Problem 1: DNA sequence conversion

The main idea for this problem is to use & operation to get first two bits of the byte data:

Text

Description automatically generated

Then use & operation to get the rest 6 bits of the byte data:



Some information about this problem:

1. DNASeqConversion.java is the source file. DNASeqConversion.class is the class file.

2. Please run on Windows OS.

3. To make things simple, I presume the Input file size is normal/not too big

4. Didn't use buffer to process file

5. Didn't use asynchronous methods to process file

6. Test.txt is the input file whose content is different from the example you give me. Its content is :

Graphical user interface, text, application, email

Description automatically generated

7. Run with the following command:

Text

Description automatically generated

8. outputFile.txt is the output file:

Text

Description automatically generated

Problem 3: Algorithm

Algorithm.java is the java file, Algorithm.class is the class file.

Text

Description automatically generated

The main idea for this problem is to first make the right boundary, then make the left boundary.

A tuple is composed of Leftbound, Rightbound, sum. For example (I,j,sum) (3,3, 1.5) means left boundary is s[3], right boundary is s[3], the sum is s[3] (1.5).

To make right boundary, we try to find some index k so that s[j+1]+s[j+2]+….s[k]>0. Then we can set k as the new right boundary. Keep searching until to the end of the array of the input data, then we get the new tuple for example (3, 7, 9.8).

A screenshot of a computer screen

Description automatically generated with medium confidence

Similarly, we want to move the left boundary to right as possible as we can. We try to find some index k, so that s[i]+s[i+1]+….+s[k] <=0, Then we can set k+1 as the new left boundary, keep searching until reach the right boundary, then we get the new tuple for example (5,7,13) which is the best optimized tuple for the original tuple (3,3, 1.5)

Then we consider the rest part of the input array, from the right boundary+1, and get a new optimized tuple which can be compared the one we get before.

Keep doing this to the end of the array.

For the space complexity: the main space is used to store the input data, so it’s o(n).

For the time complexity:

The best condition, the right boundary is at the end of the array, we search the array once to make the right boundary, then search once to make the left boundary, the complexity is 2n.

The worst condition, each time we try to find an optimized tuple, we need search the rest of the array and can’t extend the right boundary. The complexity is o(n^2)

If we do some preprocess to bind the contiguous negative/positive/zero numbers together, the efficiency should be improved.