Decipher Substitution Cipher

Standard frequency order:

e t a i

Most frequent character in encoded maps to e
2nd " " " " " " t
3rd " " " " " " a
... and so on.

Step 1: Sort the characters in encoded by frequency.

ex: =>  p m b l l l
       e maps to r
       e t a i...

Step 2: Use mapping to decode.

Initialize empty string (str)
Read each character in encoded string:
  - find what the character maps to (dec)
  - add dec to str.

ex: Read a p, Decode to e
    m, t...
Questions:
- How to do Step 1?
- "" deal with non letters? w, etc.
- "" "" chars with the same frequency?

Step 1 details:

1. Create array of size 26 (freqArr)

```
0 1 ··· 25
```

# of a's b's c's ··· z's.

How to map 'a' to 0

'b'   1
'c'    2
   ···

Use ASCII value!

'a' in ASCII = 97, map to 0

'b'   = 98   "" 1
'c'    = 99   "" 2

index = ch - 97.
Now we'll have freqArr:

\[
\begin{array}{cccc}
0 & 1 & 2 & \cdots 25 \\
10 & 25 & 5 & \cdots 100 \\
\# \text{ of a's} & \# \text{ of b's} & \# \text{ of p's} & \# \text{ of z's}
\end{array}
\]

(Need to map to standard frequency array)

2. Sort freqArr:

Arrays.sort(freqArr) sorts in ascending order

\[
\begin{array}{cccc}
5 & 10 & 25 & \cdots 100 \\
\# \text{ of p's}
\end{array}
\]

3. Reverse sorted freqArr:

\[
\begin{array}{cccc}
100 & 75 & 25 & \cdots 10 & 5 \\
\# \text{ of p's} & \# \text{ of m's} & \# \text{ of b's}
\end{array}
\]

4. Find mapping:

p maps to e ≤ 100 (highest freq) corresponds to p
m " " t ≤ 75 (2nd " " ) " " to m
b a

\[
\text{How to find this?}
\]

⇒
Search for 100 in (original) freqArray

get index of 100

convert index to letter (p).

mapping:

```
| p | m | b | . | . | . | 1 |
```