|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Long form** | **Short form** | **Function** | **Data type (default)** | **cmd\_line field** |
| --remove bubbles | -a | remove bubbles from graph (max length of the branch is hardwired) | bool (false) | remove\_bubbles |
| --mem\_width | -b | max size of bucket in hash table | int (100) | bucket\_size |
| --clip\_tip *n* | -c *n* | remove tips of length *n*  or shorter from the graph | bool (false) | tip\_clip |
| int (100) | tip\_length |
| --output\_contigs *<filename>* | -d *<filename>* | dump contigs using double-Y code (equivalent to –f *<filename>* -A Y\_PATH) | bool (false) | output\_fasta |
| algorithm is set | algorithm |
| *<filename>* required | output\_fasta\_filename |
| --output\_coverages | -e | print coverages when dumping contigs (it only makes sense if we are dumping contigs) | bool (false) | output\_coverages |
| --output\_fasta *<filename>* | -f *<filename>* | dumps contigs in fasta format (as defined by -A entry) | bool (false) | output\_fasta |
| *<filename>*  required | output\_fasta\_filename |
| --help | -h | print usage text |  |  |
| --input *<filename>* | -i *<filename>* | file of filenames to be processed (format *<filename> <read\_start> <read\_end>*) | bool(false)  *<filename>* required | input\_file  input\_filename |
| --health\_check | -j | health check graph | bool (false) | health\_check\_binary |
| --kmer\_size *n* | -k *n* | kmer size (has to be odd) | int (21) | kmer\_size |
| --log\_file *n* | -l *<filename>* | filename of log file to write to. | bool (false)  <filename> required | output\_log  log\_filename |
| --mem\_height *n* | -n *n* | Max number of buckets in hash table in bits (2*n*) | int (10) | number\_of\_buckets\_bits |
| --dump\_binary*<filename>* | -o *<filename>* | dumps graph in a binary format | bool (false) | dump\_binary |
| *<filename>* required | output\_ctx\_filename |
| --quality\_score\_offset *n* | -p *n* | quality score offset (to distinguish between Illumina 63 offset from traditional 33) | int (33) | quality\_score\_offset |
| --quality\_score\_threshold *n* | -q *n* | Threshold for quality score, any kmer containing one base below the score is discarded. | int (0) | quality\_score\_threshold |
| --remove\_low\_coverage\_supernodes *n* | -s *n* | Removes any supernodes where the coverage of all the kmers is equal or below *n.* | bool (false) | remove\_low\_coverage\_supernodes |
| int (required) | remove\_low\_coverage\_supernodes\_threshold |
| --input\_format *<format>* | -t *<format>* | Defines format of input files. All files have to be of the same format. Supported formats:  binary | fasta | fastq | hash | bool (false) | input\_file\_format\_known |
| *<format>* required (enum(FASTA,FASTQ,CTX,HASH) | input\_file\_format |
| --remove\_seq\_errors | -u | Removes supernodes with coverage 1 (this option is here for simplicity)  It sets  remove\_low\_coverage\_supernodes= true  remove\_low\_coverage\_supernodes\_threshold=1 |  |  |
| --verbose | -v | Verbose (we are not very consistent with this) | bool (true) | verbose |
| --detect\_bubbles *d,l* | -w | Detect bubbles, searching paths to a depth *d* and length *l*. | bool (false)  int (required)  int (required) | detect bubbles  bubble\_max\_depth  bubble\_max\_length |
| --singleton\_length | -x | Minimum size of a contig with both sides being blunt to be printed. | Int (100) | singleton\_length |
| --remove\_low\_coverage\_kmers *n* | -z *n* | Remove all kmers with coverage smaller or equal than *n.* | bool (false)  int (required) | low\_coverage\_node\_clip  node\_coverage\_threshold |
| --algorithm *<algorithm>* | -A | Specify walking algorithm. | enum(PERFECT\_PATH, BRANCHES, Y\_WALK, BUBBLES, READ\_PAIR)  (PERFECT\_PATH) | algorithm |
| --read\_pair\_min\_bits *n* | -B | Minimum number of bits common bits required for a read pair match. | bool (false)  int (3) | read\_pair\_enabled  read\_pair\_min\_bits |
| --read\_pair\_max\_coverage *n* | -C | Maximum average coverage of the starting Y-path – for paths over this, we don’t try to find a mate. | bool (false)  int (20) | read\_pair\_enabled  read\_pair\_coverage |
| --read\_pair\_max\_distance *n* | -D | Maximum number of nodes to walk looking for matching mate. | bool (false)  int (100) | read\_pair\_enabled  read\_pair\_distance |
| --read\_pair\_max\_paths *n* | -E | Maximum number of paths to store, from which best match is chosen. | bool (false)  int (16) | read\_pair\_enabled  read\_pair\_max\_paths |
| --output\_base\_space | -F | Translate the reads from colour space to base space (just valid with SOLiD). | Bool (false) | output\_base\_space |
| --output\_graph *<filename>* | -G | Output a graph file. Currently outputs Graphviz file for debugging. | bool (false)  *<filename>* required | graphviz  output\_graphviz\_filename |
| --input\_reference <*filename*> | -H | Input reference. Has to be fasta. Used to get the coverage | *<filename>* required  bool (false) | input\_reference  input\_reference\_known; |
| --pair\_info *<filename>* | -I | Specifies the read pair info file. | bool (false)  *<filename>* required | read\_pair\_enabled  read\_pair\_filename |
| -- output\_kmer\_coverage*<filename>* | -J | Specifies the filename where the coverage of the graph is going to be printed. | *<filename>* required  bool(false) | output\_kmer\_coverage  output\_kmer\_coverage\_known |
| --read\_pair\_min\_unvisited\_kmers *n* | -K | Minimum number of unvisited kmers in read pair start path. | bool (false)  int (0? kmer\_size+1?) | read\_pair\_enabled  read\_pair\_min\_kmers |
| --remove\_spurious\_links *d,c* | -L | Remove spurious edges from the graph. Look for maximum coverage difference of *d* and minimum coverage of *c*. | bool (false)  int (required)  int (required) | remove\_spurious\_links  remove\_spurious\_links\_max\_difference  remove\_spurious\_links\_min\_coverage |
| --output\_reference\_coverage\_file | -M | Outputs the coverage of a given reference. The first step is to load the graph, then iterating with the kmers of the reference, we print the reference (from --input\_reference) | <filename> required  bool(false) | output\_reference\_coverage\_file  output\_reference\_coverage\_file\_known |
| --print\_stack\_as\_n | -N | If set, causes read pair stack to be printed as ‘N’. Otherwise, will be printed as lower case.  Currently, behaviour is reversed – need to change! | bool (true) | print\_uncertain\_as\_n |
| --hash\_output\_file <filename> | -O | Dumps the hash table with all the information required to restore the hash table in a new cortex instance. | *<filename>* required | output\_hash\_filename  dump\_hash |
| --tip\_clip\_iterations | -P *n* | Sets a maximum number of iterations. The default is 100. The tip clip will stop once the algorithm didn’t chage the graph or this limit is reached | *Int (100)* | tip\_clip\_iterations |
| --qual\_file *<filename>* | -Q | This will be the parameter to read the 454 data. | *<filename>* required | qual\_filename |
| --454 | -R |  |  | input\_file\_format\_known  input\_file\_format |
| --read\_pair\_min\_start\_length *n* | -S | Minimum length of read pair start path required for pairs finding to begin. | bool (false)  int (1000) | read\_pair\_enabled  read\_pair\_start\_length |
| --threads *n* | -T | Number of threads to use | int (1) | threads |
| --max\_read\_len *n* | -Z *n* | Maximum read length for fastq entries | int(1000) | max\_read\_len |

Options to remove

-M seems to be the same as –c.

-P seems to be the same as -s

The following options are in the code at the moment, but missed out from Mario’s list – do we need any of them?

-m –-mem

-p –-branch\_length

-r –-prune\_single-coverage\_edges

-t –mem\_status

Reverted to –dump\_binary, as it was like that in the code.

Remove\_spurious\_links is not yet in the cmd\_line.c. Is it working?