Tutorial 10 Exercises

COMP526: Efficient Algorithms

09–10 December, 2024

Exercise 1. In our lectures on parallel algorithms, we saw a PRAM algorithm that solves string matching for searching for a pattern $P[0..m)$ in a text $T[0..n)$ with span $\Theta(m)$ and work $\Theta(n)$. The output of this algorithm, however, was different from the original setting of pattern matching we discussed earlier in the semester. In particular, the output of a parallel algorithm was an array $M[0..n)$ such that $M[i] = 1$ if T contains a match to P at index i and $M[i] = 0$ otherwise.

(a) Devise a PRAM algorithm that modifies the array M such that after applying your algorithm, $M[n-1]$ stores the total number of matches of P in T. The span of your procedure should be $O(\log n)$ and its work should be $O(n)$.
(Hint: try a divide and conquer approach.)

(b) Explain how your procedure from part (a) can be modified (or extended) to produce the index of the first instance of P in T (assuming there is a match). The span and work of the updated procedure should be (asymptotically) no worse than your first procedure.

For simplicity, you may assume that n, the length of the text, is a power of 2, say $n = 2^k$.

Exercise 2. Consider the text $T =$ `abbabbaa$`. What is n here? (Exactly follow the convention from the lecture!) Construct/draw the

(a) standard (not compacted) trie of all suffixes of T,

(b) suffix tree of T (human version) with string labels on edges and leaves,

(c) suffix tree of T (computer version) as it is stored, i.e., offsets in nodes, starting index in leaves, first characters on edges.