1:

I used the default parameters for bowtie2-build and bowtie2.

For bowtie2-build: –bmaxdivn 4 –dcv 1024 –offrate 5 –ftabchars 10

And the command for build index:

```
bowtie2-build HCV_gene.fa HCV_index
```

For bowtie2: -N 0 -L 22 -i (S,1,1.15) -D 15 -R 2

And the command for build align:

```
bowtie2 -f -x HCV_index –U virus.fa –S virus.sam
```

After apply bowtie2, there are 1249 reads can be aligned to the references in HCV_genes. And there are 864 reads can be aligned to the references in HIV_genes

2:

(a)

I create BWT index from suffix array. I first recruited all suffix for the input reads, then sorted them. After sort, I constructed bwt by finding the previous character for suffix array index. the main bottleneck for this process is to sort all input reads. Assume we have N reads and the average read length is K, directly sort will give us $\mathcal{O}(NK \log NK)$.

(b)

To recruit all overlapped reads, I first create a set in Python3. At each iteration, I will test whether the reported overlapped reads from my program in the set or not (this is $\mathcal{O}(1)$). So I will only add those reads that not already in the set. I also create a separate set to record all the reads I found in this iteration, so next iteration I only need to use these reads in the set to serve as the query read for BWT.

For the HIV_gene, my program found 19000 reads. And for the HCV_gene, my program found 28500 reads. Noticed that actually there are two reference genes in HCV_gene.fa. If I only use the real HCV_gene, my program will return 9500 reads. The number is exactly the difference of the number of the run for those two files. I test several different threshold, only when threshold is too high (like 249) the number of reads I recruited will drop. This suggest the result is consistent as the threshold should not impact the relationship between reads is the two gene not share great similarities.

3:

I use the default parameters for IDBAUD 1.1.1 .

The default parameeers are list here: –mink 20
–maxk 100
–step 20

–inner_mink 10
–inner_step 5
And the command for assemble:

```
idba_ud -r XXX.fasta -o result/
```

(a)

For HIV reads, I only have 1 contig and the read length is 9144. Compared to the reference gene (using BLAST), the assembled contig can cover the whole gene and got 100%identity

(b) For HCV reads, I have 2 contigs and the read length are 9467, 9144. So the N50 is 9467 and mean is 9305. Compared to the reference gene (using BLAST), the assembled HCV contig (There is contig for HIV) can cover the whole gene and got 100%identity.

4:

I also compare bewteen the contigs generated from different pipeline(using BWT or directly use IDBAUD). The BLAST result show that the contig is 100% identity and 100%cover. So the result is same.