ASSEMBLER PIPELINE I/O

This document is the **defining document** for the precise information content of every message that flows through the Celera Assembler pipeline. As such, it contains precise message specifications for the input and output of the assembler. The document describes the messages in order of their introduction along the assembler pipeline, and provides an Appendix with concrete examples of each input/output message.

1 Conventions

The Assembly Group has adopted an organic software development strategy, rapidly prototyping pipeline components and then evolving them into robust components. To this end, the individual modules are engineered to communicate via a simple ASCII-based encoding of the pipeline messages, which can be switched over to a more compact and efficient binary representation in production situations. The requirement for the ASCII encoding was that it be easy to read by a human (aiding debugging), while also being trivial to parse. The result is the 3-code formatted messages described within this document.

1.1 3-code format

The format of all messages is called 3-code because every field name and message type name is compressed to a 3 letter abbreviation, with the added convention that type names are all capital-letters and field names are all lower-case letters. A record is encoded across several lines of input where the first line has a '{' in column 1 and the last line consists solely of a '}' in column 1. The 3-code for the message type name is in columns 2-4 of the header line, followed immediately by a new-line. The lines between the header and tail encode the fields of the record. Each field-line has the 3-code for the field in columns 1-3 and a ':' in column 4, followed immediately by the relevant data in columns 5 to the end-of-line, or in subsequent lines in the case of multi-line fields.

1.2 Data formats

1.2.1 Numeric and character data

For the messages specified in this document, the Pascal data structure is given (at left) along with the corresponding 3-code (in shaded area at right). The encoding of the data for each field is given by specifying the *scanf* UNIX format that would correctly read the input. Thus, for example, '%ld' reads a long and '%1[AD]' reads one character, from the two characters in square brackets. By convention, scalar values are encoded as a single capital letter and the sequence of letters corresponds directly to the sequence of value names in the scalar definition.

1.2.2 String Data

Arbitrary length strings, as needed for 'Quality', 'Sequence' and 'Source' fields, are encoded as a series of new-line terminated strings ending with a line containing a '.' in column 1. In the *scanf* translation specification of such message components, we take the liberty of using regular expression syntax for these fields, with the symbol '↵' denoting a new-line. The encoded string is the concatenation of all the characters save the new-lines and the terminating period.

1.3 Sequence Intervals

Sequence intervals are specified as a pair of positions within a sequence and positions are the points between symbols of the sequence. The leftmost position is numbered 0, so that for example, (0,4) specifies

the first 4 symbols of a sequence, while (2,2) specifies the position between the second and third symbols.

```
SeqInterval: record bgn, end: int32 end
```

1.4 Alignment Positions

For every fragment in a contig/unitig alignment, a complete record of its alignment to the gapped consensus is supplied. The endpoints of the fragment in the alignment are specified as a SeqInterval, while the detailed alignment is given by a 'delta encoding'. This delta is a series of positive integers indicating the positions within the fragment at which to insert a dash to align to the consensus sequence. Note that in aggregate, these records specify the complete layout of the contig/unitig, and a precise representation from which to reconstruct the multi-alignment with just a bit of additional effort.

1.5 Quality Values

Phred quality values are integers in the range [0,60]. An encoding based on a series of, say blank-separated integer constants, is too space consumptive for even our modest prototyping requirements, so we choose to encode these numbers as a series of printable ASCII characters. To wit, a value *i* is mapped to the ASCII symbol *'0'+i*. Thus a sequence of Phred numbers is mapped to a sequence of characters and encoded in a 3-code record as a string.

2 Inputs

The input to the assembler is prefixed with an "audit record" for tracking purposes. Following the audit record, there are three main components to the Assembler input: fragments, links and distance records, and screen and repeat items.

2.1 Audit Records

Every pipeline transmission batch will have an audit record as its first item. An audit record will consist of a list of the agents that produced the batch in the sequence they were applied, and for each agent the name of the agent, time of completion, version number, and a possibly empty comment are specified:

```
AuditMesg:
  list of AuditLine
```

```
{ADT
(<ADL-record>↵)*.
}
```

```
AuditLine:
record
 name:     string
 complete: time_t
 version:  string
 comment:  string
end
```

```
{ADL

who:%s
ctm:%d
vsn:%s
com:%s
}
```

2.2 Fragments (reads and guides)

The primary input to the assembler is a **fragment message**, which can describe a Celera read or a guide. Guides are sub-categorized as either being (a) BAC-ends, (b) pseudo-reads from unfinished BACs, (c) pseudo reads from finished BACs, and (d) STSs. The differences between read and guide fragment

messages are as follows.

1. **Quality**: Every read has a quality vector and sequence clear range, whereas guides need not. In the absence of quality values, the quality field is NULL and the clear range is the entire fragment.
2. **Locale UID**: Every guide has an associated locale whereas this field is undefined for reads. The interpretation of the locale is different for each kind of guide, but is always expected to be a 64-bit UID produced by the Celera database. For the BAC-based guides it is a UID assigned to the particular BAC from which the guide came. The assembler no requirement on this UID other than that a distinct integer be given to each BAC. If over time, end reads, then contigs, and finally finished sequence for a BAC become available, the same BAC number should be given to the associated guides for that BAC. For an STS guide, a distinct locale number should be assigned to each bin formed when a sufficiently high LOD score is used to order STSs. We suggest 6.
3. **Locale Position**: Each contig of an unfinished BAC and each finished BAC is assumed to have been partitioned into a set of neatly overlapping pseudo-reads that are given as guides to the assembler. For these pseudo-reads, the assembler requires the interval of the underlying BAC from which the pseudo-read was excised, communicated in the locpos field. This field is defined only for unfinished and finished BACs. In the case of the several unordered contigs for a given unfinished BAC, one would simply assign a disjoint interval to each contig and then give the position of each pseudo-read of a contig with respect to the assigned interval, (see for interval specification).

```
FragMesg:                                          {FRG
record
 action:    scalar (AS_ADD,AS_DELETE)              act:%1[AD]
 accession: Fragment_ID                            acc:%ld
 variant of action:
  AS_ADD:
   record
    type: scalar (AS_READ,AS_EXTR,AS_TRNR,         typ:%1[RXTELUFS]
                  AS_EBAC,AS_LBAC,AS_UBAC,
                  AS_FBAC,AS_STS)
     locale:     Locale_ID                         loc:%ld
     locpos:     SeqInterval                       pos:%d,%d
     source:     "description of data source"      src:↵(%[^\n]↵)*.
     entry_time: time_t                            etm:%d
     sequence:   string(char)                      seq:↵(%[^\n]↵)*.
     quality:    string(bytes)                     qlt:↵(%[^\n]↵)*.
     clear_rng:  SeqInterval                       clr:%d,%d
    end
 end
end                                                }
```

2.3 Links and associated distance records

After the relevant fragments have been added to the system one may then add (or delete) pairwise distance constraints or **links** between them. A link message contains the type of link being added or deleted and the two fragments involved. If a link is being added then one also needs to specify the time of entry, a reference to the distance record specifying the distance range between the fragments, and a scalar indicating whether the fragments are in the same, opposite orientation, or unknown orientation. Note that mates and BAC guides are always in the opposite orientation with respect to each other. The distance constraint always refers to the distance between the 5' end of the two fragments, regardless of orientation. The last two link types model user input constraints and may be between any pair of fragments in the system.

Links are divided into six categories according to the source of the link:

- **AS_MATE** links are for mated pairs of 2K, 10K, 50K, and transposon library end reads from the Celera sequencing pipeline and external sources of whole genome shotgun sequence, or from UBAC fragments that were sequences from opposite end of subclones.
- **AS_BAC_GUIDE** links are between end-sequenced BACs.
- **AS_STS_GUIDE** links are between paired STSs.
- **AS_REREAD** links permit one to specify that two reads were sequenced from the same end of an insert. These are rereads that were resequenced for some reason, e.g. the PCR-prep encountered a mononucleotide repeat and stuttered, and thus was resequenced with a plasmid prep. In this case neither the distance or orientation fields convey any information.
- **MAY_JOIN** links represent single links that may be incorporated if there is not conflicting information.
- **MUST_JOIN** links represent infinitely weighted links that will be followed if at all possible.

The distance between guides and mates are specified in **distance records** that are passed to the assembly system as records requesting that a given distance entity be added or deleted. The distance record specifies the action, the ID of the distance entity, and (in the case of insertion) the normal distribution from which the distances were sampled. The fields `mean` and `stddev` give the mean standard deviation of the distribution. Thus, for example, 99% of all links referring to a particular distance message will be of length in `[mean-3stddev, mean+3stddev]`. Note that there should be exactly one distance record for each insert library, even if the library was designed to have insert sizes equal to that of another library. The reason for this is that the assembler will be empirically determining a distribution of observed mate distances and these distributions will be different, even for libraries designed to have the same mean distance.

The orientation field specifies the relative orientation of the two fragments. Links representing sequence of opposite ends of some type of insert (AS_MATE or AS_BAC_GUIDE) must specify an AS_INNIE orientation (3' ends are adjacent) except in the case of mated pairs of reads from a transposon library, which must specify an AS_OUTTIE orientation. Specifying an AS_UNKNOWN orientation is equivalent to specifying 4 links, each with one of the possible orientations.

```
Distance_ID, Fragment_ID, Screen_Item_ID: int64

LinkMesg:                                              {LKG
record
 action: scalar (AS_ADD,AS_DELETE)                     act:%1[AD]
 type:   scalar (AS_MATE,AS_BAC_GUIDE,                 typ:%1[MBSRYT]
                 AS_STS_GUIDE, AS_REREAD,
                 AS_MAY_JOIN,AS_MUST_JOIN)
 frag1: Fragment_ID                                    fg1:%ld
 frag2: Fragment_ID                                    fg2:%ld
 variant of action:
  AS_ADD:
   record
    entry_time:  time_t                                etm:%d
    distance:    Distance_ID                           dst:%ld
    orientation: scalar (AS_NORMAL, AS_ANTI            ori:%1[NAIOU]
                         AS_INNIE,AS_OUTTIE,
                         AS_UNKNOWN
   end
  end
end                                                    }


DistanceMesg:                                          {DST
```

```
record
 action:    scalar (AS_ADD,AS_DELETE)
 accession: Distance_ID
 mean:      float32
 stddev:    float32
end
```

```
act:%1[AD]
acc:%ld
mea:%f
std:%f
}
```

2.4 Screen Items

Screen items are used to specify vector, contaminant, or repeat sequences that should be masked or tagged for the purposes of assembly. The repeat_id field indicates a user determined classification of the item, while the relevance bit-vector field is used to indicate how matches should be handled within the screener or in subsequent programs. For example, in the case of ubiquitous repeats, a relevance value that binary-ands with AS_OVL_HEED_RPT (1) instructs the overlapper not to base overlaps on sequences in matching intervals. Simple repeats (and heterochromatin) should have relevance fields that binary-and with AS_URT_IS_SIMPLE (8) to instruct the screener to consolidate matches more effectively.

```
ScreenItemMesg:
record
 type:       scalar(AS_UBIQREP,AS_CONTAMINANT)
 accession:  ScreenItem_ID
 repeat_id: [0…Num_of_Repeat_Types-1]
 relevance:  int32
 source:     "description of data source"
 sequence:   string(char)
 variation:  float32 in [0,.1]
 min_length: int32
end
```

```
{SCN

typ:%1[UC]
acc:%ld
rpt:%d
rel:%d
src:↵(%[^\n]↵)*.
seq:↵(%[^\n]↵)*.
var:%f
mln:%d
}
```

```
RepeatItemMesg:
record
 repeat_id: [0…Num_of_Repeat_Types-1]
 which:     string(char)
 length:    int32
end
```

```
{RPT

rpt:%d
wch:%s
len:%d
}
```

3 Outputs

The assembler output consists essentially of the Extended Unitig Graph, the Extended Contig Graph and the Scaffold. The nodes of the unitig graph are output as Unitig messages, and the edges are output as UnitigLink messages. Similarly, the contig graph is output as Contig and ContigLink messages. The scaffold is repesented by a series of Scaffold messages. These basic snapshot components are accompanied by a) Augmented Fragment messages, which present the assembler's determination of each fragment's screen matches, mate status, chimeric status and clear range, and b) Mate Distribution messages, which summarize the mean and standard deviation of the mate distances *AS ASSEMBLED* (versus as input).

3.1 The Augmented Fragment Messages

Assembler annotation of individual fragments is provided by augmented fragment messages (`AFG`), emitted for each fragment that the assembler has processed. The accession indicates the Celera UID assigned to the fragment being augmented. Any Screen Matches found during the assembler's screening phase are provided, along with the assembler's evaluation of the fragment's mate status and chimerism (all chimeric

fragments will appear in singleton unitigs). The mate status field should be interpreted as follows.

- 'G' GOOD_MATE: the mate is confirmed by the assembly
- 'B' BAD_MATE: the mate is inconsistent with the assembly
- 'N' NO_MATE: no mate for the fragment was input into the assembler
- 'U' UNRESOLVED_MATE: mate is neither confirmed nor inconsistent with the assembly

The clear range field reflects any changes to the clear range made within the assembler.

```
AugFragMesg:                                        {AFG
record
 accession:     Fragment_ID                         acc:%ld
 screened:      list of ScreenMatch                 scn:↵<SMA-record>*.
 mate_status:   scalar (GOOD_MATE, BAD_MATE,
                 NO_MATE,UNRESOLVED_MATE)           mst:%1[GBNU]
 chimeric:      Boolean                             chi:%d
 clear_rng:     SeqInterval                         clr:%d,%d
end                                                 }
```

Each Screen Match record provides the affected interval of the fragment ('where'), which Screen Item was matched ('what'), the repeat identifier and relevance (described in) and the matching interval and orientation within the Screen Item

```
ScreenMatch:                                        {SMA
record
 where:      SeqInterval in fragment                whr:%d,%d
 what:       Screen_ID                              wht:%ld
 repeat_id:  [0…Num_of_Repeat_Types-1]              rpt:%d
 relevance:  int32                                  rel:%d
 portion_of: SeqInterval                            pof:%d,%d
 direction:  scalar (AS_FORWARD,AS_REVERSE)         dir:%1[FR]
end                                                 }
```

3.2 The Extended Unitig Graph

A Unitig message provides a unique accession number assigned by the assembler, and communicates several calculated statistics on the unitig, as well as its multi-alignment. The assembler passes along the unitig coverage statistic, which is used internally to determine whether a unitig should be classified as unique. The unitig status should be interpreted as follows.

- 'U' `AS_UNIQUE:`appears in only one contig
- 'C' `AS_CHIMER`: represents a single fragment that has been deemed chimeric
- 'N' `AS_NOTREZ:` an repeat containing some fragments that can not be assigned uniquely
- 'S' `AS_SEP:` an repeat whose fragments have all been assigned to unique contigs (a.k.a.separable repeat)
- 'X' `AS_UNASSIGNED:` status as yet undetermined

An interesting branch point at either end of the unitig is output using the a_branch_point and b_branch_point fields. The magnitude of the field value specifies how far from the respective unitig ends the branch point is located, while the sign indicates the orientation of the branch points. A positive value indicates a branch from repeat into unique, while a negative value from unique into repeat. If no interesting branch point is detected (i.e., none within a certain fixed distance from the end, currently 1000 b.p.), the field will be set to zero.

The remaining fields provide the multi-alignment of the unitig, including the gapped consensus and quality strings (and their length), an indicator of whether any fragments were forced into the alignment, and the complete encoding of the multi-alignment as described in .

```
Chunk_ID: int64

UnitigMesg:                                    {UTG
record
 accession:      Chunk_ID                      acc:%ld
 source:         "description of data source"  src:↲(%[^\n]↲)*.
 coverage_stat:  float32                       cov:%f
 status:         scalar (AS_UNIQUE,AS_CHIMER,
                         AS_NOTREZ,AS_SEP,     sta:%1[UCNSX]
                         AS_UNASSIGNED)
 a_branch_point: int32                         abp:%d
 b_branch_point: int32                         bbp:%d
 length:         int32                         len:%d
 consensus:      string(char)                  cns:↲(%[^\n]↲)*.
 quality:        string(bytes)                 qlt:↲(%[^\n]↲)*.
 forced:         boolean                       for:%d
 num_frags:      int32                         nfr:%d
 f_list:         list of MultiPos              (<MPS-record>↲)*
end                                            }

MultiPos: record                               {MPS
 type: scalar (AS_READ,AS_EXTR,AS_TRNR,        typ:%1[RXTELUFS]
               AS_EBAC,AS_LBAC,AS_UBAC,
               AS_FBAC,AS_STS)
 ident:        IntFragment_ID                  mid:%d
 source:       "description of data source"    src:↲(%[^\n]↲)*.
 position:     SeqInterval                     pos:%d,%d
 delta_length: int32                           dln:%d
 delta:        list of int16                   del:↲((%d )*↲)*
end                                            }
```

Each Unitig Link Message identifies a pair of unitigs and the orientation of the overlap between them. The orientation is encoded as a scalar with the following interpretation:

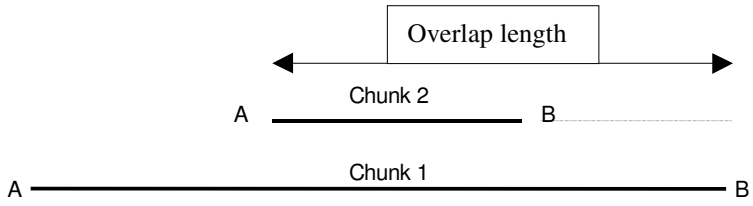
- ‘N’ Normal
- ‘I’ Innie
- ‘A’ Anti-Normal
- ‘O’ Outie

In the case of the containment overlaps, there are only two possible orientations for each overlap, innie or outie (the unitigs are either aligned in the same direction, or in opposite directions).

The overlap_type specifies the relationship between the unitigs:

- 'N' No overlap
- 'O' Regular overlap
- 'T' Tandem overlap
- 'C' chunk1 contains chunk2
- 'I' chunk2 contains chunk1

For contained unitigs, the overlap distance is specified as if the contained unitig was extended past the B end of the containing unitig.



If the number of contributing edges is two, and a single read is required for both edges, then `is_possible_chimera` is set to true. This will happen if a read is part of a mate in the other chunk and is also required for the chunks to overlap. If the edge includes a guide, the `includes_guide` is set to TRUE. The `mean_distance` and `std_deviation` fields describe the distribution of the edge distances represented in the link (a negative distance means the unitigs overlap). The number of edges (mates and potentially a chunk overlap) contributing to the mate edge is given by the field num_contributing. The status field, determined late in the process after a best scaffold has been chosen, gives the status of the link with respect to the assembly. Finally, the `jump_list` gives a list of all pairs contributng to the edge. The length of the jump_list corresponds to the number of contributing edges if `overlap_type` takes the value AS_NO_OVERLAP. Otherwise the length of the jump_list will be num_contributing – 1.

```
UnitigLinkMesg:                                    {ULK
record
 unitig1                                           ut1:%ld
 unitig2:              Chunk_ID                    ut2:%ld
 orientation:          scalar {AB_AB, BA_BA,
                               BA_AB, AB_BA}       ori:%1[NAOI]
 overlap_type:         scalar (AS_NO_OVERLAP,
                        AS_OVERLAP,
                        AS_TANDEM_OVERLAP,
                       AS_1_CONTAINS_2,
                       AS_2_CONTAINS_1)            ovt:%1[NOTCI]
 is_possible_chimera:  boolean                     ipc:%d
 includes_guide:       boolean                     gui:%d
 mean_distance:        float32                     mea:%f
 std_deviation:        float32                     std:%f
 num_contributing:     int32                       num:%d
 status:               scalar (AS_IN_ASSEMBLY,
                        AS_POLYMORPHISM,
                        AS_BAD,AS_CHIMERA,
                        AS_UNKNOWN_IN_ASSEMBLY)    sta:%1[APBCU]
 jump_list:            list of Mate_Pairs          jls:↲(%d,%d↲)*
end                                                }

Mate_Pairs:
record
 in1, in2:             Fragment_ID
```

```
 type:                 scalar (AS_MATE,
                        AS_BAC_GUIDE,
                       AS_STS_GUIDE,
                       AS_MAY_JOIN,
                       AS_MUST_JOIN)
end
```

```
%d,%d,%1[MBSYT]
```

3.3 Extended Contig Graph

Contigs are ordered collections of Unitigs (and Surrogates, pending placement of all fragments) that cover contiguous regions of the genome. A contig is composed of fragments from the contained Unitigs, as well as fragment-like "surrogates", which are subsets of repeat Unitigs introduced to span gaps in contigs in the absence of complete repeat resolution. A Contig Message provides a unique accession number assigned by the assembler and a representation of the multi-alignment of the contig. The consensus sequence is **gapped**, that is, it will contain dash characters as needed to allow the alignment of all fragments. For each of the 'num_pieces' component fragments, a MultiPos record is given specifying its location in the multi-alignment and a delta encoding of the alignment to the consensus. Further, for each 'num_unitigs' component unitig, an ElementPos record is given specifying the extent of the unitig within the gapped consensus.

```
ConConMesg:
record
 accession:       Contig_ID
 length:          int32
 consensus:       string(char)
 quality:         string(bytes)
 forced:          boolean
 num_pieces:      int32
 num_unitigs:     int32
 pieces:          list of MultiPos
 unitigs:         list of ElementPos
end
```

```
{CCO

acc:%ld
len:%d
cns:↲(%[^\n]↲)*.
qlt:↲(%[^\n]↲)*.
for:%d
npc:%d
nou:%d
(<MPS-record>↲)*
(<EPS-record>↲)*
}
```

```
ElementPos: record
 type:            scalar (AS_UNITIG)
 ident:           Chunk_ID
 position:        SeqInterval
end
```

```
{EPS
typ:%1[U]
lid:%d
pos:%d,%d
}
```

The edges in the contig graph are represented by Contig Link edges, direct analogs of the Unitig Link Edges in the unitig graph. The only difference is in the objects being related.

```
ContigLinkMesg:
record
 contig1:                Contig_ID
 contig2:                Contig_ID
 orientation:            scalar {AB_AB, BA_BA,
                          BA_AB, AB_BA}
 overlap_type:           scalar
(AS_NO_OVERLAP,
                          AS_OVERLAP,
                          AS_TANDEM_OVERLAP)
 is_possible_chimera:    boolean
 includes_guide:         boolean
```

```
{CLK

co1:%ld
co2:%ld

ori:%1[NAOI]

ovt:%1[NORT]
ipc:%d
gui:%d
mea:%f
```

```
 mean_distance:              float32
 std_deviation:              float32
 num_contributing:           int32
 status:                     scalar
(AS_IN_ASSEMBLY,
                              AS_POLYMORPHISM,
                              AS_BAD,AS_CHIMERA,

AS_UNKNOWN_IN_ASSEMBLY)
 jump_list:                  list of Mate_Pairs
end
```

```
    std:%f
    num:%d


    sta:%1[APBCU]
    jls:↵(%d,%d↵)*
}
```

3.4 The Scaffold

The Assembler's best choice for the scaffolds is output as "the" assembly. Each scaffold is given as a list of the pairs of adjacent contigs in the scaffold with the chi-squared estimate of the distance and standard deviation between the pair of contigs. The unitig and contig links supporting the scaffold are flagged by their AS_IN_ASSEMBLY status value.

In the list of contig pairs, the contigs are ordered from left to right across the scaffold. For example, if the first three contigs in a scaffold have ids 1, 2, & 3. Then in the list of contig pairs, the first pair of contigs would have contig1 = 1 and contig2 = 2, and the second pair would have contig1 = 2 and contig2 = 3. The orientation field describes the pairwise orientation of the two contigs within the scaffold. A scaffold may consist of a single contig, in which case the num_contig_pairs will be zero and the id of the second contig will repeat the first, with arbitrary orientation and distance.

```
ScaffoldMesg:
record
 accession:          Scaffold_ID
 num_contigs_pairs:  int32
 contig_pairs:       list of ContigPairs
end
```

```
    {SCF

    acc:%ld
    noc:%d
    (<CTP -record>↵)*
    }
```

```
ContigPairs:
record
 contig1:            Contig_ID
 contig2:            Contig_ID
 orientation:        scalar {AB_AB, BA_BA,
                      BA_AB, AB_BA}
 mean:               float32
 stddev:             float32
end
```

```
    {CTP

    ct1:%ld
    ct2:%ld

    ori:%1[NAOI]
    mea:%f
    std:%f
    }
```

3.5 Mate-Distance Distribution Messages

These messages are emitted to provide information on the distribution of mate lengths observed for those pairs whose mates lie in the same unitig or contig (and thus whose distance in the assembly is known precisely). For each mate-link distance record provided as input to the assembler with such a contributing

pair, a message describing the observed distribution of corresponding mates in the current assembly is produced, with the 'refines' field referencing the original input record. The observed mean and standard deviation for mate pairs of this type in the current assembly are given, as well as the minimum and maximum distances observed for this type. Further, a histogram is provided of the number of contributing pairs within each distance subrange of the entire range from `min` to `max`.

```
MateDistMesg:
record
 refines:     Distance_ID
 mean:        float
 stddev:      float
 min:         int32
 max:         int32
 num_buckets: int32
 histogram:   list of int32
end
```

```
{MDI
ref:%ld
mea:%f
std:%f
min:%d
max:%d
buc:%d
his:↵(%d↵)*
}
```

4 Intermediate messages

The following table identifies each stage of the pipeline, and the component messages output from that stage. (The input is presumed to be the output from the previous stage.

Input	Gatekeeper	Screener	Overlapper	Unitiger	Scaffolder	Terminator
.frg	.inp	.urc	.ovl	.cgb	.cgw	.asm
ADT	ADT	ADT	ADT	ADT	ADT	ADT
FRG	IFG	SFG	OFG		IAF	AFG
LKG	ILK	ILK	ILK	ILK		
DST	IDT	IDT	IDT	IDT	IDT	MDI
SCN			OVL	IUM	IUM	UTG
RPT				UOM	IUL	ULK
					ICM	CCO
					ICL	CLK
					ISF	SCF

4.1 Gatekeeper

The assembler modules require consecutive IDs beginning at 1 for efficient indexing of internal and disk-based data structures. These 32-bit “IID’s” are assigned and added to every record containing a UID supplied by the external DMS with the exception of Repeat_Ids, which already have this property. Thus the GateKeeper module augments all input messages -- FRG, LKG, SCR, and DST -- with internal ID assignments and passes them on as IFG, ILK, ISN, and IDT messages. These messages are identical to the input counterparts save that:

All `acc`–fields that contain external references are converted to (External,Internal) accession number pairs, encoded in 3-format as “(%ld,%d)”. In the corresponding C-structs, the single field, say “<X>” to the external ID, is replaced with two fields “e<X>” and “i<X>” to the appropriate sized ints.

All other fields that contain external references are converted to internal accession numbers encoded in 3-code format as "%d" with a suitably modified field name, ("xyz" becomes "ixz", for example).

The Gatekeeper further checks all input for semantic consistency as described in the defining document for that stage.

4.2 Repeat Tagger/Contaminant Screener

The URT/URC module consumes ISC messages, passes IDT messages through unaltered, and adds to the IFG message. The consumed ISC messages are recorded in a screen index store on disk. To avoid any ambiguity the augmented fragment records are called “ScreenedFragMesg” and their 3-code header is “SFG”. The component IntScreenMatch records are the internal equivalents of the output ScreenMatch records described in .

```
IntScreen_ID: int32

ScreenedFragMesg:                                    {SFG
Record
 “As Before”
 clear_rng:   SeqInterval
```

```
 screened:    sorted list of IntScreenMatch
 "As Before"
end
```

```
scn:↵<ISM-record>*.

}
```

4.3 Overlapper

The Overlapper module stores screened fragments in a fragment store, passes on relevant fragment information to the subsequence stages in an OFG fragment message, and adds overlap messages (OVL) to the stream. An OFG fragment message message the SFG, save that:

1) the type name is `OFG` instead of `SFG`

2) the `seq` and `qlt` fields are absent.

Overlaps between fragments are described in "OverlapMesg" records as follows. It would be preferrable if the overlaps for a given fragment followed its `OFG` message and if that fragment were the `A_frag` for the relevant overlap records.

```
OverlapMesg:
record
 aifrag:       IntFrag_ID
 bifrag:       IntFrag_ID
 orientation:  scalar (AS_NORMAL,AS_INNIE,
                       AS_OUTTIE,AS_ANTI)
 overlap_type: scalar (AS_DOVETAIL,
                       AS_CONTAINMENT,
                       AS_SUPERREPEAT)
 a_hang:       int32
 b_hang:       int32
 quality:      float32
 min_offset,
 max_offset:   int32
 polymorph_ct: int32
 delta:        string(int)
end
```

```
{OVL

afr:%d
bfr:%d

ori:[NAIO]


olt:[DCM]
ahg:%d
bhg:%d
qua:%f
mno:%d
mxo:%d
pct:%d
del:↵((%d)*↵)*.
}
```

4.4 Unitiger

See output messages for description of the Unitig message. The internal version merely substitutes an internal ID for the Celera UID.

A series of independent UnitigOverlapMesg messages specify chunk graph edges. Each edge identifies a pair of chunks and the orientation of the overlap between them. Details of the orient and ovrelap_type fileds are identical to the UnitigLink description of . The UnitigOverlapMesg also provides the best, minimum, and maximum overlap length between the pairs of unitigs. The minimum and maximum equal, except in the case of edges induced by a tandem repeat, where the extent of possible 'slippage' in the overlap is thus indicated. Note that not every tandem edge will be detected by overlap alone, and the CGB will transitively infer when the overlapping parts of an edge involve tandem satellites. This inference will be reflected in the `overlap_type` field. In this internal context, the `AS_NO_OVERLAP` value is not exercised.

```
UnitigOverlapMesg:
record
 chunk1:              IntChunk_ID
 chunk2:              IntChunk_ID
 orient:              scalar (AB_AB, BA_BA,
                              BA_AB, AB_BA)
```

```
{UOM

ck1:%d
ck2:%d

ori:%1[NAOI]
```

```
  overlap_type:        scalar (AS_NO_OVERLAP,
                               AS_OVERLAP,
                               AS_TANDEM_OVERLAP,
                                AS_1_CONTAINS_2,
                                AS_2_CONTAINS_1)
  source:              "description of data"
  best_overlap_length: int32
  min_overlap_length:  int32
  max_overlap_length:  int32
  quality:             float32
  polymorph_ct:        int32
end
```

```
ovt:%1[NOTCI]
src:↵(%[^\n]↵)*.
len:%d
min:%d
max:%d
qua:%f
pct:%d
}
```

4.5 Scaffolder

The scaffolder reads Internal Unitig Messages, and outputs an internal representation of the extended unitig graph, the extended contig graph and the scaffold, as well as mate distance summaries. The internal versions of these messages correspond directly to their external counterparts.

A. Input Examples

```
{ADT
{ADL
who:dewim
ctm:939937796
vsn:0
com:
Genome Length is 130000000
}
{ADL
who:gatekeeper
ctm:942871386
vsn:(blank)
com:
     $Id: IOSpec.rtf,v 1.1.1.1 2004/04/14 13:41:50 catmandew Exp $
     $Id: IOSpec.rtf,v 1.1.1.1 2004/04/14 13:41:50 catmandew Exp $
     $Id: IOSpec.rtf,v 1.1.1.1 2004/04/14 13:41:50 catmandew Exp $
     $Id: IOSpec.rtf,v 1.1.1.1 2004/04/14 13:41:50 catmandew Exp $
     $Id: IOSpec.rtf,v 1.1.1.1 2004/04/14 13:41:50 catmandew Exp $
     $Id: IOSpec.rtf,v 1.1.1.1 2004/04/14 13:41:50 catmandew Exp $
     $Id: IOSpec.rtf,v 1.1.1.1 2004/04/14 13:41:50 catmandew Exp $
Complete call: gatekeeper -e 1000000 -P -f celera.Store celera
Started: Wed Nov 17 15:43:06 1999
Working directory: /dev4/data/drosophila/celera.Oct29/QV_WINDOW_NEW_PARAMS
.
}
}

{FRG
act:A
acc:17000001585819
typ:R
src:
.
etm:925759534
seq:
CCGTTGTGGCTTAGCTGGTTTGTGTGTGTTGGCTTGGACATTCTAAAACAAACTCGCACTGCTGCTACTT
CTCTAAAATCTGCATGCTTAATCTCAACCTTCTAGATTTTATAGTCATTGATAACATAACGAAGCTATTG
ACTTCCTTTGCAATAATAAAGCAAGTCATTTTGCAACAGTTTTCGTAGCAGCTACATGTGATTTTGATCA
TATAAAGATCAAGTTTTTAAATTGTAAACAATTTTTTAAATACAAATAAGTTATCTTACACTTAGATAAA
AAAAAATCAGTTTCGGTTCCATTTCAATACTTGTTTGACGTTTGGAACAAGCATTTGTTCAACAGCGTGT
AAAGGTAAAATGAAACGTTTTCCACCCTATAACACCAGTCGTTATGATTAAAATTCAGCATGAAGATTCT
GGAGACAAAGTTTGTTTTAGAAAAATTACCACGTACTCCCAGTTTACAAAACGTGCAAAAAACGTTTGGT
CACGCCCACTCTATCGCTTACAAACATCTGTTCTTGATCAAAAAGTATTCTAACTCAGTTAAAAACAGTT
TATTCAACAACAAGAGCGAATGCTAGAGTGTTGTAAATTTTTATCGTTTTCAAAAAAATTTCCACGCCCA
```

```
GTCTAACTATAGGCCGGTACGCCCTACATTGGAACATTTTTAAGGTTTTTTATTTTATTTTATTCCCCTT
ATCTATCGCCTCCCTAAAAATTATTAACTTCGCGTTCGCTTCCTGAGAACGGGGTTTGTTGGCGGGGGAC
TCACTTTAGAATCTCTCTGGTCTTTAGCTCACCCCCTTTACCTCCCAATTTTTTTAAAAAGAATATTTAT
TTTTAAAGGGAAAACCAATTTAAAAAATTTTTTTTTGGAATAAATTTCCGGGGCCCTTCTTGGGGGAATT
AAAAGGTTGGGAAGGTAAACCACCCCGGATTGTTGGGGGGGGGGTGGTTTTAACCACCGGGGGG
.
qlt:
0007:7<?FF?<;;;<FFKPPA=DJPPPPPPPPRDHPOOOOOPPPPPPPPPPPPPPPOTOPPPPPPP
PPPPPPPPPKB<<BGF@EPF@FPFBFPPPPPPPPPPPPPPE;;>=;;;;<@PPLFKPPPPPKF>BB>;>
AOPPPPPPPPPPPPK;<E<;;<F@FBEDEPPPPPPPPZZZJFKVOOOPPPPPPPPPKBEPPPPEBEPPPP
PPPPPPPPPPPPPPPPPPPPPPB@BPPPPPPPPPPPPPPPZZZZPPPPPPPPE?CPPPPPPPI@E?DPPP
PPPPPPPPEBEPPPPPPPPPPPPPPPPPPPPPPPPFPI@BIIPPPPPPPPPPPPPPPPPPPPPPPFDFI=BKK
PPPPPDPKIFDFPPPPPPPPPPPPPPPPPPPPPPQQPPPPPKTPPPTPPPPPPPPPPPPPPPPPPPPPPP
PPQPPPPPPPPPPMPPQFIPPQQPPPPPPPQPPPCA;@GPPPPPPPPPPPPPPPPPTPPPPZZPPPPPPPO
PEPPPPFBEPPPPPPPPPPPPPPPPPPPPPIPPPKDKPPQPPPPPPPPPPPPPPPPPOPPPNNTPPPGOCL
PPPPPPPKFKFKPPPPPPPPPPPDPPPPPD>;>P@BLPPLGJPFG=A>;;>>;AIPQOKPPLDEDPPEHIP
PPPPPPPPI>;;BB>==CPPQPPPPPPQPD<>;>AB<FDO<;<<;;;=>;BBBIDDF;;;;?;LBE;<;<
=RRPC=;;;;;;;?C?;;;?CF?;?;?F;;>DDPHDA;;;;AA?@=;;;<;<<;;;;H@;@;B=B;;>>:
;;;A;;;<<F;;?C??=?<;<>;;BPPP>;;;;;;;;;;<;;;A?<<><;;;;;<;;;;;;;;;;<;;<
;?;B<;;;;;;;;;;;;;;;;;;;;;;;;;;;;<;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;<;;;;;;;;;;;
.
clr:18,639
}
{FRG
act:A
acc:17000035336745
typ:T
loc:1
pos:47,550
src:
<This is an UBAC guide>
.
etm:941811186
seq:
TATACGAGACCGGCAGATCTGATATCATCGCCACTGTGCTCTAAAGAGCCTAATAAGAAGAACTCGGGAA
TAATGTATGGAAGCATCTAAGAATGTCTATATATCACCGTCAAGATATATATTTTTTGTAAAAATTGTCT
TTTGTAGTCAATTTACTCTGTAATTACAATTTTTTAAAATATTAAGATAAGAAATATAGTTAACTTGAAA
TGAATGTGAAATTCTAAAACATATCTTTTGCCGCCTAAAATCTTCTAAAATATTTTCGTGGTTTTGAACT
CTCTCTGTTGTTGGCCAGTGCAATCGACCTCGTTGGCAGCTTGTTGGCGACAGGAGCAGGAGGTGCGCCA
CACTCGGCTGCCCACATCATCTTTTCGGGGCCACAATGGGCGCACTTACGGGCAGCGGCTCCGTGTTCCG
GCCCAGGTCACAGTTCGTCCGGGTTCAGCTGGGCTCCTGATACCTGGAGGTGGCAGTAAGTTAAATTGCA
TTCGCCCCGTTGCCGATAACTTTCTAATGAGTGGTAATCGACGTTCGGCTTTATGTGTGTCCTGGCCTGG
TGGATACGGGGCGTATGTGGTATAAGGTTTGCCGCTGGTGTCAATGTGGATGGCAAAACTCGGAAAGGGT
CCTTTTTGTGGGAAACATAAACCTTTCCTTTTTGGGGGAAAAAAAAAAATTTTTAATTTGCCTTTGTGAA
GCGCCACAAATGGTCGCTGCAATGCCACCGA
.
qlt:
9?9999CAJJJLUSMMMMMRWXRLIIIIMMMMMMPc^^ZUSSSSXXXXWWSUSPPPSZZ_hhhhhhh_ZZ
XXXXZ^^cc^hhhh^[[ZWSJJSWZ^^^[hh^^c]X]XSSZTTXX]]hhZZZZ[[hhhbbbhhhbb\\\b
bbbbbbZUSUUXZYY]YYYYYY]YYYYY\\__bbbbbbhhhhZZZZ\ZZSSSSbhYYYVVVVVVVVXbb\
__bbZZZ^^b\\YZZ[[bb__ZbZbbbXYSMMLOTXZZ\_b\\\\\bb^[[ZZZbWYYSSTZ\XXUWUUQ
ZQ\UXXXXXSSSXXSSPPPSbbXUSSSSSSPSSSSSSSSUUUUXXUXUUUXZZ`\\\\UQQQSSSUUUUU
UUSXSSUPSJJJSTTTTZZZZh\ZZUUSTTXXXZZhhZUSSSSSVZQQLLLNLLMMMSSMMIIIIMIMIC
CGIKPTXXUQQMLNHBB=>>MMQQSMSRRSSSMMFFFOQLLLMKUMUMMDDDMMHE@<:@CDPPPXUUNN
KMFFGPPSQMMMMMSVUXVSRRPMMMMMQOKJJHHCFMEDCCBBGQQRRRQQUUKKHLIC>:>:=?FAA>
A>GEC@<<?@EED@B@>>HAC:==>::<=B@B<??A@;:999;B@9:<GDD<;;?DDHGEEEC==?999A
?=<:?9999888887::<888<<89;B<8898778:7799ACRQKE@<<77977@DM>A>>>C7766688
:>;98889C<=877:89>@=;9999:;7777
.
clr:47,550
}

{DST
act:A
acc:17000001001566
mea:2667.000000
std:341.000000
}

{LKG
act:A
typ:M
fgl:17000001594981
```

```
fg2:17000001585968
etm:925759632
dst:17000001001566
ori:I
}
```

```
{RPT
rpt:3
wch:simple repeats
len:0
}

{SCN
typ:U
acc:300001
rpt:3
rel:0
src:
(CA)n#Simple_repeat
.
seq:
CACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACA
CACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACACA
.
var:0.020000
min:40
}
```

B. Output Examples

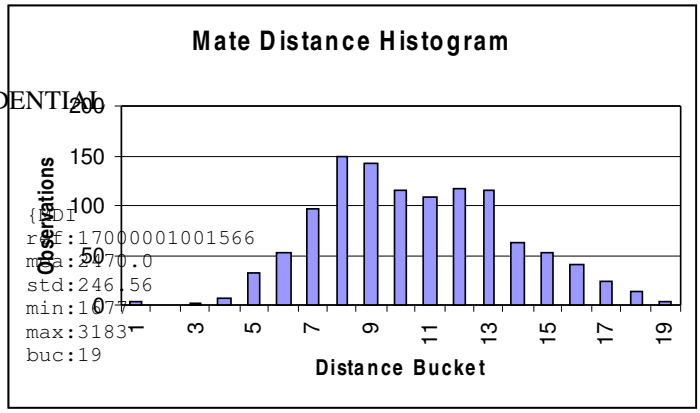
```
{AFG
acc:17000006003882
scn:
{SMA
whr:0,56
wht:600002
rpt:6
rel:0
pof:139,197
dir:R
}
{SMA
whr:0,43
wht:600027
rpt:6
rel:0
pof:438,481
dir:R
}
.
mst:U
chi:0
clr:23,625
}
{UTG
acc:6000006770087
src:
gen> uu [0,0]
.
cov:-0.495
sta:N
abp:0
bbp:0
len:706
cns:
GGCGATGGCGAAGCTAGATGTAGGTCGTGTGTACATCAAGTGGTTTCGACCACAAGGTCAGC-GAATGCA
GGCAAAAGGACAATGTTTGTAGCCAGTGCGGGCAACAAGGCGACACCGCGGCAAAGTGCCAAAACCCGGT
GAACTGCCGGAACTGCCGTCACAGAGGGCAACCTACGGGGCATTATATGCTCTCGAATGTATGCCCAATA
TACGGAGCGTTGCTAGCAAAGGTTGCAAGCTAGATTTTAATGTTTAGCTTCATCCAAGAGAACTGTGCCC
GAGGCCGAGCTGTGACCATCGAGCTCGGAGTCCGACTCAGGAGATCGGAGCTAATGTTTGCGCTGGTGCA
GGAACCGTACCTCGGCGGGGATGGAATGGATGTGCTGCCTGAAGGAATGAGGGTTTTCCCCGACCGGCGA
GGGAAGGCAGCCATCCTAGTGGATCATCAGGAAGCCATCTGTATACCAGTGGAAACCCTCATCACAGATA
ATGCCGTATGTCTGGTCGTTTAAGGGAGTTTTGGCTCAATCATTTTGCGCCGCATACTGCCATTCTGTTC
GACGCACCTCTGGAACCGTACCTCCGGTACATGGATGCGGTCCTTCTGCAGGCCAGCAGAACCCCCGCAA
TCCTGGGCCTCGACGCAAATGCAGTGTCCCTCAGTGGCTTAGCAAACTCTTTCGTCATGCCGAGGGGCAA
GCTAAA
.
qlt:
HHHHHCQOlllflllllllei111111111111111111111111111111111111111111jjce;lllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
llllllllllllllllllllllllllllllllllllllllllllkalllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllldjfbllllllllllll_lllllllllllllllllllllhebccllllll
lllllllllhjlllllllllllckjkjlllllllllfbfedelllllllklllllllg`hlgkllllll
llQQQQQQQQQQQQQQQMPRQQKQQQQLMOLQQQQQQQAELQHL>PQQQQSPRQQQQLLLKQQFCMQQFER
QQQQGE
.
for:0
nfr:2
{MPS
typ:R
mid:17000017604749
src:

lab>BE
.
```

```
pos:0,632
dln:1
del:
62
}
{MPS
typ:R
mid:17000018443345
src:

lab>BE
.
pos:8,706
dln:0
del:
}
}
```

```
{ULK
ut1:6000006408321
ut2:6000006408435
ori:O
ovt:T
ipc:0
gui:0
mea:-386.000
std:108.871
num:8
sta:B
jls:
17000010581032,17000011052168,M
17000004869118,17000004338777,M
17000004378897,17000004440217,M
17000006001388,17000006020913,M
17000004353835,17000004239070,M
17000007247406,17000007323381,M
17000010558554,17000010942531,M
}
```

```
{CCO
acc:6000007191259
len:709
cns:
TAAAAAAATATACTCTCTCTTAGCGTCTATCACCTTTTGAAGTCGTTTTGGTACAGACTATACCAAGCTG
TCGATATAATCTATTGAAATATCCTTCCATAAGGTTTGAATTTCCAAAATCGTTTCTTTGCGACTTCTAG
ATAAGTTTGCTAAGTTTGTGAGGTGGCTAATCCAATGTGTTTACGCAAACATCTTTCAGAAACTGCGTAA
TCAGTTTGGCCTTATGACAGGGAGCATTATCCTGCTGGAGTATGAAGGACTCTCCAATCAATTTATCGCC
AGAGGGGAATGCATGATTATTAAGGACATCTAAATATTTTCCTTGATTCATGGTTCCATCAACAGGAACC
AAATCTCCCAAGCCAGTGAAGGCTACGCATATCCAAAACATAACAGACCCTCCTGAATGATTTACTCTCT
GACATACTGGTGCCCCTTTTTTCCGATATTGTTTAGGTAAAAGAGCTTTCATC-GCTCCACAAAACATTG
TTCCAGAATAATACAGGCTGACCAACATATTTTTTTCCAAAATCGGCATCCTTAAGACGACTGGGAACCG
TTCTTTACTGATATCGATTTCTGCTCCTTCTTTTAGTTTTTTGGCAACCGATCGGGGAGTTTGTAGAACA
TTCTGCTTAAACTTCCGTAAAATATAACCATCGTCCTTTTGAGTTGTTTTACGTGGTTGGCCACTACGCG
CAACGTTTT
.
qlt:
GQQRHPNEFQQNNQQQQQQM<IQQQQQQQQKQELCRPC=??>@QQQQQMLLLLDQQQQQQQQQBEHEFQQl
lllllllllclllllifldfllch`hdlllllllllllglllllllllllllllllllllllllllllllll
lllllllllllllllllllllklllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
llllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllllll
lllgbW<^^^clllkllllllllllllllllllhjllllllllllllllllllllllaVZQQQQQQQQQQ
F>LQQQQRI
.
for:0
npc:3
nou:1
{MPS
typ:R
mid:17000009478835
src:
.
pos:617,0
dln:1
del:
596
}
{MPS
typ:R
mid:17000003389122
src:
.
pos:690,69
dln:1
del:
404
}
{MPS
typ:R
mid:17000003557317
src:
.
pos:117,709
dln:1
del:
356
}
{EPS
typ:U
lid:6000006408686
pos:0,709
}
}
```

```
{CLK
co1:6000007191211
co2:6000007191213
ori:N
ovt:N
ipc:0
gui:0
mea:-5049.000
std:146.312
num:4
sta:B
jls:
17000005169778,17000005573432,M
17000005113856,17000005257824,M
17000004862206,17000004804742,M
17000005301085,17000005302243,M
}

{SCF
acc:6000007958935
noc:3
{CTP
ct1:6000007191211
ct2:6000007191212
mea:693.091
std:310.470
ori:N
}
{CTP
ct1:6000007191212
ct2:6000007191213
mea:814.491
std:251.357
ori:N
}
{CTP
ct1:6000007191213
ct2:6000007191214
mea:389.617
std:308.193
ori:N
}
}
```




```
{MDI
rff:17000001001566
mea:2470.0
std:246.56
min:1677
max:3183
buc:19
his:
3
0
1
7
33
53
97
150
143
116
109
117
115
62
52
41
24
14
3
}
```