Home exam 1 in4200

Candidate 15836

March 2021

Abstract

Explanations for the functions in each .c program, beeing that this pdf exists tha programs are not very heavily commented. Compilation instructions is taken care of by makefile

1 read_from_file1

1.1 Idea

Reads number of nodes, N, and number of edges, n_edges, in addition to the details, from-node and to-node for each of the edges. Puts this information in a boolean NxN matrix where 1 at position matrix[n][m] signifies a edge between node n and node m. As the edges are non-directional the matrix is symetric. Self-linkage is not permitted.

1.2 Algorithm

Allocates a NxN matrix of zeroes. iterates through the edges of the file and checks if from_node and to_node are legal values, and populates the matrix with a 1 at index [from_node][to_node] and [to_node][from_node and], if they are. Clears potential self linkages after this, to avoid an additional if-test.

2 read_from_file1

2.1 Idea

Reads file as before and puts this information in a CRS-format. That is an array row_ptr of length N+1 and an array col_idx of length n_edges. This format can be thought of as a condensed version of the format described in the first section, where the content of row_ptr and col_idx together points to the 1s in the matrix of the last section. Row_ptr slices col_idx and col_idx contains the indexes of the 1s for each row.

2.2 Algorithm

allocates the two arrays, and two more of size n_edge called to and from. Iterates through the egdes in the file and increases row_ptr's value by 1 at index to_node+1 and from_node+1 and stores to_node in to and from_node in from. preform a cumsum at row_ptr to get desired format.

Runs a for loop of i up to N, that for each iteration runs through to and from to check for nodes nummbered i. When found col_index is populated with the corresponding node in the oposite array (to and from). This way Col index is sorted for the rows but not internaly in the rows.

3 create_SNN_graph1

3.1 Idea

Gets the 2d table from readFromFile1 as input and creates a new 2d table like the input but where the edges are weighted by the connected nodes number of shared nearest neighbours.

3.2 Algorithm

Iterates through the input-table, for each value, if the value is 1, iterates through the values of the ith and the jth row, and checks (multiplies the values that are either 0 or 1) for matching, nonzero, values ie SNNs. updates SNN_table accordignly.

4 create_SNN_graph2

4.1 Idea

Pretty much the same as the last one but for CRS-format. To create weights for the edges in col_idx.

4.2 Algorithm

Iterates through row_ptr uses those entries to iterate through each value of col_idx. Each entry here corresponds to a node and for each node here the corresponding index of row_ptr is used to make two slices of col_idx corresponding to each of the neighbours of the initial node from the first loop and the node from the third loop. These are compared to search for SNNs. SNN_val is updated accordingly.

5 create_SNN_graph1_parallel

Same as before but with extra input for N_threads. Considerable speedup ¿2 is achieved. though depending on N and n_egde and overheadtime. See test.c.

6 create_SNN_graph2_parallel

Same as before but with extra input for N_threads. Severe speedup is achieved through parralellisation. Speeduptime and if CRS-is prefered over the 2d case is decided both by the amount of nodes, and the sparcity of the matrix

7 checknode

Takes a node as input and iterativley checks for neighbours with tau or more SNNs.

loops through SNN_val to see if any first neighbours of the node meets the tau crieteria. If so, keeps checking the newly added members of the cluster untill no more are added in the iteration.

No more time to write more on performance. see test.c