Reference Documentation for ODGI 0.4.1

Table of Contents

1. odgi (1)

1.1. NAME

odgi - dynamic succinct variation graph tool

1.2. SYNOPSIS

odgi build -g graph.gfa -o graph.og

odgi stats -i graph.og -S

odgi sort -i graph.og -o graph.sorted.og -p bSnSnS

odgi view -i graph.og -g

odgi kmers -i graph.og -c -k 23 -e 34 -D 50

odgi unitig -i graph.og -f -t 1324 -l 120

odgi viz -i graph.og -o graph.og.png -x 1920 -y 1080 -R -t 28

odgi paths -i graph.og -f

odgi prune -i graph.og -o graph.pruned.og -c 3 -C 345 -T

odgi unchop -i graph.og -o graph.unchopped.og

odgi normalize -i graph.og -o graph.normalized.og -I 100 -d

odgi subset -i graph.og -o graph.subsetted.og -n 1 -t 28

1.3. DESCRIPTION

odgi, the **Optimized Dynamic (genome) Graph Interface**, links a thrifty dynamic in-memory variation graph data model to a set of algorithms designed for scalable sorting, pruning, transformation, and visualization of very large genome graphs. **odgi** includes python bindings that can be used to directly interface with its data model. This **odgi** manual provides detailed information about its features and subcommands, including examples.

1.4. COMMANDS

Each command has its own man page which can be viewed using e.g. **man odgi_build.1**. Below we have a brief summary of syntax and subcommand description.

odgi build [**-g, --gfa**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi build(1) command constructs a succinct variation graph from a GFA. Currently, only GFA1 is supported. For details of the format please see https://github.com/GFA-spec/GFA-spec/blob/master/GFA1.md.

odgi stats [**-i, --idx**=*FILE*] [*OPTION*]...
The odgi stats(1) command produces statistics of a variation graph. Among other metrics, it can calculate the #nodes, #edges, #paths and the total nucleotide length of the graph. Various histogram summary options complement the tool. If [**-B, --bed-multicov**=*BED*] is set, the metrics will be produced for the intervals specified in the BED.

odgi sort [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi sort(1) command sorts a succinct variation graph. Odgi sort offers a diverse palette of sorting algorithms to determine the node order:

- A topological sort: A graph can be sorted via breadth-first search (BFS) or depth-first search (DFS). Optionally, a chunk size specifies how much of the graph to grab at once in each topological sorting phase. The sorting algorithm will continue the sort from the next node in the prior graph order that has not been sorted, yet. The cycle breaking algorithm applies a DFS sort until a cycle is found. We break and start a new DFS sort phase from where we stopped.
- A random sort: The graph is randomly sorted. The node order is randomly shuffled from Mersenne Twister pseudo-random generated numbers.
- A sparse matrix mondriaan sort: We can partition a hypergraph with integer weights and uniform hyperedge costs using the Mondriaan partitioner.
- A 1D linear SGD sort: Odgi implements a 1D linear, variation graph adjusted, multi-threaded version of the Graph Drawing by Stochastic Gradient Descent algorithm. The force-directed

graph drawing algorithm minimizes the graph's energy function or stress level. It applies stochastic gradient descent (SGD) to move a single pair of nodes at a time.

- An eades algorithmic sort: Use Peter Eades' heuristic for graph drawing.

Sorting the paths in a graph my refine the sorting process. For the users' convenience, it is possible to specify a whole pipeline of sorts within one parameter.

odgi view [**-i, --idx**=*FILE*] [*OPTION*]…
The odgi view(1) command can convert a graph in odgi format to GFAv1. It can reveal a graph's internal structures for e.g. debugging processes.

odgi kmers [**-i, --idx**=*FILE*] [**-c, --stdout**] [*OPTION*]…
Given a kmer length, the odgi kmers(1) command can emit all kmers. The output can be refined by setting the maximum number of furcations at edges or by not considering nodes above a given node degree limit.

odgi unitig [**-i, --idx**=*FILE*] [*OPTION*]…
The odgi unitig(1) command can print all unitigs of a given odgi graph to standard output in FASTA format. Unitigs can also be emitted in a fixed sequence quality FASTQ format. Various parameters can refine the unitigs to print.

odgi viz [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi viz(1) command can produce a linear, static visualization of an odgi variation graph. It aggregates the pangenome into bins and directly renders a raster image. The binning level depends on the target width of the PNG to emit. Can be used to produce visualizations for gigabase scale pangenomes. For more information about the binning process, please refer to odgi bin. If reverse coloring was selected, only the bins with a reverse rate of at least 0.5 are colored. Currently, there is no parameter to color according to the sequence coverage in bins available.

odgi paths [**-i, --idx**=*FILE*] [*OPTION*]…
The odgi paths(1) command allows the investigation of paths of a given variation graph. It can calculate overlap statistics of groupings of paths.

odgi prune [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi prune(1) command can remove complex parts of a graph. One can drop paths, nodes by a certain kind of edge coverage, edges and graph tips. Specifying a kmer length and a maximum number of furcations, the graph can be broken at edges not fitting into these conditions.

odgi unchop [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi unchop(1) command merges each unitig into a single node.

odgi normalize [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi normalize(1) command unchops a given variation graph and simplifies redundant furcations.

odgi subset [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
Extracting a node subset of a variation graph is the task of the odgi subset(1) command. Users can specify a node, a list of nodes or a the context of which to generate a subset from.

1.5. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

1.6. AUTHORS

Erik Garrison from the University of California Santa Cruz wrote the whole **odgi** tool. Despite small code contributions, Simon Heumos from the Quantitative Biology Center Tübingen wrote **odgi pathindex**, **odgi panpos**, **odgi server**, and the documentation.

1.7. RESOURCES

Project web site: https://github.com/vgteam/odgi

Git source repository on GitHub: https://github.com/vgteam/odgi

GitHub organization: https://github.com/vgteam

Discussion list / forum: https://github.com/vgteam/odgi/issues

1.8. COPYING

2. odgi build(1)

2.1. NAME

odgi_build - construct a dynamic succinct variation graph

2.2. SYNOPSIS

odgi build [**-g, --gfa**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

2.3. DESCRIPTION

The odgi build(1) command constructs a succinct variation graph from a GFA. Currently, only GFA1 is supported. For details of the format please see https://github.com/GFA-spec/GFA-spec/blob/master/GFA1.md.

2.4. OPTIONS

2.4.1. Graph Files IO

-g, --gfa=***FILE***

GFA1 file containing the nodes, edges and paths to build a dynamic succinct variation graph from.

-o, --out=***FILE***

Write the dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

2.4.2. Graph Sorting

-s, --sort

Apply a general topological sort to the graph and order the node ids accordingly. A bidirected adaptation of Kahn's topological sort (1962) is used, which can handle components with no heads or tails. Here, both heads and tails are taken into account.

2.4.3. Processing Information

-p, --progress

Print progress updates to stdout.

-d, --debug

Verbosely print graph information to stderr. This includes the maximum node_id, the minimum node_id, the handle to node_id mapping, the deleted nodes and the path metadata.

--trace

Include backtrace information when reporting errors.

-v, --verbose

Verbosely print processing information to stderr, including debug-level log messages.

-w, --warnings

Turn on script warnings (applies to executed code).

-t, --timings

Print timings report to stderr (time to read, parse, and convert).

2.4.4. Program Information

-h, --help

Print a help message for **odgi build**.

2.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

2.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

2.7. AUTHORS

odgi build was written by Erik Garrison.

3. odgi stats(1)

3.1. NAME

odgi_stats - metrics describing variation graphs

3.2. SYNOPSIS

odgi stats [**-i, --idx**=*FILE*] [*OPTION*]...

3.3. DESCRIPTION

The odgi stats(1) command produces statistics of a variation graph. Among other metrics, it can calculate the #nodes, #edges, #paths and the total nucleotide length of the graph. Various histogram summary options complement the tool. If [**-B, --bed-multicov**=*BED*] is set, the metrics will be produced for the intervals specified in the BED.

3.4. OPTIONS

3.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to create statistics from. The file name usually ends with *.og*.

3.4.2. Summary Options

-S, --summarize

Summarize the graph properties and dimensions. Print to stdout the #nucleotides, #nodes, #edges and #paths of the graph.

-b, --base-content

Describe the base content of the graph. Print to stdout the #A, #C, #G and #T of the graph.

-C, --coverage

Provide a histogram of path coverage over bases in the graph. Print three tab-delimited columns to stdout: **type**, **cov**, **N**. **type** is one of *full* or *uniq* and determines if the histogram corresponds to the full graph or only to a unique paths graph. **cov** implies the #paths. **N** implies the #nucleotides.

-V, --set-coverage

Provide a histogram of coverage over unique set of paths. Print two tab-delimited columns to stdout: **cov**, **sets**. **cov** implies #nucleotides. **sets** lists the unique set of paths in a comma separated list. Sets with a **cov** of one and no paths in **sets** are listed, too.

-M, --multi-coverage

Provide a histogram of coverage over unique multiset, the combination with possible repetition of paths. Print two tab-delimited columns to stdout: **cov**, **sets**. **cov** implies #nucleotides. **sets** lists the unique multisets of paths in a comma separated list. Multisets with a **cov** of one and no paths in **sets** are listed, too.

3.4.3. BED Interval

-B, --bed-multicov=*BED*

For each BED entry, provide a table of path coverage over unique multisets of paths in the graph. Each unique multiset of paths overlapping a given BED interval is described in terms of its length relative to the total interval, the number of path traversals and unique paths involved in these traversals.

3.4.4. Threading

-t, --threads=*N*

Number of threads to use.

3.4.5. Program Information

-h, --help

Print a help message for **odgi stats**.

3.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

3.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

3.7. AUTHORS

odgi stats was written by Erik Garrison.

4. odgi sort(1)

4.1. NAME

odgi_sort - sort a variation graph

4.2. SYNOPSIS

odgi sort [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

4.3. DESCRIPTION

The odgi sort(1) command sorts a succinct variation graph. Odgi sort offers a diverse palette of sorting algorithms to determine the node order:

- A topological sort: A graph can be sorted via breadth-first search (BFS) or depth-first search (DFS). Optionally, a chunk size specifies how much of the graph to grab at once in each topological sorting phase. The sorting algorithm will continue the sort from the next node in the prior graph order that has not been sorted, yet. The cycle breaking algorithm applies a DFS sort until a cycle is found. We break and start a new DFS sort phase from where we stopped.
- A random sort: The graph is randomly sorted. The node order is randomly shuffled from Mersenne Twister pseudo-random generated numbers.
- A sparse matrix mondriaan sort: We can partition a hypergraph with integer weights and uniform hyperedge costs using the Mondriaan partitioner.
- A 1D linear SGD sort: Odgi implements a 1D linear, variation graph adjusted, multi-threaded

version of the Graph Drawing by Stochastic Gradient Descent algorithm. The force-directed graph drawing algorithm minimizes the graph's energy function or stress level. It applies stochastic gradient descent (SGD) to move a single pair of nodes at a time.

- An eades algorithmic sort: Use Peter Eades' heuristic for graph drawing.

Sorting the paths in a graph my refine the sorting process. For the users' convenience, it is possible to specify a whole pipeline of sorts within one parameter.

4.4. OPTIONS

4.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to sort. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the sorted dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

-s, --sort-order=*FILE*

File containing the sort order. Each line contains one node identifier.

4.4.2. Topological Sorts

-b, --breadth-first

Use a (chunked) breadth first topological sort.

-B, --breadth-first-chunk=*N*

Chunk size for breadth first topological sort. Specify how many nucleotides to grap at once in each BFS phase.

-z, --depth-first

Use a (chunked) depth first topological sort.

-Z, --depth-first-chunk=*N*

Chunk size for the depth first topological sort. Specify how many nucleotides to grap at once in each DFS phace.

-w, --two-way

Use a two-way topological algorithm for sorting. It is a maximum of head-first and tail-first topological sort.

-n, --no-seeds

Don't use heads or tails to seed topological sort.

-c, --cycle-breaking

Use a cycle breaking sort.

4.4.3. Random Sort

-r, --random

Randomly sort the graph.

4.4.4. Mondriaan Sort

-m, --mondriaan

Use the sparse matrix diagonalization to sort the graph.

-N, --mondriaan-n-parts=*N*

Number of partitions for the mondriaan sort.

-E, --mondriaan-epsilon=*N*

Set the epsilon parameter for the mondriaan sort.

-W, --mondriaan-path-weight

Weight the mondriaan input matrix by the path coverage of edges.

4.4.5. 1D Linear SGD Sort

-S, --linear-sgd

Apply 1D linear SGD algorithm to sort the graph.

-O, --sgd-bandwidth=*sgd-bandwidth*

Bandwidth of linear SGD model. The default value is *1000*.

-Q, --sgd-sampling-rate=*sgd-sampling-rate*

Sample pairs of nodes with probability distance between them divided by the sampling rate. The default value is *20*.

-K, --sgd-use-paths

Use the paths to structure the distances between nodes in SGD.

-T, --sgd-iter-max=*sgd_iter-max*

The maximum number of iterations for the linear SGD model. The default value is *30*.

-V, --sgd-eps=*sgd-eps*

The final learning rate for the linear SGD model. The default value is *0.01*.

-C, --sgd-delta=*sgd-delta*

The threshold of the maximum node displacement, approximately in base pairs, at which to stop SGD.

4.4.6. Eades Sort

-e, --eades

Use eades algorithm.

4.4.7. Path Sorting Options

-L, --paths-min

Sort paths by their lowest contained node identifier.

-M, --paths-max

Sort paths by their highest contained node identifier.

-A, --paths-avg

Sort paths by their average contained node identifier.

-R, --paths-avg-rev

Sort paths in reverse by their average contained node identifier.

-D, --path-delim=*path-delim*

Sort paths in bins by their prefix up to this delimiter.

4.4.8. Pipeline Sorting

-p, --pipeline=*STRING*

Apply a series of sorts, based on single character command line arguments given to this command. The default sort is *s*. The reverse sort would be specified via *f*.

4.4.9. Additional Parameters

-d, --dagify-sort

Sort on the basis of a DAGified graph.

-O, --Optimize

Use the MutableHandleGraph::optimize method to compact the node identifier space.

4.4.10. Threading

-t, --threads=*N*

Number of threads to use for parallel sorting in SGD. Only specify this argument in combination with **-S, --linear-sgd**. No multi-threading support for any other sorting algorithm.

4.4.11. Processing Information

-P, --progress

Print sort progress to stdout.

4.4.12. Program Information

-h, --help

Print a help message for **odgi sort**.

4.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

4.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

4.7. AUTHORS

odgi sort was written by Erik Garrison.

5. odgi view(1)

5.1. NAME

odgi_view - projection of graphs into other formats

5.2. SYNOPSIS

odgi view [**-i, --idx**=*FILE*] [*OPTION*]...

5.3. DESCRIPTION

The odgi view(1) command can convert a graph in odgi format to GFAv1. It can reveal a graph's internal structures for e.g. debugging processes.

5.4. OPTIONS

5.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

-g, --to-gfa

Write the graph in GFAv1 format to standard output.

5.4.2. Summary Options

-d, --display

Show the internal structures of a graph. Print to stdout the maximum node identifier, the minimum node identifier, the nodes vector, the delete nodes bit vector and the path metadata, each in a separate line.

5.4.3. Program Information

-h, --help

Print a help message for **odgi view**.

5.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

5.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

5.7. AUTHORS

odgi view was written by Erik Garrison.

6. odgi kmers(1)

6.1. NAME

odgi_kmers - show and characterize the kmer space of the graph

6.2. SYNOPSIS

odgi kmers [**-i, --idx**=*FILE*] [**-c, --stdout**] [*OPTION*]...

6.3. DESCRIPTION

Given a kmer length, the odgi kmers(1) command can emit all kmers. The output can be refined by setting the maximum number of furcations at edges or by not considering nodes above a given node degree limit.

6.4. OPTIONS

6.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

-c, --stdout=

Write the kmers to standard output. Kmers are line-separated.

6.4.2. Kmer Options

-k, --kmer-length=*N*

The kmer length to generate kmers from.

-e, --max-furcations=*N*

Break at edges that would induce this many furcations when generating a kmer.

-D, --max-degree=*N*

Don't take nodes into account that have a degree greater than *N*.

6.4.3. Threading

-t, --threads=*N*

Number of threads to use.

6.4.4. Program Information

-h, --help

Print a help message for **odgi kmers**.

6.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

6.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

6.7. AUTHORS

odgi kmers was written by Erik Garrison.

7. odgi unitig(1)

7.1. NAME

odgi_unitig - output unitigs of the graph

7.2. SYNOPSIS

odgi unitig [**-i, --idx**=*FILE*] [*OPTION*]...

7.3. DESCRIPTION

The odgi unitig(1) command can print all unitigs of a given odgi graph to standard output in FASTA format. Unitigs can also be emitted in a fixed sequence quality FASTQ format. Various parameters can refine the unitigs to print.

7.4. OPTIONS

7.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

7.4.2. FASTQ Options

-f, --fake-fastq

Write the unitigs in FASTQ format to stdout with a fixed quality value of *I*.

7.4.3. Unitig Options

-t, --sample-to=***N***

Continue unitigs with a random walk in the graph so that they have at least the given *N* length.

-p, --sample-plus=***N***

Continue unitigs with a random walk in the graph by *N* past their natural end.

-l, --min-begin-node-length=***N***

Only begin unitigs collection from nodes which have at least length *N*.

7.4.4. Program Information

-h, --help

Print a help message for **odgi unitig**.

7.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

7.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

7.7. AUTHORS

odgi unitig was written by Erik Garrison.

8. odgi viz(1)

8.1. NAME

odgi_viz - variation graph visualizations

8.2. SYNOPSIS

odgi viz [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

8.3. DESCRIPTION

The odgi viz(1) command can produce a linear, static visualization of an odgi variation graph. It aggregates the pangenome into bins and directly renders a raster image. The binning level depends on the target width of the PNG to emit. Can be used to produce visualizations for gigabase scale pangenomes. For more information about the binning process, please refer to odgi bin. If reverse coloring was selected, only the bins with a reverse rate of at least 0.5 are colored. Currently, there is no parameter to color according to the sequence coverage in bins available.

8.4. OPTIONS

8.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

-o, --out=***FILE***

Write the visualization in PNG format to this file.

8.4.2. Visualization Options

-x, --width=*N*

Set the width in pixels of the output image.

-y, --height=*N*

Set the height in pixels of the output image.

-P, --path-height=*N*

The height in pixels for a path.

-X, --path-x-padding=*N*

The padding in pixels on the x-axis for a path.

-R, --path-per-row

Display a single path per row rather than packing them.

-L, --link-path-pieces=*FLOAT*

Show thin links of this relative width to connect path pieces.

-A, --alignment-prefix=*STRING*

Apply alignment related visual motifs to paths which have this name prefix.

-S, --show-strand

Use red and blue coloring to display forward and reverse alignments. This parameter should only be set in combination with [**-A, --alignment-prefix**=*STRING*].

8.4.3. Threading

-t, --threads=*N*

Number of threads to use.

8.4.4. Program Information

-h, --help

Print a help message for **odgi viz**.

8.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

8.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

8.7. AUTHORS

odgi viz was written by Erik Garrison.

9. odgi paths(1)

9.1. NAME

odgi_paths - embedded path interrogation

9.2. SYNOPSIS

odgi paths [**-i, --idx**=*FILE*] [*OPTION*]...

9.3. DESCRIPTION

The odgi paths(1) command allows the investigation of paths of a given variation graph. It can calculate overlap statistics of groupings of paths.

9.4. OPTIONS

9.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to investigate the paths from. The file name usually ends with *.og*.

-O, --overlaps=*FILE*

Read in the path grouping file to generate the overlap statistics from. The file must be tab-delimited. The first column lists a grouping and the second the path itself. Each line has one path entry. For each group the pairwise overlap statistics for each pairing will be calculated and printed to stdout.

9.4.2. Investigation Options

-L, --list-paths

Print the paths in the graph to stdout. Each path is printed in its own line.

-H, --haplotypes

Print to stdout the paths in an approximate binary haplotype matrix based on the graph's sort order. The output is tab-delimited: **path.name**, **path.length**, **node.count**, **node.1**, **node.2**,

node.n. Each path entry is printed in its own line.

-D, --delim=*CHAR*

The part of each path name before this delimiter is a group identifier. This parameter should only be set in combination with [**-H, --haplotypes**]. Prints an additional, first column **group.name** to stdout.

-f, --fasta

Print paths in FASTA format to stdout.

9.4.3. Threading

-t, --threads=*N*

Number of threads to use.

9.4.4. Program Information

-h, --help

Print a help message for **odgi paths**.

9.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

9.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

9.7. AUTHORS

odgi paths was written by Erik Garrison.

10. odgi prune(1)

10.1. NAME

odgi_prune - remove complex parts of the graph

10.2. SYNOPSIS

odgi prune [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

10.3. DESCRIPTION

The odgi prune(1) command can remove complex parts of a graph. One can drop paths, nodes by a certain kind of edge coverage, edges and graph tips. Specifying a kmer length and a maximum number of furcations, the graph can be broken at edges not fitting into these conditions.

10.4. OPTIONS

10.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to load in. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the pruned graph to *FILE*. The file name should end with *.og*.

10.4.2. Kmer Options

-k, --kmer-length=*N*

The length of the kmers to consider.

-e, --max-furcations=*N*

Break at edges that would induce *N* many furcations in a kmer.

10.4.3. Node Options

-d, --max-degree=*N*

Remove nodes that have a higher node degree than *N*.

-c, --min-coverage=*N*

Remove nodese covered by fewer than *N* number of path steps.

-C, --max-coverage=*N*

Remove nodes covered by more than *N* number of path steps.

-T, --cut-tips=*N*

Remove nodes which are graph tips.

10.4.4. Edge Options

-E, --edge-coverage

Remove edges outside of the minimum and maximum coverage rather than nodes. Only set this argument in combination with [**-c, --min-coverage**=*N*] and [**-C, --max-coverage**=*N*].

-b, --best-edges=*N*

Only keep the *N* most covered inbound and output edges of each node.

10.4.5. Path Options

-D, --drop-paths

Remove the paths from the graph.

10.4.6. Threading

-t, --threads=*N*

Number of threads to use.

10.4.7. Program Information

-h, --help

Print a help message for **odgi prune**.

10.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

10.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

10.7. AUTHORS

odgi prune was written by Erik Garrison.

11. odgi unchop(1)

11.1. NAME

odgi_unchop - merge unitigs into single nodes

11.2. SYNOPSIS

odgi unchop [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

11.3. DESCRIPTION

The odgi unchop(1) command merges each unitig into a single node.

11.4. OPTIONS

11.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to unchop. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the unchopped dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

11.4.2. Program Information

-h, --help

Print a help message for **odgi unchop**.

11.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

11.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

11.7. AUTHORS

odgi unchop was written by Erik Garrison.

12. odgi normalize(1)

12.1. NAME

odgi_normalize - compact unitigs and simplify redundant furcations

12.2. SYNOPSIS

odgi normalize [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

12.3. DESCRIPTION

The odgi normalize(1) command unchops a given variation graph and simplifies redundant furcations.

12.4. OPTIONS

12.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to normalize. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the normalized dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

-I, --max-iterations=*N*

Iterate the normalization up to *N* many times. The default is *10*.

12.4.2. Program Debugging

-d, --debug

Print information about the normalization process to stdout.

12.4.3. Program Information

-h, --help

Print a help message for **odgi normalize**.

12.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

12.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

12.7. AUTHORS

odgi normalize was written by Erik Garrison.

13. odgi subset(1)

13.1. NAME

odgi_subset - extract subsets of the graph as defined by query criteria

13.2. SYNOPSIS

odgi subset [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

13.3. DESCRIPTION

Extracting a node subset of a variation graph is the task of the odgi subset(1) command. Users can specify a node, a list of nodes or a context of which to generate a subset from.

13.4. OPTIONS

13.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to subset. The file name usually ends with *.og*.

-o, --out=***FILE***

Write the subset to this file.

13.4.2. Traversal Options

-l, --node-list=***FILE***

A file with one node identifier per line. All nodes specified here will be extract as a subset.

-n, --node=***N***

Specify the node identifier from which our traversal should begin.

-c, --context=***N***

The number of steps away from which to begin our traversal.

13.4.3. Threading

-t, --threads=***N***

Number of threads to use.

13.4.4. Program Information

-h, --help

Print a help message for **odgi subset**.

13.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

13.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

13.7. AUTHORS

odgi subset was written by Erik Garrison.