Explanation of the algorithm computing probability for a given seed - Yufeng Wu

In class, I described the algorithm that computes the probability for a given seed hitting a region. This notes is for a more detailed explanation of this algorithm. Note that since the seeds being used are not very long, so we may be able to just enumerate all possible seeds (with specified length $M$ and the number of cared positions $W$). This allows us to find, for a given region with certain similarity level $p$, the seed that gives us the highest probability of hitting the region. Also note that spaced seeds are known to give higher probability of hitting a region than the consecutive seed. See Keich, et al. for some of these results. So here we focus on the algorithm to compute the probability of a seed $D$ hitting a region $R$. I want to make this clear: we want to compute the probability that the seed hits at least once in the region. We have totally $L - M + 1$ positions where the match can start. If at any of these position we see a match, then we consider the seed hits the region.

Let $L$ be the length of region $R$ (with two sequences $S_1, S_2$). The first step is to reduce $S_1, S_2$ into a binary sequence $S$, where $S[i] = 1$ if $S_1[i] = S_2[i]$, and 0 otherwise. Since the similarity level of region is $p$, $S[i] = 1$ with probability $p$, and $S[i] = 0$ with probability $1 - p$. Since the length of $S$ is $L$ that could be large, we do not want to enumerate all possible $S$. Here is where the dynamic programming helps.

Now define $f[i, b]$ to be the probability that seed $D$ hits the prefix of $S$ with length $i$ such that this prefix ends with a binary string $b$. Here, $i$ stands for the position in $S$ with value from $M$ to $L$, and $b$ is a binary string whose length is fixed to $M$ (same length as the seed $D$). You should draw a picture here to make sure you understand the definition. Again, the seeds can hit $S[1..i]$ anywhere from 1 to $iM + 1$. If we know $f[i, b]$ for all possible $i$ and $b$, then the result we are seeking is simply:

$$\sum_b f[L, b] * \text{prob}(b)$$

where $\text{prob}(b) = (1 - p)^{n_0}p^{n_1}$, and $n_0$ (resp. $n_1$) is the number of 0 (resp. 1) in $b$.

It is easy to initialize $f[M, b]$ (What is it? Think about it.). We consider $i > M$. Two cases.

Case (a). The seed hits the region at position $i - M + 1$. That is, there is a match between the seed $D$ and the rightmost region of length-i prefix of $S$. Specifically, this means for any position $k$, if $D[k] = 1$ (that means we care about this position in seed), then $S[i - M + k] = 1$ (assume we start from array index 1 here). This is because $S[i] = 1$ indicates there is a match at position $i$. What is $f[i, b]$ in this case? It is 1! This is because the seed $D$ already hits the region at $i - M + 1$ and so the seed hits the length-i prefix with probability one. Note that in the above formulation, $b$ is required to have fixed length $M$, so this event can be easily checked with the given seed $D$ and the known $b$.

Case (b). The seed does not hit the region at position $i - M + 1$. In this case, $S[i]$ is not useful (since position $i - M + 1$ is the only possible hit position that might need $S[i]$). This naturally leads to the following recursion:

$$f[i, b] = (1 - p) * f[i - 1, 0b'] + p * f[i - 1, 1b']$$

where $b' = b >> 1$ (right shift by 1, or drop the last bit of $b$). Why is this? In this case, only $S[1..i - 1]$ can lead to a hit. That is exactly what $f[i - 1, b*]$ is for. But since we already fixed the suffix of $b^*$, $b^*$ can only be $0b'$ or $1b'$. The above equation follows from that in $S$, a position takes a value 1 with probability $p$ and a value 0 with probability $1 - p$.

\footnote{Here we do not consider the situations where you have a match from position 1 in $S_1$ and position 3 from $S_2$. That is, we only consider the matches starting at the same position in two strings.}