DOCUMENTATION FOR PHRAP AND CROSS_MATCH (VERSION 0.990319)

phrap ("phragment assembly program", or "phil's revised assembly
program"; a homonym of "frappe" = French for "swat") -- a
program for assembling shotgun DNA sequence data. Key features:
allows use of entire read (not just trimmed high quality part); uses a
combination of user-supplied and internally computed data quality
information to improve accuracy of assembly in the presence of
repeats; constructs contig sequence as a mosaic of the highest quality
parts of reads (rather than a consensus); provides extensive information
about assembly (including quality values for contig sequence) to
assist trouble-shooting; able to handle very large datasets.
 N.B. phrap does not provide editing or viewing capabilities; these
are available with consed and phrapview. It is strongly recommended
that phrap be used in conjunction with the base calls and base quality
values produced by the basecaller, phred; and with the sequence
editor/assembly viewer, consed.

cross_match -- a general-purpose utility (based on a "banded" version
of SWAT, an efficient implementation of the Smith-Waterman algorithm)
for comparing any two sets of (long or short) DNA sequences. For
example, it can be used to compare a set of reads to a set of vector
sequences and produce vector-masked versions of the reads; a set of
cDNA sequences to a set of cosmids; contig sequences found by two
alternative assembly procedures (e.g. phrap and xbap) to each other;
or phrap contigs to the final edited cosmid sequence. It is slower but
more sensitive (because it allows gaps) than BLASTN.

CONTENTS

I. INSTALLATION

I. INSTALLATION

The source code for the swat/cross_match/phrap package will be sent to
you in the form an email message containing a uuencoded .tar.Z file;
you will need to have access to a Unix system for the initial
unpacking, but once you've uudecoded it and unpacked the .tar file
(steps i and ii below), you should be able to compile the programs
on computers running other operating systems -- they should be portable
to almost anything with a decent C compiler and adequate memory (64 Mb
RAM or more is desirable). Here are the steps needed to unpack and
install the programs:

i. Save the email message as a file (for example, "temp.mail"). If
possible, do this using the Unix mail command, rather than another
mail program -- some mail programs (e.g. Pine) remove trailing spaces

on each line of incoming messages, which will corrupt a uuencoded
message.
 Do not attempt to modify the saved mail message in any way. That is
unnecessary and may corrupt the message.

ii. To unpack the saved email message, execute the following two
commands on a Unix workstation, in the directory containing the file
created in step i above:

> uudecode temp.mail

> zcat distrib.tar.Z | tar xvf -

If either of these commands results in an error message, it is likely that
the email message was corrupted by your mail program (see step i above).

iii. To produce working versions of the programs, move (if necessary)
all of the files produced by the above command to the computer on
which you wish to run the programs (which must have a C compiler!),
and execute the following command in the directory that contains the
files:

> make

If your compiler does not recognize the -O2 optimization flag (which
should be evident from warning messages it produces), you should change
the line

        CFLAGS= -O2

in the file "makefile" to

        CFLAGS= -O

Then remove all files ending in .o produced by the original make, and
recompile.

iv. If you have datasets with more than 64,000 reads, or that include
sequences longer than 64,000 bp, you will need the .manyreads and/or
.longreads versions of phrap and cross_match. These are created using the
command

> make manyreads

v. If you are operating a non-commercial (academic or government)
computer facility which provides access to several independent
investigators, you are required by the licensing agreement to set the
permissions on the executables and source code to allow execute but
not read access, so that the programs may not be copied.

II. RUNNING PHRAP & CROSS_MATCH

1. Phrap.

 i. PhredPhrap script. Generally if you are assembling reads
basecalled by phred you should run phrap via the phredPhrap script
rather than directly. PhredPhrap runs phred; runs another script,
phd2fasta, to make the fasta format sequence file from the .phd files

generated by phred; runs cross_match to mask vector sequence; runs
phrap; and finally runs some other programs that provide useful
information to consed (e.g. tagging of repeats). It will accept any
of the phrap command line options described below (section IV).
PhredPhrap and its documentation are included in the consed
distribution.

   Although phredPhrap automates most of the steps in the assembly, you
will still need to do several things: (a) set up the directory
structure that is assumed by phredPhrap and make sure your
chromatogram files are appropriately located in this structure; (b)
adhere to the read naming convention assumed by phrap, or make other
arrangements to get template/chemistry/read direction information into
the .phd or fasta file (THIS IS IMPORTANT -- SEE section III.1 below);
and (c) make sure that appropriate clone and subclone (sequencing)
vector sequences are included in a fasta file used for screening with
cross_match (see section III.4 below).

 ii. The following instructions pertain to "standalone" operation of
phrap.   Before the run, you may want to increase the amount of memory
available to phrap: see the instructions for using the Unix
"limit" command under IX.1, "Insufficient memory" below.
The command line should be of the form

> phrap seq_file1 [seq_file2 ...] [-option value] [-option value] ...

where each seq_file is the name of a sequence file in FASTA format as
described below in III.2. Available options are described in section IV
below. The standard output is extensive and should be redirected to a
file (phrap.out in the following example).

Example:

> phrap C05D11.reads.screen -minmatch 20 -new_ace > phrap.out

Note that the quality file is not specified on the command line, but
instead must be named appropriately (as described below in III.3) to be
recognized by phrap; in this example its name would need to be
C05D11.reads.screen.qual. If more than one seq_file is provided,
then reads in the first file are assembled only against sequences in
the 2d (and later) files; no sequences are compared to other sequences
in the same file. [N.B. THE ABILITY TO USE MULTIPLE SEQUENCE FILES IS
NOT AVAILABLE AT PRESENT -- YOU SHOULD PROVIDE ONLY A SINGLE SEQUENCE
FILE TO PHRAP]. In this case, many features of phrap (e.g. checking
for anomalies, vector, etc.) which rely on comprehensive read-read
comparisons are turned off, SWAT scores are used in place of
LLR_scores, and implied merges are not performed -- i.e. each read in
the first file is merged with at most one sequence in the 2d
file. This usage of phrap is convenient for aligning a set of reads
against a known "finished" sequence in order to look for
polymorphisms, for example.

2. Cross_match.

The command line should be of the form

> cross_match seq_file1 [seq_file2 ...] [-option value] [-option value] ...

where each seq_file is the name of a sequence file in FASTA format as
described below in III.2. Available options are described in section

IV below. The standard output should generally be redirected to a
file. If there are at least two files, all sequences in the first
file are compared to those in the second (and any subsequent) files;
if there is only one file, all sequences in this file are compared to
each other (N.B. at present entries are not compared to their own
complements, but are compared to themselves in the same orientation,
and to all other entries in both orientations). Matches meeting
relevant criteria (see section IV) are written to the standard output.

III. INPUT FILES

Input files to cross_match and phrap are of two types: sequence files
(required), and quality files (optional, but strongly recommended for
phrap). If you are doing phrap assemblies of reads generated by the
basecalling program phred, then it is recommended that you generate
these input files automatically using the phd2fasta or phredPhrap
scripts distributed with phred and consed (see II.1.i above).
Even in this case however you should read the following section on the
read naming convention, since you will either need to make sure that
the read names attached to your chromatogram files adhere to the
naming convention, or you will need to modify the phredPhrap script
(or create your own script) so that correct template names and read
orientation information is appended to the .phd files after they are
produced by phred.

1. Read naming convention (phrap only)

 In addition to the sequence and quality data, phrap needs to know
three things for each read:
 (i) the name of the subclone (or other template) from which the read
is derived; this is used, for example, in checking for chimeric
subclones (for which purpose it is necessary to know when two reads
are from the same subclone so that they are not regarded as
independently confirming each other) and for certain other data
anomalies;
 (ii) the orientation of the read (forward or reverse) within the
subclone, in cases where data is acquired from each end of a subclone
insert (at present this information is used only for consistency
checking following assembly, and not in the assembly itself, but this
will change in future versions);
 (iii) the chemistry used to generate the read (which influences
phrap's decisions as to how to treat discrepancies between potentially
overlapping reads, and as to how to adjust qualities of reads which
confirm each other -- confirmation of a read by another read with
different chemistry counts as significantly as confirmation by an
opposite-strand read).

The above information is generally conveyed via the read names using
the "St. Louis" read naming convention described below. If your
laboratory uses a different naming system, you have several options:

 (i) provide the relevant information in the description field of the
.fasta file instead (see below), or (if you are using the phd2Fasta
script) as tags in the .phd files for each read;
 (ii) see whether appropriate values of the command line options
-subclone_delim and -n_delim (described below) can be used to bring
your own naming convention into conformity with the St. Louis
convention -- if your naming convention is such that the subclone name

always occurs as the first part of the read name, and the end of the
subclone name occurs at the first (or n-th for some fixed value of n)
occurrence of a particular character, then you may be able to use
these options without having to change your read names;
 (iii) create a script which translates your read names into St. Louis
form (this does not mean that you cannot retain the original names as
well -- for example, for purposes of running phrap you could create a
directory of "links" to your chromatogram files, such that the link
names adhere to the naming convention, and then run the phredPhrap
script on this directory);
 (iv) alter the subroutines in the source module "names.c" in order to
parse your read names correctly; or
 (v) ignore the issue and hope that it does not cause problems. This
may have unpredictable adverse effects on assembly and is NOT
recommended.

 The "St. Louis" read naming convention assumed by phrap is as follows:

  The portion of the read name up to the first '.', if any, is
the name of the subclone from which the read is derived (i.e. the DNA
template used in the sequencing reaction). If desired, the command line
option -subclone_delim can be used to change the character which
delimits the end of the subclone name to something other than '.',
and/or the option -n_delim may be used to indicate that the subclone
name contains a fixed number of occurrences of this character within
it.   See section IV below for a description of command line options.
  The orientation of the read within the subclone, and the chemistry,
are indicated by the first letter following the '.', as follows:

 "s" forward direction read on single stranded (SS) template, dye primer chemistry
 "f" forward read on double stranded (DS) template, dye primer chemistry
 "r" DS reverse read, dye primer chemistry
 "x" SS forward read, standard dye terminator chemistry
 "z" DS forward read, standard dye terminator chemistry
 "y" DS reverse read, standard dye terminator chemistry
 "i" SS forward read, big dye terminator chemistry
 "b" DS forward read, big dye terminator chemistry
 "g" DS reverse read, big dye terminator chemistry
 "t" for T7 (cDNAs)
 "p" for SP6 (cDNAs)
 "e" for T3 (cDNAs)
 "d" for special
 "c" consensus pieces
 "a" assembly pieces

 The remainder of the name is ignored. If the read name ends in ".seq"
the .seq is removed.

 Example: the name BAC112_a11b3.x3_pg indicates a forward read with
(old) dye terminator chemistry from the subclone BAC112_a11b3.

2. Sequence files.

 With both phrap and cross_match, either a single sequence file or
multiple sequence files can be specified (N.B. WITH CURRENT VERSION OF
PHRAP ONLY A SINGLE SEQUENCE FILE SHOULD BE SPECIFIED). If a single
file is specified, all sequences in it are compared to each other; if
more than one file is specified, sequences in the first file are
compared to sequences in the second (and subsequent) files. Typically

phrap is used with a single input sequence file containing the reads,
unless one wants to assemble one set of reads "against" another
sequence or set of sequences (see below). Cross_match is typically
used with two input sequence files, the first containing the "query"
sequences and the second containing the "subject" sequences, but it
may also be used to compare the sequences in a single input file to
each other.

 Phrap is designed to be able to use all of each read sequence in the
assembly, not just the trimmed (highest quality) part, so the full
sequence of each read should be provided when available. If
phred-generated qualities are not available however then the default
quality parameter -default_qual should be set appropriately (section
IV).

Sequence files must be in FASTA format:
   (i) Each sequence entry has a single header line as follows: The
first character is '>'. This is followed immediately (i.e. with no
intervening spaces or other characters) by the sequence name, and then
(optionally) by a space or spaces, followed by descriptive information
about the sequence.
   (ii) The header line is followed by a separate line or lines
containing the sequence.

The sequence names, descriptive information, and sequences can be of
arbitrary length (up to a combined total size for each type of
about 2 billion characters), since space for them is allocated
dynamically. If there are more than 64,000 entries in the first
(query) file, or if any of the sequences are longer than about 64,000
characters, the .manyreads version of phrap or cross_match must be
used.

Descriptive information on the header line may optionally be used to
indicate template (subclone), read orientation, read chemistry and
dye. If present, this information overrides whatever would be implied
by the read name. The template name is indicated by including the word
"TEMPLATE:", followed by a space, followed by the name of the
template. Orientation is indicated by the word "DIRECTION:", followed
by the word "fwd" or "rev". Chemistry is indicated by the word
"CHEM:", followed by the word "prim" (for dye primer), "term" for dye
terminator, or "unknown". Dye is indicated by the word "DYE:" followed
by "rhod", "big", "ET", "d-rhod", or "unknown". The chemistry and dye
information is provided automatically by newer versions of phred
(version 0.980904.a or later), which obtain it from the chromatogram
file. (Use the script phd2fasta distributed with phred and consed to
create the .fasta file from the .phd files created by phred).

Example:

>read#5 DIRECTION: rev CHEM: prim DYE: ET TEMPLATE: a23f1 any_other_information
GAAAGATCTCATTGATCACTCTATTCAAGTGGGAGTCTCCGGTCTTT
ATGATCGATTTGTGAATCTTCGTATCAAAGTTGGAGCTGACAAGTATCCA
TTGCTTGCGAAATGGGCTCAAATTTTCACTCAGGGAGTCGTCTTCGATCC
TTCAAGAATTCATCAATGTGCTCAAAAGTTGGCTGGAGAAGCTCGTGATC
GGAAGAGAGATGGATGTACTGTGGCATCAACTGCAGTAGCTTCAATGGTT
TATGGAAAGAGTATGTTATTt

All descriptive information in the header line is written back out to
the .ace and .singlets files.

Non-alphabetic characters (including '*', and digits) in the sequence
lines are automatically stripped out when the file is read in, except
for '-' which is converted to 'N'; '>' must not appear anywhere within
the sequence since it is assumed to start the header of a new sequence
(even if it is not at the beginning of a line). Lower case letters are
converted to upper case on readin, so it is not possible (as it was in
old phrap versions) to use case as a crude indication of quality.

3. Quality files.

   For each input sequence file in FASTA format, you may optionally
include a corresponding file of data quality information. This is
strongly recommended for phrap runs since it greatly improves the
accuracy of assembly and of the consensus sequence; with cross_match,
the qualities are at present used only for purposes of annotating and
tabulating discrepancies and not in the alignment scoring. The name of
this file must consist of the name of the FASTA file, with ".qual"
appended.
 The format of the .qual file is similar to that of the corresponding
FASTA file. For each read there should appear a header line identical
(except possibly for the "description" field) to that in the FASTA
file. This is followed by one or more lines giving the qualities for
each base. Quality values should be integers between 0 and 99
(inclusive), and should be separated by spaces. No other (non-digit,
non-white space) characters should appear. The total number of quality
values for each read must match the number of bases for that read in
the FASTA file. Also, the orders of the reads in the FASTA and
corresponding .qual file must be identical. N's are automatically
assigned quality 0, overriding any value present in the .qual file;
however X's retain whatever value is present in the .qual file.
   If provided, quality information is used by phrap both in the
assembly itself (where discrepancies between high quality bases are
used to discriminate repeats from true overlaps) and in determining
the contig consensus sequence (which is formed as a mosaic of the
highest quality parts of the reads). Because phrap generates its own
quality measures (based on read-read confirmation) it performs
reasonably well even if no input quality is provided; however when it
is available, it is important to provide input data quality
information since this can substantially increase the accuracy of the
consensus, particularly in regions where there are strand-specific
effects on quality (e.g. compressions) -- in such cases the input
quality information may constitute the only basis for choosing one
strand over the other.
   If your sequencing data is from automated sequencing machines it is
strongly recommended that you generate the read sequences using the
basecalling program phred, which will produce an appropriate quality
file for input to phrap. Phred generally gives more accurate base
calls overall, for a longer distance, than the ABI software does. On
the basis of the trace characteristics, phred computes a probability p
of an error in the base call at each position, and converts this to a
quality value q using the transformation $q = -10 \log_{10}(p)$. Thus a
quality of 30 corresponds to an error probability of $1 / 1000$, a
quality of of 40 to an error probability of $1 / 10000$, etc.
   The quality value 99 is reserved for base calls that have been
visually inspected and verified as "highly accurate" (during editing),
and 98 for bases that have been edited but are not highly accurate
(these are converted to quality 0 in phrap). If two reads appear to

match but have discrepancies involving bases that have quality 99 in
both reads, phrap does not allow them to be merged during assembly.
At present this is the only way to break false joins made by phrap.
   If you wish to generate your own quality values you should be
prepared to experiment. It is particularly important that possible
insertion errors (which are fairly frequent late in the trace, with ABI
basecalls) receive low quality, because in choosing the "consensus"
phrap tends to favor the reads with the highest total quality in a
given region. Also, if at all possible you should use quality values
that are defined in terms of error probabilities in the manner
described above, since to some extent phrap is tuned to expect these
(and its output qualities will then have a similar interpretation).
 If all input quality values are relatively small (less than 15),
phrap assumes that they do not correspond to error probabilities and
attempts to rescale them so that the largest quality value is
approximately 30. This allows different input scales to be used.

4. Vector screening

 Following creation of a FASTA file with the raw read sequences, it is
important to remove or screen out sequencing (subclone) vector
sequence before running phrap. (It is unnecessary to remove the
cloning (e.g. cosmid or BAC) vector, unless the inserts from multiple
overlapping clones are being assembled simultaneously; however it is
generally useful to remove at least the central part of the cloning
vector, since this then provides a natural point at which phrap can
break the circular sequence into a linear one). Unremoved sequencing
vector may cause reads to be identified as "possible chimeras", or
otherwise interfere with proper assembly. Some vector removal
programs may be unreliable at identifying vector sequences that are
rearranged or found in the lower quality part of the read, and I
recommend that you screen for sequencing vector using cross_match as
described here, whether or not you have already screened them using
another program.

To carry out the screening, first create a FASTA file, say "vector",
with all of the vector sequences you want to screen for (I use one
that contains all vector sequences used in any of the ongoing
sequencing projects, in case the vector is misidentified in the
current project). If your read file is named "C05D11.reads" (for
example), then run the command

> cross_match C05D11.reads vector -minmatch 10 -minscore 20 -screen > screen.out

(This uses somewhat more sensitive parameter settings than the
cross_match defaults). The '-screen' option causes a file named
"C05D11.reads.screen" to be created, containing "vector masked"
versions of the original sequences: i.e. any region that matches any
part of a vector sequence is replaced by X's. This ".screen" file is
what should be provided as input to phrap. The output from
cross_match (which has been redirected to the file screen.out in the
above example) lists the matches that were found (see below).

N.B. If you have created a .qual file for your reads (say
C05D11.reads.qual), be sure to rename or copy it to
C05D11.reads.screen.qual so that it will be recognized by phrap when
C05D11.reads.screen is used as the input FASTA file.

The phredPhrap script automatically carries out the above steps

(i.e. vector screening using cross_match, and renaming of the .qual
file); however it is your responsibility to make sure the vector
sequence file includes vector sequences appropriate for your cloning
and sequencing vectors.

IV. COMMAND LINE OPTIONS

 This section describes all command line options available with
cross_match and phrap, grouped by category. Each option name is
followed immediately by the "default value" assumed by the program,
and a brief description. A '*' appearing as the default value
indicates that the option is a flag rather than a parameter; in this
case no value should be included on the command line after the option
name, and by default it is turned off. [None] means there is no
default, i.e. the parameter is not relevant when the option is
omitted.

 Unless otherwise indicated, an option can be used with either
cross_match or phrap.

1. Scoring of pairwise alignments

 The following options control the scoring of pairwise sequence
alignments. By default, matching residues receive a reward of +1,
mismatches get a penalty of -2, gap opening residues a penalty of -4
(= -2 - 2), and gap extension residues a penalty of -3 (= -2 - 1). The
highest scoring word-nucleated local alignment between a pair of
sequences is found, and its score is then complexity-adjusted
(i.e. scores of matches between sequence regions of biassed nucleotide
composition are adjusted downwards). These parameters are chosen such
that regions of two sequences that are about 70% or more identical
will tend to have a positive alignment score. For more stringent
comparisons, -penalty should be made more negative (e.g. a value of -9
will tend to find alignments that are 90% identical), and/or -minscore
should be increased; for more sensitive comparisons, a score matrix
should be used instead.

option name & default value

 -penalty -2
Mismatch (substitution) penalty for SWAT comparisons.

 -gap_init penalty-2
Gap initiation penalty for SWAT comparisons.

 -gap_ext penalty-1
Gap extension penalty for SWAT comparisons.

 -ins_gap_ext gap_ext
 Insertion gap extension penalty for SWAT comparisons (insertion in
subject relative to query).

 -del_gap_ext gap_ext
 Deletion gap extension penalty for SWAT comparisons (deletion in
subject relative to query).

 -matrix [None]
 Score matrix for SWAT comparisons (if present, supersedes -penalty)

Matrix format: (TO BE ADDED)

 -raw *
 Use raw rather than complexity-adjusted Smith-Waterman scores.

2. Banded search

 The following options control the region of the Smith-Waterman matrix
that is searched. In phrap and cross_match (as opposed to swat, which
performs full Smith-Waterman comparisons), word-nucleated banded
Smith-Waterman comparisons are performed: the set of input sequences
is scanned to find pairs of perfectly matching subsequences ("word
matches") meeting certain criteria, and for each such word match a
band in the Smith-Waterman matrix that is centered on the diagonal
defined by the match is searched to find an optimal-scoring alignment.
If there are multiple matching words for a given pair of sequences,
the union of the corresponding bands is searched. The search is
"recursive" in the sense that if an optimal-scoring alignment is found
whose score exceeds -minscore the remainder of the band is searched
again; as a result multiple alignments may be found within a single
band.
   The word matches that are considered are "maximal" in the sense that
they cannot be extended in either direction without introducing a
discrepancy in one of the two sequences. If the length of the word
match is at least -maxmatch, it is accepted; otherwise if the
complexity-adjusted length is less than -minmatch, or if the matching
sequence occurs more than -max_group_size times on the forward strands
of the query file, it is rejected. The latter criterion effectively
downweights frequently occurring words. It can be turned off by
setting -maxmatch equal to -minmatch.

 -minmatch 14
 Minimum length of matching word to nucleate SWAT comparison. if
minmatch = 0, a full (non-banded) comparison is done [N.B. NOT
PERMITTED IN CURRENT VERSION]. Increasing -minmatch can dramatically
decrease the time required for the pairwise sequence comparisons; in
phrap, it also tends to have the effect of increasing assembly
stringency. However it may cause some significant matches to be
missed, and it may increase the risk of incorrect joins in phrap in
certain situations (by causing implied overlaps between reads with
high-quality discrepancies to be missed).

 -maxmatch 30
 Maximum length of matching word. For cross_match, the default value
is equal to minmatch, instead of 30.

 -max_group_size 20
 Group size (query file, forward strand words)

 -word_raw *
 Use raw rather than complexity-adjusted word length, in testing
against minmatch (N.B. maxmatch always refer to raw lengths). (The
default is to adjust word length to reflect complexity of matching
sequence).

 -bandwidth 14
 1/2 band width for banded SWAT searches (full width is 2 times
bandwidth + 1). Decreasing bandwidth also decreases running time at

the expense of sensitivity. Phrap assemblies of clones containing long
tandem repeats of a short repeat unit (< 30 bp) may be more accurately
assembled by decreasing -bandwidth; -bandwidth should be set such that
2 bandwidth + 1 is less than the length of a repeat unit. -bandwidth 0
can be used to find gap-free alignments.

3. Filtering of matches

 The following options control which matches are displayed (cross_match)
or used in assembly (phrap).

 -minscore 30
 Minimum alignment score.

 -vector_bound 80
 Number of potential vector bases at beginning of each read. Matches
that lie entirely within this region are assumed to represent vector
matches and are ignored. For cross_match, the default value is 0
instead of 80.

 -masklevel 80
 (cross_match only). A match is reported only if at least (100 -
masklevel)% of the bases in its "domain" (the part of the query that
is aligned) are not contained within the domain of any higher-scoring
match.
 Special cases:
    -masklevel 0   report only the single highest scoring match for each query
    -masklevel 100   report any match whose domain is not completely contained
                          within a higher scoring match
    -masklevel 101   report all matches

4. Input data interpretation

 The following options influence how input data (including read names)
are interpreted.

 -default_qual 15
 Quality value to be used for each base, when no input .qual file is
provided. Note that a quality value of 15 corresponds to an error rate
of approximately 1 in 30 bases, i.e. relatively accurate sequence. If
you are using sequence that is substantially less accurate than this
and do not have phred-generated quality values you should be sure to
decrease the value of this parameter.

 -subclone_delim .
 (phrap only). Subclone name delimiter: Character used to indicate end
of that part of the read name that corresponds to the subclone name

 -n_delim 1
 (phrap only). Indicates which occurrence of the subclone delimiter
character denotes the end of the subclone name (so for example
    -subclone_delim _ -n_delim 2
means that the end of the subclone name occurs at the
second occurrence of the character '_'). Must be the same for all
reads!

 -group_delim _

(phrap only). Group name delimiter: Character used to indicate end of
that part of the read name that corresponds to the group name
(relevant only if option -preassemble is used); this character must
occur before the subclone delimiter (else it has no effect, and the
read is not assigned to a group).

 -trim_start 0
 (phrap only). No. of bases to be removed at beginning of each read.

5. Assembly

 The following options are used to control completeness and stringency
of assembly; note that the options -minmatch, -bandwidth, -penalty,
and -minscore discussed above also affect assembly stringency.

 -forcelevel 0
 (phrap only). Relaxes stringency to varying degree during final
contig merge pass. Allowed values are integers from 0 (most
stringent) to 10 (least stringent), inclusive.

 -bypasslevel 1
 (phrap only). Controls treatment of inconsistent reads in merge.
Currently allowed values are 0 (no bypasses allowed; most stringent)
and 1 (a single conflicting read may be bypassed).

 -maxgap 30
 (phrap only). Maximum permitted size of an unmatched region in
merging contigs, during first (most stringent) merging pass.

 -repeat_stringency .95
 (phrap only). Controls stringency of match required for joins. Must
be less than 1 (highest stringency), and greater than 0 (lowest
stringency).

 -revise_greedy *
 (phrap only). Splits initial greedy assembly into pieces at "weak
joins", and then tries to reattach them to give higher overall score.
Use of this option should correct some types of missassembly.

 -shatter_greedy *
 (phrap only). Breaks assembly at weak joins (as with -revise_greedy)
but does not try to reattach pieces.

 -preassemble *
 (phrap only). Preassemble reads within groups, prior to merging with
other groups. This is useful for example when the input data set
consists of reads from two distinct but overlapping clones, and it is
desired to assemble the reads from each clone separately before
merging in order to reduce the risk of incorrect joins due to
repeats. The preassemble merging pass is relatively stringent and not
guaranteed to merge all of the reads from a group.
 Groups are indicated by the first part of the read name, up to the
character specified by -group_delim.

 -force_high *
 (phrap only). Causes edited high-quality discrepancies to be ignored
during final contig merge pass. This option may be useful when it is
suspected that incorrect edits are causing a misassembly.

6. Consensus sequence construction

 The following options affect the weighted directed graph that is used
to find the consensus sequence. Increasing their values will reduce
the size of the graph, which should reduce memory requirements for the
phrap run (substantially in some cases) but may decrease the accuracy
of the consensus sequence that is found.

 -node_seg 8
 (phrap only). Minimum segment size (for purposes of traversing
weighted directed graph).

 -node_space 4
 (phrap only). Spacing between nodes (in weighted directed graph) .

7. Output

 The following options control the creation or formatting of certain
output files, and of the standard output.

 -tags *
 Tag selected lines in the standard output, to facilitate
parsing. (N.B. this does NOT refer to the phrap-generated tags that
are included in the .ace file and viewable in consed.)

 -screen *
 (cross_match): Create a ".screen" file. This is a FASTA format
sequence file, in which any sequence region in a sequence in the first
input sequence file that matches some region in a sequence in the
second (or later) file is replaced by X's. This option is primarily
used to create "vector-masked" copies of the reads prior to phrap
assembly.
 (phrap): when the -old_ace or -new_ace option is specified (see
below), this option causes parts of the read sequences that consist of
phrap-inferred sequencing vector and chimeric segments to be replaced
by X's in the .ace file. (The "phrap-inferred sequencing vector"
consists of the beginning part of the read that either does not match
any other read, or matches only the beginning parts of other reads;
"phrap- inferred chimeric segments" consist of segments of the read
that match other reads but do not match the consensus at that
location.)

 -old_ace *
 (phrap only). Create ".ace" file in old style format.

 -new_ace *
 (phrap only). Create ".ace" file in a new style format (STRONGLY
recommended over old_ace !!)

 -ace *
 (phrap only). Same as -new_ace.

 -view *
 (phrap only). Create ".view" file suitable for input to phrapview.

 -qual_show 20
 (phrap only). Cutoff for flagging "low_quality" regions in contig

sequence and "high quality" discrepancies between read and
contig. Bases in the .contigs and .ace file are lower case if and only
if their LLR-converted quality values are below this value.

 -print_extraneous_matches *
 (phrap only). Print information about non-local matches between
contigs.

 -exp [None]
 (gcphrap only). Name of a directory in which output experiment files
are to be placed.

 -alignments *
 (cross_match only). Display the alignment for each match.

 -discrep_lists *
 (cross_match only). Give list of discrepancies for each match.

 -discrep_tables *
 (cross_match only). Give table of discrepancies (by quality) for each
match (default is to display a single table, that combines results for
all matches).

8. Miscellaneous

 -retain_duplicates *
 (phrap only). Retain exact duplicate reads, rather than eliminating
them.

 -max_subclone_size 5000
 (phrap only). Maximum subclone size -- for forward-reverse read pair
consistency checks.

 -trim_penalty -2
 (phrap only). Penalty used for identifying degenerate sequence at
beginning & end of read.

 -trim_score 20
 (phrap only). Minimum score for identifying degenerate sequence at
beginning & end of read.

 -trim_qual 13
 (phrap only). Quality value used in to define the "high-quality" part
of a read, (the part which should overlap; this is used to adjust
qualities at ends of reads.

 -confirm_length 8
 (phrap only). Minimum size of confirming segment (segment starts at
3d distinct nuc following discrepancy).

 -confirm_trim 1
 (phrap only). Amount by which confirming segments are trimmed at
edges.

 -confirm_penalty -5
 (phrap only). Penalty used in aligning against "confirming" reads.

 -confirm_score 30
 (phrap only). Minimum alignment score for a read to be allowed to

"confirm" part of another read.

 -indexwordsize 10
 Size of indexing (hashing) words, used in finding word matches
between sequences. The value of this parameter has a generally minor
effect on run time and memory usage.

V. VIEWING PHRAP ASSEMBLIES WITH OTHER PROGRAMS

1. Consed

 Consed is the recommended sequence editor for use with phrap. Its
development has been closely tied to the development of phrap and it
has been designed to take advantage of the information regarding the
assembly that is generated by phrap (e.g. phrap's consensus sequence
and quality values, and tags indicating potential assembly problems
and various data anomalies such as chimeras and compressions).
 To use consed with phrap, ace files must be generated during the phrap
run using the flag -new_ace or -old_ace on the command line. This is done
automatically if you use the script phredPhrap to carry out all of the
data processing steps.
 For additional information, consult consed documentation.

2. Phrapview

  Phrapview is a graphical tool that provides a "global" view of the
phrap assembly, complementary to the "local" view provided by the
sequence editor CONSED; it will become obsolete when a similar globabl
view is added to CONSED.

a. Installation
 Phrapview is written in perl/Tk; to run it you will need to have
installed on your system a recent version of perl (perl5 or later),
together with the perl/Tk perl module written by Nick Ing-Simmons. The
perl/Tk scripts can be obtained from any CPAN site in
modules/by-authors/Nick_Ing-Simmons/, for example:

ftp://ftp.perl.org/pub/CPAN/modules/by-authors/Nick_Ing-Simmons/

The latest version of perl/Tk is the most recent Tk800.xxx.tar.gz file
(where xxx is the most recent version number). The latest version of
perl can also be obtained from any CPAN site in /src, for example

ftp://ftp.perl.org/pub/CPAN/src/perl5.005_02.tar.gz

(Perl 5.005_02 was the most recent version when this was written.)

If you have problems with running phrapview, please make sure that you
have both a recent version of perl installed and a recent version of
perl/Tk. If you are unsure whether your version is recent enough, you
or your system administrator should download the latest sources and
install them.

b. Running phrapview

Phrapview is invoked with the command

  phrapview filename

where filename is the name of a ".view" file that was created by
running phrap with the -view option.

N.B. The format of the .view file is still under development. To
ensure that phrapview will read correctly the .view file produced by
phrap, make sure that both programs (phrap and phrapview) were part of
the same release.

The following information can be displayed:

  (i) Basic information concerning the numbers of reads, singletons,
contigs, chimeras, etc. Each contig, and each chimeric read, is
represented by a horizontal black line proportional in length to the
contig or read size. Note that although they are shown separately in
this display, chimeric reads in fact generally are incorporated by
phrap into one of the contigs, in a region that corresponds to one of
the two chimeric pieces; the location of this is indicated by the
black "chimera match" line connecting the chimera to a contig (see
below).

The following information requires pressing an appropriate "button" to
display:

  (ii) Depth of coverage. Graphs (in yellow) above and below the
contig line indicate the depth of coverage (i.e. number of reads
aligned against the contig) on each strand at each position; the
horizontal orange lines correspond to a depth of 10. In addition to
the ordinary depth of coverage (which counts all reads in the contig,
and is mainly useful to indicate regions where an abnormal deficit or
excess occurs), the "reduced depth" can be displayed. This counts only
reads that are non-chimeric, have a positive LLR score against the
contig, and have no positive LLR scoring match to a read elsewhere in
the assembly -- i.e. the "confidently placed" reads. (See phrap.doc
for a description of LLR scores). Misjoins usually occur in places
where the reduced depth is 0 on both strands.
 Regions of the contig where the depth is 0 on one or both strands are
indicated in red.

  (iii) "Matches" and "links". "Matches" are pairwise read matches
between reads in different parts of the assembly, and are indicated by
a single (curved or straight) line connecting the midpoints of the two
matching regions. Contig matches (between two non-chimeric reads in
the assembled contigs) and chimera matches (between a chimeric read
and itself or another read) can be displayed separately. "Links"
connect the startpoints of two reads that derive from the same
subclone: forward-reverse links correspond to reads from opposite ends
of a subclone insert, while same-strand links correspond to walking or
duplicated reads in the same direction on the insert. Links will only
be indicated properly if the read naming convention assumed by phrap
is used (see section III.1 above).
 Each match or link is considered to be in one of three classes,
indicated by color: "problems" (red), which indicate serious
discrepancies or a significant possibility of incorrect, incomplete,
or non-unique assembly; "ok" (black), which tend to confirm the
assembly or are otherwise consistent with it; and "grayzone" (blue or
green), which may in some cases indicate problems but are probably ok.

Links are considered "ok" if the two read starts meet the expected
spacing and orientation constraints, otherwise they are marked as
"problems" (in many cases however these reflect subclone tracking
errors rather than assembly errors). Matches are considered "ok", and
are not shown, if they are between reads assembled in the same
location, or have LLR scores below the cutoff. Matches are considered
"problems" if they have LLR scores above the cutoff, are non-local,
and involve regions where the reduced depth is 0. "Problem" matches
between or within large contigs often indicate the presence of
near-perfect repeats, which may or may not have been misassembled; or
(in some cases) a join that was missed. "Problem" matches between a
small (e.g. singleton) contig and a large one typically represent
cases where a read could not be assembled into its proper location by
phrap, due generally to some anomaly involving it (e.g. an internal
region of low quality data). Non-local matches with LLR scores above
the cutoff which do not lie in regions of reduced depth 0 are
considered to be "grayzone". Usually these involve isolated pairs of
reads each of which extends part way into a repeat, such that the
overlapping region is too small to contain high-quality discrepancies.
The fact that the reduced depth is non-zero means that there are other
reads in the same region which do not have any positive LLR scores to
the other region, so it is unlikely that there is a significant
misassembly in such cases, although it is possible that one of the two
matching reads may have been incorrectly placed.
 Chimeras generally have one "ok" match (to the site where the read
was assembled) and one or more "problem" matches to the region from
which the other part of the chimera derives (sometimes these matches
all have negative LLR scores and are thus not shown with the default
LLR cutoff). Some chimeras actually represent subclones with internal
deletions; these tend to be obvious since the two pieces map to nearby
locations in some contig and are in the same orientation there.
 Double-headed arrows on matches indicate that the matching regions
are on opposite strands; double-headed arrows on links indicate that
the orientation of the two reads is inconsistent with the read names
(i.e. forward-reverse pairs that point in the same direction, or same
strand pairs that point in opposite directions; these are colored as
"problems" if the reads are in the same contig).

   (iv) Low quality contig bases (bases having phrap qualities that are
lower than a specified threshold), and high quality discrepancies
(positions where there is an aligned read discrepant with the contig
base and having a phrap quality that exceeds the threshold). A
horizontal green line above the contig indicates the quality
threshhold value ("qual cutoff"). Low quality contig bases are
indicated by a vertical blue line drawn from a height corresponding to
the contig quality value, up to the threshhold line; high quality
discrepancies are indicated by a vertical blue line drawn from a
height corresponding to the discrepant read's quality, down to the
threshhold line. Thus in each case longer lines are more "serious"
(reflect a larger deviation from the threshhold, in the wrong
direction).

   Passing the mouse pointer (slowly!) over an item of types ii-iv
above results in its being highlighted (converted to yellow); it
remains highlighted until the pointer is moved over another item, or a
contig line. Textual information about the highlighted item is shown
in a box in the top right region of the display.
 Several parameters can be changed by entering new values in
appropriate boxes at the bottom of the display. Four different
magnifications can be changed: horizontal magnification; the "spacing

magnification" which determines the vertical spacing between contigs;
and the depth and quality magnifications, which affect the scales used
for coverage depth and quality. Values of each of these are specified
as percents of the original (startup) magnification. The role of LLR
cutoff, and qual cutoff are described above. The unalign parameter
refers to the size of the largest segment involved in the pairwise
read match that is unaligned against the contig sequence. If large,
such segments may indicate a misplaced read or (occasionally) an
otherwise undetected chimera. Min fwd-rev and max fwd-rev refer to
the minimum and maximum allowed separation (in bp) between (the starts
of) forward-reverse read pairs; i.e. these represent the minimum and
maximum allowed subclone insert size. Separations outside this range
(or in the wrong orientation) are marked as problems. Min ss and max
ss are the minimum and maximum spacing between same-strand reads (i.e.
size of a walking step).
 The display is updated to reflect changes made in the above
parameters only after a relevant button (Show Contig Matches, etc.) is
pressed.

 Colors can be changed by altering the variable definitions that occur
near the beginning of the phrapview source file, and re-running the
program.

3. Gap

  A version of phrap ("gcphrap", for "gap-compatible phrap") suitable
for use with the Staden gap4 package may be created as follows: Obtain
relevant source files from the Staden web site
(http://www.mrc-lmb.cam.ac.uk/pubseq/index.html). Put these in the
same directory as the phrap source files, and enter the command

> make gcphrap

 Consult the Staden web site for instructions on using gcphrap in
conjunction with gap4, including input and output file formats. In
addition to the usual phrap command line options, gcphrap accepts an
option -exp, which is used to specify the name of a directory in which
the output experiment files will be placed.

VI. PHRAP OUTPUT

Phrap output includes the "standard output", the "standard error", and
a series of output files. The latter receive names that consist of the
input read file name, with an appropriate suffix (e.g. ".ace")
appended.

The standard output is described below. The other files
are

 (i) The standard error (which is normally written to the terminal,
but may be redirected to a file). This contains information
indicating the point in the run that phrap has reached, summary
results for some of the steps, and various warning and error messages.
 (ii) The .contigs file. This is a FASTA file containing the contig
sequences (without pads; bases in this file are upper case if and only
if the quality is >= qual_show). These include singleton contigs
consisting of single reads with a match to some other contig, but that

couldn't be merged consistently with it.
 (iii) The .contigs.qual file. This has the phrap-generated qualities
for the contig bases.
 (iv) The .singlets file. A FASTA file containing the singlet reads
(i.e. the reads with no match to any other read).
 (v) The .log file. This includes various diagnostic information, and
a summary of aspects of the assembly; it is probably only of interest
to me, for trouble-shooting purposes.
 (vi) The .problems file. Again this is probably only of interest to
me.
 (vii) The .ace file (produced when the options -new_ace or
-old_ace is used). This file is required for viewing the assembly
using consed. It's format is described in the consed documentation.
 (vii) The .view file (produced when the option -view is
used). Required for viewing the assembly using phrapview.

Standard output:

 The standard output has the goal of giving as complete a summary as
possible of the final assembly, including data anomalies and possible
sites of assembly error. The information it provides includes a list
of the contigs and the reads they contain, information about the
quality of the alignments of the reads against the contig sequence,
matching regions within and between contigs, suspect reads (probable
deletions or chimeras), sites of possible errors in the assembly or
contig sequence (e.g. discrepancies between reads and contig
sequence), and results of the forward/reverse read consistency checks.

The output includes, in order:

  (i) Summary information about the run conditions (command line,
phrap version number, and parameter settings) and the input data
file(s) (sequence and quality, template/read counts and read
orientations, chemistry). It is worth reviewing this information
routinely for possible problems with the input data, such as parameter
misspecification, read nomenclature that is causing faulty inferences
by phrap regarding templates or chemistry, or a missing quality file.

  (ii) Low-quality regions at beginnings or ends of reads which
consist largely (>50 %) of a single nucleotide, due (usually) to
degenerate signal. These are internally converted to N's, to prevent
them from causing spurious matches which could interfere with
assembly. (Note that the .ace and .singlets files output by phrap will
contain the N's in place of the original sequence).

  (iii) Exact and near-exact duplicate reads. Exact duplicates, which
probably represent duplicate entry of the same data, are excluded from
assembly (this feature may be turned off using the option
-retain_duplicates). The near-exact duplicates are only listed if they
are from different subclones (insofar as this information is available
to phrap!).

  (iv) Probable unremoved sequencing vector, detected as regions at
beginnings of reads which match only the beginnings of other reads.
Phrap's method for identifying these is not foolproof and should not
be considered a substitute for screening out sequencing vector as
described above (III.4). If vector has been removed in that manner,
the "unremoved vector" detected here may simply be inaccurate vector
sequence, that did not match the correct sequence in the original
cross_match screen but does match other reads in which similar errors

occur. You may want to consider adding the (potentially erroneous)
sequence that is displayed in the phrap output to your vector file,
and redo the cross_match vector screen, to increase the likelihood
that you catch all of the vector. I would appreciate hearing of any
examples of sequence that is not vector being erroneously labelled as
such by phrap.

   (v) "Internal read matches", i.e. reads which have off-diagonal same
orientation matches to themselves (due e.g. to short tandem repeats).
This information is ignored at present, but ultimately will be used to
improve assembly of tandem repeat regions. Due to the use of
complexity-corrected Smith-Waterman scores some commonly occurring
short microsatellites may not be detected here.

Node-rejected pairs.

   (vi) Multi-segment reads, defined as reads whose confirmed parts
break into two or more pieces. These include chimeras and non-chimeric
"single links". They are omitted from the initial merging passes to
reduce the risk of false joins in the assembly, but are incorporated
in later merging passes; as a result, chimeric reads will generally be
assembled into a contig, but should be tagged as chimeric.

   (vii) Probable deletion reads, identified as reads which match some
other read but have a gap with respect to it of >10 bp (more
specifically, read 1 is determined to have a deletion with respect to
read 2 if there are two pairs of matching segments between the reads
such that the segments in read 1 are nearly adjacent (within 20 bases)
and the separation between the segments in read 2 is at least 10bp
larger than the separation in read 1), and for which there is no other
clone that shows the same deletion (the latter condition is needed to
rule out the possibility of repeated sequences for which some copies
have deletions). For each deletion clone and each such matching read,
the apparent size of the deletion, location of the deletion and (in
parentheses) name of the matching (undeleted) read and extent of the
segment that was deleted from in it are given.
   The probable deletions are not used in assembly and instead appear
as singleton contigs.

Revised quality (= phrap quality), and LLR score histograms, for two passes.

   (viii) Summary of "rejected" pairwise alignments; i.e. matches which
either prematurely terminate (do not extend as far as they should,
given the apparent sequence quality), or include mismatches between
high quality bases. In such cases the reads are probably from
different repeats, and so the alignments are not used in the assembly
(N.B. at present incompletely extended matches are considered during
the assembly, since a more sensitive criterion used there will reject
them if warranted).

   (ix) (N.B. This part of output needs revision). Summary of
information about accepted pairwise alignments, including no. of
confirmed reads (i.e. reads matching some other read), their average
lengths (total, confirmed, "strongly confirmed" (i.e. matching a
reverse sense read) and trimmed); a crude estimate of the clone size
(assuming all non-rejected matches are real -- if there are
near-perfect repeats, the size may be underestimated) and depth of
coverage; a histogram of depths (letting the depth of a read be the
maximum depth that occurs in it); a summary of the discrepancies
between matching reads (by strand sense, nucleotide and quality); and

a histogram of spacings between adjacent indel pairs. By sense of the
aligned pair (reads in same direction, reads in reverse directions).

Blocked reads
reads lacking a high-quality segment
Bypassed reads

   (x) Isolated singlets. These are reads having no non-vector match
(with score at least minscore) to any other read. Shown for each read
are its length, and the number of high-quality non-X bases. For most
reads, the latter number should be 0, indicating that the read is
either entirely vector (its high-quality part has been completely X'd
out) or is of very low quality; reads for which this value is positive
may be contaminants.

   (xi) Contigs (including "singleton contigs" consisting of a single
read -- these are cases in which the read did have a match with some
other read(s), but could not be consistently assembled with it). The
contigs are sorted by (increasing) number of reads, so that the
singleton contigs appear first. The following information appears for
each contig: the number of reads; the total length of the contig
sequence (not including pads); the length of the trimmed sequence
(i.e. after regions consisting entirely of lower case letters, N, and
X have been removed from either end of the contig); and a list of the
reads in the contig and information about their alignments. For each
read the following information is given: a 'C' if the read is in
reverse orientation; the starting and ending positions for the read
(i.e. its full length including vector and low-quality data, not just
the part that is alignable) with respect to the contig sequence;
the read name; the score of the alignment of the read against the
contig sequence; (in parentheses) the highest score of a match of the
read against some other read in a different contig or elsewhere in the
same contig (so reads for which this number is non-zero are those
which overlap a repeat, or an incorrectly or incompletely assembled
region); the per cent mismatch, insertion, and deletion rates for the
alignment of the read against the contig sequence; the number of bases
at the beginning of the read which were not included in the alignment;
(in parentheses) the number of bases at the beginning of the read
which did not align against any other read (i.e. were not confirmed);
and similarly for the bases at the end of the read. If the
unparenthesized number is substantially greater than the parenthesized
number that accompanies it, there could be a problem with the
alignment. Note however that different penalties are used for aligning
a read against another read as opposed to aligning it against the
contig sequence.

   Following the read list is information to be used in assessing the
quality of the assembly. This includes:

   A description of regions in the contig sequence whose quality has
been adjusted on the basis of alignments of reads to the contig (e.g.
regions that had double-stranded confirmation in the initial pairwise
comparison may not have when the contig is assembled, due to
resolution of repeats; and regions that were not confirmed in the
initial pairwise comparisons may be confirmed in the assembled contig,
due to the use of a lower gap penalty which gives more complete
alignments).

Histogram of the qualities of the contig bases; the extents of the
leading and trailing quality 0 regions of the contig (such regions are

likely to be highly inaccurate); and a list of the bases that have
quality < qual_show.

Slack and HQ Mismatch histograms (not yet documented here).

   A table showing "gaps" on each strand (i.e. regions for which there
is no aligned part of any read, so that additional data, or possibly
editing to permit alignment of existing data, is required); the
closest read that could potentially be extended or edited to cover the
gap; whether or not the inaccurate, unaligned part of that read
already covers the gap (in which case editing might be sufficient to
close it); and the length of an accurate extended read (from the same
start in the same clone) that would be required to cover the gap. The
table appears following the list of reads in the contig and their
positions. An example (from a contig in C05D11):

Gap	Size	Closest read (Start)		Covers now?	Read length required to cover
Top strand:					
left – 244	244+				
1809 – 1880	72	x72c5.s1	(1382)	No	499
2967 – 2993	27	ae82b07.s1	(2562)	Yes	432
5392 – 5536	145	z17c5.s1	(4965)	No	572
6036 – 6320	285	z13f6.s1	(5537)	No	784
8137 – right	0+	z26c9.s1	(7667)	No	469+
Bottom strand:					
left – 0	0+	z17b4.s1	(563)	No	563+
1138 – 1139	2	z14f3.s1	(1143)	Yes	6
6940 – 7230	291	z17c10.s1	(7666)	No	727
7717 – right	420+				

   The gaps that say "left" or "right" are the ones at the left and
right ends of the contig (the gap size in this case is the part of the
existing contig sequence at that end that is not double stranded, with
the plus indicating that there is an additional gap to the left or the
right of the contig; these gaps are always designated not covered
("No")), and the relevant read in this case is the one that should be
extended to close gaps between contigs, while the other gaps are
internal gaps and the relevant reads are the ones required to get
double-stranding. Note one problem in this example: the bottom strand
gap of size 2 lies in the inaccurate 5' part of the z14f3.s1 read, so
that would probably not be an appropriate read to edit or get more
data from. Need appropriate cutoff values (i.e. size of 5' part of
read to ignore; this obviously depends on whether the 5' end has
already been trimmed prior to inputting it to phrap). Also, note that
since the full read lengths are used by phrap, the ends of the contigs
may include some very inaccurate sequence; so the size of the end gaps
(i.e. the total amount of inaccurate sequence there) could be larger
than indicated; this isn't particularly important though since in any
case one will always want to extend those reads to try to close the
gaps between contigs.

 Phrap uses a lower gap penalty (-2) in aligning reads to the contig
sequence (to allow more complete double-stranding than the -9 default
used for aligning reads to each other).

Gaps with "No" in the Covers? column will always require more data (an
extended read or walk step, depending on the required length). Those

with "Yes" may be coverable by editing of the appropriate read; but it
may be simpler to automatically get longer reads for them also,
particularly if the gap size is large, since substantial editing would
probably be required in that case to use the existing data. Walking
primers could be output directly from OSP analysis of the contig
sequence (high quality part, i.e. all upper case letters) in cases
where the gap looks too large to close with an extended read
(presumably "finish" already does this?)

Following the gap table, there is a table giving, for each quality
value, the total number of bases of that quality in the reads for the
contig; the number which are aligned (i.e. included in the SWAT
alignments of the reads against the contig); the cumulative no. of
aligned bases (for this and higher qualities); the number of bases not
included in the alignment (but potentially alignable -- i.e. not
lying before the beginning or after the end of the second sequence);
the number of discrepancies of each type (substitution, deletion,
insertion); the total # of discrepancies (for this quality level) and
their percentage (of the aligned bases of the given quality), and the
cumulative # of discrepancies and their percentage (of the cumulative
no. of aligned bases). The regions at the ends of the sequence that
consist entirely of quality 0 are reassigned quality values of -1,
since these usually correspond to the lowest quality data at the
extreme ends of reads. (Not done in the contig quality histogram.)
 An 'N' in either a read or the contig sequence is always counted as a
substitution error.

Depth 0 regions.

Following this there is more specific information about possible
problems with the assembly:

 Unaligned segments: parts of reads of length at least 10 that did not
align to the contig sequence (the part of the read having quality -1
is not considered).

 High quality discrepancies: sites in the contig sequence for
which at least one read with a quality at least qual_show disagrees
with the contig sequence. Most errors in the contig sequence will be
found either among these sites or in the regions where the contig
quality is < qual_show.

 Low quality suspects: sites in the contig sequence of quality <
qual_show (but greater than 0) for which there is a discrepant read of
equal or greater quality. Such sites are particularly error-prone.
The list gives the site position, base in the contig sequence and its
quality, and quality of the discrepant read.

[Formerly also printed out: for each of the two strands, the # of
reads on that strand having a substitution, insertion or deletion
(with respect to the contig sequence) at that position, and depth (#
of reads whose alignments extend across that position). The site
position is preceded by a '<' (resp. '=') if there is a read with a
discrepant higher (resp. equivalent) quality base at that position.]

 There then is given a list of the regions that are not covered by
"unique-reads" (i.e. reads with no non-rejected matches); and of
regions that match (as detected by one or more non-rejected pairwise
alignment between reads) other contigs or other regions in the same
contig. These are likely to be either true near-perfect repeats or

reflections of an error (of omission) in the assembly. To help
distinguish these possibilities, such regions are designated spanned
(if some read in the current contig extends all the way across it and
includes sequence to either side of it; a list of all such reads is
given) or "UNSPANNED" (if no such read exists). Misassembly errors of
commission generally arise from true near-perfect unspanned repeats,
and should result in an "UNSPANNED" flag for at least one of the two
matching regions. However not all of them do -- while the fact that
repeats exist should generally be detected by phrap, the full extent
of the repeat may in some cases not be detected, and the apparent
repeat may be "spanned" even though the full (true) repeat is not.
Moreover in some cases of misassembly only one of the matching contig
regions may be flagged as unspanned. Misassembly errors of omission
should always result in "UNSPANNED" matching regions at the end of one
or both contigs.
 Regions of a non-singleton contig matching an anomalous (chimeric or
deleted) read are indicated only in the output for the singleton contig
that contains the anomalous read.

 (xii) Following the contig information, there is a summary of the
forward/reverse read consistency checks. For each pair of reads with
the same clone name are given the contig no. and orientation within the
contig (0 = top strand, 1 = bottom strand) for each read; an "L" (for
"link") if the reads occur in different contigs; a '*' if there is an
inconsistency (e.g. forward and reverse reads which are not pointing
towards each other, or two forward reads which are not on the same
strand); distance between read starts for forward/reverse pairs
(should correspond to clone insert size, and be positive); and
distance between starts for forward/forward or reverse/reverse pairs.

N.B. IN ALL PLACES WHERE A POSITION IS GIVEN, THE POSITION IS WITH
RESPECT TO THE UNPADDED SEQUENCE, NOT THE PADDED SEQUENCE.

VII. CROSS_MATCH OUTPUT

 In addition to the standard output, the files produced include the
standard error and .log files (as for phrap); and the .screen file
(produced when the option -screen is used).

Standard output

  The standard output lists matches between any sequence in the first
input file (the "query" sequences) and a sequence in the second (or
later) input file (the "subject" sequences); or, if a single input
file is provided, the matches between any two sequences in this file.
The matches that are reported are controlled by the command line
options -minscore and -masklevel, as well as by the options that
control scoring of the alignments and the band search (see section IV
above). The reported matches are ordered by query, and for each query
by the position of the start of the alignment within the query.
  For each reported match, an initial output line gives summary
information:

 Example:

440 2.38 1.39 0.79 hh44a1.s1   33   536 (0) C 00311   (3084) 8277   7771 *

Interpretation:
  440 = smith-waterman score of the match (complexity-adjusted, by default).
  2.38 = %substitutions in matching region
  1.39 = %deletions (in 1st seq rel to 2d) in matching region
  0.79 = %insertions (in 1st seq rel to 2d) in matching region
  hh44a1.s1 = id of 1st sequence
  33 = starting position of match in 1st sequence
  536 = ending position of match in 1st sequence
  (0) = no. of bases in 1st sequence past the ending position of match
        (so 0 means that the match extended all the way to the end of
          the 1st sequence)
  C 00311 : match is with the Complement of sequence 00311
  (3084) : there are 3084 bases in (complement of) 2d sequence prior to
        beginning of the match
  8277 = starting position of match in 2d sequence (using top-strand
        numbering)
  7771 = ending position of match in 2d sequence
  * indicates that there is a higher-scoring match whose domain partly
includes the domain of this match.

Following this line there appears (if the flag -discrep_lists is
specified) a listing of the discrepancies between the aligned regions
of the two sequences, giving for each discrepancy its type, position,
nucleotide, and quality (in the query sequence), and its position in
second sequence. This list is followed (if -discrep_tables is
specified) a table giving, for each quality value, the total number of
bases of that quality in the first sequence; the number which are
included in the SWAT alignment; the cumulative no. of aligned bases
(for this and higher qualities); the number of bases not included in
the alignment (but potentially alignable -- i.e. not lying before the
beginning or after the end of the second sequence); the number of
discrepancies of each type (substitution, deletion, insertion); the
total # of discrepancies (for this quality level) and their percentage
(of the aligned bases of the given quality), and the cumulative # of
discrepancies and their percentage (of the cumulative no. of aligned
bases). This information is useful for computing accuracy as a
function of quality, for automatically generated contig sequences.
 In the discrepancy listing and table, the regions at the ends of the
sequence that consist entirely of quality 0 are reassigned quality
values of -1, since these usually correspond to the lowest quality
data at the extreme ends of reads. (Not done in the phrap output...)
 An 'N' in either sequence is always counted as a substitution error.
 If the flag -alignments is specified, a complete alignment for each
reported match is displayed.

VIII. EST ASSEMBLIES
 (TO BE ADDED)

IX. PROBLEMS

1. Insufficient memory.

  If the run stops prematurely, displaying the message

 FATAL ERROR: REQUESTED MEMORY UNAVAILABLE

then you need to increase the amount of memory alloted to you by the operating system. This can usually be done without obtaining additional physical RAM for your computer -- it is usually enough just to have the operating system allocate additional virtual memory to your process. (However if you are substantially exceeding physical RAM the run may take an exceptionally long time to complete). On Unix systems this can be done using the "limit" command, as follows:

i. Run "limit" with no arguments to list the system resources currently available to you. For example (this output is for a DEC alpha computer -- the output on other computers may look somewhat different):

```
> limit
cputime         unlimited
filesize        unlimited
datasize        131072 kbytes
stacksize       2048 kbytes
coredumpsize    unlimited
memoryuse       699392 kbytes
descriptors     4096 files
addressspace    1048576 kbytes
```

The relevant parameter here is datasize, currently set to about 131 megabytes.

ii. Now run limit with the option -h to see what the maximum possible values for these parameters are:

```
> limit -h
cputime         unlimited
filesize        unlimited
datasize        1048576 kbytes
stacksize       32768 kbytes
coredumpsize    unlimited
memoryuse       699392 kbytes
descriptors     4096 files
addressspace    1048576 kbytes
```

This shows that datasize can be increased to 1048576 kbytes, or about 1 Gb. It may be possible to increase this even further by altering the operating system configuration (this may require rebuilding the kernel and is best left to an expert).

iii. Run limit once again, providing the desired value for datasize, e.g.:

```
> limit datasize 500000
```

which increases datasize to 500 megabytes. Any value smaller than the value returned in step ii can be used.

iv. Now rerun phrap. Note that the new datasize value will only apply to the particular shell (window) in which you execute the limit command, so you may need to repeat step iii next time you log in or if you switch to a different window.

2. Other phrap- or cross_match-generated error messages
   (TO BE ADDED)

3. Phrap- or cross_match-generated warning messages

 These can generally be ignored, unless there is something obviously
wrong or incomplete about the assembly
 (TO BE ADDED)

4. "Crashes" reported by operating system

 True "crashes" (e.g. segmentation fault, floating point exception)
generally indicate a bug in the program. I would greatly appreciate
having any such problem reported to me, as described below.

5. Long running time.

 If the run appears to be "stuck" or takes an exceptionally long time
to complete, try using a larger value for the parameter -minmatch
(above, section IV.2)

6. Misassemblies, incomplete assemblies, incorrect selection of read for
consensus sequence.
 Marked discrepancy not split ...
 (TO BE ADDED)

7. Reporting problems

 If you have a problem that is not addressed by any of the above,
first be sure you have read sections I-IV of the documentation and
that you are using a current version of the programs. Then repeat the
run in which the problem occurred, capturing the standard error to a
file. For example, the following command, run on a Unix computer in a
C shell, captures the standard output to a file phrap.out and the
standard error to a file stderr.file:

    (phrap reads.screen > phrap.out) >& stderr.file

Then send a copy of the standard error (only! -- not the standard
output), i.e. the file stderr.file in the above example, to me at the
following email address:

    phg@u.washington.edu

(If the standard error is more than 1 or 2 pages you have probably
inadvertently captured the standard output.) Please do NOT send the
standard output, datasets, or any other large file without making
prior arrangements to do so with me! Large files cannot be sent to the
above address, which has a limited disk quota on a Univ. of Washington
computer.

APPENDIX: ALGORITHMS

[N.B. MUCH OF THE FOLLOWING DESCRIPTION IS SOMEWHAT OUT-OF-DATE].

Outline of phrap assembly:

0) Read in sequence & quality data, trim off any near-homopolymer runs
at ends of reads, construct read complements.

1) Find pairs of reads with matching words. Eliminate exact duplicate
reads. Do swat comparisons of pairs of reads which have matching
words, compute (complexity-adjusted) swat score.

2) Find probable vector matches and mark so they aren't used in assembly.

3) Find near duplicate reads.

4) Find reads with self-matches.

5) Find matching read pairs that are "node-rejected" i.e. do not have
"solid" matching segments.

6) Use pairwise matches to identify confirmed parts of reads; use these to
compute revised quality values.

7) Compute LLR scores for each match (based on qualities of discrepant and
matching bases).

(Iterate above two steps).

8) Find best alignment for each matching pair of reads that have more
than one significant alignment in a given region (highest LLR-scores
among several overlapping).

9) Identify probable chimeric and deletion reads (the latter are
withheld from assembly).

10) Construct contig layouts, using consistent pairwise matches in
decreasing score order (greedy algorithm). Consistency of layout is
checked at pairwise comparison level.

11) Construct contig sequence as a mosaic of the highest quality parts
of the reads.

12) Align reads to contig; tabulate inconsistencies (read / contig
discrepancies) & possible sites of misassembly. Adjust LLR-scores of
contig sequence.

 Phrap adjusted quality values/error probabilities: Phrap computes
adjusted quality values for each read on the basis of read-read
confirmation information, as follows. If a read is confirmed by an
opposite-strand or different-chemistry (dye terminator vs. dye primer)
read at a given position, that position is given a quality which is
the sum of the two input read qualities; when more than one
opposite-strand read confirms a given position, only the single
highest quality from all opposite-strand matching reads is used. If
qualities are related to error probabilities as described above, this
procedure can be interpreted as computing an error probability for
each base which is the product of the error probabilities for the two
reads; since error profiles for opposite-strand or different chemistry
reads are essentially independent, this is reasonable, although it is

somewhat conservative in that it does not take into account
same-strand matches (which clearly should count for something,
although they certainly are not independent).
  Contig positions are assigned quality values equal to the highest
adjusted quality of any read at that position, and then adjusted
downward to take into account any discrepancies with other reads. As a
result, when phred input qualities are used, the output phrap
qualities associated to each contig position have a natural
interpretation as (conservative) error probabilities. Such error
probabilities provide an extremely useful guide to where editing or
additional data collection is needed.
 The phrap adjusted qualities are used in computing "LLR scores" for
each pairwise match between two reads. These scores take into account
the qualities of the base calls in the reads, and are (approximate)
log-likelihood ratios for comparing the hypothesis that the reads
truly overlap to the hypothesis that they are from 95% similar
repeats. The point is that discrepancies between overlapping reads are
due to base-calling errors and thus tend to occur in low-quality
bases, whereas reads from different repeats can have high-quality
discrepancies that are due to sequence differences between the
repeats; and the probability of the observed data under each
hypothesis can be quantified using the interpretation of the phrap
qualities in terms of error probabilities. A pairwise match tends to
have a positive LLR score if the two reads overlap, whereas it tends
to have a negative LLR score if the two reads are from different
repeats (unless the repeats are nearly identical).

 Description of algorithms:

   0) The procedure used in both phrap and cross_match to find
sequence matches is the following. First, (in phrap only) any region
at the beginning or end of a read that consists almost entirely of a
single letter is converted to 'N's; such regions are highly likely to
be of poor data quality which if not masked can lead to spurious
matches. Reads are then converted to uppercase (in order to allow
case-insensitive word matches). All matching words of length at least
minmatch between any pair of sequences are then found, by (i)
constructing a list of pointers to each position (in each sequence)
that begins a word of at least minmatch letters not containing 'N' or
'X'; (ii) sorting the list (using a modified version of quicksort,
with string comparison as the comparison function + a few tricks to
improve speed by keeping track of the parts of the words that are
known already to match); (iii) scanning the sorted list to find pairs
of matching words. For each such pair, a band of a specified width
(in the imaginary dot matrix for the two sequences), centered on the
diagonal defined by the matching words, is defined. Overlapping bands
(for the same pair of sequences) are merged. Following construction
and merging of bands, a "recursive" SWAT search of each band is used
to find matching segments with score greater than or equal to
minscore: "recursive" here means that if such a match is found, the
corresponding aligned segments in each sequence are (conceptually) X'd
out and the process repeated on the remaining portions (if any) of
each sequence. This procedure allows (at the cost of some redundant
calculation) detection of multiple matching pieces in different
locations, and will usually find most copies of repeats (since they
generally occur in separate bands).
 By default, SWAT scores are complexity-adjusted (so that matches for
which the collection of matching nucleotides has biassed composition
have their scores significantly penalized.)

1) In phrap, sequence pairs with score >= minscore are considered for
possible merging. A critical issue here is the appropriate score
matrix for SWAT. [Given the generally high accuracy of reads and the
desirability of minimizing false matches due to imperfect repeats, a
relatively high penalty seems desirable. Currently I use +1 for a
match, -9 for a mismatch involving A,C,G, or T, 0 for a match or
mismatch involving N, -1 for a match or mismatch involving X, -11 for
the gap initiating penalty (the first residue in a gap), and -10 for
the gap extension penalty (each subsequent residue).] Setting the
indel penalties slightly higher than the mismatch penalties gives
better alignments by favoring mismatches in compression regions. Since
SWAT uses profiles, one has the option of distinguishing different
quality levels by use of different symbols (e.g. upper case for more
accurate calls, lower case for less accurate calls) and setting the
penalties appropriately; this would involve using the quality levels
to adjust the symbols used in the reads, which should be done AFTER
the word matching routine. (At present it does not seem particularly
useful to use differing positive scores for different nucleotides to
reflect their different frequencies).

   1a) Determine "confirmed" part of each read (i.e. the part which
appears in a SWAT alignment against some other read; a read with the
same name -- up to an internal '.' if any -- is not considered
confirming). Probable chimeras are detected as reads for which the
confirmed part can be separated into two non-overlapping pieces,
separated by at most MAX_CHIMERA_GAP bp (currently 30) such that the
part confirmed by a given read lies in one or the other piece but not
both; AND such that each piece has a prematurely terminating alignment
with some other read. Chimeric reads may arise from i) chimeric
clones; ii) gel mistracking across lanes; iii) unremoved sequencing
vector; (iv) deletion clones. Reads having two non-overlapping
confirmed pieces (but failing the "premature termination" condition)
are often non-chimeras that are the only link between two
non-overlapping contigs, and thus should be permitted in the assembly.
   Identify "strongly confirmed" regions in each read: having matches
to a reverse sense read.
   Identify deletions.
   Identify "rejected" alignments -- those which don't extend as far as
they should, or that have mismatches involving high quality bases.

   2) Sort all matching pairs by decreasing LLR score, and assemble
layout by progressively merging pairs with high score. Consistency of
merge is required: the SWAT alignment implies relative offset of one
contig with respect to another, and hence a potential implied overlap.
The SWAT alignment(s) should extend over essentially the entire region
of implied overlap, apart from an allowable gap (necessary to allow
for the fact that the ends of the alignments are somewhat uncertain).
Probable deletion clones (identified as reads which have two adjacent
pieces matching two separated pieces of another read, and having no
confirming read across the breakpoint), and chimeras are not used in
any merges. [N.B. Following is obsolete: Potential chimeras are
deferred to a second pass (as well as being flagged in output), so
that true reads will assemble first.]

   3) Construct contig "consensus" as a mosaic of individual reads.
Strategy: ends of alignments, and midpoints of perfectly matching
segments of sufficient length, define crosslinks - pursue crosslinks
which increase accuracy and extend read. Formally this is done by
constructing a weighted directed graph whose nodes consist of

(selected) positions in reads; there are bidirectional edges with
weight 0 between aligned bases in overlapping reads, and
unidirectional edges from 5' to 3' positions within a single read,
with weight equal to the total quality of the sequence between the two
nodes. Standard C.S. algorithms (dating back to Tarjan) permit
identification of a path with maximal weight, in time linear w.r.t.
number of nodes.

 The quality values for the resulting sequence are inherited from the
read segments of which it is composed.