Ref: Chapter 14 of [EK] text.
Suppose you type “UAlbany” to Google.
The web page for UAlbany is among the top few results displayed.

Search engines use automated methods to rank pages.
These methods are generally based on link analysis.
Search engines also maintain and try to get clues from a user’s search history.

Common difficulties:
- **Synonymy:** Multiple ways to describe the same thing (e.g. “scallions” vs “green onions”).
- **Polysemy:** Multiple meanings for the same word (e.g. “mercury” may refer to the planet, a car model, the chemical element or newspaper).
Information Retrieval – Then and Now

**Pre-web era:** Problem of *scarcity*.

**Example:** A lawyer searching for certain types of cases could only locate a few documents.

**Now:** Problem of *abundance*.

- The search engine should try to produce the most relevant information (from a whole lot of information).
- A popular area of research.
- **Focus:** Use of link analysis in ranking.

**Some basic issues:**

- Suppose a user types a one word query “Cornell” into a search engine.
- Are there clues within the web to suggest that cornell.edu is a good answer to the query?
Idea 1 – Voting by in-links:

- If many other pages link to cornell.edu, one can think of that page as receiving **collective endorsement**.
- Some of those pages may actually express negative opinions about cornell.edu.

Idea 2 – List finding:

- Consider the query “newspapers” to a search engine.
- There is no single “best” answer to this query.
Idea 2: List finding (continued):

- Suppose we collect a set of web pages that have the word “newspapers” and then check which pages they “endorse” (i.e., to which pages they have in-links).

- The answers typically consist of the following:
  - High scores for web pages of prominent newspapers.
  - High scores for other web pages such as Google, Amazon, Facebook, etc.

Note: Web pages for Google, Amazon, Facebook, etc. generally receive a high score no matter what the query is.
Pages that contain lists of resources relevant to a topic are also useful.

For the query “newspapers”, we may try to find pages that have lists of links to newspapers.

We can try to compute a measure that represents the value of a page as a list.

One possible measure: The list value of a page $X$ is the sum of the votes received by the pages voted for by $X$. 
Information Retrieval ... (continued)

Example (with list values):

```
10–8 / 27
```
Idea 3 – Principle of iterative improvement:

- Since pages with high list values are important, their votes should be weighted more heavily. (Endorsements from more important people should count more.)

- So, it is useful to tabulate the votes again, using the list values.

- After this, we can recompute the list values again; that is, repeat the vote count and list count steps.

- The resulting algorithm (due to Kleinberg) is called HITS (Hyperlink-Induced Topic Search).
Example (with list values and new vote counts):

- SJ Merc News: new score: 19
- Wall St. Journal: new score: 19
- New York Times: new score: 31
- USA Today: new score: 24
- Facebook: new score: 5
- Yahoo!: new score: 12

10–10 / 27
A Description of the HITS Algorithm

Definitions:

- **Authorities** for a query: Pages that are prominent and highly endorsed answers.
- **Hubs** for a query: Pages that have high list values.

Preliminary ideas:

- For each page $p$, we maintain two numerical values denoted by $\text{auth}(p)$ and $\text{hub}(p)$. Initially, $\text{auth}(p) = \text{hub}(p) = 1$.
- Two update rules are used.

1. **Authority update rule (or voting step):** For each page $p$, update $\text{auth}(p)$ to be the sum of the hub scores for all the pages that point to $p$. 
A Description of the HITS Algorithm (continued)

Update rules (continued):

2. **Hub update rule (or list finding step):** For each page $p$, update $\text{hub}(p)$ to be the sum of the authority scores for all the pages to which $p$ points.

Outline of the HITS Algorithm:

1. For each page $p$, set $\text{auth}(p) = \text{hub}(p) = 1$. Choose a value for the number of steps $k$.

2. **Repeat** the following steps $k$ times:
   - Apply the Authority update rule.
   - Apply the Hub update rule.

3. Normalize the scores and output pages in non-increasing order of their authority scores.
HITS Algorithm ... (continued)

Result produced by the HITS Algorithm:

- SJ Merc News: 0.249
- Wall St. Journal: 0.199
- New York Times: 0.304
- USA Today: 0.205
- Facebook: 0.043
- Yahoo!: 0.008
- Amazon: 0.018

Limits: SJ Merc News 0.199, Wall St. Journal 0.199, New York Times 0.304, USA Today 0.205, Facebook 0.043, Yahoo! 0.008, Amazon 0.018.
Final remarks:

- Kleinberg [1999] shows that the scores converge to appropriate limits as $k \to \infty$ (except in some degenerate cases).
- It is possible to express the HITS Algorithm as an iterative algorithm on matrices $MM^T$ and $M^T M$, where $M$ is the adjacency matrix formed by the initial pages.
- The authority scores and hub scores of pages converge to specific eigenvectors of $MM^T$ and $M^T M$ respectively.
- The resulting authority and hub scores represent a form of equilibrium (under the authority update and hub update rules).
HITS Algorithm works well in commercial contexts where competing firms don’t (generally) link to each other.

In other contexts (e.g. academic pages, scientific literature), page rank algorithm generally outperforms the HITS Algorithm.

Page rank computation uses ideas similar to those of HITS:

- A page rank update rule.
- Idea of iterative improvement.

A physical model for page rank:

- Think of page rank as a \textit{fluid} that circulates through the links of the web network.

- The fluid accumulates at nodes that are “most important”.
Notation: For any page $u$, 
- $\text{PR}(u)$ denotes its page rank.
- $\text{OD}(u)$ denotes its outdegree.

Outline of the algorithm:

1. Let $n$ denote the number of pages. For each node $u$, let $\text{PR}(u) = 1/n$.
2. Choose a value for $k$ (the number of iterations).
3. Repeat the following step $k$ times:
   - Apply the Basic Page Rank Update Rule to all the nodes in parallel.
Basic Page Rank Update Rule: For any node $u$,

1. **(Flow generation step)**
   - If $\text{OD}(u) = 0$ then $u$ sends $\text{PR}(u)$ to itself.
   - If $\text{OD}(u) \geq 1$, then $u$ sends $\text{PR}(u)/\text{OD}(u)$ along each of its outgoing edges.

2. **(Flow accumulation step)**
   - Suppose node $u$ has $r$ incoming edges and the flow along the $i^{th}$ edge is $\alpha_i$.
   - If $\text{OD}(u) = 0$, then $\text{PR}(u) = \text{PR}(u) + \alpha_1 + \alpha_2 + \cdots + \alpha_r$.
   - If $\text{OD}(u) \geq 1$, then $\text{PR}(u) = \alpha_1 + \alpha_2 + \cdots + \alpha_r$. 
Examples for the flow generation step:

Example 1:

- Suppose $PR(u) = 1/2$.
- Since $OD(u) = 3$, $u$ sends $1/6$ along each of the three outgoing edges.

Example 2:

- Suppose $PR(u) = 1/10$.
- Since $OD(u) = 0$, $u$ sends $1/10$ to itself.
Examples for the flow accumulation step:

Example 3:

Here, $OD(u) = 0$.

Let the current value of $PR(u)$ be $1/10$.

New value of $PR(u) = \frac{1}{10} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} = \frac{53}{60}$.

Example 4:

Let the current value of $PR(u)$ be $1/10$.

Since $OD(u) = 2$, $u$ has already sent $\frac{1}{20}$ to each of $a$ and $b$.

New value of $PR(u) = \frac{1}{3} + \frac{1}{4} + \frac{1}{5} = \frac{47}{60}$.
A more detailed example:

- Initially, \( \text{PR}(a) = \text{PR}(b) = \text{PR}(c) = \text{PR}(d) = \frac{1}{4} \).
- Each node has outdegree > 0. So, in every step, each node sends out its page rank along the outgoing edges.

**Step 1:**

- Node \( a \) receives 1/8 from \( b \) and 1/4 from \( c \). So, \( \text{PR}(a) = \frac{1}{8} + \frac{1}{4} = \frac{3}{8} \).
- Node \( b \) receives 1/8 each from \( a \) and \( d \). So, \( \text{PR}(b) = \frac{1}{8} + \frac{1}{4} = \frac{3}{8} \).
- Node \( c \) receives 1/8 from \( a \). So, \( \text{PR}(c) = \frac{1}{8} \).
- Node \( d \) receives 1/8 from \( b \). So, \( \text{PR}(d) = \frac{1}{8} \).
A more detailed example (continued):

At the end of Step 1, \( \text{PR}(a) = \text{PR}(b) = \frac{3}{8} \) and \( \text{PR}(c) = \text{PR}(d) = \frac{1}{8} \).

Step 2:

- Node \( a \) receives \( \frac{3}{16} \) from \( b \) and \( \frac{1}{8} \) from \( c \). So, \( \text{PR}(a) = \frac{3}{16} + \frac{1}{8} = \frac{5}{16} \).
- Node \( b \) receives \( \frac{3}{16} \) from \( a \) and \( \frac{1}{8} \) from \( d \). So, \( \text{PR}(b) = \frac{3}{16} + \frac{1}{8} = \frac{5}{16} \).
- Node \( c \) receives \( \frac{3}{16} \) from \( a \). So, \( \text{PR}(c) = \frac{3}{16} \).
- Node \( d \) receives \( \frac{3}{16} \) from \( b \). So, \( \text{PR}(d) = \frac{3}{16} \).
Table showing successive page rank values:

<table>
<thead>
<tr>
<th>Step</th>
<th>PR(a)</th>
<th>PR(a)</th>
<th>PR(a)</th>
<th>PR(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>1</td>
<td>3/8</td>
<td>3/8</td>
<td>1/8</td>
<td>1/8</td>
</tr>
<tr>
<td>2</td>
<td>5/16</td>
<td>5/16</td>
<td>3/16</td>
<td>3/16</td>
</tr>
</tbody>
</table>

Remarks:

- There is no normalization here; the total page rank is always 1.
- It can be shown that (except for degenerate cases), the page rank values converge to a limit as $k \to \infty$. 
Example for equilibrium state (or fixed point):

- Suppose $\text{PR}(a) = 0$, $\text{PR}(b) = 1/2$ and $\text{PR}(c) = 1/2$.
- These values won’t change; that is, this is an equilibrium state.

Another form of equilibrium:

- Initially, let $\text{PR}(a) = 0$, $\text{PR}(b) = 3/4$ and $\text{PR}(c) = 1/4$.
- At the end of Step 1: $\text{PR}(a) = 0$, $\text{PR}(b) = 1/4$ and $\text{PR}(c) = 3/4$.
- At the end of Step 2: $\text{PR}(a) = 0$, $\text{PR}(b) = 3/4$ and $\text{PR}(c) = 1/4$ (which is the initial state).

Remark: If the network is strongly connected, it can be shown that there is a **unique** equilibrium state.
A drawback: In some networks, the page rank update rule allows “wrong” nodes to end up with all the page rank.

- One would expect node \( a \) to have a high page rank.
- However, the current page rank update rule cause all the page rank to flow out of \( a \).
- All the page rank accumulates at \( g \) and \( h \); it doesn’t flow back to the other nodes.

Remedy: Modify the page rank update rule.
Scaled Page Rank Update Rule

**Steps:**

1. Pick a **scaling factor** $s$, where $0 < s < 1$.
2. Apply the basic page rank update rule.
3. Scale down **all** the page rank values by the factor $s$.
   (This step reduces the total page rank from 1 to $s$.)
4. Divide the residual $1 - s$ units of page rank equally among the $n$ nodes; that is, add $(1 - s)/n$ units of page rank to each node.
   (This step restores the total page rank value to 1.)

**Remarks:**

- It can be shown that (except for degenerate cases), the scaled page rank values converge to a limit as $k \to \infty$.
- It is believed that the value of $s$ used by Google is in the range $0.8$ to $0.9$. 
Basic page rank update rule and random walks:

1. Suppose we have $n$ web pages $p_1, p_2, \ldots, p_n$.
2. Choose an initial page: each page is chosen with probability $= 1/n$. Let $p_i$ be the chosen page.
3. **Repeat** $k$ times:
   - Suppose the OD($p_i$) $= r$.
     - If $r = 0$, stay at $p_i$ itself.
     - If $r \geq 1$, choose one of the outgoing edges of $p_i$ with probability $= 1/r$.
     - Update $p_i$ to the other end point of that edge.

**Theorem:** For each $i$, $1 \leq i \leq n$, the probability that the above random walk is at node $p_i$ is equal to the page rank of $p_i$ after $k$ applications of the basic page rank update rule.
Notes:

- The random walk approach provides another way to estimate page ranks.
- The approach can also be extended to the **scaled page rank update rule**. In the body of the loop for Step 3, do the following:
  - With probability $s$ (the chosen scale factor) continue the random walk as before.
  - With probability $1 - s$ choose another node, say $p_j$, with all nodes being equally likely and continue the random walk from $p_j$.
- Search engine companies are generally very secretive about the exact methods they use for computing page ranks.