課題 1-3（塩基使用頻度その 3）

0.1　課題 1-3

二連続塩基 (dinucleotide) の頻度解析を行うようにプログラムを変更する。マイコプラズマ全ゲノム DNA 配列の二連続塩基の頻度パターンを調べ、1-2 の結果からの期待値と比較、考察する。

この問題の中核は「期待値をどうやって求めるか」と「実際の値と期待値をどう比較するか」ということになると思います。観測値 (observation) を期待値 (expectation) で割ったものを O/E 値と言います。これを使うと例えば「AA は実際には期待値の半分しかなかった。しかし CT は 2 倍以上ある。何か理由があるに違いない」などという議論ができますので、今回は比較の方法としてこれを採用しましょう。

1　　各 dinucleotide の頻度を数える

1.1　データ変数の用意

```
my %dinuc;   #dinucleotide の数を数える
my %diexp;   #dinucleotide の期待値
my %diobs;   #dinucleotide の実頻度
my %oe;       #dinucleotide の O/E 値
```

こういう変数が必要になることが予想されます。ここで % が先頭についたものはハッシュ（連想配列）というものです。ハッシュとは、基本的には先頭に @ がつく配列と似たような機能を持っていますが、配列と違う点は引数に文字列が使える点です。ですから、例えば配列では

```
@array = (1, 2, 3);
```

としたら $array[1] は 2 となりますが、ハッシュの場合は

```
%hash = {'string'=>1, 'message'=>2, 'line'=>3}
```

としたら $hash{'message'} が 2 となるように、任意の文字列を引数に使えます。このハッシュを使えば引数として dinucleotide 自身を使えるようになりますので、プログラムが非常に楽になります。Perl ではこのハッシュを使う機会が多いので、是非しっかり覚えてください。

1.2　カウント部分の改良

基本は変わらないのですが、二連続塩基ですので 1 文字置換をする tr/// を使って数えることはできません。複数置換の s/// では置換された数を返さないのでこれを使うこともできません。しかたがないので、ここでは for 文を使って 2 文字の並びを少しずつずらして読んでいきましょう。ここでは塩基配列の最初から 2 文字の並びを一文字ずつずらして読むのですから、例えば "atgcggctg" という並びなら一つ目が "at"、二つ目が "tg" というようになります。つまりこれは [一文字目] から [配列の長さ -1 文字目] まで for 文で繰り返すことになります。ここで注意しなければいけないことは、Perl では先頭の番号は 1 ではなく 0 から始まる点です。つまり、for 文で繰り返す場合は [0 文字目] から [配列の長さ -2 文字目] までとなります。

文字列から一部分、例えばこの場合は 2 連続塩基を切り出すには、substr() 関数を使います。substr() は

```
$parts = substr($seq, 0, 2);
```

のように（文字列 , 切り出す場所 , 切り出す文字数）という引数を与えて使います。　さて、ここでハッシュが大変役に立ちます。ハッシュの引数は文字列でいいので、

```
$diobs{$parts} ++;
```

としてやることで、$parts の中身が何であるかを気にせずにカウントすることができます。ここで、あくまでカウントしているのはハッシュの中身でありハッシュではないので、

```
%diobs{$parts} ++;
```

ではなく

```
$diobs{$parts} ++;
```

となっている点に注意してください。

2　O/E 値を求める

2.1　期待値を求める：基本

期待値ってなんでしょう。文字通り「期待される値」です。たとえば dinucleotide の場合何も考えなければ、4 × 4=16 通りあるわけですから各 dinucleotide の期待値は一律 1/16 です。しかし、1-2 で見たように生物は a,t,g,c をそれぞれ均等に使っているわけではありません。したがって、もともと使用頻度が低い塩基（仮に C とします）を含む組合せの期待値も当然低くならなければいけません。

したがって、単に一律 1/16 とするのではなく、各生物ごとの期待値というもので議論する必要があるわけです。

この時、期待値を求める式は以下のようになります。

1 文字目の塩基の頻度 × 2 文字目の塩基の頻度

したがってまず、a,t,g,c を頻度に直すことから考えましょう。頻度は小数で表したいですね。うまく変換して下さい。

2.2　期待値を求める：応用

この部分の計算は

```
$diexp{'aa'} = $percent_a * $percent_a;
$diexp{'at'} = $percent_a * $percent_t;
                    .
                    .
                    .
```

のように、普通に書くと式が 16 本できてしまいます。これをどうにかしてまとめてみましょう。Perl には C や Java にはない foreach 文というものがありますが、これを使えばどうにかなりそうです。

foreach 文は、配列やハッシュの全ての要素を順番に計算します。例えば

```
foreach $content (@array){
  print $content;
}
```

とすれば、配列の各要素が順次 $content の中に入れられるので、次々とその中身が出力されます。ハッシュの場合も同じように

```
foreach $key (sort keys %hash){
  print "$key = " , $hash{$key};
}
```

とすることで毎回ハッシュのキー（引数となる文字列）が $key に入れられます。sort keys というのは、foreach で計算する順序を、キーを ASCII 順に並べた順番にする、という意味です。

それでは、foreach を使って、%diobs の全ての要素を順次とりだしながら、毎回 $key に入力されるキー（ニ連続塩基）の期待値と O/E 値も同時に計算されるようにプログラムを改良してみましょう。

2.3　O/E 値を求める

O/E 値はその名の通り、実測値 / 期待値です。同じ foreach 文の中であらかじめ計算してある期待値と実測値をもとに O/E 値を計算し、これもハッシュに代入しましょう。

3　　出力

ここまで来れば %oe というハッシュに aa から順番に求めるデータが格納されているはずです。後は出力させるだけですね。

「組合せ、O/E 値、実際値、期待値」と 1 行に出せばいいですね。最後の 2 つはおまけです。これも foreach 文の中でやってしまいましょう。

［出力例］

```
     O/E: Observation/Expectation
aa  *.22:        0.*5/0.*2
at  *.77:        *.*9/*.*2
                 .
                 .
                 .
```

前回同様 printf を使って整形します。

```
print ("     O/E: Observation/Expectation\n");
printf ("%s  %.2f:       %.2f/%.2f\n", $key, $oe{$key}, $diobs{$key}, $diexp{$key});
```

ちなみに出力例を出したプログラムは上のようになっています。こういうやり方もあるということを一応知っておいて損はないでしょう。