## 1: Number of global alignments given two $n$ reads

### (a)

Given two sequence with length $n$, there are several different situations:

All $n$ bases in one sequences align to "-", so this means all bases in the other one will also align to "-". So we only need to choose n position from all 2n positions, which is ${}^{2n}C_n$.

Then we discuss if there is only one base from each reads align together. So we should have 2n - 1 positions now and we still need to select n position to host one read. This give us ${}^{2n-1}C_n$. However, in the other read there are n-1 bases need to align to gap, and we need to determine by using ${}^{n}C_{n-1}$. So for this case we finally have ${}^{n}C_{n-1} \cdot {}^{2n-1}C_n$

Then next case is we have two bases from each reads align together. Following the above discussion, so we have ${}^{n}C_{n-2} \cdot {}^{2n-2}C_n$

Considering all cases, so the number of global alignments is given by:

$$ {}^{2n}C_n + {}^{n}C_{n-1} \cdot {}^{2n-1}C_n + {}^{n}C_{n-2} \cdot {}^{2n-2}C_n + ... + {}^{n}C_1 \cdot {}^{n+1}C_n + {}^{n}C_0 \cdot {}^{n}C_n $$

### (b)

For "AG" and "CT":

1. - -AG
   CT- -
2. AG- -
   - -CT
3. A-G-
   -C-T
4. A- -G
   -CT-
5. -A-G
   C-T-
6. -AG-
   C- -T
7. -AG
   CT-
8. AG-
   -CT

9. A-G
   CT-
10. AG-
    C-T
11. A-G
    -CT
12. -AG
    C-T
13. AG
    CT

---

**2: Align with sequence with N**

---

**Algorithm:** Assume we have read a and read b, and read b has Ns in the sequence.

read **a** and **b** with Ns
**score**$[m+1][n+1]$ ▷ Initialize DP score matrix
**N**$[m+1][n+1]$ ▷ matrix to record choice of Ns

$score[0][0] \leftarrow 0$
**for** $i \leftarrow 1$ to $m$ **do**
  $score[i][0] \leftarrow score[i-1][0] - gappenalty$
**end for**
**for** $j \leftarrow 1$ to $n$ **do**
  $score[0][j] \leftarrow score[0][j-1] - gappenalty$
**end for**

**for** $i \leftarrow 1$ to $m$ **do**
  **for** $j \leftarrow 1$ to $n$ **do**
    **if** **b**$[j]$ == N **then**
      **score**$[i][j]$ =
max(**score**$[i-1][j-1] + match$, **score**$[i][j-1] - gappenalty$, **score**$[i-1][j] - gappenalty$)
given N can be A,T,C,G ▷ So there is 4 times 3 options
      **N**$[i][j] = \arg\max$ **score**$[i][j]$ for A, T, C, G
    **else**
      **score**$[i][j]$ is the max of match(mismatch), insertion, and deletion ▷ only 3 options without N
    **end if**
  **end for**
**end for**

$i \leftarrow m+1$, $j \leftarrow n+1$
**while** $i \geq 0$, $j \geq 0$ **do**

i,j ← pointer from **score**$[i][j]$ to the previous cell given the maximum score ▷ backtrack
**if b**$[j]$ == N **then**
**b**$[j] \leftarrow$ **N**$[i][j]$
**end if**
**end while**

**Running time:** Although we need to try A,C,G,T for each N. we only times a small constant to $M \cdot N$. So the running time is still $\mathcal{O}(MN)$

---

**3: Overlap Alignment**

---

**Algorithm:** Assume we have two read w, v. And we want to find overlap between the suffix of v and the prefix of w.

**score**$[m+1][n+1]$ ▷ Initialize DP score matrix

$score[0][0] \leftarrow 0$
**for** $i \leftarrow 1$ to $m$ **do**
$score[i][0] \leftarrow 0$
**end for**
**for** $j \leftarrow 1$ to $n$ **do**
$score[0][j] \leftarrow 0$
**end for**

**for** $i \leftarrow 1$ to $m$ **do**
**for** $j \leftarrow 1$ to $n$ **do**
**score**$[i][j]$ is the max of match(mismatch), insertion, and deletion
**end for**
**end for**

$i \leftarrow$the highest score in the n+1 column (or smallest i if has more than two highest scores),
$j \leftarrow n+1$
**while** $i \geq 0$ and $j \geq 0$ **do**
i,j ← pointer from **score**$[i][j]$ to the previous cell given the maximum score ▷ backtrack
**end while**
the final alignment is the optimal alignment to align the suffix of v to the prefix of w

**Correctness:** There are two major modification compared to the global alignment algorithm.

First, we remove the penalty for the first row and first column so we can have the alignment not start from the beginning of the sequence, then we can align other sequence to suffix.

Second, we start our backtrack from highest score in last column so that we do have optimal overlap alignment.