Lingfu Zhang February 15, 2015

6.046 Problem 2-1

Collaborators: *none*

(a) The input of this problem are two strings S and P, with length n and m, respectively. S contains only a and b, and P contains only a, b and $*$. We need to output a sorted integer listi M, which are indexs j such that the continuing substring of S with length m starting from $S[j]$ matches P, under conditions that $*$ can match either a and b.

Naively we compare every contunuing substring with length m of S, from the one starting from $S[0]$ to the one starting from $S[n-m]$. It's obviously correct, and if we do this in the order, the list M we get is surely sorted.

As there are $n-m+1$ such substrings, and every comparing contains at most m compares between characters. The total time is $O(mn)$

(b) Same problem as (a).

We first convert S and P to polynomials:

$$S(x) = \sum_{i=0}^{n-1} s_i x^i$$

where $s_i = 1$ if $S[i] =$ 'a', and $s_i = \imath$ if $S[i] =$ 'b'.

$$P(x) = \sum_{i=0}^{m-1} p_i x^i$$

where $p_i = 1$ if $P[m-1-i] =$ 'a', $p_i = \imath$ if $P[m-1-i] =$ 'b', and $p_i = 0$ if $P[m-1-i] =$ '$*$'. Then compute $R(x) = P(x)S(x)$. For i from 0 to $n-m$, if the coefficient of x^{i+m-1} in R has no imaginary part, add i to M.

The proof is simple. First, we can have

$$R(x) = \sum_{i=0}^{n+m-2} \sum_{j=0}^{m-1} s_{i-j} p_j x^i$$

Here we assume that $s_k = 0$ for $k < 0$ or $k > n-1$. Then $s_{i-j}p_j = \imath$ if and only if $S[i-j]$ and $P[m-1-j]$ are exactly 'a' and 'b' or 'b' and 'a', which indicates that P doesn't match the continuing substring of S with length m and starting from $S[i-m+1]$. On the other side, if $\sum_{j=0}^{m-1} s_{i-j}p_j$ has no imaginary part, none of $s_{i-j}p_j$ is $\imath$, and the two strings match. As we chack the coefficients of R one by one, M must be sorted.

Computing $R(x) = S(x)P(x)$ takes $O(mn)$ time, other operations all take linear time. Thus this algorithm takes $O(mn)$ time.

For the example given, we have

$$S(x) = 1 + \imath x + x^2 + \imath x^3 + \imath x^4 + x^5 + \imath x^6$$

$$P(x) = \imath x + x^2$$

Then we can get

$$R(x) = S(x)P(x) = \imath x + 2\imath x^3 + (-1-\imath)x^5 + 2\imath x^6 + \imath x^8$$

Examing coefficients of x^2, x^3, x^4, x^5 and x^6, we find only the ones of x^2 and x^4 have no imaginary part. Then $M = \{0, 2\}$.

(c) Let k be the least 2 power no less than $m+n$. Then using FFT we need to convert $S(x)$ and $P(x)$ to samples, computing $R(x) = S(x)P(x)$ and converting $R(x)$ back. The whole process needs to treat $R(x)$, $P(x)$ and $S(x)$ as polinomials with degree k. Then $O(k \log k) = O((m+n)\log(m+n))$ time is needed. Considering $m \ll n$, the time taken is $O(n \log n)$.

(d) The problem is basically the same as (a), except that characters in D and P can be A, G, C,T and $*$ for P.

This time we make two pairs of polynomials:

$$D'(x) = \sum_{i=0}^{n-1} d'_i x^i$$

where $d'_i = 1$ if $D[i] =$ 'A' or 'G', and $d'_i = 1$ if $D[i] =$ 'C' or 'T'.

$$D''(x) = \sum_{i=0}^{n-1} s''_i x^i$$

where $d'_i = 1$ if $D[i] =$ 'A' or 'C', and $d'_i = 1$ if $D[i] =$ 'G' or 'T'.

$$P'(x) = \sum_{i=0}^{m-1} p'_i x^i$$

where $p'_i = 1$ if $P[m-1-i] =$ 'A' or 'G', $p'_i = 1$ if $P[m-1-i] =$ 'C' or 'T', and $p'_i = 0$ if $P[m-1-i] =$ '$*$'.

$$P''(x) = \sum_{i=0}^{m-1} p''_i x^i$$

where $p''_i = 1$ if $P[m-1-i] =$ 'A' or 'C', $p''_i = 1$ if $P[m-1-i] =$ 'G' or 'T', and $p''_i = 0$ if $P[m-1-i] =$ '$*$'.

Then compute $R'(x) = P'(x)S'(x)$ and $R''(x) = P''(x)S''(x)$.

Similiarly, we need to add i to M if coefficients of x^{i+m-1} in both R' and R'' has no imaginary part.

To prove that this algorithm works, we first see that if $P[i]$ and $D[j]$ doesn't match, at least one of $p'_{m-1-i}d'_j$ and $p''_{m-1-i}d''_j$ is 1. Then similiar to (b),

P matches the continuing substring of D with length m and starting from $D[i]$ if and only if the coefficients of x^{i+m-1} in both $R'(x)$ and $R''(x)$ has no imaginary part. As we chack the coefficients of R' and R'' one by one, M must be sorted.

The time used here is twice as much as that in (b), thus is $O(mn)$ (not using FFT) or $O(n \log n)$ (using FFT).