**Complexity of some operations**

Converting between COO, CSR and CSC: Linear

“All conversions among the CSR, CSC, and COO formats are efficient, linear-time operations.” <https://docs.scipy.org/doc/scipy/reference/sparse.html#usage-information\>

<https://github.com/scipy/scipy/blob/3b36a574dc657d1ca116f6e230be694f3de31afc/scipy/sparse/sparsetools/coo.h>

CSR matrix multiplication: O(n\_row \* K^2 + max(n\_row, n\_col))

where K is the **maximum nnz** in a row of A and column of B. This is max n-1 for a fully connected node.

So O(n3), though in practice it might feel quicker since not all nodes have lots of connections. In fact, given the power law distribution of degrees, we know very few of them do.

Src: <https://github.com/scipy/scipy/blob/701ffcc8a6f04509d115aac5e5681c538b5265a2/scipy/sparse/sparsetools/csr.h#L542-L544>

Transposing: Linear because it converts from CSR to CSC or creates a new COO behind the scenes, and those operations are linear.

Coo does coo\_matrix((self.data, (self.col, self.row)), shape=(N, M), copy=copy)

inverting self.col and self.row and the shape

<https://github.com/scipy/scipy/blob/v1.7.1/scipy/sparse/coo.py#L291-L299>

CSR and CSC call eachother’s constructors

<https://github.com/scipy/scipy/blob/v1.7.1/scipy/sparse/csr.py#L135-L145>

<https://github.com/scipy/scipy/blob/v1.7.1/scipy/sparse/csc.py#L108-L118>

Extracting a diagonal from a CSR -> O(nnz)

Src: <https://github.com/scipy/scipy/blob/3b36a574dc657d1ca116f6e230be694f3de31afc/scipy/sparse/sparsetools/coo.h#L31>

**Complexity of Q1** – Will depend on what code we run

a) Depends

c) Time complexity will depend on the chosen algorithm

connected compo uses

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.1707&rep=rep1&type=pdf>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csgraph.connected_components.html#scipy.sparse.csgraph.connected_components>