Questions on Li and Durbin (2009) *Bioinformatics* 25:1754-1760.

Fast and accurate short read alignment with the Burrows-Wheeler transform.

1. What where the practical challenges that prompted this paper? How did requirements for sequence aligners evolve compared to those that prompted development of the MAQ aligner?

2. What are the four basic designs for read mappers summarized in the introduction? What are the weaknesses of designs that are not based on the Burrows-Wheeler transform?

3. What is the main reason for the efficiency of algorithms based on the Burrows Wheeler transform?

4. The paper talks about *tries*… and it is not a spelling mistake! What is a trie?

5. When using a Burrows-Wheeler based indexing scheme, which data structures must be kept in memory? Can you estimate the approximate sizes of each of these structures?

6. What does the `CalculateD` function in BWA do?

7. Consider the following pseudo-code in the next page, adapted from Figure 3 in the original manuscript. What do each of the highlighted lines do (for example, the first highlighted line allows for an inserted base in the read that is not aligned to any base in the reference).

8. Why is the speed of the proposed algorithm sensitive to the mismatch rate?

9. What are the most striking features of Tables 1 and 2?

10. The authors discuss alignment to a hybrid reference genome, consisting of the concatenation of the human genome and the chicken genome. How is this useful?

11. What struck you most about the paper?

PSEUDO-CODE, adapted from Figure 3 in Li and Durbin (2009)

```
InexactRecursion(W, i, z, k, l)
     // Parameter W is the word being searched for
     // Parameter i is an index within the word
     // Parameter z is the maximum allowed number of differences
     // Parameters k and l denote the current suffix array interval

     if (z < D[i]) then return {};

     if (i < 0) then return {{k,l}};

     RESULT <- {}

     RESULT <- RESULT ∪ InexactRecursion(W, i-1, z-1, k, l)

     Foreach base ∈ {A,C,G,T} do

          k* <- C(base) + O(base,k-1) + 1
          l* <- C(base) + O(base,l)

          if (k <= l) then

               RESULT <- RESULT ∪ InexactRecursion(W, i, z-1, k*, l*)

               if (base == W[i])
                  RESULT <- RESULT ∪ InexactRecursion(W, i-1, z, k*, l*)
               else
                  RESULT <- RESULT ∪ InexactRecursion(W, i-1, z-1, k*, l*)

     return RESULT
```