Lecture 06 - Matrix Re-Ordering

June 18, 2025

Outline

Outline

Key Idea

Symmetric Permutations

- The special case of symmetric permutation is of particular importance.
- A symmetric permutation is when we replace A with PAP^T, i.e., we permute the rows and columns in the same way ($Q = P^T$).
- Symmetric permutations naturally preserve symmetry for symmetric matrices (more generally when the sparsity pattern is symmetric).

Key Idea

Q: What does a symmetric permutation do to the graph of A?
A: The structure will be unchanged; the nodes will be re-numbered.
Given a particular reordering, what is P for a symmetric permutation? Reordering gives a list of before/after labels, e.g.,

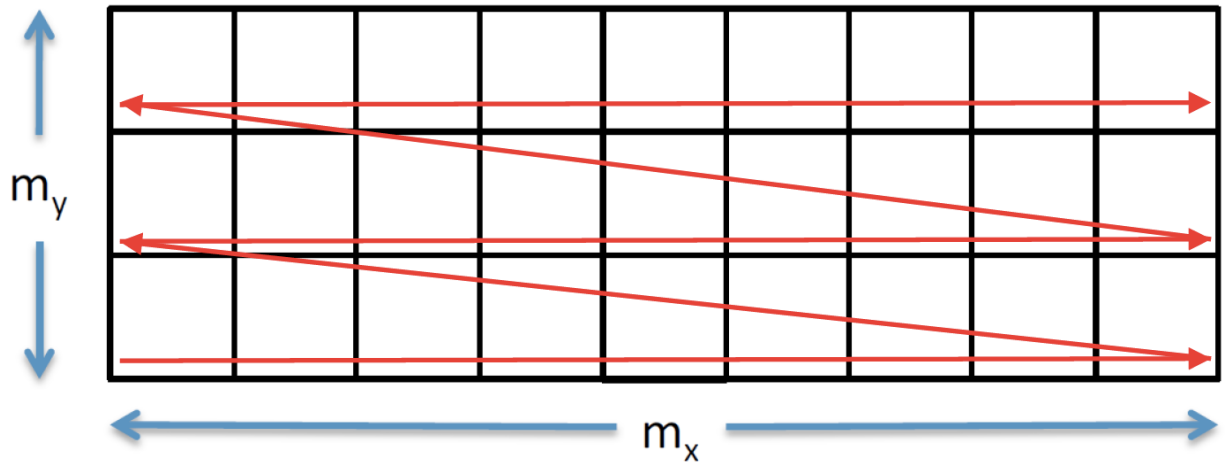
$$1 \to 2,$$
$$2 \to 3,$$
$$3 \to 1,$$
$$4 \to 4.$$

This says which old row moves to which new row. We apply the desired row swaps to the identity matrix I to get the permutation P. In the example above,

$$I = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \quad \Longrightarrow \quad P = \begin{bmatrix} & & 1 & \\ 1 & & & \\ & 1 & & \\ & & & 1 \end{bmatrix}.$$

Example with Natural Ordering

Idea: We saw earlier that bandwidths are preserved when we factorize A. Thus, when we have some choice of how to write A, making a choice which affords **minimum bandwidth** is desirable. Consider the 2D Laplacian matrix on a non square domain, with $m_x \gg m_y$, and natural rowwise ordering. This domain and rowwise ordering (first along x-axis, then y-axis) is depicted below.

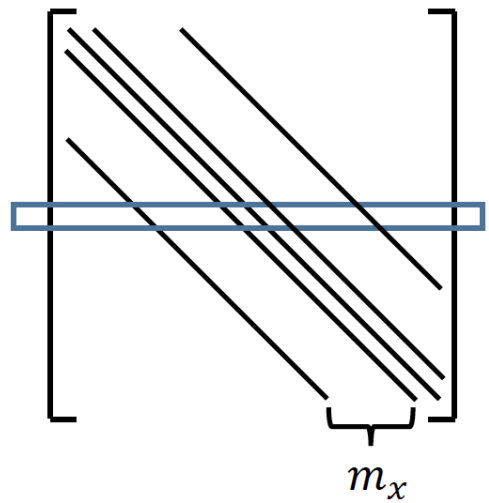


Example with Natural Ordering

What is the bandwidth of the 2D Laplacian matrix with natural rowwise ordering? Consider row i, which has entries at columns

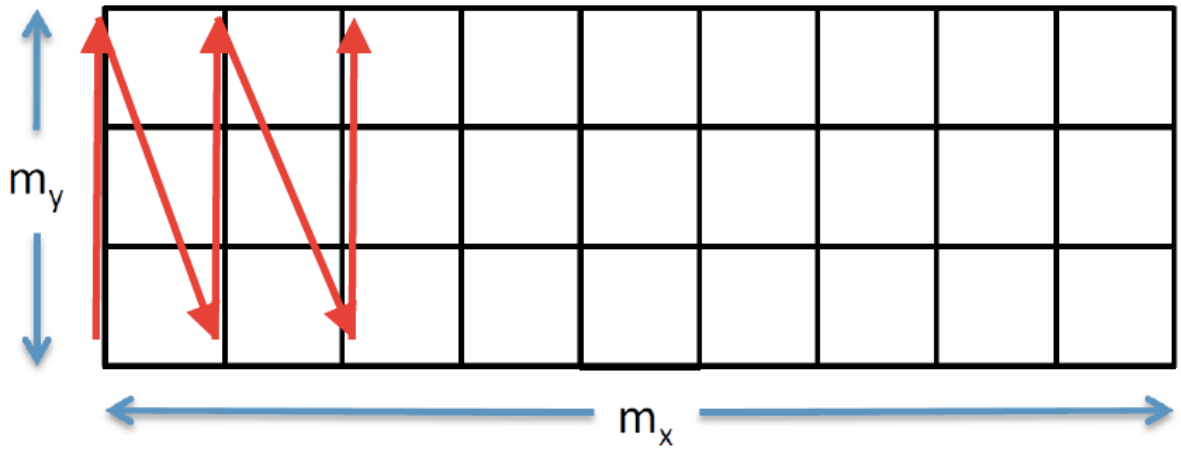
- i (diagonal),
- $i-1, i+1$ (inner bands),
- $i-m_x, i+m_x$ (outer bands).

Hence, the bandwidth is m_x.



Example with Natural Ordering

Consider instead columnwise ordering along y-axis first, and then x-axis. In this case, the bandwidth becomes m_y instead!



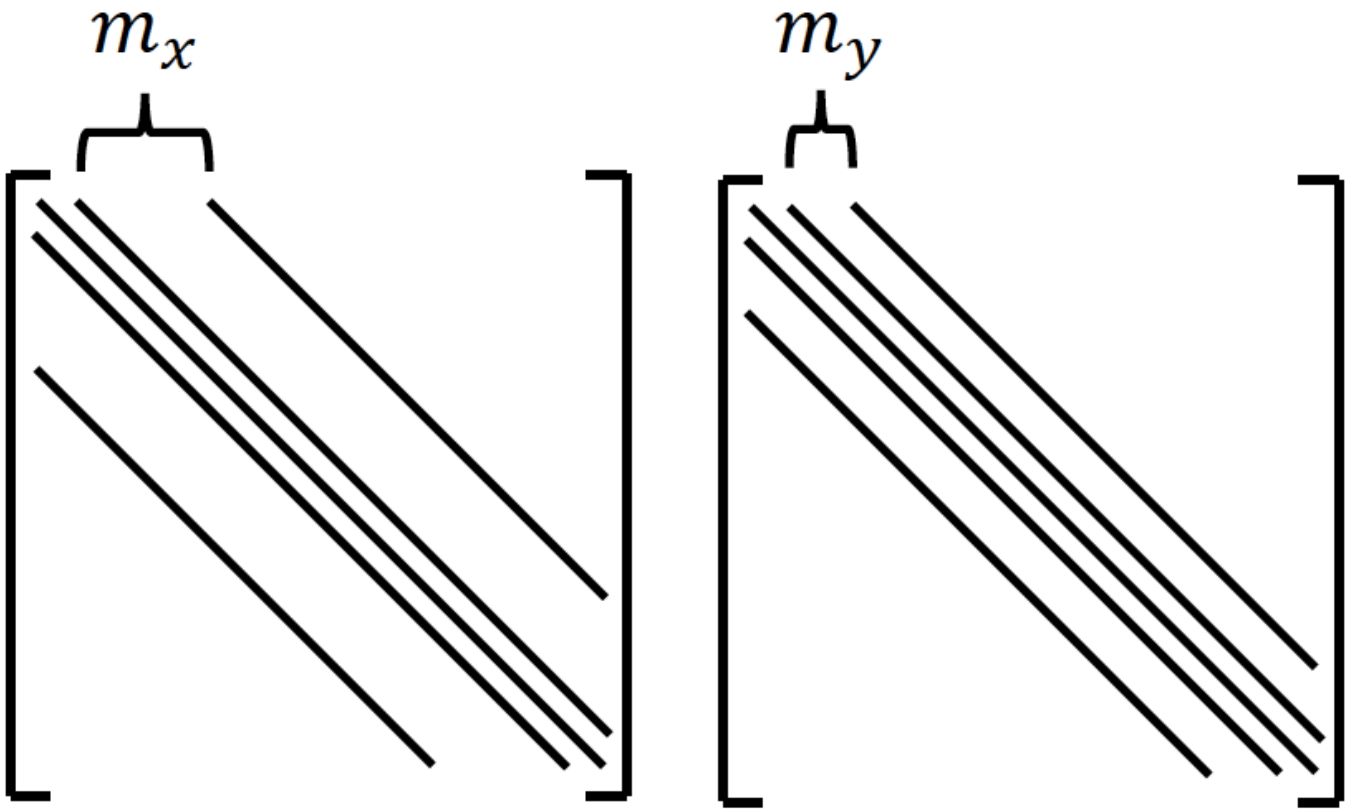
Consider row j, which has entries at columns

- j (diagonal),
- $j-1, j+1$ (inner bands),
- $j-m_y, j+m_y$ (outer bands).

Hence, the bandwidth is m_y.

Example with Natural Ordering

Ordering along the y-axis first gives narrower bandwidth (in this case) since $m_x \gg m_y$.



The columnwise ordering of the 2D Laplacian matrix produces less fill.

Example with Natural Ordering

Recall when factoring banded matrices the L and U factors of a band matrix have the same lower and upper bandwidths, respectively, as the input A. That is, the widths of the bands are preserved. Therefore, fill can only occur between the outermost bands (e.g., see Figure 1).

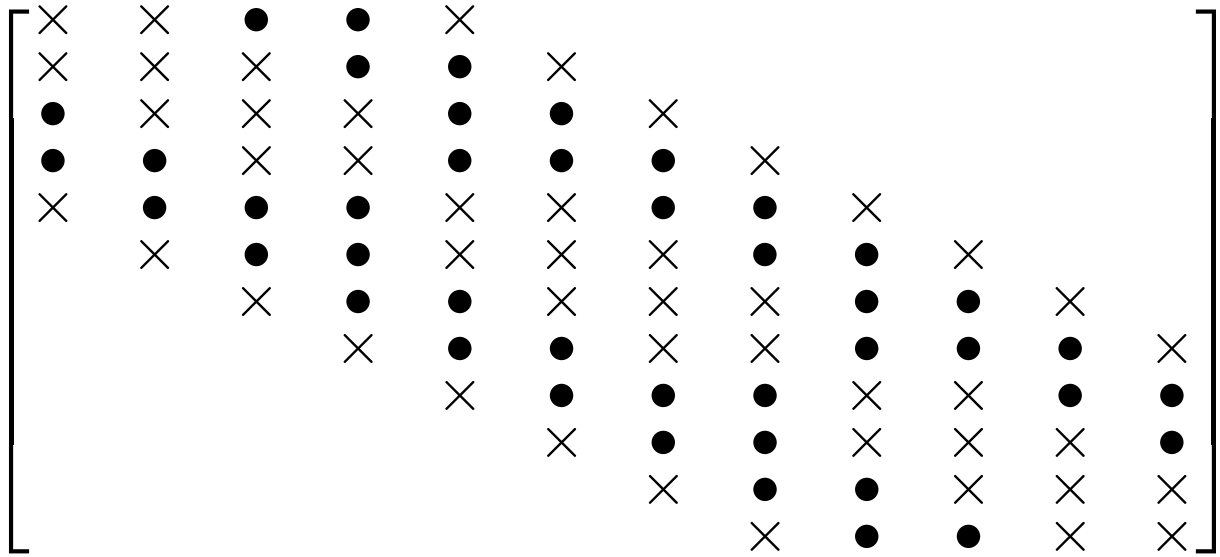


Figure: Example of where fill can occur for banded matrices. The possible fill occurs where • are depicted.

Example with Natural Ordering

- One can verify that flops(banded GE) $\approx O(npq)$ for bandwidths p, q and n gridpoints.
- Therefore, the cost is $O(m^2 n)$ for bandwidth m.
- For the rowwise ordering we have flops $= O(m_x^2 n)$, whereas for the columnwise ordering we only have flops $= O(m_y^2 n)$.
- But what can we do for more general sparsity patterns?
  - Finding the true optimum ordering of the graph is NP complete (i.e. it is hard).
  - There do exist many ordering algorithms based on good heuristics.
- We will look at some common algorithms next:
  - Envelope/Level set methods,
  - (Reverse) Cuthill McKee,
  - Markowitz,
  - Minimum Degree.

Example with Natural Ordering

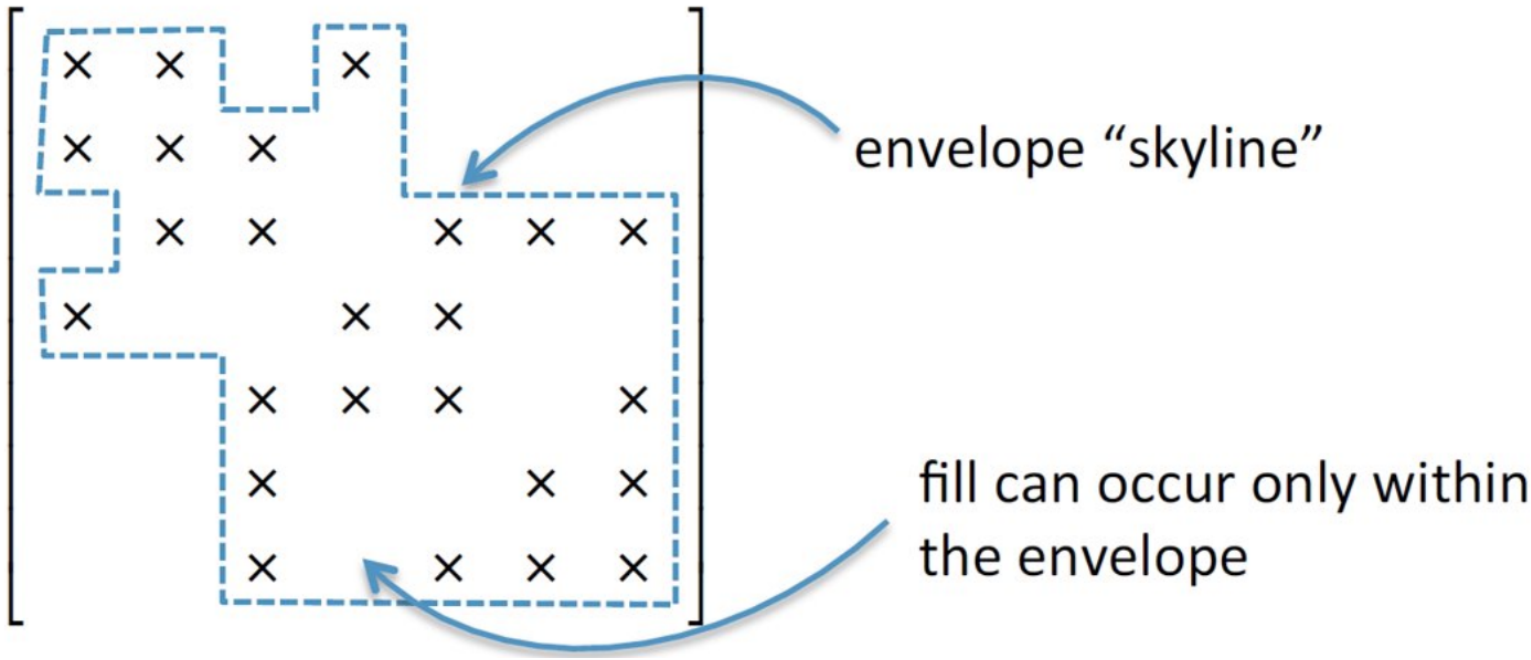
Q & A

1. Last week we discussed upper- and lower-bandwidths. What does "bandwidth" mean, unqualified?
   A: If upper and lower bandwidths are equal, then writing "bandwidth", unqualified, is clear enough.

Envelope Reordering

- In this lecture we will look at the first two types of methods.
- The next lecture will discuss Markowitz and Minimum degree reorderings.
- In practice, bandwidth may vary a lot between individual rows.

Envelope Reordering



The **envelope** is the contiguous part of the matrix containing all non-zero entries, indicated by dotted lines in the picture.
Note, we are not asserting that all entries inside the envelope are non-zero; we are asserting that all entries outside the envelope are zero.

Envelope Reordering

Remarks:

1. Recall that fill-in during factorization is bad, hence it is to be minimized.
2. In each row of L, fill can only occur between 1st non-zero entry (from the left) and the diagonal entry.
3. This motivates us to limit fill by keeping the envelope as close to the diagonal as possible.
4. This further motivates the next section, on **Level Sets**.

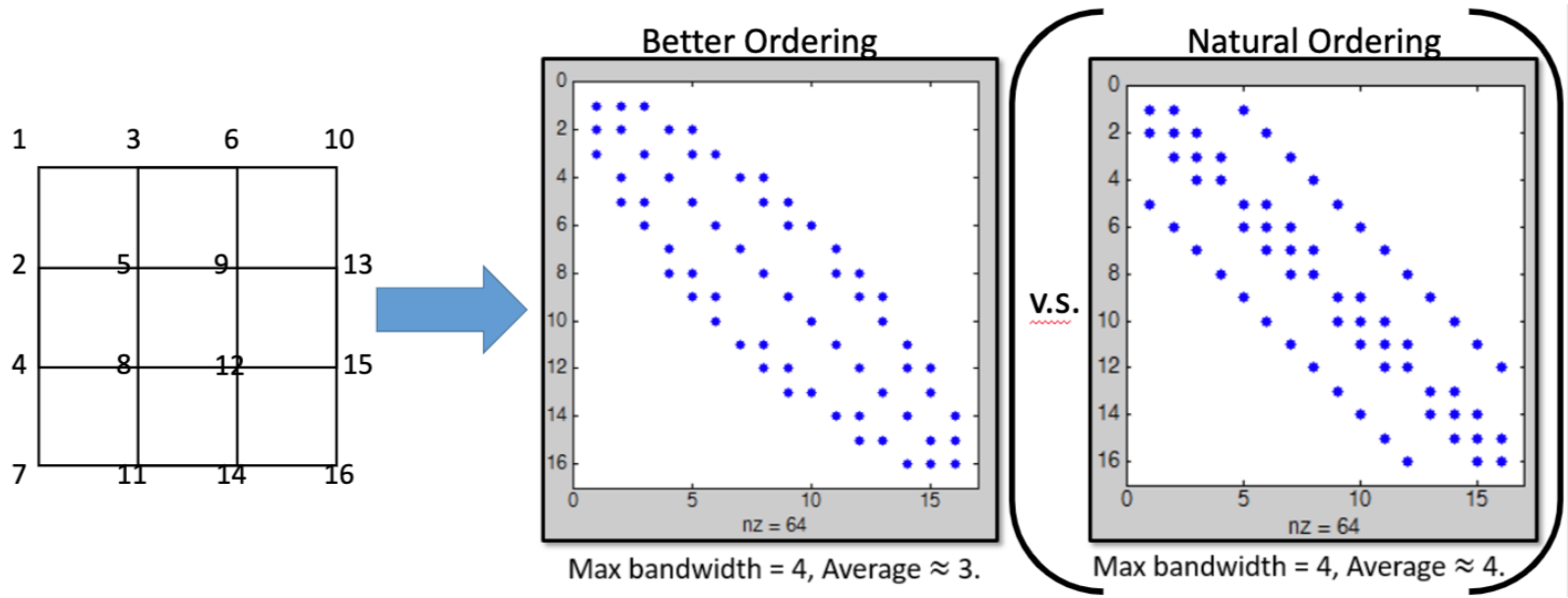
Envelope Reordering

Q & A:

1. Does our matrix have to be symmetric?
   A: All graphs coming up are undirected, which requires A to be symmetric. So although our setup does not require A to be symmetric, we will demand that A be symmetric to agree with the convention in the notes.
2. Can the envelope sit both above and below the diagonal?
   A: Yes, as in the above picture.

Envelope Reordering

Q: What does this imply about a **good** numbering of nodes?
A: The graph neighbours should have numbers as close together as possible.



Level sets

We assume the sparsity pattern of the matrix is symmetric, i.e., the underlying graph representation is undirected. Envelope methods are based on **graph level sets** S_i defined below.

Definition 1.1

Let A be a symmetric matrix. Let $\{s_j\}$ be the nodes of the graph of A. Then the **level sets** *S_i are the sets of nodes that are the same graph distance from some starting point. That is,*

$$
\begin{aligned}
S_1 &= \{\textit{the single starting node}\},\\
S_2 &= \{\textit{all immediate neighbours of the node in } S_1\},\\
S_3 &= \{\textit{all immediate neighbours of nodes in } S_2, \textit{not in } S_1 \textit{ or } S_2\},\\
&\vdots\\
S_i &= \{\textit{all immediate neighbours of nodes in } S_{i-1}, \textit{not in } S_1, S_2, \ldots, S_{i-1}\}.
\end{aligned}
$$

Level sets

Figure 2 shows an example of the level sets of a graph.

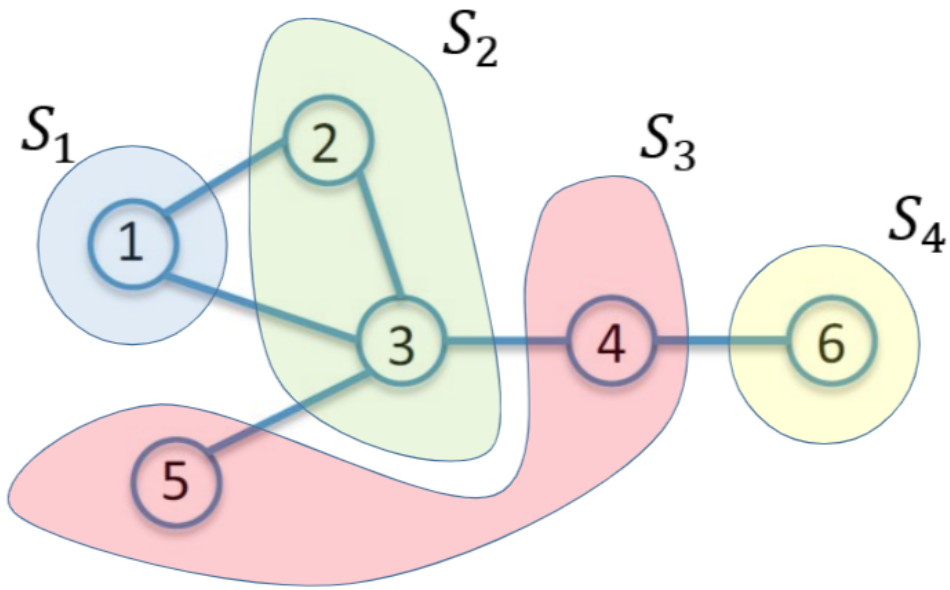


Figure: Level sets of an example graph.

Level sets

Q: Why do we care about level sets?
A: Envelope methods:

1. order the nodes in $S_2, S_3, \ldots, S_k$, and
2. "do something" with the ordered list of nodes.

Level sets

- Envelope methods order nodes in S_2, then nodes in S_3, and so on.
- This is similar to a breadth first traversal (BFS).

Cuthill-McKee

How do we order nodes **within** each level set? In the **Cuthill-McKee (CM)** algorithm a heuristic is used based on the **degree** of a node.

Definition 1.2

The degree of a node v, denoted $\deg(v)$, is the number of adjacent nodes (i.e. the number of incident edges).

For example, in the graph in Figure 2 we have $\deg(3) = 4$ and $\deg(5) = 1$.

Cuthill-McKee

The Cuthill-McKee ordering heuristic is as follows. When visiting a node during traversal, order its neighbors (yet to be visited) in increasing order of degree and add them to the queue in this order. Cuthill-McKee algorithm is given in Algorithm 1, which consists of the following:

1. pick an arbitrary starting node and number it 1,
2. find all un-numbered neighbours of node 1 and number them in increasing order of degree,
3. for each of node 1's neighbours, order their neighbours in increasing order of degree,
4. continue recursively until all nodes have been numbered.

Ties are broken in an arbitrary manner, e.g., based on the initial node ordering.

Cuthill-McKee

Remarks:

1. This assumes that the graph is connected.
2. If the graph is not connected, then the corresponding matrix (possibly under some permutation of rows/columns) can be decomposed into diagonal blocks, each of which corresponds with a connected subgraph.
3. Then we could solve each subsystem independently of the others.
4. Thus we lose no generality by assuming that our graph is connected.

Cuthill-McKee

Algorithm 1 : Cuthill-McKee Ordering

Input: undirected graph $G = (V, E)$
Output: ordered level sets S_i
choose starting node s ▷ for each connected component
$S_1 \leftarrow \{s\}$, mark s
$i = 1$
while $S_i \neq \emptyset$
    $S_{i+1} \leftarrow \emptyset$
    for each $u \in S_i$ ▷ in order of increasing degree
        for each unmarked v adjacent to u ▷ in order of increasing degree
            $S_{i+1} \leftarrow S_{i+1} \cup \{v\}$
            mark v
        end for
    end for
    $i = i + 1$ ▷ move on to the next level set
end while

Cuthill-McKee - Q & A

1. **Q:** What is a good choice of starting node for CM?
   A: A vertex with as low a degree as possible.

Reverse Cuthill-McKee (RCM)

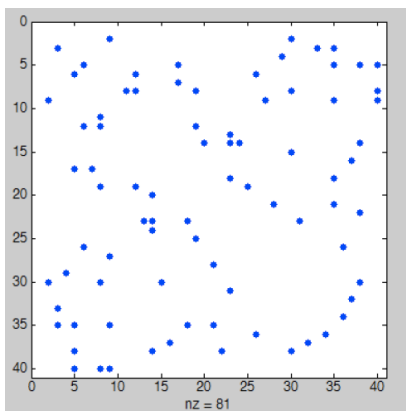
- The **reverse Cuthill-McKee (RCM)** algorithm is exactly what it sounds like!
- You compute the CM numbering then reverse it, i.e.,

$$\text{node}_i^{\text{RCM}} = \text{node}_{n-i+1}^{\text{CM}} \quad \text{for} \quad i = 1, \ldots, n.$$

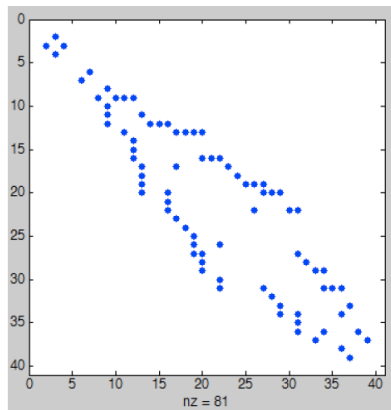
- This is simple, but why perform the reversal? John Alan George ("Computer Implementation of the finite element method", 1971) observed by simply reversing the Cuthill-McKee ordering, we can reduce the amount of fill-in for many graphs (matrices).
- The RCM has the same envelope of CM but better observed behavior in practice.
- The patterns produced by RCM are more like the low fill downward arrow matrix, rather than the upward arrow.
- Figure 3 shows an example on a random symmetric matrix.
- As expected the CM algorithm produces a matrix with a smaller bandwidth.
- The RCM algorithm has the same bandwidth but is more "down-arrow" like.

Cuthill-McKee

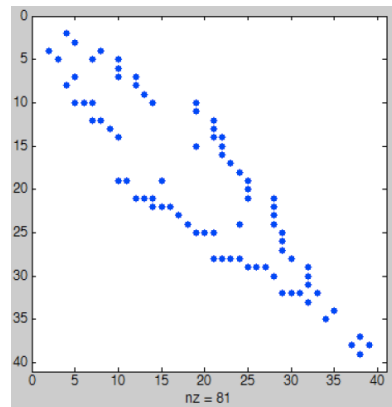
Figure 3 shows an example on a random symmetric matrix. As expected the CM algorithm produces a matrix with a smaller bandwidth. The RCM algorithm has the same bandwidth but is more "down-arrow" like.



Random Symmetric Matrix



Cuthill-McKee

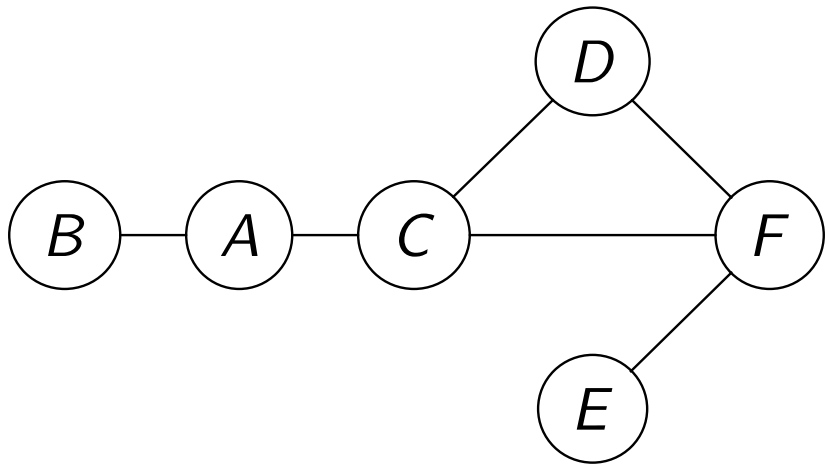


Reverse Cuthill-McKee

Figure: Comparison of CM (middle) vs RCM (right) for a symmetric matrix (left).

Cuthill-McKee

Example: In this example, we will compute the CM ordering, and the RCM ordering, for this graph:



Assumptions:

1. Select node A as your starting node.
2. Break any ties with respect to the degrees of nodes in a level set according to the usual alphabetic order of the nodes.

Cuthill-McKee

Computing the CM and RCM Orderings:

1. Start by numbering:
   1. A
2. Number the un-numbered neighbours of node A in ascending order of their degrees:
   2. B (degree 1)
   3. C (degree 3)
3. B has no un-numbered neighbours. Number the un-numbered neighbours of node C in ascending order of their degrees:
   4. D (degree 2)
   5. F (degree 3)
4. D has no un-numbered neighbours. Number the un-numbered neighbours of node F in ascending order of their degrees:
   6. E (degree 1)
5. This yields the Cuthill-Mckee ordering:

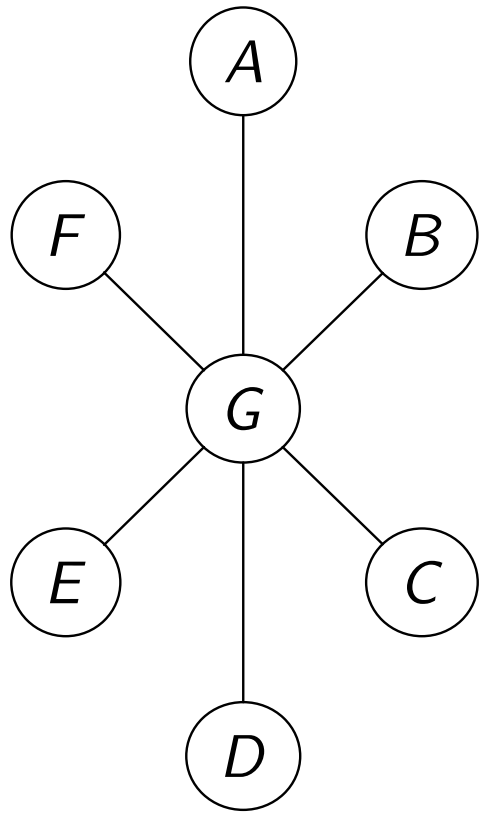
1	2	3	4	5	6
A	B	C	D	F	E

6. This yields the Reverse Cuthill-Mckee ordering:

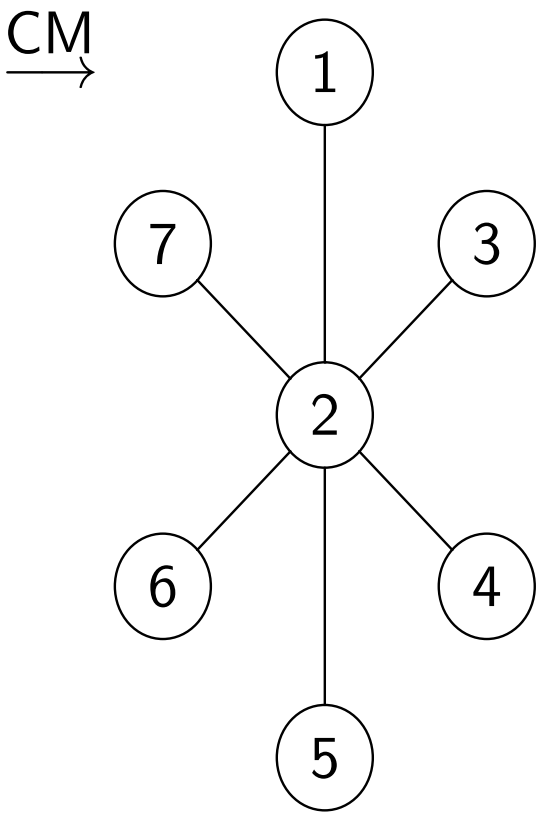
1	2	3	4	5	6
E	F	D	C	B	A

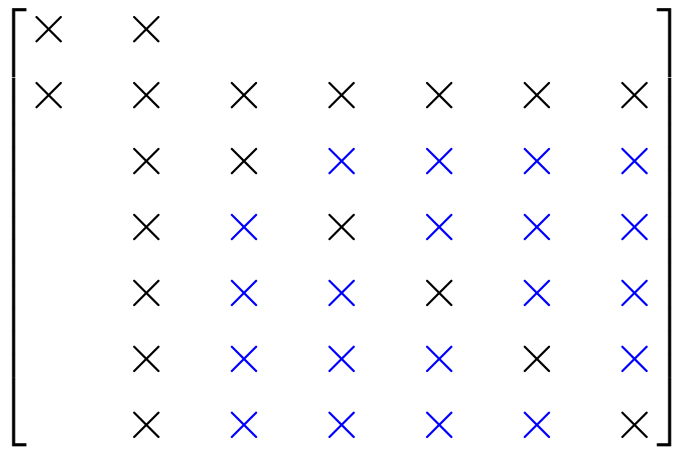
Cuthill-McKee

Initial Graph:



Cuthill-McKee





Lots of fill-in!

Cuthill-McKee

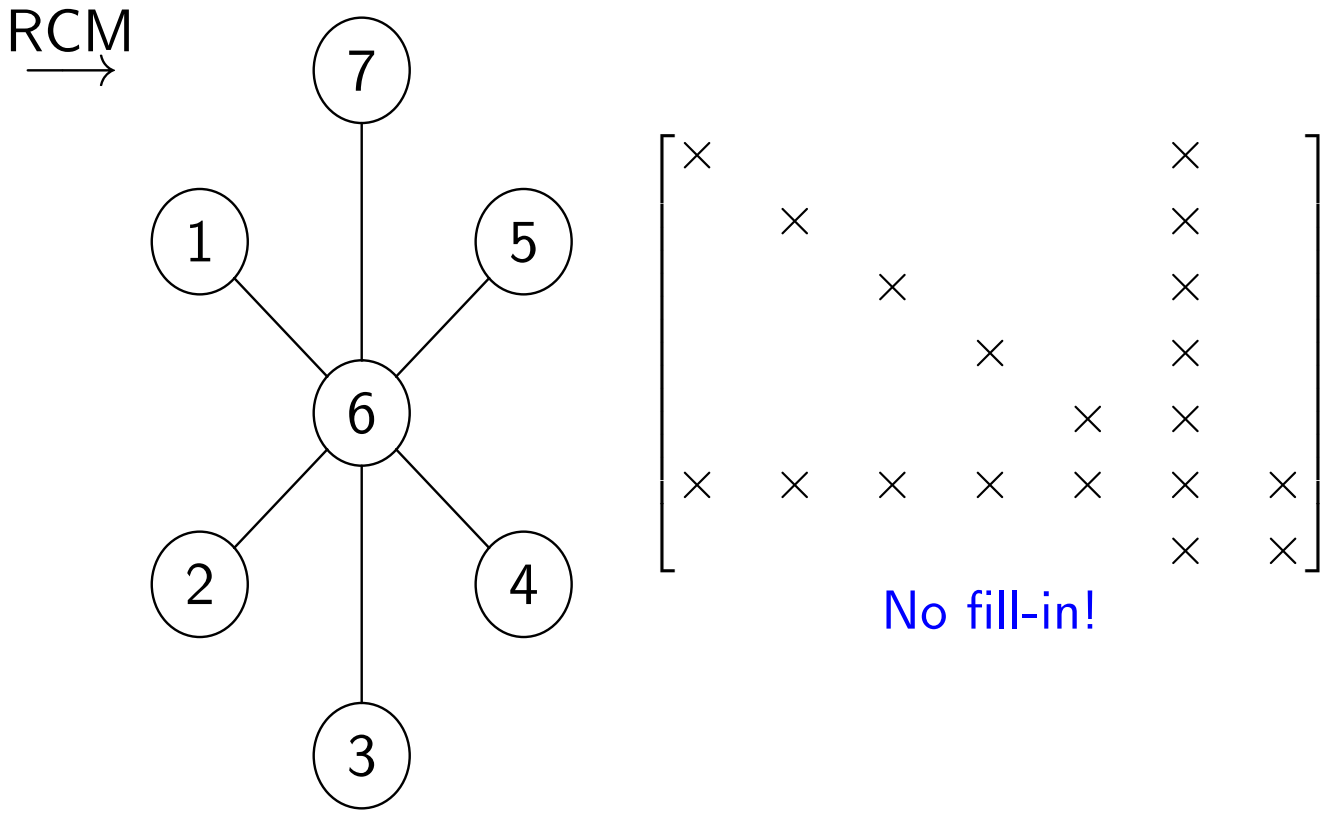


Figure: CM and RCM orderings for the initial graph. The CM algorithm is started at node A and ties are broken alphabetically.

Cuthill-McKee

Detailed Explanation of Fig 4:

1. Start by numbering:
   1. A
2. Number the un-numbered neighbours of node A in ascending order of their degrees:
   2. G (degree 6)
3. Number the un-numbered neighbours of node G in ascending order of their degrees:
   3. B (degree 1)
   4. C (degree 1)
   5. D (degree 1)
   6. E (degree 1)
   7. F (degree 1)
4. This yields the Cuthill-Mckee ordering:

1	2	3	4	5	6	7
A	G	B	C	D	E	F

5. This yields the Reverse Cuthill-Mckee ordering:

1	2	3	4	5	6	7
F	E	D	C	B	G	A

Cuthill-McKee

Theorem 1

(RCM is optimal on trees.) On a tree, no matter which node we start with, RCM ordering produces no fill.

Proof.

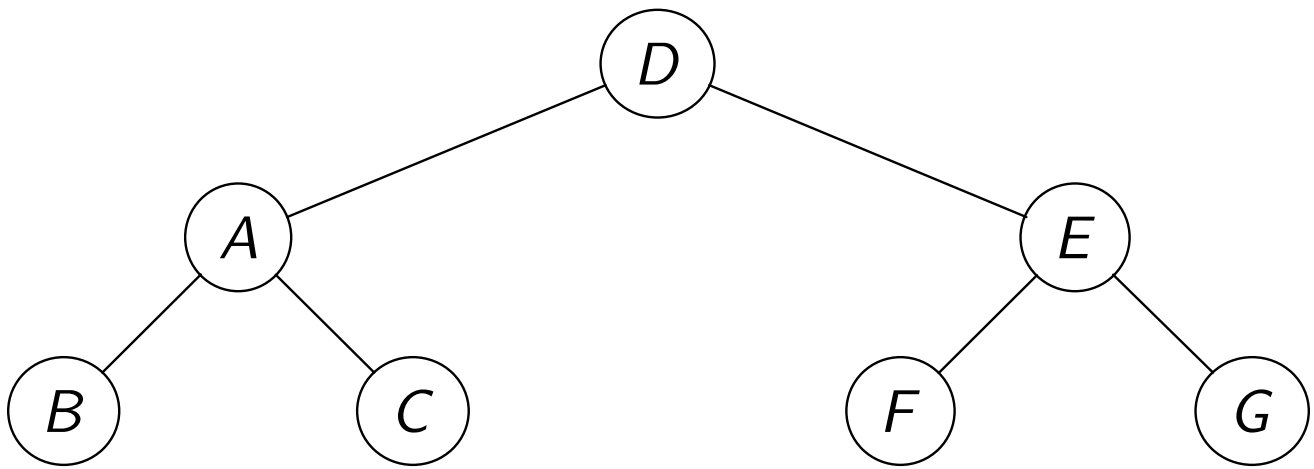
We argue that the first node in the RCM ordering has to be a leaf (degree 1).

- Towards a contradiction, suppose the starting node is s and the first node in RCM is a non-leaf node u.
- Then u is connected to at least two nodes v and w.
- In constructing the CM ordering we may have reached (from s) at most one of v and w before reaching u (otherwise we have a cycle $v \cdots s \cdots wuv$, which can not happen for a tree).
- But then after we have reached u we need to continue the CM ordering to the other (or both) child node, so u cannot be the last node in CM (i.e. u cannot be the first note in the RCM ordering).
- This is a contradiction, so u must be a leaf node.
- Now note that if we remove the node u, and rerun CM, we would get the same ordering (without u).
- Thus, recursively, we see that in the elimination of the graph by following RCM, we are always removing leaf nodes hence will not introduce any new edges.

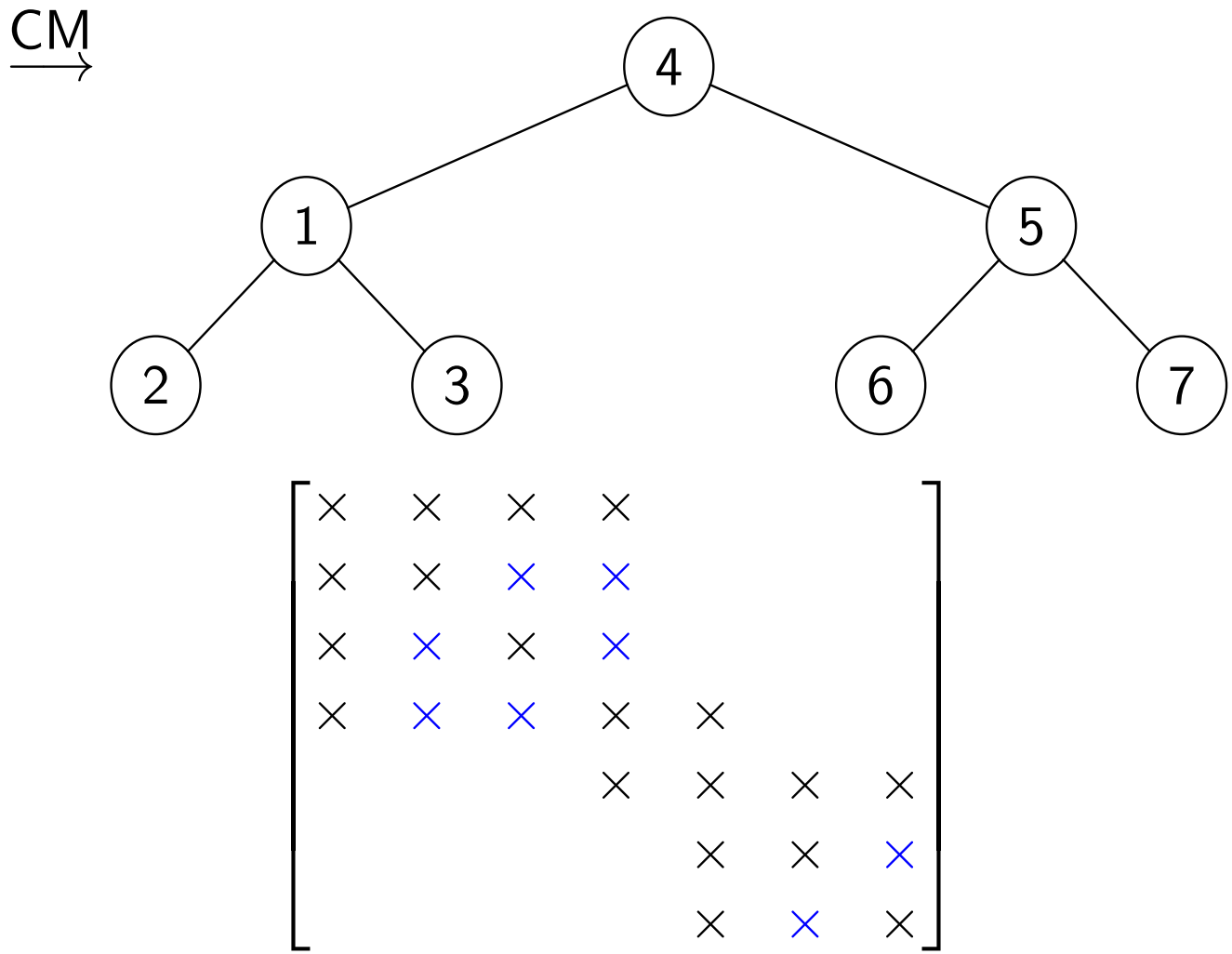
Cuthill-McKee

- Note that Theorem 1 does not hold in general for the CM ordering.
- Further, RCM does not necessarily produce the optimal ordering for general graphs.
- Determining the optimal ordering that introduces the least amount of fill-in is NP-complete [Yannakakis 1981].
- The example in Figure 5 shows the optimality of RCM on a small tree.

Initial Graph:



Cuthill-McKee



Cuthill-McKee

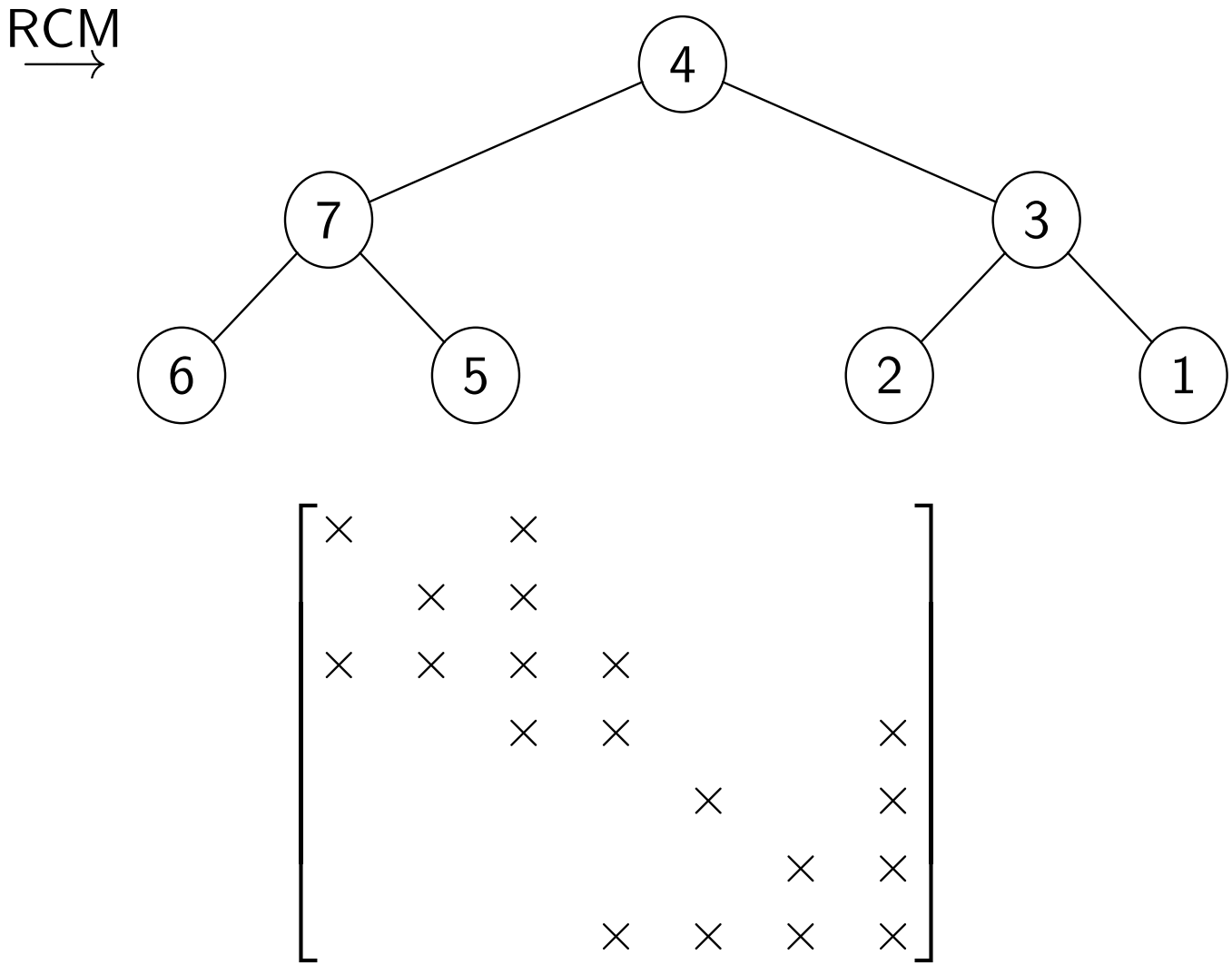


Figure: CM and RCM ordering for a tree graph. Notice that RCM produces a reordering with no fill (shown as $\times$).