Homework 6, additional problems, solution set

1. Each string will be stored in a 2–3 tree. The leaves will store the string characters in the order they occur in the string, i.e. the contents of the leaves in left to right order is exactly the string. For each subtree, we keep a count of the number of leaves in the subtree, storing this value at the parent. As it happens there will be no need for guides in these 2–3 trees. Now we show how to support the various desired operations.

Cut(σ, i, j). σ is the string being cut.
We split the 2–3 tree storing σ into three pieces, $\sigma_\ell, \sigma_{mid}, \sigma_r$, comprising the first $i-1$ characters, the next $j-i+1$ characters, and the remaining characters. So the middle pieces consists of the ith through the jth characters in σ. We do this by splitting the 2–3 tree at the ith character, and the remainder at its $j-i+1$th character. σ_{mid} is the cut. We then join σ_ℓ and σ_r, forming the cut version of σ. Each of the join and cut operations takes $O(\log n)$ time. Thus the cut also takes $O(\log n)$ time.

Paste(τ, σ, k). This operation pastes σ after the kth character in τ.
We split the 2–3 tree storing τ into two pieces τ_ℓ and τ_r at the kth character in τ. We then join τ_ℓ and σ, and join the resulting string with τ_r, thereby forming the string $\tau_\ell\sigma\tau_r$. Again, each of the join and cut operations takes $O(\log n)$ time. Thus the paste also takes $O(\log n)$ time.

2.a.i. We store T using a Bloom filter. This will use an additional $O(|T|) = O(n)$ bits of space. Then, in turn, we test each item in S for membership in T. If an item is in T the answer will be a correct "yes". If the item is not in T, there is at most a 2% probability that it will be reported at being in T. So if $S \not\subseteq T$, the probability that it is reported as being a subset of T is at most 2%. This procedure runs in worst-case time $O(|S| + |T|)$, i.e. in linear time.

ii. Suppose we store T using a Bloom filter and we then test each item of S for membership in T. The problem is that for each item not in T, there is a small constant probability p that it is reported at being in T. The probability that no item in S is reported as being in T (assuming the sets are disjoint) is at most $(1-p)^{|S|} \approx e^{-p|S|}$, and this is very small for large $|S|$; i.e. the sets are almost always reported as being non-disjoint.

b.i. Here we store T using a hash table. We then test for each item in S whether it is in T in expected $O(1)$ time. This takes a total of expected $O(|S| + |T|)$ time and reports the correct answer.
ii. The same solution applies to this part.