CS648A : Randomized Algorithms

Department of Computer Science and Engineering, IIT Kanpur

Randomized algorithm for 2-dimensional pattern matching

Sketch of the solution is given on the last page. Use it only after making sincere attempt(s).

In one of the lectures, we discussed a randomized algorithm for 1-dimensional pattern matching using fingerprinting technique. Our aim is to design a randomized algorithm for 2-dimensional pattern matching which was also described in the class. For this purpose, first we shall discuss a different finger printing technique to solve 1-dimensional pattern matching problem. The idea is to map any bit string s into a 2×2 matrix $M(s)$, as follows.

- For the empty string ϵ, $M(\epsilon) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$
- $M(0) = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$
- $M(1) = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$
- For non-empty strings x and y, $M(xy) = M(x) \times M(y)$.

Convince yourself that this fingerprint function has the following properties.

- $M(x)$ is well defined for all $x \in \{0,1\}^*$.
- $M(x) = M(y) \Rightarrow x = y$.

Starting with the two properties of M listed above, solve the following problems.

1. By using the matrix $M(x)$ for a bit-string x, design an algorithm for the pattern matching problem in 1-dimension (text and pattern are bit-strings of length n and m respectively). The algorithm should take $O(n+m)$ time. You may make use of the following assumption.
   Assumption: Any arithmetic operation involving two numbers of arbitrary length can be executed in $O(1)$ time.
2. It can be shown that for a bit string x of length n, entries of $M(x)$ can be quite long; however, they will be bounded by Fibonacci number F_n. Unfortunately, this fact suggests that the assumption mentioned above is not practical. In order to circumvent this problem, we need to work with small numbers - numbers of $O(\log n)$ bits only. Recall that the word-RAM model of computation which is very realistic requires this condition Therefore, taking inspiration from one of the lectures, we keep entries of M modulo a prime number p picked randomly uniformly from a suitable range. This is now a practical but randomized Monte Carlo algorithm for 1-dimensional pattern matching. Do proper analysis to find the range from which you need to pick the prime number randomly to achieve error probability $< 1/n^4$.
   (*Most of you might be finding this pattern matching algorithm more complicated than the one we discussed in the class. However, this algorithm has the merit that it can be extended to solve 2-dimensional pattern matching very easily. You will do this extension as the last part of this problem.*)
3. Consider the two-dimensional version of the pattern matching problem. The text is an $n \times n$ bit-matrix X, and the pattern is an $m \times m$ bit-matrix Y. Of course, $m \leq n$. A pattern match occurs if Y appears as a (contiguous) sub-matrix of X. Get inspired from the randomized algorithm for part (b) above to design a randomized Monte Carlo algorithm for this 2-dimensional pattern matching problem. The running time should be $O(n^2)$ and the error probability should be $< 1/n^4$.

Sketch:
In order to apply the randomized algorithm described in step 2, we convert the matrix Y into an m^2-bit vector using the row-major format. The possible occurrences of Y in X are the m^2-bit vectors $X(j)$ obtained by taking all $(n-m+1)^2$ sub-matrices of X in a row-major form. Observe that the earlier algorithm can now be applied to this scenario. Analyze the error probability in this case, and explain how the fingerprints of each $X(j)$ can be computed at a small incremental cost.