ then $\lambda(G \circledR H) \leq 1 - \frac{\epsilon\delta^2}{24}$
- This shows that replacement product does not worsen the expansion by too much.

Recap

- Path Product - Improves expansion but increases degree

Recap

- Path Product - Improves expansion but increases degree
- Replacement Product - Decreases degree and does not worsen expansion by too much

$UPATH \in L$ when G is an expander

- Let G be a d-regular graph, whose every connected component is an expander.

$UPATH \in L$ when G is an expander

- Let G be a d-regular graph, whose every connected component is an expander.
- Observe that for such graphs, there is a number $l = O(\log n)$ such that if s and t are connected, they are connected with a path of length at most l. (due to Lemma 1)

$UPATH \in L$ when G is an expander

- Let G be a d-regular graph, whose every connected component is an expander.
- Observe that for such graphs, there is a number $l = O(\log n)$ such that if s and t are connected, they are connected with a path of length at most l. (due to Lemma 1)
- Put $t = \frac{\log \frac{1}{n^2}}{\log \lambda}$ in Lemma 1 to get $||A^t p - 1|| \leq \lambda^t \leq \frac{1}{n^2}$. Put $p = \hat{e_u}$ (vector with all zeros except 1 at starting vertex u). After t steps, the entry corresponding to final vertex v in $A^t p$ must be non zero (otherwise $\left\|A^t p - 1\right\|_2 \geq \frac{1}{n}$). Thus, there must be some path of length t between u and v.

$UPATH \in L$ when G is an expander

- Let G be a d-regular graph, whose every connected component is an expander.
- Observe that for such graphs, there is a number $l = O(\log n)$ such that if s and t are connected, they are connected with a path of length at most l. (due to Lemma 1)
- Put $t = \frac{\log \frac{1}{n^2}}{\log \lambda}$ in Lemma 1 to get $||A^t p - 1|| \leq \lambda^t \leq \frac{1}{n^2}$. Put $p = \hat{e_u}$ (vector with all zeros except 1 at starting vertex u). After t steps, the entry corresponding to final vertex v in $A^t p$ must be non zero (otherwise $\left\|A^t p - 1\right\|_2 \geq \frac{1}{n}$). Thus, there must be some path of length t between u and v.
- We can now enumerate over all paths of length $O(\log n)$ (instead of $O(n)$). This can be done in logspace now as the number of walks are $O(d^{O(\log n)}) = poly(n)$, which can be enumerated in logspace.

Reingold's Theorem [2005]

$$\texttt{UPATH} \in \mathbf{L}$$

Motivation

- It is easy to check whether s and t are connected in an expander graph. This is because if s and t are connected, then there is a $O(\log n)$ length path between s and t.

Motivation

- It is easy to check whether s and t are connected in an expander graph. This is because if s and t are connected, then there is a $O(\log n)$ length path between s and t.
- Then we can check all walks of length $O(\log n)$ from s to see whether it hits t. Assuming that the degree of each vertex is atmost some constant d, there will be $d^{O(\log n)}$ walks (as there are d choices at every vertex), which is $poly(n)$ and takes logspace to enumerate.

Motivation

- It is easy to check whether s and t are connected in an expander graph. This is because if s and t are connected, then there is a $O(\log n)$ length path between s and t.
- Then we can check all walks of length $O(\log n)$ from s to see whether it hits t. Assuming that the degree of each vertex is atmost some constant d, there will be $d^{O(\log n)}$ walks (as there are d choices at every vertex), which is $poly(n)$ and takes logspace to enumerate.
- So, the main idea is to convert the given graph into an expander graph while maintaining connectivity properties.

Converting into 4-regular graph

- We can convert any graph into a 4-regular graph, preserving the connectivity properties.

Converting into 4-regular graph

- We can convert any graph into a 4-regular graph, preserving the connectivity properties.
- If a vertex has degree $d" < 3$, we can add self loops to increase multiplicity.

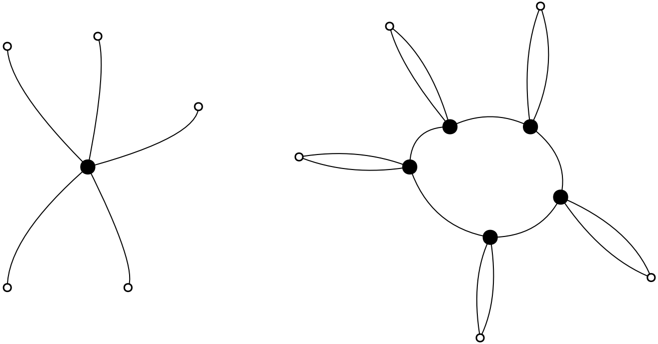
Converting into 4-regular graph

- We can convert any graph into a 4-regular graph, preserving the connectivity properties.
- If a vertex has degree d" < 3, we can add self loops to increase multiplicity.
- If vertex has degree $d' > 3$, then we can replace the cycle by a cycle containing d' vertices, each of the d' vertices that were incident to the old vertices attach to one of the cycle nodes.

Converting into 4-regular graph

- We can convert any graph into a 4-regular graph, preserving the connectivity properties.
- If a vertex has degree $d" < 3$, we can add self loops to increase multiplicity.
- If vertex has degree $d' > 3$, then we can replace the cycle by a cycle containing d' vertices, each of the d' vertices that were incident to the old vertices attach to one of the cycle nodes.
- This transformation does not change connectivity properties.

Converting into 4-regular graph



Recursive Algorithm

- We can convert the 4-regular graph into a d^{50}-regular graph by adding self loops (assuming d is even).

Recursive Algorithm

- We can convert the 4-regular graph into a d^{50}-regular graph by adding self loops (assuming d is even).
- Let H be a $(d^{50}, d/2, 0.01)$-expander graph. Note that H is the same for all problems.

Recursive Algorithm

- We can convert the 4-regular graph into a d^{50}-regular graph by adding self loops (assuming d is even).
- Let H be a $(d^{50}, d/2, 0.01)$-expander graph. Note that H is the same for all problems.
- Let G_0 be our d^{50}-regular graph. And let us define

$$G_k = (G_{k-1} \circledR H)^{50}$$

.

Number of Vertices

- If G_0 has n vertices and is d^{50}-regular, then $G_0 \textcircled{R} H$ has $d^{50}n$ vertices is d-regular. Therefore, $G_1 = \left((G_0 \textcircled{R} H)^{50}\right)$ has $d^{50}n$ vertices and is d^{50}-regular.

Number of Vertices

- If G_0 has n vertices and is d^{50}-regular, then $G_0 Ⓡ H$ has $d^{50}n$ vertices is d-regular. Therefore, $G_1 = \left((G_0 Ⓡ H)^{50}\right)$ has $d^{50}n$ vertices and is d^{50}-regular.
- Therefore in general, G_k has $d^{50k}n$ vertices, and is d^{50}-regular

Claim

For all $\epsilon < 1/20$, if $\lambda(F) \leq 1 - \epsilon$, then

$$\lambda\left((F \circledR H)^{50}\right) \leq 1 - 2\epsilon$$

.

Claim

For all $\epsilon < 1/20$, if $\lambda(F) \leq 1 - \epsilon$, then

$$\lambda\left((F \text{Ⓡ} H)^{50}\right) \leq 1 - 2\epsilon$$

. **Proof:**

- This is because if $\lambda(F) \leq 1 - \epsilon$, then

$$\lambda(F \text{Ⓡ} H) \leq 1 - \frac{\epsilon(1 - 0.01)^2}{24} \leq 1 - \frac{\epsilon}{25}$$

.

- Then $\lambda\left((F \text{Ⓡ} H)^{50}\right) \leq (1 - \epsilon/25)^{50} \leq 1 - 2\epsilon$.

G_k is an expander graph

- Recall that for a D-regular graph containing self loops, every connected component of G_0 has expansion parameter of atmost $1 - \frac{1}{4Dn^2}$. Here $D = d^{50}$.

G_k is an expander graph

- Recall that for a D-regular graph containing self loops, every connected component of G_0 has expansion parameter of atmost $1 - \frac{1}{4Dn^2}$. Here $D = d^{50}$.
- By previous claim, we get that expansion parameter of G_1 is atmost $\max\left(1 - \frac{1}{20}, 1 - \frac{2}{4Dn^2}\right)$. Similarly, we get expansion parameter of G_k is atmost $\max\left(1 - \frac{1}{20}, 1 - \frac{2^k}{4Dn^2}\right)$.

G_k is an expander graph

- Recall that for a D-regular graph containing self loops, every connected component of G_0 has expansion parameter of atmost $1 - \frac{1}{4Dn^2}$. Here $D = d^{50}$.
- By previous claim, we get that expansion parameter of G_1 is atmost $\max\left(1 - \frac{1}{20}, 1 - \frac{2}{4Dn^2}\right)$. Similarly, we get expansion parameter of G_k is atmost $\max\left(1 - \frac{1}{20}, 1 - \frac{2^k}{4Dn^2}\right)$.
- Therefore, for $k = O(\log n)$, we get $\lambda(G_k)$ is atmost $1 - \frac{1}{20}$.

Analysis

- Here $N = d^{50c \log n} n$. Therefore, we get that there is a $O(\log n)$ length path between s and t.

Analysis

- Here $N = d^{50c \log n} n$. Therefore, we get that there is a $O(\log n)$ length path between s and t.
- Now we explore each path starting from s of length atmost $O(\log n)$ to check whether it hits t. As the degree of each vertex is d^{50}, we get that there are atmost $d^{O(\log n)}$ (polynomially many) paths to explore.

Analysis

- Here $N = d^{50c \log n} n$. Therefore, we get that there is a $O(\log n)$ length path between s and t.
- Now we explore each path starting from s of length atmost $O(\log n)$ to check whether it hits t. As the degree of each vertex is d^{50}, we get that there are atmost $d^{O(\log n)}$ (polynomially many) paths to explore.
- Assuming we know G_k, we can do this in log space as we only have to maintain the current vertex, number of edges traversed in current path and number of paths traversed.

Analysis

- We don't know G_k explicitly. So finding the neighbours of a vertex in G_k is not trivial.

Analysis

- We don't know G_k explicitly. So finding the neighbours of a vertex in G_k is not trivial.
- Observe that we only need to be able to find the i^{th} neighbour of v in log space.

Analysis

- We don't know G_k explicitly. So finding the neighbours of a vertex in G_k is not trivial.
- Observe that we only need to be able to find the i^{th} neighbour of v in log space.
- If we can take a single step in log space, then we can take l steps in log space by reusing the space.

Analysis

- Recall that $G_k = (G_{k-1} \textcircled{R} H)^{50}$, thus it suffices to show that we can take a single step in the graph $G_{k-1} \textcircled{R} H$ in logspace.

Analysis

- Recall that $G_k = (G_{k-1} \circledR H)^{50}$, thus it suffices to show that we can take a single step in the graph $G_{k-1} \circledR H$ in logspace.
- Suppose we are at some vertex $\langle u, v \rangle$ (where u is a vertex of G_{k-1} and v is a vertex in H). We want to take a step from this vertex.

Analysis

- Suppose we want to take a step on the edge labelled $\langle b, i \rangle$.

Analysis

- Suppose we want to take a step on the edge labelled $\langle b, i \rangle$.
- If $b = 0$, then the edge is inside a copy of H, so this requires us to access the rotation map of H, which takes $O(1)$ space.

Analysis

- Suppose we want to take a step on the edge labelled $\langle b, i \rangle$.
- If $b = 0$, then the edge is inside a copy of H, so this requires us to access the rotation map of H, which takes $O(1)$ space.
- If $b = 1$, then the edge is a cross edge between clusters, so this requires us to access the rotation map of G^{k-1}

Analysis

- This leads to a recursive algorithm where we need $O(1)$ space at every step.

Analysis

- This leads to a recursive algorithm where we need $O(1)$ space at every step.
- If s_k is the space needed to compute the rotation map of G^k, we have $s_k = s_{k-1} + O(1)$.

Analysis

- This leads to a recursive algorithm where we need $O(1)$ space at every step.
- If s_k is the space needed to compute the rotation map of G^k, we have $s_k = s_{k-1} + O(1)$.
- Hence, $s_k = O(\log n)$ as $k = O(\log n)$.

Recap

- In a graph with n vertices, there is a path with length $\leq n$ between any two connected vertices.

Recap

- In a graph with n vertices, there is a path with length $\leq n$ between any two connected vertices.
- The naive way to derandomize in general graphs fails since there are d^n possible walks to search, and enumerating these walks will take $O(n)$ space. (assuming d to be constant)

Recap

- In a graph with n vertices, there is a path with length $\leq n$ between any two connected vertices.
- The naive way to derandomize in general graphs fails since there are d^n possible walks to search, and enumerating these walks will take $O(n)$ space. (assuming d to be constant)
- Fortunately, in graphs with good expansion there is a $O(\log n)$ length path between any two connected vertices.

Recap

- This leads to a natural question - Can we transform *any* graph into an expander? (that is, every connected component of the final graph will be an expander).

Recap

- This leads to a natural question - Can we transform *any* graph into an expander? (that is, every connected component of the final graph will be an expander).
- Graph product improves expansion, but increases degree (which is a problem).

Recap

- This leads to a natural question - Can we transform *any* graph into an expander? (that is, every connected component of the final graph will be an expander).
- Graph product improves expansion, but increases degree (which is a problem).
- Replacement product reduces degree, and does not worsen expansion by too much. We can get away with this by using a graph H with very good expansion.

Recap

- Finally, we use rotation maps to implicitly keep track of the different graphs in logspace.
- The final algorithm is as follows:

Recap

- Given G, implicitly construct G_k for appropriate k (that is, you don't actually construct G_k but treat it's adjacency list/ rotation map as a recursive lookup function).
- We start enumerating all walks of $O(\log n)$ length in G_k implicitly, using rotation maps to ensure logspace.
- Effectively, this gives a complete derandomization (logspace constructible universal exploration sequences for general graphs).

Extra

- Every permutation can be represented by a permutation matrix, hence every rotation map can be represented by a permutation matrix.

Extra

- Every permutation can be represented by a permutation matrix, hence every rotation map can be represented by a permutation matrix.
- One can algebraically represent replacement product as

$$A \textcircled{R} A' = \frac{1}{2}\hat{A} + \frac{1}{2}(I_n \otimes A')$$

where A and A' are corresponding random walk matrices.

Extra

- Lemma: Let M be a random walk matrix of an (n, d, λ) expander graph G. Let J be the random walk matrix of the n clique with self loops, that is every entry is $\frac{1}{n}$. Then, $M = (1-\lambda)J + \lambda M'$ where $\left|M'\right| \leq 1$ (can check that $M' = \frac{1}{\lambda}\left(M - (1-\lambda)J\right)$ works).

Extra

- Lemma: Let M be a random walk matrix of an (n, d, λ) expander graph G. Let J be the random walk matrix of the n clique with self loops, that is every entry is $\frac{1}{n}$. Then, $M = (1-\lambda) J + \lambda M'$ where $\left| M' \right| \leq 1$ (can check that $M' = \frac{1}{\lambda}\left(M - (1-\lambda) J\right)$ works).
- Let A be the $n \times n$ random walk matrix of G (with $\hat{A}$ as the $nD \times nD$ permutation matrix)

Extra

- Lemma: Let M be a random walk matrix of an (n, d, λ) expander graph G. Let J be the random walk matrix of the n clique with self loops, that is every entry is $\frac{1}{n}$. Then, $M = (1-\lambda)J + \lambda M^{'}$ where $\left|M^{'}\right| \leq 1$ (can check that $M^{'} = \frac{1}{\lambda}\left(M - (1-\lambda)J\right)$ works).
- Let A be the $n \times n$ random walk matrix of G (with $\hat{A}$ as the $nD \times nD$ permutation matrix)
- Let B be the $D \times D$ random walk matrix of H, and let C be the $nD \times nD$ random walk matrix of $(G \textcircled{R} H)^3$

Extra

- Using the algebraic definition of replacement product, we get,
  $C = \left(\frac{1}{2}\hat{A} + \frac{1}{2}\left(I_n \otimes B\right)\right)^3$

Extra

- Applying the lemma on B, we obtain $B = (1 - \delta) B^{'} + \delta J$ where $||B^{'}|| \leq 1$

Extra

- Applying the lemma on B, we obtain $B = (1 - \delta) B^{'} + \delta J$ where $||B^{'}|| \leq 1$
- Substituting and manipulating,

$$C = \left(1 - \frac{\delta^2}{8}\right) C^{'} + \frac{\delta^2}{8} (I_n \otimes J) \hat{A} (I_n \otimes J)$$

Extra

- Applying the lemma on B, we obtain $B = (1-\delta)\, B^{'} + \delta J$ where $||B^{'}|| \leq 1$
- Substituting and manipulating,

$$C = \left(1 - \frac{\delta^2}{8}\right) C^{'} + \frac{\delta^2}{8} \left(I_n \otimes J\right) \hat{A} \left(I_n \otimes J\right)$$

- One can algebraically check that

$$\left(I_n \otimes J\right) \hat{A} \left(I_n \otimes J\right) = A \otimes J$$

and

$$\lambda(A \otimes J) \leq max(\lambda(A), 0)$$

Extra

- Plugging this back, we obtain $\lambda\left((G\circledR H)^3\right)$.

Extra

- Plugging this back, we obtain $\lambda\left((G \circledR H)^3\right)$.
- This implies that $\lambda(G \circledR H) \leq\ 1 - \frac{\epsilon\delta^2}{24}$ since $\lambda(G^3) = \lambda(G)^3$.