NAME

mcx erdos − compute shortest paths in a graph

SYNOPSIS

mcx erdos [options]

mcxerdos is not in actual fact a program. This manual page documents the behaviour and options of the mcx program when invoked in mode *erdos*. The options **-h**, **--apropos**, **--version**, **-set**, **--nop**, **-progress** *<num>* are accessible in all **mcx** modes. They are described in the **mcx** manual page.

mcx erdos [-query <fname> (*query input stream*)**] [-abc** <fname> (*specify label input*)**] [-imx** <fname> (*specify matrix input*)**] [-tab** <fname> (*use tab file*)**] [-o** <fname> (*output file name*)**] [--is-directed** (*input graph is directed*)**] [--is-undirected** (*input graph is directed*)**] [-write-path** <fname> (*path matrix file*)**] [-write-step** <fname> (*step matrix file*)**] [-h** (*print synopsis, exit*)**] [--apropos** (*print synopsis, exit*)**] [--version** (*print version, exit*)**]**

DESCRIPTION

mcx erdos computes shortest paths in graphs. It can read a graph either in label format with **-abc** or in native format with **-imx**. It reads pairs of node indices from an input stream, and for each pair outputs a data structure describing the full set of shortest paths between the two nodes. Edge weights are not taken into account, so an edge always represents a unit step size between two nodes irrespective of its weight. A mode to compute shortest paths while taking into account edge weights will be implemented later as **mcx dijkstra**.

Note that the full set of shortest paths between two nodes in a graph can be described as a directed acyclic graph (DAG), and this is how **mcx erdos** operates. It is easy to construct graphs and node pairs for which the number of shortest paths between the two nodes becomes exponential in the size of the graph, whereas the lattice description is always garantueed to map to a subset of the graph edge set.

By default it is assumed that the input graph should be treated as undirected. To this end a transformation step is applied to ensure that the graph in memory is undirected. It is possible to compute shortest paths in directed graphs by using **--is-directed**, and it is possible to omit the transformation step by using **--is-undirected**. If the latter is specified while the input graph is in native format and in fact directed, results will be erroneous. This could in theory be mitigated by checking that the input graph is undirected. However, the reason to use **--is-undirected** is simply to increase speed of operation, whereas such a check would be equally expensive as the transformation step that is omitted with **--is-undirected**.

The input graph/matrix, if specified with the **-imx** option, has to be in mcl matrix/graph format. You can use label input instead by using the **-abc** option. Refer to **mcxio(5)** for a description of these two input formats. By default **mcx erdos** reads from STDIN *and expects matrix format*. To specify label input from STDIN use **-abc -**.

OPTIONS

-query <fname> (*query input*)
The name for the file from which queries are read. A query consists of two white-space separated node indices or two white-space separated labels. Labels can only be used if either **-abc** or **-tab** is specified.

-abc <fname> (*label input*)
The file name for input that is in label format.

-imx <fname> (*input matrix*)
The file name for input that is in mcl native matrix format.

-o <fname> (*output file name*)
The name of the file to write output to.

-tab <fname> (*use tab file*)
This option causes the output to be printed with the labels found in the tab file. With **-abc** this option will, additionally, construct a graph only on the labels found in the tab file. If this option is used in conjunction with **-imx** the tab domain and the matrix domain are required to be identical.

--is-directed (*compute directed shortest paths*)
The input graph is not transformed and assumed to be directed. Shortest paths are computed taking this into account.

--is-undirected (*skip symmetrification step*)
The input graph is not transformed and assumed to be undirected. Shortest paths are computed on the assumption that the input is undirected. Use this option only if you are sure the input is undirected and need to have faster execution.

-write-path <fname> (*path matrix file*)
-write-step <fname> (*step matrix file*)
The path matrix enumerates the nodes that take part in all shortest paths. The first list contains those nodes that lie at distance 1 of the source node, the second list contains nodes lying at distance 2, and so on. The step matrix contains all the edges that make up the lattice of shortest paths between the two query nodes.

SEE ALSO

mcxio(5), and **mclfamily(7)** for an overview of all the documentation and the utilities in the mcl family.