P.2. Search for best matches on sequences

\[
\text{match score } = 0.7 \times 0.1 \times \ldots \times 0.05
\]

\[
\begin{array}{cccc}
A & C & G & T \\
1.7 & 1.7 & 1.5 & 1.4 \\
1.1 & 1.1 & 1.5 & 1.4 \\
1.1 & 1.1 & 1.5 & 1.4
\end{array}
\]

\[
\begin{array}{cccc}
A & C & T & G \\
0.6 & 0.5 & 0.5 & 1.0 \\
0.1 & 0.1 & 0.5 & 0.5 \\
0.1 & 0.1 & 0.5 & 0.5
\end{array}
\]

Return location and substring that has highest score (prob) on each seq.

\[
\begin{array}{cccc}
S_1 & S_2 & S_3
\end{array}
\]

P.3. a & b common: evaluate each pattern

Candidate pattern (for example)

- 2 mismatches
- 1 mismatch 0 MM
- 2 MM
- 1 MM
- 3 MM

Best match

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Best score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 MM</td>
<td>( S_1 )</td>
</tr>
<tr>
<td>1 MM</td>
<td>( S_2 )</td>
</tr>
<tr>
<td>3 MM</td>
<td>( S_3 )</td>
</tr>
</tbody>
</table>

Total # MM = ?
3a. Candidate pattern generation

pattern 1  pattern R
pattern 2  pattern k+1

patterns

All candidate patterns from above

# mismatches = 10

# mismatches = 20

Which pattern has smallest # of mismatches?
P3b. Similar to P3a.

diff in how to generate all candidate patterns.

Need to enumerate all possible patterns of length k.

Convert each integer \([0, 4^{k-1}]\) to a pattern.

get Pattern \((i)\)

```
char[k] P = new char[k];
for \(j = 0 \rightarrow k-1\)
    \(P[j] = (i / 4^j) \, \text{mod} \, 4;\)
```

integer division