1. Readwriter / Encoders. (sequence file: F, out files: O

Readwriter: 1) reads a chunk of data from F (if needed).
Go to step 10 if inbuffer data are empty ⟹ 2)
3) waits for a call from an Encoder.
4) locates the Encoder or any Encoder in idle.
5) copies outbuffer of the Encoder (if necessary)
6) sends part of chunk to Encoder (if needed).
7) notifies Encoder of that data ready.
8) writes outbuffer to files O (if needed)
9) Go to step 1.
10) waits for a call from an Encoder
11) locates any Encoder in idle.
12) copies outbuffer of the Encoder (if needed)
13) let the Encoder exit if no outbuff data.
14) writes outbuffer to files O (if needed)
15) Go to step 10 if Encoders are working.
16) END

Encoder: 1) notifies Readwriter of its need of inbuffer.
2) waits for a wake-up call from Readwriter.
3) Go to step 6 if inbuffer & outbuffer empty
4) works on encoding of inbuffer
5) Go to step 1 if empty inbuffer or full outbuffer.
6) END.

2. Hashing for each partition

Similarly to Readwriter/Encoder thread model, Readwriter reads from a partition, and "Encoders" works on updating a hash table. Use lock-free mechanism to allow multiple worker threads to update the hash table without waiting.
END.

3. Search kmer in the find output file or files. (F

kmer m → I wish to retrieve the # of occurrence.

1) $i \leftarrow h(m) \% n_i$.
2) $j \leftarrow h(m) / n_i \% n_p$
3) Locate i-iteration & j-partition part hash table in F.
4) Read the array (that must be sorted) to search it for the kmer m.