Evan Gary - HW3 parts a and b

Java uses a variation of following function to calculate hashes for strings:

s[0]\*P^(n-1) + s[1]\*P^(n-2)+…+s[n-1]

s[i] represents the integer representation of a character, where i is the index of the character in the string. P is any prime number larger than 2. Java’s function uses 31. My program will use 5 to avoid overflow. n is the number of characters in the string.

Part a uses three methods for calculating hash values, the first two are essentially the same thing, and the third is an application of Method 2.

**(Part a)**

**Method 1:**

To stay consistent with the homework notation, rewrite using A0 in place of s[0]

H = A0P^(n-1) + A1P^(n-2) + … + An-1

Let’s take n=4 and make the formula for the hash function:

H4 = A0P^3 + A1P^2 + A2P + A3

**Method 2:**

We can calculate this value a much simpler (in implementation) way by starting with the hash of a single character string:

H1 = A0

For the two character string, all we need to do is multiply H1 by P and add A1:

H2 = P\*H0 + A1 = PA0 + A1

Repeat for H3:

H3 = P\*H1 + A2 = P\*(PA0 + A1) + A2 = A0P^2 + A1P + A2

Repeat for H4:

H4 = P\*H2 + A3 = P\*(A0P^2 + A1P + A2 ) = A0P^3 + A1P^2 + A2P + A3

Which is the same as the H4 calculated above with the formula.

**Method 3:**

Using either method with n = k-1 calculates the hash value for AiAi+1Ai+2…Ai+k-1:

H = AiP^(k-1-1) + Ai+1P^(k-1-2) + … + Ai+k-1

Where k is the length of the string fragment. We now need to find the hash value for Ai+1Ai+2…Ai+k

Notice that Ai+k immediately follows AiAi+1Ai+2…Ai+k-1

So to calculate AiAi+1Ai+2…Ai+k we only need to multiply by P and add Ai+k (Method 2):

(AiP^(k-1-1) + Ai+1P^(k-1-2) + … + Ai+k-1) \* P + Ai+k = AiP^(k-1) + Ai+1P^(k-2) + … + Ai+k-1P+ Ai+k

Then, calculating Ai+1Ai+2…Ai+k  is simply a matter of subtracting the Ai term, AiP^(k-1)

Once the initial AiAi+1Ai+2…Ai+k-1  is known, there are only three operations needed to get to AiAi+1Ai+2…Ai+k

Multiply by P

Add AI+k

Subtract Ai term

Therefore, the algorithm works in constant time.

**(Part b)**

Now we are considering two strings:

P1P2P3…Pk  and A1A2A3…AN

Now, the calculation of both P1P2P3…Pk and A1A2A3…Ak are done using Method 2. Each of these calculations is O(k) since they depend on the length of the pattern string. From there, the remaining A substrings can be calculated using Method 3. The process is repeated N-k times. Therefore, the running time is 2k + N – k, or O(N + k)