Producing a wig file with the trace of physical coverage

Let's consider the beginning of a sam file sorted by read name:

```
Lactobacillus_3072417_3070178_0_0_0_0_0:2:2_0:4:0_0   67    genome  3070108  60    100M      =     3072347  2240  ...
Lactobacillus_3072417_3070178_0_0_0_0_0:2:2_0:4:0_0   131   genome  3072347  60    75M2D25M  =     3070108 -2240  ...
Lactobacillus_868142_866162_0_0_0_0_0:0:0_0:2:0_1     67    genome  866162   60    100M      =      868142  1981  ...
Lactobacillus_868142_866162_0_0_0_0_0:0:0_0:2:0_1     131   genome  868142   60    100M      =      866162 -1981  ...
Lactobacillus_1348887_1350827_1_1_0_0_0:1:0_0:3:0_2   115   genome  1336757  60    100M      =     1334817 -1941  ...
Lactobacillus_1348887_1350827_1_1_0_0_0:1:0_0:3:0_2   179   genome  1334817  60    100M      =     1336757  1941  ...
Lactobacillus_1286327_1288378_1_1_0_0_0:1:1_0:1:0_3   115   genome  1274308  60    100M      =     1272257 -2051  ...
Lactobacillus_1286327_1288378_1_1_0_0_0:1:1_0:1:0_3   179   genome  1272257  60    35M1D65M  =     1274308  2051  ...
Lactobacillus_232519_234829_1_1_0_0_0:0:0_0:1:0_4     115   genome  234829   60    100M      =      232519 -2311  ...
Lactobacillus_232519_234829_1_1_0_0_0:0:0_0:1:0_4     179   genome  232519   60    100M      =      234829  2311  ...
Lactobacillus_2138648_2140765_1_1_0_0_0:2:0_0:3:0_5   115   genome  2139995  60    100M      =     2137878 -2118  ...
Lactobacillus_2138648_2140765_1_1_0_0_0:2:0_0:3:0_5   179   genome  2137878  60    100M      =     2139995  2118  ...
Lactobacillus_2836017_2837629_1_1_0_0_0:1:0_0:2:0_6   115   genome  2837559  60    100M      =     2835947 -1613  ...
Lactobacillus_2836017_2837629_1_1_0_0_0:1:0_0:2:0_6   179   genome  2835947  60    100M      =     2837559  1613  ...
...
```

… and the beginning of the same sam file sorted by genomic position:

```
Lactobacillus_4_2256_1_1_0_0_0:1:0_0:2:0_c5f17        179   genome  4        60    100M      =        2256  2253  ...
Lactobacillus_4_1911_1_1_0_0_0:2:0_1:2:2_118dd7       179   genome  4        60    100M      =        1911  1910  ...
Lactobacillus_2061_8_0_0_0_0_0:3:0_0:1:0_12043c       67    genome  8        60    100M      =        2061  2054  ...
Lactobacillus_2013_9_0_0_0_0_0:4:0_0:1:0_43d20        67    genome  9        60    100M      =        2013  2005  ...
Lactobacillus_10_2338_1_1_0_0_0:1:0_0:2:0_129232      179   genome  10       60    100M      =        2338  2329  ...
Lactobacillus_12_2042_1_1_0_0_0:2:0_0:2:0_332fb       179   genome  12       60    100M      =        2042  2031  ...
Lactobacillus_12_2246_1_1_0_0_0:2:0_0:2:0_118b07      179   genome  12       60    100M      =        2246  2235  ...
Lactobacillus_13_1806_1_1_0_0_0:2:0_0:2:0_1230c5      179   genome  13       60    100M      =        1806  1794  ...
Lactobacillus_14_2057_1_1_0_0_0:2:0_0:3:0_c9b1c       179   genome  14       60    100M      =        2057  2044  ...
Lactobacillus_16_1951_1_1_0_0_0:3:0_0:2:0_191dc       179   genome  16       60    100M      =        1951  1936  ...
Lactobacillus_2199_20_0_0_0_0_0:1:0_0:3:0_da31d       67    genome  20       60    100M      =        2199  2180  ...
Lactobacillus_20_2158_1_1_0_0_0:2:0_0:1:0_8bf2b       179   genome  20       60    100M      =        2158  2139  ...
Lactobacillus_23_2141_1_1_0_0_0:2:0_0:3:0_9f694       179   genome  23       60    100M      =        2141  2119  ...
Lactobacillus_24_1711_1_1_0_0_1:3:0_0:3:0_f9146       179   genome  24       60    100M      =        1711  1688  ...
Lactobacillus_2103_27_0_0_0_0_0:1:0_0:3:0_e236e       67    genome  27       60    100M      =        2103  2077  ...
...
```

We should know from the sam specifications that the start position of the alignment is in column 3 (I.e the 4th column) and the extent of the alignment in column 5 (called "CIGAR"), where 100M means 100 matches while 75M2D25M means 75 Matches, followed by 2 Deletion and 25 Matches. Here the meaning of the term "Match" is limited to the extent of the alignment, not to the actual correspondence of the bases (See SAM specification).
As we can see in the above figure, each mate pair is reported twice because each of the two mates is independently aligned on the genome. However, BWA is not only aligning the reads, being also able to pair the mates and to report the reciprocal information in the appropriate field of the sam file. For instance, the length of the "insert" is reported in column 8 while the mapping position of the paired mate is in column 7. It can be seen that in one case the length of the insert is positive and in the other case is negative.

How can we calculate the values for the physical coverage and how can we produce a wig file, considering that the sam file is possibly very large?

Question 1: Is it better to sort the alignments by read name or by genomic position?
Question 2: Can we create the output as we read the file, or we must first read the entire file?

Answer to Question 1
First proposal by the teacher: since the alignments come in pairs (i.e. two lines) and we need to consider only one of the two lines, it could be better to sort by read name, thus the 2 alignments of each pair will come one after the other and one will be taken while the other will be discarded. Since the two alignments have respectively positive and negative "length of inserts", the easiest to take the alignment with the positive insert length.
Counter proposal by the classroom: We do not need to sort by read name. Since the length of the two alignments is in one case positive and in the other case negative, we can consider only the alignment with the positive value. We can discard the lines with negative values even if they are not next to they corresponding mate.

Answer to Question 2
First proposal by the teacher: create an array of counters, one counter for each genomic position, and initialize each counter to zero. Then, for each mate increase the counters corresponding to the region covered by the insert.
Second proposal by the teacher: using the above method, if an insert is 2000 bases long we will have to repeat 2000 times the increment operation throughout the length of the insert. To make the process more efficient, we can memorize the changes of coverage rather than the coverage itself. Thus, no matter which insert length, we will need only 2 operations: increase the counter at the beginning of the insert and decrease the counter at the end on the insert.
The perl code shown in the box implements this method. Note (see the last few lines) that the coverage needs to be calculated at every position, starting from a value of zero and then adding the positive or negative number found in @genome_change.
Counter proposal by Anna Danese: As discussed above (question #1) we do not need to sort the reads according to their name, thus we can sort them according to their position, then we can start at the beginning of the genome and for every position we can dynamically calculate the coverage considering the value of the previous position and the change of value reported in the array of counters.

```
#!/usr/bin/perl
die"Sintax: physical.pl samfile genomelen" unless $#ARGV>0; # if no arguments print instructions
@genome_change = (0) x $ARGV[1];                            # initialize genome_change, all to 0

open(INFILE, $ARGV[0]);                                     # open sam file: INFILE as file handler
while($line = <INFILE>)                                     # put in $line the next line of the file
{   if($line =~ /^@/) {next;}                               # if $line starts with @ then skip it
    @item = split("\t", $line);                             # split $line at tab, put results in @item
    if((($item[1] & 3) == 3) && ($item[8] > 0) )            # both reads align correctly and length > 0
    {  $genome_change[$item[3]]++;                          # increment start position
       $genome_change[ $item[7] + length($item[9]) ]--;     # decrement end position
    }
}
close INFILE;                                               # the sam file has been fully read

print("fixedStep chrom=genome start=1 step=1 span=1\n");    # print the heading of the wiggle file
$current_coverage=0;
for($i=0; $i<$ARGV[1]; $i++)                                # output the results
{   $current_coverage += $genome_change[$i];
    print "$current_coverage\n";
}
```

As we read the sam file, at every genomic position given by the start of the alignment, we can output the final coverage of the previous genomic position because we know that all the next alignments will be at most starting from the same position and not before. As described above, for each mate pair we have to update the counters indicating the changes of coverage, however, we only need to keep track of the stretch of genome immediately following the current position, where "immediately" means the maximal length that we may expect for an insert. Therefore we do not need to memorize the counters for the whole genome but only for the region spanning from the current position to the maximal length of the insert. This opportunity is very interesting bcause it would require a fixed amount of memory, no matter what the length of the genome is. The problem is that the position of the array of counters keeps changing together with the current position of the analysis. But certainly there are ways to manage this problem.