15-451 Assignment 08

Karan Sikka
ksikka@cmu.edu
Collaborated with Dave Cummings and Sandeep Rao.
Recitation: A
November 13, 2014

1: Streaming Medians

(a)

(b)

(c)

(d)

2: Counting Substrings

3: LDIS

(a)

Choosing a random prime: Let $|\Sigma|$ be the size of the alphabet. Note that it is upper-bounded by a constant.

Turn the string into a binary string by replacing each character with a binary string of p bits where $p = lg(|\Sigma|)$.

If the size of the original string was T, it is now t where $t = Tlg(|\Sigma|)$ which is in the order of T since log sigma is upper bounded by a constant.

Choose a random prime between 1 and $K = 5 * lg(\Sigma) * n * ln(lg(\Sigma)n)$.

You can do this by picking a random integer, checking if it's prime,
and trying again if not. This algorithm is expected $O(log(K))$ which is $O(log(p) + log(t) + log(log(p) + log(t))$ which is in $O(log(t))$

Modified Karp-Rabin: Now we will compute the Karp-Rabin hashes of each prefix and the string of the same length following it.
Starting at i = 1, compute h(s[0:i]), h(s[i:2i])
Increment i and compute the hashes again. Note that this takes constant time since $s[0 : i] \implies s[0 : i + 1]$ and $s[i : 2i] \implies s[i + 1 : 2i + 2]$ so to compute the hash we only have to do a constant time adjustment to the previously computed hash.

Repeat until $i > t/2$. Return the max i for which the two hash computed at the ith iteration are equal.

Karan Sikka
ksikka@cmu.edu

Proof that Pr[false positive] <1/2: Next we will show the probability of a false positive is less than 1/2.

Let the length of the string be n.
Suppose for any fixed locations i and $2i$ the probability of an incorrect match is upper-bounded by some δ.
Then by summing over $n/2$ locations the probability of any incorrect match is at most $n/2 * \delta$.
We want $n/2 * \delta < 1/2$ or equivalently $\delta < 1/n$.

We make an incorrect match when the hash a of one substring is equal to the hash b of another, modulo some random prime q.
Formally, this occurs when $q|a-b$.
A string like a or b is at most $n/2$ characters which is represented in binary with at most $lg(\Sigma) * n/2$ bits. Let $p = lg(\Sigma) * n/2$.

Then $a-b$ has at most p distinct prime divisors since it's a p-bit number and each prime divisor is at least 2.
In the case of a false answer, $q|a-b$, then q must have been one of the p prime divisors of $a-b$.

We want to choose a prime such that the chance that it's one of the p prime divisors is less than $1/n$, so that the total probability of failure is less than $n/2 * 1/n$ which is less than 1/2.
Equivalently we want to choose K large enough so that there are at least pn primes between 2 and K.

If there are $\pi(x)$ primes between one and x, then $\pi(x) \geq \frac{7}{8}\frac{n}{ln(n)}$. Thus we want to choose K such that $\pi(K) \geq \frac{7}{8}\frac{K}{ln(K)} > pt$.

Setting $K = 5 * lg(\Sigma) * n * ln(lg(\Sigma)n)$ acheives this result.

(b)

Construct a suffix tree for s in linear time.

We will preprocess the suffix tree to store at each node, the following value:
Let p be the pattern represented by the node.
Let L be the length of the pattern.
Store the index of the last occurence of p in the string, which starts on or before index L.

How we do the above is explained later.

Then, the algorithm proceeds as follows:

Do a DFS where you keep track of the length of the prefix at a node, called L, cache the index of the last occurence of P starting on or before index L+1.

Ask your children, what's your last occurence, and since your L+1 is strictly greater than my L+1, I'll filter out the invalid answers and still have the correct last occurence before L+1.

Traverse the suffix tree starting from the root, taking the edge which leads you to a node representing a pattern which is a prefix of the original string. At each node, keep track of the length of the prefix, and check whether the length of the value we computed earlier is equal to L. Keep a running max length of the prefix for which this condition is true.

TODO Correctness TODO details of how to do the precomputation