PageRank & HITS (circa 1998)
Ranking Webpages by Popularity

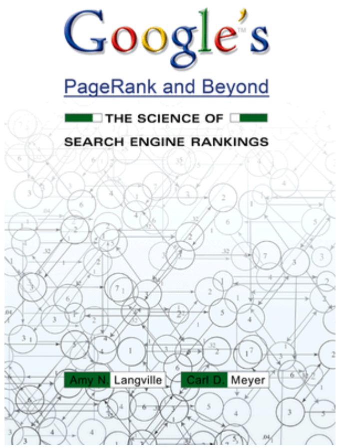
- Concepts underlying PageRank (Google) & HITS (Teoma, Ask) algorithms
  - ▷ Web graph (directed graph, nodes: webpages, directed arcs, edges: hyperlinks)
  - ▷ inlink (hyperlinks pointing into a webpage)
  - ▷ outlink (hyperlinks pointing out of a webpage)
- PageRank & HITS assign a score to each webpage, a measure of its popularity and relevance to a search query

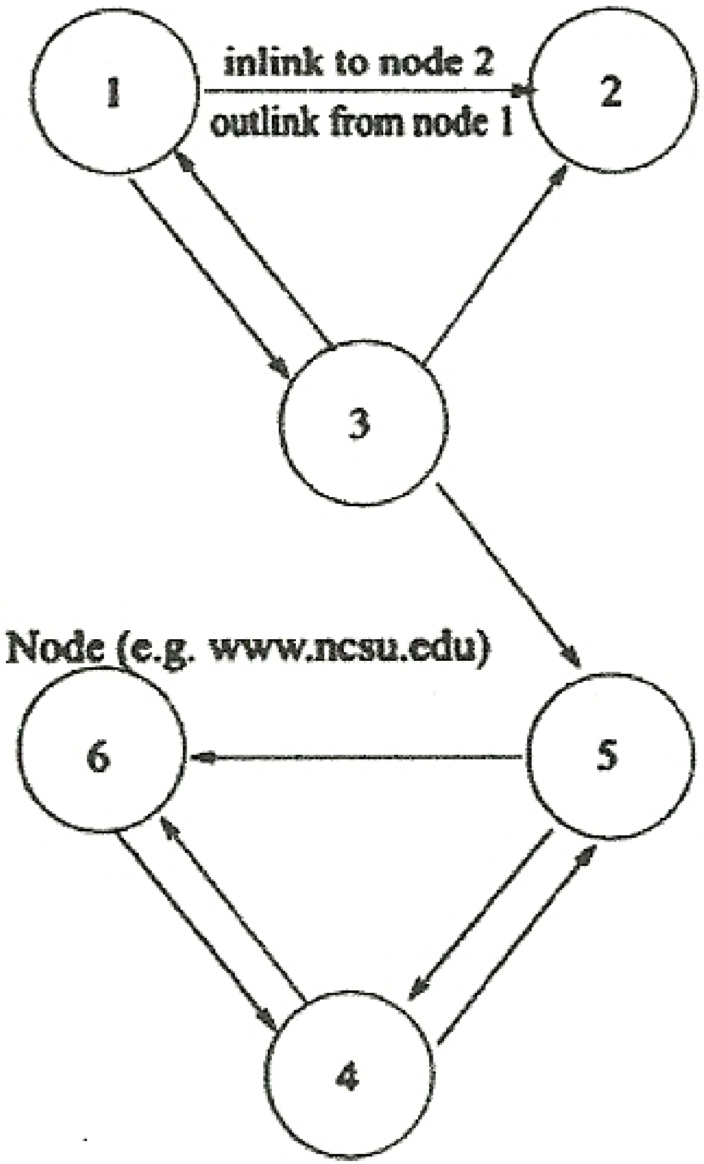
Google's PageRank and Beyond:

The Science of Search Engine Rankings

by Amy N. Langville & Carl D. Meyer

PUP 2006




inlink to node 2
outlink from node 1
1
2
3
Node (e.g. www.ncsu.edu)
6
5
4

PageRank thesis

A webpage is important if it is pointed to by other important pages.

PageRank assigns a score to each webpage.

Comparing the PageRank scores of two pages gives an indication of the relative importance of the two pages.

Google Toolbar

Displays the PageRank score as an integer from 0 to 10.
Most important pages receive a score of 10.

Query-Independence

- A webpage ranking is called **query-independent** if the popularity score for each webpage is determined off-line and remains constant (until the next update) regardless of the query.
- At query time, no time is spend computing the popularity scores for relevant pages. These scores are found by table lookup, in the previously computed popularity table.
- PageRank is query-independent, i.e. it produces a global ranking of the importance of all pages in Google's index ($\approx$ 8.1b pages)
- HITS is query-dependent, in its original version.
- Both PageRank, HITS can be modified to become q-dep, q-indep resp.

PageRank Mathematical Formalism

- PageRank equation: (derived from bibliometrics research: analysis of citation structure among scientific research papers)

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}$$

  - ▷ $r(P_i)$ is the PageRank of page P_i
  - ▷ B_{P_i} is the set of pages pointing into P_i
  - ▷ $|P_j|$ is the number of outlinks from page P_j

- The values $r(P_j)$ (PageRanks of pages inlinking to P_i) are unknown.
- Iterative procedure, Assumption: $r(P_i) = \frac{1}{n}, \; i = 1, \ldots, n$

- $r_{k+1}(P_i)$ is the PageRank of page P_i at iteration $k+1$

- iterative PageRank equation:

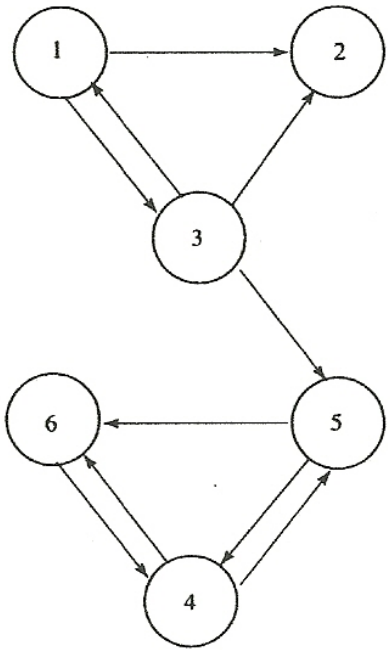
$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|}$$

  with initialization values: $r_0(P_i) = \frac{1}{n},\ i = 1, \ldots, n$

- n is the total number of pages indexed.

- The process is applied iteratively, hoping that it will eventually converge to some stable values,
  i.e. the PageRank scores of all pages P_i.

PageRank Toy Example



Iteration 0	Iteration 1	Iteration 2	Rank at Iter. 2
$r_0(P_1) = 1/6$	$r_1(P_1) = 1/18$	$r_2(P_1) = 1/36$	5
$r_0(P_2) = 1/6$	$r_1(P_2) = 5/36$	$r_2(P_2) = 1/18$	4
$r_0(P_3) = 1/6$	$r_1(P_3) = 1/12$	$r_2(P_3) = 1/36$	5
$r_0(P_4) = 1/6$	$r_1(P_4) = 1/4$	$r_2(P_4) = 17/72$	1
$r_0(P_5) = 1/6$	$r_1(P_5) = 5/36$	$r_2(P_5) = 11/72$	3
$r_0(P_6) = 1/6$	$r_1(P_6) = 1/6$	$r_2(P_6) = 14/72$	2

Matrix Representation of PageRank Equation

- the Σ-PageRank equations compute PageRanks one page at a time.
- the matrix-PageRank equation computes a **PageRank vector**.
- this is a $1 \times n$ row vector π^T that holds the PageRank values of all pages in the index. (T denotes transposition)
- the **hyperlink matrix** H is a $n \times n$ square matrix with

$$H_{ij} = \begin{cases} 1/|P_i| & \text{if there is a link from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

- H exhibits the same non-zero element structure as the adjacency matrix of the graph, but the non-zero elements are probabilities.

Matrix-PageRank Toy Example

$$
\mathbf{H} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{array}
\begin{array}{c} \begin{array}{cccccc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \end{array} \\
\begin{pmatrix}
0 & 1/2 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1/2 & 0 & 1/2 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix} \end{array}.
$$

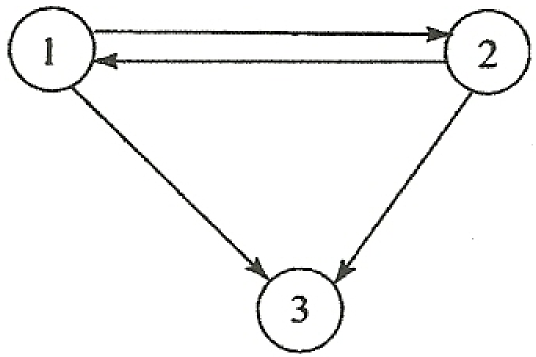
- In H,
  the non-zero elements of row i corresp. to the outlinking pages of P_i
  the non-zero elements of col i corresp. to the inlinking pages of P_i
- $\pi^{(k)T}$ denotes the PageRank vector at the k^{th} iteration.
- Matrix-PageRank equation: $\pi^{(k+1)T} = \pi^{(k)T} H$ Power Method
  - ▷ Each iteration requires one vector-matrix multiplication, $O(n^2)$
  - ▷ H is a very **sparse** matrix, lots of 0 elements, most pages link to only a few other pages, v-m mult. complexity: $O(nnz(H))$
  - ▷ estimate: the average webpage has about 10 outlinks $\rightsquigarrow$ H has approx. $10n$ non-zero elements $\rightsquigarrow$ v-m mult. complexity: $O(n)$
  - ▷ **nondangling nodes**: row sums are equal to 1: **stochastic rows**
  - ▷ **dangling nodes**: (pages w/out any outlinks) create rows of n zeros: H **substochastic**

Issues with the Matrix Representation

- will the iterative process converge?
- under what properties of H is it guaranteed to converge?
- will it converge to something sensible in the PageRank context?
- will it converge to just one vector or multiple vectors?
- does the convergence depend on the starting vector $\pi^{(0)T}$?
- how many iterations can we expect, to achieve convergence?

The Rank Sinks Problem

- Let e^T be the $1 \times n$ row vector of 1's
- Start the iterations with $\pi^{(0)T} = \dfrac{1}{n} e^T$
- **rank sinks** are those pages that accumulate more and more PageRank at each iteration, monopolizing the scores and refusing to share.



- the dangling node 3 is a rank sink
- the cluster of nodes $4, 5, 6$ is a rank sink, $\pi^{(13)T} = [0, 0, 0, \frac{2}{3}, \frac{1}{3}, \frac{1}{5}]$

The Cycles Problem

- page 1 points only to page 2 and vice versa, infinite loop, cycle

- the iterates will flip-flop indefinitely, there is no convergence

Overcoming Rank Sinks, Cycles Markov Chains Theory

- For any starting vector, the power method applied to a Markov matrix P converges to a unique positive vector called the **stationary vector**, as long as P is **stochastic**, **irreducible** and **aperiodic**.
- We can overcome convergence problems caused by rank sinks and cycles, if H is modified slightly, so that it is a Markov matrix with the desired properties.

(1) A unique positive PageRank vector exists when the **Google matrix** is stochastic and irreducible.

(2) In addition, if the Google matrix is aperiodic, then the (iterative) power method will converge to this PageRank vector, regardless of the starting vector.

Adjustments to the basic model

- Notion of a **Random Surfer (RS)**
  Bounces along randomly following the hyperlink structure of the Web. Arrives at a page, chooses randomly a hyperlink and follows it. Continues this random decision process indefinitely.
- In the long run, the proportion of time the RS spends on a given page is a measure of the importance of that page.
- Pages that the random surfer revisits often must be important, because they must be pointed to by other important pages.
- PB The RS gets trapped upon entering a dangling node (.pdf file, image file, data table, etc)
- FIX **stochasticity adjustment**
  replace all 0^T rows in H by $\frac{1}{n}e^T$ rows $\quad \rightsquigarrow \quad$ stochastic matrix

- Effect of the stochasticity adjustment on the RS after entering a dangling node: they can visit any page at random.
-

$$\text{stochasticity adjustment:} \qquad \boldsymbol{S} = \boldsymbol{H} + \boldsymbol{a}\left(\frac{\boldsymbol{1}}{\boldsymbol{n}}\boldsymbol{e}^{\boldsymbol{T}}\right)$$

where a is the $n \times 1$ column **dangling node vector** $a_i = 1$ if P_i is a dangling node, 0 otherwise.

- S is created from a **rank-one update** $\left(\frac{1}{n}e^T\right)$ to H.

$$\mathbf{S} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

- **primitivity adjustment** RS argument: occasionally, the RS gets bored and decides to visit a page by entering a URL in the browser.
-

$$\text{mathematical model:} \qquad \boldsymbol{G} = \boldsymbol{\alpha S} + (\mathbf{1} - \boldsymbol{\alpha})\left(\frac{\mathbf{1}}{\boldsymbol{n}}\boldsymbol{e}\boldsymbol{e}^{\boldsymbol{T}}\right)$$

  where α is a number in $[0,1]$ and G is the **Google matrix**.
- the parameter α controls the proportion of time the RS follows hyperlinks vs URLing.
- suppose $\alpha = 0.6$, then 60% of the time the RS follows hyperlinks and the other 40% of the time uses a URL to visit a new page randomly.
- URLing is random, because the corresp. matrix $\boldsymbol{E} = \left(\frac{\mathbf{1}}{\boldsymbol{n}}\boldsymbol{e}\boldsymbol{e}^{\boldsymbol{T}}\right)$ is uniform, i.e. the RS is equally likely to jump to any page.

Consequences of the Adjustments

- G is stochastic, convex combination of two stochastic matrices S, E
- G is irreducible,
- G is aperiodic, $G_{ii} > 0$
- G is primitive, $G > 0$

and therefore the power method converges to a unique PageRank vector.

G is completely dense, but can be written as a rank-one update to the very sparse H.

For the twice-modified G, a unique PageRank vector exists and is a remarkably good way of assigning global importance value to webpages.

Notation for the PageRank Problem

$\mathbf{H}$	very sparse, raw substochastic hyperlink matrix
$\mathbf{S}$	sparse, stochastic, most likely reducible matrix
$\mathbf{G}$	completely dense, stochastic, primitive matrix called the Google Matrix
$\mathbf{E}$	completely dense, rank-one teleportation matrix
n	number of pages in the engine's index = order of $\mathbf{H}$, $\mathbf{S}$, $\mathbf{G}$, $\mathbf{E}$
α	scaling parameter between 0 and 1
π^T	stationary row vector of $\mathbf{G}$ called the PageRank vector
$\mathbf{a}^T$	binary dangling node vector

Google's Adjusted PageRank Toy Example

- Google's adjusted PageRank is the power method $\pi^{(k+1)T} = \pi^{(k)T} G$ applied to the Google matrix G.
- Set $\alpha = 0.9$ and compute $G = 0.9H + (0.9a + 0.1e)\frac{1}{6}e^T$
- Google's PageRank vector is the stationary vector of G given by

$$\begin{array}{cccccccc} & & 1 & 2 & 3 & 4 & 5 & 6 \\ \pi^T & = & (.03721 & .05396 & .04151 & .3751 & .206 & .2862) \end{array}$$

- So the 6 webpages can be ranked by their importance as $(4\,6\,5\,2\,3\,1)$, i.e. page 4 is the most important and page 1 is the least important, according to the PageRank definition of importance.

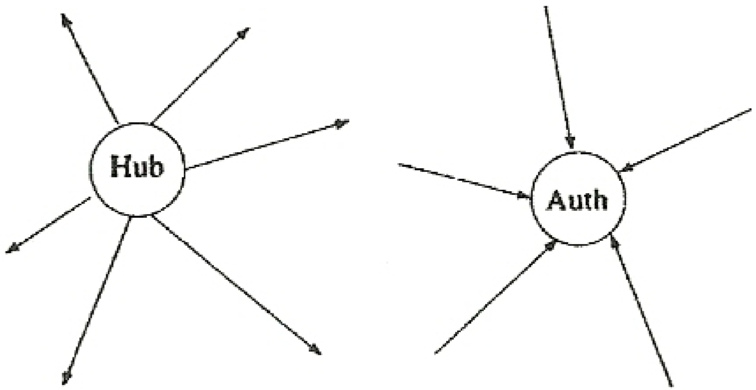
$$\mathbf{G} = .9\mathbf{H} + \left(.9\begin{pmatrix}0\\1\\0\\0\\0\\0\end{pmatrix} + .1\begin{pmatrix}1\\1\\1\\1\\1\\1\end{pmatrix}\right) 1/6\begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}$$

$$= \begin{pmatrix} 1/60 & 7/15 & 7/15 & 1/60 & 1/60 & 1/60 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 19/60 & 19/60 & 1/60 & 1/60 & 19/60 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 7/15 & 7/15 \\ 1/60 & 1/60 & 1/60 & 7/15 & 1/60 & 7/15 \\ 1/60 & 1/60 & 1/60 & 11/12 & 1/60 & 1/60 \end{pmatrix}.$$

HITS theses

A **hub** is a webpage containing many outlinks

An **authority** is a webpage containing many inlinks `link:` feature



(1) A webpage is a good hub[a] if it points to good authorities.

(2) A webpage is a good authority[b] if it is pointed to by good hubs.

---

[a] and therefore deserves a high hub score

[b] and therefore deserves a high authority score

- A webpage can be both a hub and an authority.
- HITS uses the Web's hyperlink structure to assign popularity scores to webpages.
- HITS assigns **two** scores to each webpage: authority score & hub score.
- Good authorities are pointed to by good hubs.
- Good hubs point to good authorities.
- HITS acronym (Hypertext Induced Topic Search)

HITS Mathematical Formalism

- every page i has both an authority score x_i and a hub score y_i

- let E denote the set of all directed edges in the web graph

- let e_{ij} denote the directed edge from node i to node j

- each page (node) has been assigned an initial a-score $x_i^{(0)}$ and an initial h-score $y_i^{(0)}$

- HITS successively refines these initial scores by computing:

$$x_i^{(k)} = \sum_{j:e_{ji} \in E} y_j^{(k-1)} \text{ and } y_i^{(k)} = \sum_{j:e_{ij} \in E} x_j^{(k)} \text{ for } k = 1, 2, 3, \ldots$$

Matrix Representation of HITS Equations

$$x^{(k)} = L^T y^{(k-1)} \text{ and } y^{(k)} = L x^{(k)}$$

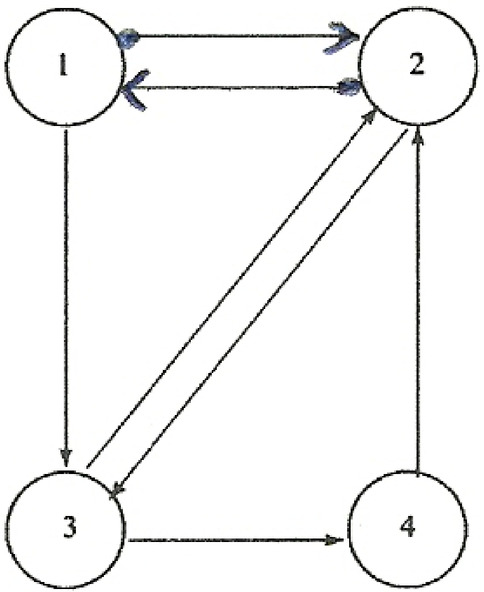
- L is the adjacency matrix of the directed web graph

$$L_{ij} = \begin{cases} 1, & \text{if there exists an edge from node } i \text{ to node } j \\ 0, & \text{otherwise} \end{cases}$$

- $x^{(k)}$ and $y^{(k)}$ are $n \times 1$ vectors holding the current (at each iteration) a-scores and h-scores for each webpage.

- Note that L is sparse.

HITS Toy Example



$$\mathbf{L} = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccc} P_1 & P_2 & P_3 & P_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}.$$

Original HITS Algorithm

1. Initialize $y^{(0)} = e$, where e is a column vector of all ones.

2. Set $k = 1$.

3. Repeat

$$x^{(k)} = L^T y^{(k-1)}$$

$$y^{(k)} = L x^{(k)}$$

$$k = k + 1$$

Until convergence

- Note that substitution simplifies the two HITS matrix equations to:

$$x^{(k)} = L^T L x^{(k-1)}$$
$$y^{(k)} = L L^T y^{(k-1)}$$

- Iterative Power method for the matrices $L^T L$ and LL^T
- Compute the **dominant eigenvectors** of $L^T L$ and LL^T
- $L^T L$ is the **authority matrix**, determines the a-scores
- LL^T is the **hub matrix**, determines the h-scores
- These are both sparse symmetric positive semidefinite matrices

HITS Implementation

Two main phases:

1. build a **neighborhood graph N** based on the query terms

2. compute the authority and hub scores for each page in **N** and establish two ranked lists accordingly

Construction of the neighborhood graph **N**:

All pages containing references to the query terms are put into **N**.

To determine these pages consult the **inverted file index**.

term 1 (lion)	n_1, n_2, n_3
term 2 (aztec)	$n_1, n_4, n_5, n_6, n_7, n_8$
$\vdots$	$\vdots$
term m (car)	$\ldots$

For each term, all pages mentioning this term are stored in a list.

A query on terms 1 and 2 would result in placing pages $n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8$ into $\mathbf{N}$.

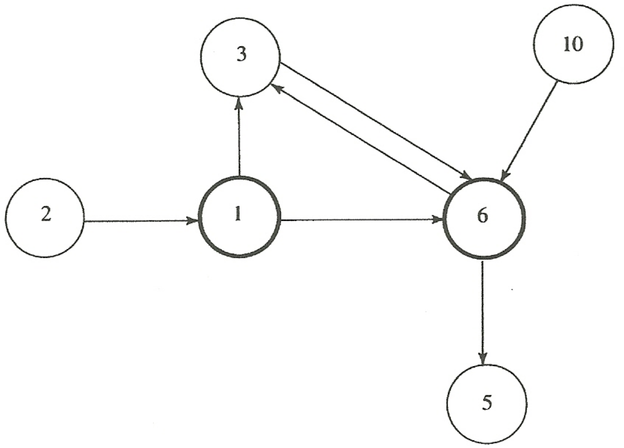
Expand $\mathbf{N}$ by adding nodes that point either to or from nodes in $\mathbf{N}$.

Form the adjacency matrix L, corresponding to the nodes in $\mathbf{N}$.

This is a much smaller matrix than the matrix L corresponding to all the nodes in the web graph.

In addition, we can compute either the a-scores vector or the h-scores vector, since they are related by $y = Lx$.

HITS Toy Example



$$\mathbf{L} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 10 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$\mathbf{L}^T\mathbf{L} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 10 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad \text{and} \quad \mathbf{L}\mathbf{L}^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 10 \end{array} \\ \begin{pmatrix} 2 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$