Cow Proximity

Background. This is an irreverent example of a problem that's easy to solve inefficiently but hard to solve efficiently. Before writing any code, think about the data structures we've discussed in class, like queues, stacks, and hash tables, and consider which support the needed operations in constant time. You can combine several data structures, and you can use any data structures provided by your programming language, such as Python's `collections` module or C++'s STL. (Not all of these structures are available in base R, so you may have to go spelunking in CRAN. The `dequer` and `hash` packages may be useful.)

Task. Suppose that cows of the same breed get into an argument with each other if they are standing too close. Two cows of the same breed are "crowded" if their positions within a line of N cows differ by no more than K, where $1 \leq K < N$.

Given an array representing the breed IDs of a line of cows, compute the maximum breed ID of a pair of crowded cows. If there are no crowded cows, return -1.

Example. If we write a function *`crowded_cows`*`(`*`cow_list`*`, `*`K`*`)`, we have

```
crowded_cows([7, 3, 4, 2, 3, 4], 3) == 4
crowded_cows([7, 3, 4, 2, 3, 10, 4], 3) == 3
crowded_cows([7, 3, 1, 0, 4, 2, 16, 28, 3, 4], 3) == -1
```

Requirements.

- ☐ Write a function *`crowded_cows`*`(`*`cow_list`*`, `*`K`*`)` that returns the maximum breed ID of a pair of crowded cows in the supplied list.
- ☐ Write unit tests that comprehensively check the correctness of *`crowded_cows`*.
- ☐ Provide a command-line driver script which takes a list of cows on standard input and prints the result of *`crowded_cows`*, so that you can run e.g. `Rscript crowded_cows.R < cows.txt` and get the answer as output.
- ☐ Test your code on the provided file of 50,000 cows, `Data/cows.txt`, using $K = 25000$. Report your answer. Optimize your code and data structure choice so it runs in under five seconds. (With the right data structures, it can run in under one second.)