Advanced Algorithms

Spectral methods and algorithms

尹一通　栗师　**刘景铖**

Recap

Previous lecture:
Cheeger's inequality on d-regular graphs

- Easy direction: a sparse cut implies λ_2 is small
- Hard direction: a small λ_2 means we can find a sparse cut from x_2
- Spectral partitioning
- Improvements of Cheeger's
- Generalizations of Cheeger's

What next?
Random walks on undirected graphs

- Speeding up bipartite matching
- Return time
- Fundamental theorem of Markov chains
- Pagerank

Random Walks on Graphs

A random walk is a simple random process.

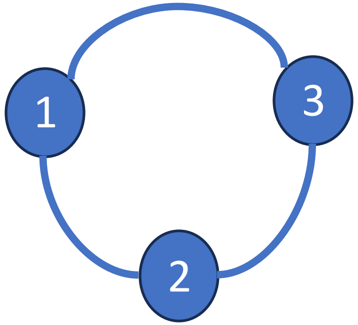
- Start from some vertex
- Move to a uniformly random neighbor of the current vertex
- Repeat many steps

We are interested in the long term behavior of this random process

For example, what is the probability of being in a vertex at a time t?

Vector Formulation of random walk

Example: Consider an undirected 3-cycle



If we start at vertex 1

with probability ½ we go to vertex 2

with probability ½ we go to vertex 3

In vector terms

our starting distribution is $p_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

after one-step of the random walk, the distribution becomes $p_1 = \begin{pmatrix} 0 \\ 1/2 \\ 1/2 \end{pmatrix}$

Common Questions

Basic questions in applying random walks in designing algorithms:

1. Is there a limiting distribution of the random walk? What does it look like? (*Stationary distribution*)
2. How long does it take for the random walk to converge to the limiting distribution? (*Mixing time*)
3. Starting from a vertex s, what is the expected number of steps to first reach vertex t? (*Hitting time*)
4. How long does it take to reach every vertex of the graph at least once? (*Cover time*)

There are two main approaches to answer the first two questions.

- One approach is probabilistic, using the technique of "**coupling**" of two random processes
- Another approach is linear algebraic, using the **eigenvalues of the transition matrix**
- We first introduce the spectral approach to answer the first two questions
- Then the probabilistic approach when we introduce the **Markov chain Monte Carlo method**
- The last two questions are best answered by viewing the graph as an "**electrical network**"

A random walk for finding bipartite matching

Matching : A subset of edges that do not overlap at any vertex

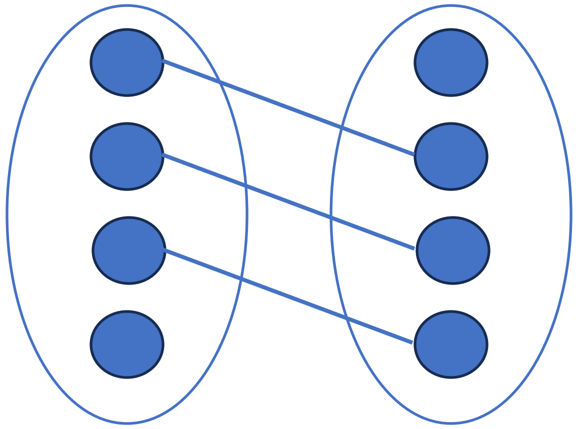
Bipartite Matching problem: Find the largest matching in a bipartite graph

We consider an O(n log n) algorithm for regular bipartite graphs (Goel, Kapralov, Khanna 2010), which is based on random walk

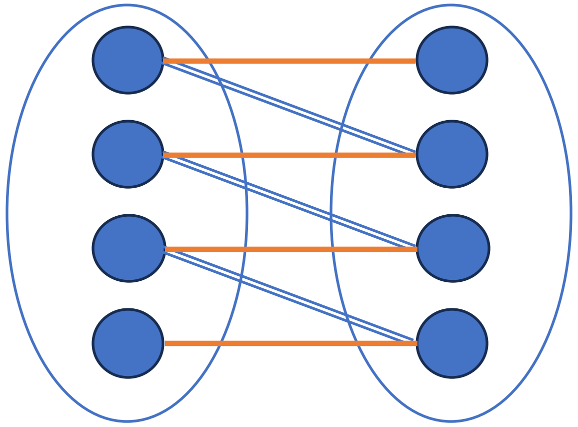
- Traditional: find augmenting paths by e.g. BFS/DFS

Augmenting path

Augmenting path for a matching M: A path in the graph that alternates between edges in M and outside of M, and starts and ends with edges outside of M

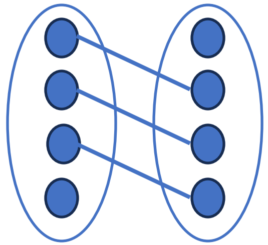


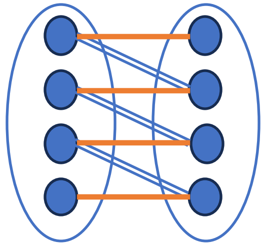
A matching M



An augmenting path for matching M

Augmenting path





Augmenting path for a matching M: A path in the graph that alternates between edges in M and outside of M, and starts and ends with edges outside of M

Berge's Theorem A matching M is maximum iff there is no augmenting path for it

Proof (sketch): If there is an augmenting path, clearly it enlarges the matching. Conversely, if there is a larger matching M', one can find an augmenting path for M using the edges of $M \cup M'$.

Remark: Given this theorem, we can find a maximum **bipartite** matching by repeatedly finding augmenting paths using e.g. DFS/BFS.

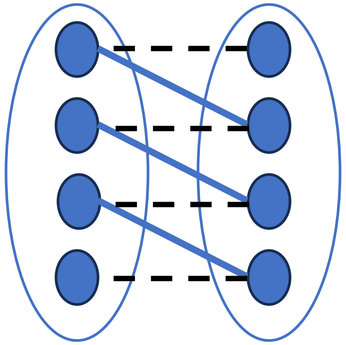
This takes $O(n)$ iterations of DFS/BFS.

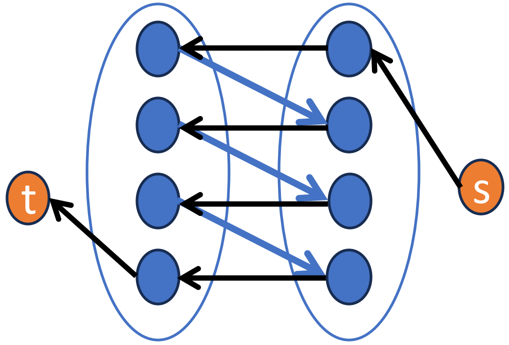
One can speed this up by finding many augmenting paths at once (Hopcroft-Karp)

A random walk for finding bipartite matching

Idea: replace BFS/DFS by a random walk (on a different **Eulerian directed graph**, i.e. indegree=outdegree for every vertex)

- Each edge in the matching points to the right
- Each edge not in the matching points to the left
- There is a source node s that connects to every unmatched vertex on the right
- A sink node t that is connected from every unmatched vertex on the left



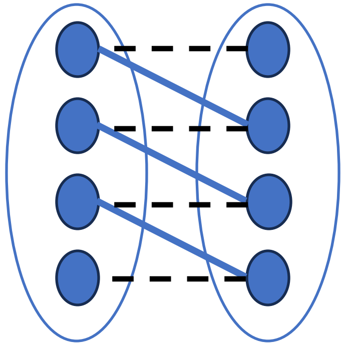


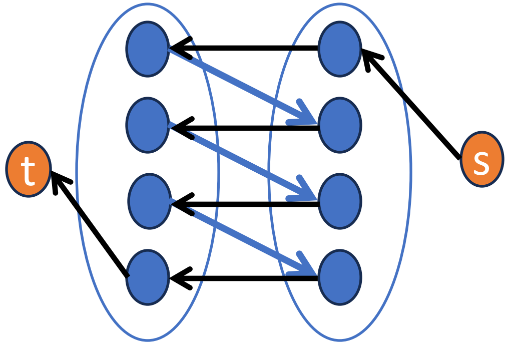
Then G_1 has an augmenting path iff G_2 has an s-t path

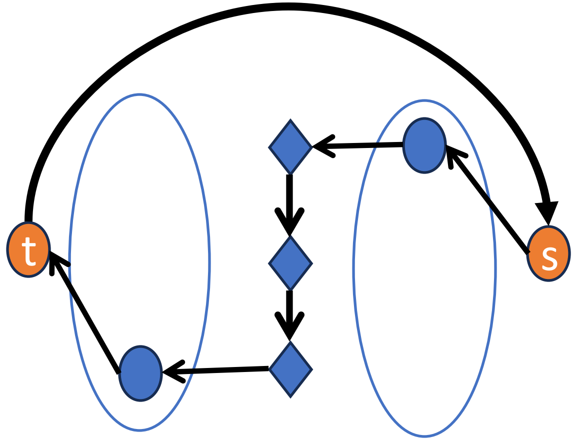
A random walk for finding bipartite matching

Idea: replace BFS/DFS by a random walk (on a different **Eulerian directed graph**, i.e. indegree=outdegree for every vertex)

- From G_2 we contract edges in the matching
- Increase parallel edges from s to every unmatched vertex v on the right to $\deg_{\text{out}}(v)$
- Increase parallel edges from every unmatched vertex u on the left to the sink t to $\deg_{\text{in}}(v)$
- Connect $\deg_{\text{in}}(t)$ parallel edges from t to s



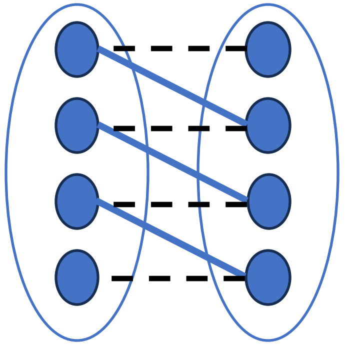


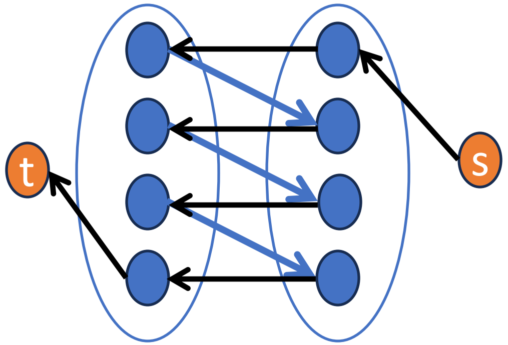


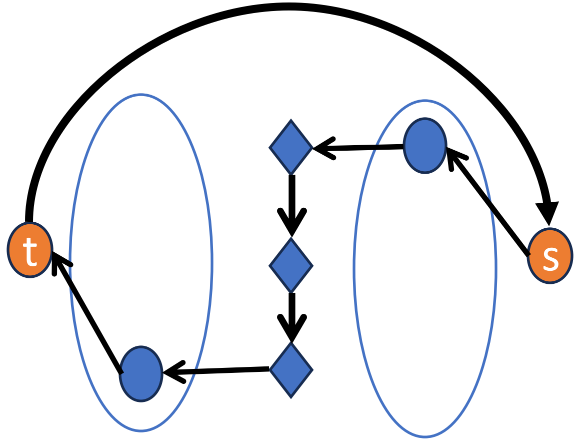
Then G_1 has an augmenting path iff G_2 has an s-t path iff G_3 has a cycle from s to s

A random walk for finding bipartite matching

Idea: replace BFS/DFS by a random walk (on a different **Eulerian directed graph**, i.e. indegree=outdegree for every vertex)







Then G_1 has an augmenting path iff G_2 has an s-t path iff G_3 has a cycle from s to s

Further, G_3 is Eulerian if G_1 is a regular graph

Question: Expected time to find a cycle from s to s, using random walk? (Return time)

Matrix Formulation of random walk

In each step, we move to a uniform random neighbor, and repeat.

Let $p_t \in \mathbb{R}^n$ be the probability distribution at time t.

Then, for all $v \in V$,

$$p_{t+1}(v) = \sum_{u:uv \in E} p_t(u) \cdot \frac{1}{\deg(u)}.$$

Let A be the adjacency matrix and D be the diagonal degree matrix.

Then $p_{t+1} = p_t(D^{-1}A)$ and thus $p_t = p_0(D^{-1}A)^t$, if p_t is a row vector.

For the spectral analysis, it will be more convenient to have p_t as a column vector.

So we write $p_{t+1} = (AD^{-1})p_t$ and $p_t = (AD^{-1})^t p_0$.

This is called a **Markov chain**, because it forgets about the past (given the current state)

Matrix Formulation of random walk

$p_{t+1} = (AD^{-1})p_t$ and $p_t = (AD^{-1})^t p_0$

Transition matrix $W := AD^{-1}$

To understand W^t, it suffices to study the powers of $\mathcal{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, because:

$$\mathcal{A} = D^{-\frac{1}{2}} W \, D^{\frac{1}{2}} \Rightarrow W^t = D^{\frac{1}{2}} \mathcal{A}^t D^{-\frac{1}{2}}$$

Further, as a real symmetric matrix $\mathcal{A} = V\Sigma V^\top$ we have:

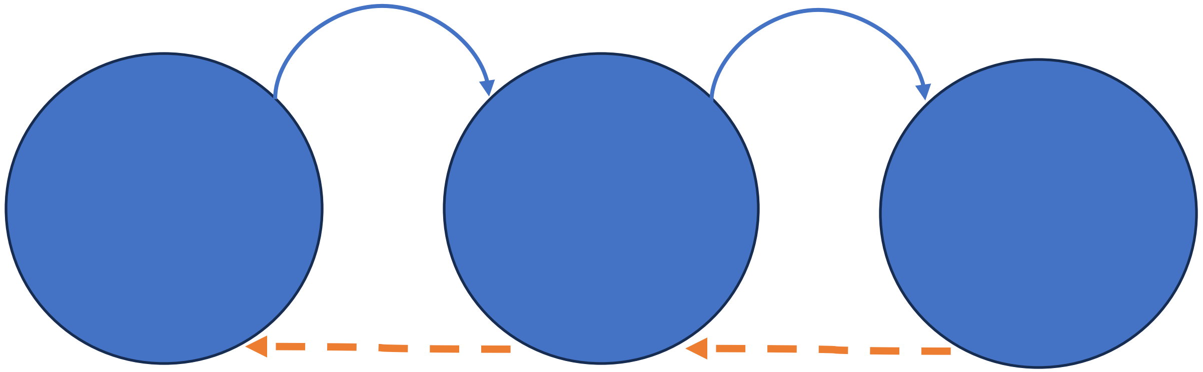
$$\mathcal{A}^k = V\Sigma^k V^\top = \sum_{1 \le i \le n} \lambda_i^k v_i v_i^\top$$

We study the convergence of the random walk, by utilizing the spectral knowledge of $\mathcal{A}$
Notice that $\mathcal{A}^k$ share the same set of eigenvectors as $\mathcal{A}$, and the eigenvalues $\lambda_i \to \lambda_i^k$

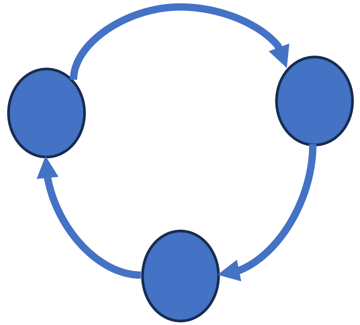
Irreducible Markov Chains

We assume all Markov chains are finite in this course.

A Markov chain is called **irreducible** if the underlying directed graph is strongly connected.



Aperiodic Markov Chains



The period of a state i is defined as $period(i) \coloneqq \gcd\{\, t \mid P_{i,i}^{t} > 0\}$.

A state is a **aperiodic** if $period(i) = 1$.

A Markov chain is **aperiodic** if all states are aperiodic; otherwise it is called periodic.

Example: What is the period of a directed 3-cyle? What about an undirected 3-cycle?
Example: What if we add a self-loop to the directed 3-cyle?

By irreducibility and aperiodicity, we have the following property.

Lemma. For any finite, irreducible, aperiodic Markov chain, there exists a $T < \infty$ such that

$(P^{t})_{i,j} > 0$ for all i, j and for all $t \geq T$.

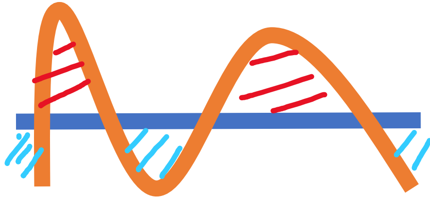
Stationary Distribution

A probability distribution $\vec{\pi}$ is a **stationary distribution** if $\vec{\pi} = \vec{\pi} \cdot P$

A stationary distribution is a “steady/equilibrium/fixed point” distribution, and $\vec{\pi} = \vec{\pi} \cdot P^t$ for any t.

A limiting distribution is a stationary distribution.

Distance and Convergence



Given two probability distributions $\vec{p}$ and $\vec{q}$, the total variation distance is defined as

$$d_{TV}(\vec{p},\vec{q}) = \frac{1}{2}\|p-q\|_1 = \frac{1}{2}\sum_{i=1}^{n}|p_i - q_i| = \max_{S\subseteq[n]}|p(S)-q(S)|\,.$$

We say $\overrightarrow{p_t}$ converges to $\vec{q}$ if

$$\lim_{t\to\infty} d_{TV}(\overrightarrow{p_t},\vec{q}) = 0.$$

Side note: we can also measure progress in the 2-norm given by

$$\left\|\frac{p}{\pi}-1\right\|_{2,\pi}^{2} := \sum_{i=1}^{n}\pi_i\left(\frac{p_i}{\pi_i}-1\right)^2$$

Similar to a "variance"

By Cauchy-Schwarz,

$$\sum_{i=1}^{n}|p_i-\pi_i| = \sum_{i=1}^{n}\pi_i\left|\frac{p_i}{\pi_i}-1\right| \le \sqrt{\sum_{i=1}^{n}\pi_i\left|\frac{p_i}{\pi_i}-1\right|^2} = \left\|\frac{p}{\pi}-1\right\|_{2,\pi}$$

This means that if 2-norm is small, so is the 1-norm (but not vice versa).

Return Time

The **return time** from i to i is defined as

$$H_i := \min\{t \geq 1 \mid X_t = i, X_0 = i\}.$$

The **expected return time** is defined as $h_i := \mathbb{E}[H_i]$.

Note that the expected return time can be seen as a special case of (expected) hitting time, where the start/end vertices are the same

Fundamental Theorem of Markov Chains

Theorem. For any finite, irreducible, aperiodic Markov chain, it holds that

1. There exists a stationary distribution $\vec{\pi}$.
2. The distribution $\overrightarrow{p_t}$ will converge to $\vec{\pi}$ as $t \to \infty$, regardless of the distribution $\overrightarrow{p_0}$.
3. There is a unique stationary distribution.
4. $\pi(i) = \frac{1}{h_i}$.

Intuition

Imagine you are playing a game of guessing where a random walk might have started, and the only information that you know is the “current location of a random walk”

Then, two random walks become indistinguishable after they meet at the same vertex, because the “current location of a random walk” can no longer be used to distinguish them

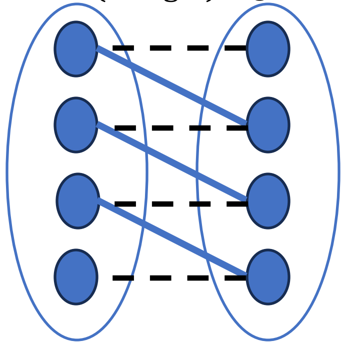
By the lemma from irreducibility and aperiodicity, after some time T, there is a non-zero probability
for two random walks to meet at the same vertex, no matter what their current vertices are.

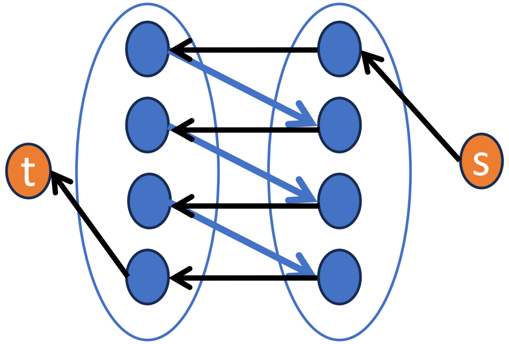
So, eventually, two random walks will meet at the same vertex with probability one,
and they would converge to the same stationary distribution.

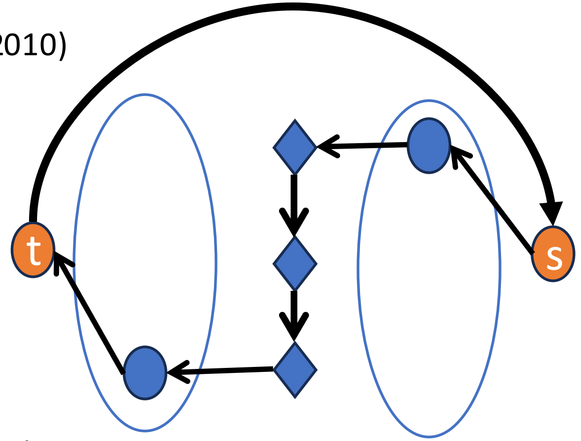
This argument can be made precise by the “coupling” technique we introduce later.

Return time for finding augmenting path

An $O(n \log n)$ algorithm for regular bipartite graphs (Goel, Kapralov, Khanna 2010)







Then G_1 has an augmenting path iff G_2 has an s-t path iff G_3 has a cycle from s to s

Further, G_3 is Eulerian if G_1 is a regular graph

Expected time to find an augmenting path => expected return time => stationary value in Eulerian directed graph

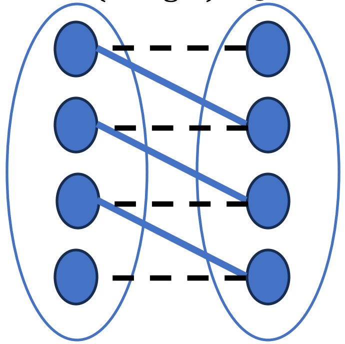
Claim: $\pi(v) = \frac{\deg_{\text{out}}(v)}{E}$ is the unique stationary distribution for an Eulerian directed graph

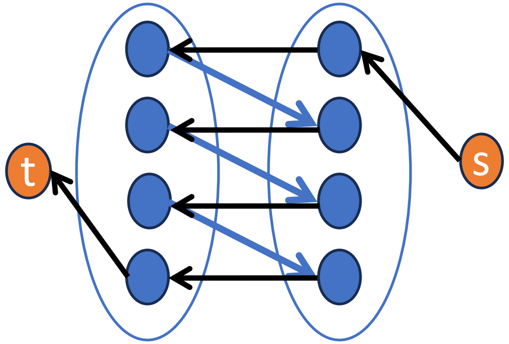
Open problem: Extension to non-regular bipartite graphs?

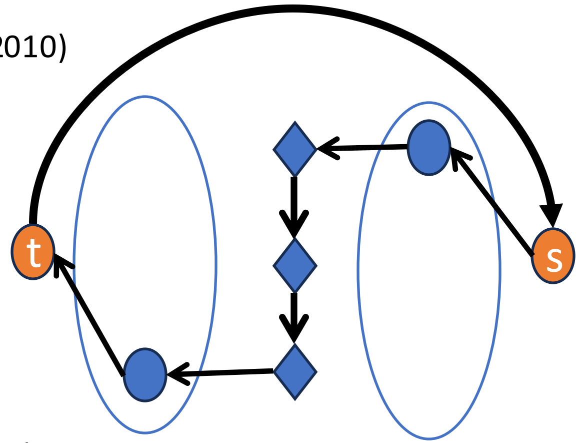
Aperiodicity: Add self-loop at t

Return time for finding augmenting path

An $O(n \log n)$ algorithm for regular bipartite graphs (Goel, Kapralov, Khanna 2010)







Then G_1 has an augmenting path iff G_2 has an s-t path iff G_3 has a cycle from s to s

Claim: $\pi(v) = \frac{\deg_{\text{out}}(v)}{E}$ is the unique stationary distribution for an Eulerian directed graph

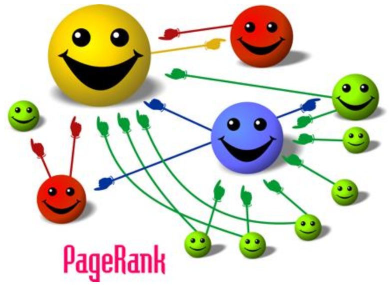
Then, by the fundamental theorem, if we are given a matching of size i

$$h_s = \frac{1}{\pi(s)} \leq O\left(\frac{nd}{d(n-i)}\right) = O\left(\frac{n}{n-i}\right)$$

We have to find at most n augmenting paths, so the total running time sums up to $\sum_{i=1}^{n} O\left(\frac{n}{n-i}\right) \leq O(n \log n)$

Open problem: Extension to non-regular bipartite graphs?

Another application: Pagerank



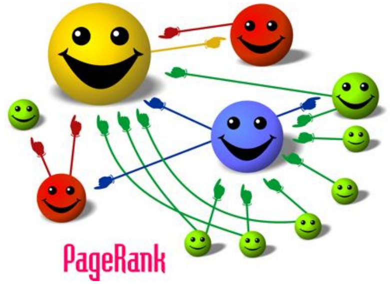
Webpages: vertices

Hyperlinks: edges

This gives a directed graph. A search engine wants to rank pages based on their "importance" or "reputation"

- A page that is being cited by many other people, it is probably more important
  - Metric: in-degree of a vertex
- A page that is being cited by other important pages, it is probably more important
  - Metric: ???

Pagerank



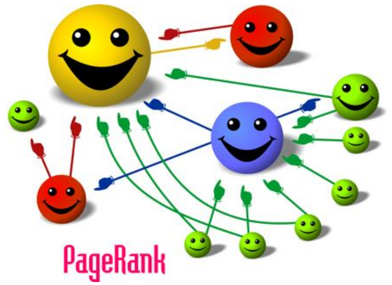
Consider the following iterative algorithm:

- Each page is initialized with a pagerank of 1/n
- Then repeat the following until convergence:
  - Each page will distribute its pagerank equally to its outgoing neighbors
  - Each page updates its pagerank by the total sum of pagerank

$$\mathrm{Pagerank}_{t+1}(j) = \sum_{i:ij\in E} \mathrm{Pagerank}_t(i)/\deg_{out}(i)$$

Let $P_{i,j} = \begin{cases} \frac{1}{\deg_{out}(i)}, & \text{if } ij \in E \\ 0, & \text{otherwise} \end{cases}$, Then, $\overrightarrow{\mathrm{Pagerank}_{t+1}} = \overrightarrow{\mathrm{Pagerank}_t} \cdot P$

Pagerank



- Can be interpreted as a random walk on directed graphs
- When the graph is finite, irreducible, aperiodic, the pagerank values are unique by the fundamental thm.
- This shows that the pagerank values is a function of the graph structure, not based on initial values.
- Also, we have some intuition about pagerank values, which are the reciprocal of the expected return time.
- Practical modification makes the graph irreducible and aperiodic, without changing the relative importance of the webpages

Application in 2-SAT

Say you are given an assignment, some clauses are violated

A random walk algorithm is to repeat until all clauses are satisfied:

- Pick an arbitrary violated clause, choose a literal uniformly at random, then flip its assignment

To analyze this algorithm, take any satisfying assignment τ, consider the evolution of $\|\tau - \sigma_t\|_1$

We would like to know the first time $\|\tau - \sigma_t\|_1 = 0$, in expectation

This is dominated by the same hitting time of a simple symmetric random walk on $[0, n] \cap \mathbb{Z}$

See Chapter 7.1.1 of *Probability and Computing* for a coupling + recursion analysis

See also **Gambler's ruin** and **optional stopping theorem** for an analysis of such hitting time

Open: Can the random walk idea give constant factor approximation to MAX-2SAT

Examples of algorithms from random walk

Finding certain objects faster

- Hitting time / return time
- Ex: Finding bipartite matching, algorithmic Lovász local lemma, 2-SAT, random 3-SAT…

Exploring graphs in space bounded computations

- Cover time
- Ex: checking undirected s-t connectivity, cat and mouse game
- Time-space trade-off (see e.g., Feige's theorem)

Rapid mixing of random walks: Markov chain Monte Carlo method

- "Local mixing" : local graph partitioning/clustering
- Mixing time
- Ex: Card shuffling, sampling random combinatorial objects, approximate counting
- Exponentially large graph, yet mixes in polynomial time $\approx O(\log N)$ where N is the size of the graph