Report sketch

The function read_graph_from_file1

Extracting number of nodes and edges

To extract the number of edges and nodes we read the 2 first lines to remove them entirely. The number of nodes N and edges N_{links} is extracted with the following code:

```
fscanf(fp,  "%*s %*s %d %*s %d", N, &N_links);
```

Filling the 2D-array

This one is straight forward. The following code reads the data from the file and fills the 2D-array with 1s for each index pair (ToNodeId, FromNodeId):

```
int FromNodeId, ToNodeId;
for (int k = 0; k < N_links; k++){
  fscanf(fp, "%d %d", &FromNodeId, &ToNodeId);
  (*table2D)[ToNodeId][FromNodeId] = (char) 1;
}
```

The function read_graph_from_file2

Sorting the ToNodeIds and counting number of elements on each row

The main algorithmic challenge here is to sort the column indices corresponding to ToNodeId. This is done by the following short code:

```
int x = 0;
for (int i = 0; i < *N; i++){
  for (int j = 0; j < *N_links; j++){
    if (tmp_row[j] == i){
      (*col_idx)[x] = tmp_col[j];
      row_count[i] += 1;
      x++;
    }
  }
}
```

What it does is essentially use two temporary arrays, tmp_col consisting of ToNodeIds and tmp_row storing FromNodeIds read from the input file, to sort the column indices according to which FromNodeId they correspond to. The nested loop also count how many elements there are on each row which will be used to create the row_ptr array.

Creating the row_ptr array

The row_ptr is straight forward to fill which is done with the following code:

```
int row_elems = 0;
for (int i = 1; i < *N+1; i++){
  row_elems += row_count[i-1];
  (*row_ptr)[i] = row_elems;
}
```

It sets the first element to zero. Then it cumulatively add up number of row elements such that the difference row_ptr$[i+1]-$row_ptr$[i]$ gives the number of elements on row $i+1$.