# Filters

The filters work by restricting data to the search term , this can be useful to allow the user to optimise their search.

LinkedList<Records> dataCollected

LinkedList<Records> output

int matchScore = 0

foreach entry in dataCollected

if entry matches filter condition

add to output

else

ignore the data

For date filter

Date dateToFilter

Date range

foreach entry in dataCollected

get entry date and

if entry date is in the range of dateToFilter+-range

add to output

else

ignore the data

The search algorithm gets the data from the filter algorithm, it then compares the hamming distance of the binary input and search term to give a closest match the closest 10 are then stored in an array to be printed out at the end of the search. For example, if the user was searching for “Rose” but entered the search term “Ros” there would still be a result returned. This means there will almost always be a result.

int maximumHammingDistance

Record[] databaseRecord;

Record[] results;

String searchTerm

foreach record x in databaseRecord

get hamming distance between x and searchTerm

if x hamming distance is less than maximumHammingDistance

foreach record y in results

if x distance is less than y distance

continue

else if x distance is greater than or equal to y distance

insert x behind y

else

continue

return results

**ADT Pseudo code**

**Recording queue**

int number of queue elements

int pointer to the front of the queue

int pointer to the back of the queue

add item to queue

insert item at the back of the queue

update pointer to the back of the queue using circular update method

update number of elements in the queue

remove item from the queue

take item off the head of the queue

update pointer to the front of the queue using circular update method

update number of elements in the queue

examine front

return the element at the head of the queue

clear

for each element in the queue

make current element equal to null

set number of elements to 0

length

return the number of elements in the queue

isEmpty

if the number of elements in the queue is 0

return true

else

return false

circular update

if a pointer is at the end of an array and the array NOT full

set the pointer to the beginning of the array (position 0)

else

increment pointer

return pointer