DM582 Solutions

Mads Anker Nielsen
`madsn20@student.sdu.dk`

April 14, 2024

This document contains written solution to exercise problems from the course DM582 (spring 2024). The solutions given here might differ from the solutions discussed in class. In class, we place more emphasis on the intuition leading to the correct answer. Please do not consider reading these solutions an alternative to attending the exercise classes.

References to CLRS refer to the book *Introduction to Algorithms, 4th edition* by Cormen, Leiserson, Rivest, and Stein.

References to KT refer to the book *Algorithm Design, 1st edition* by J. Kleinberg and E. Tardos.

References to Rosen refer to the book *Discrete Mathematics and its Applications, 8th edition* by K. Rosen.

This document will inevitably contain mistakes. If you find some, please report them to me (Mads) so that I can correct them.

Sheet 8

CLRS, 32.1-1

Exercise

Show the comparisons the naive string matcher makes for the pattern 0001 in the text 000010001010001.

Suggested solution

We did this on the blackboard in class.

CLRS, 32.1-2

Exercise

Suppose that all characters in the pattern are different. Show how to accelerate `NAIVE-STRING-MATCHER` to run in time $O(n)$ on n-character text T.

Suggested solution

When we check for an occurrence at shift s and find a mismatch, say when comparing with $P[i]$, we can skip directly to checking shift $s + i$ instead of shift $s + 1$. Indeed, if some i-length prefix $P[1 : i]$, $i \in [m]$, occurs with shift s in T, then $T[s + j] = P[j] \neq P[1]$ for $j = 2, 3, \ldots, i$ since all characters are distinct.

This approach ensures that every character in T participates in at most 2 comparisons (possibly 2 for the characters that cause a mismatch) and thus the runtime is $O(n)$.

CLRS, 32.1-3

Exercise

Suppose that pattern and text are randomly chosen strings of length m and n respectively, from the d-ary alphabet $\Sigma_d = \{0, 1, \ldots, d-1\}$, where that the expected number Show that the expected number of character-to-character comparisons made by the implicit loop in line 2 of the naive algorithm is

$$(n-m+1)\frac{1-d^{-m}}{1-d^{-1}} \leq 2(n-m+1)$$

over all executions of this loop. (Assume that the naive algorithm stops comparing characters for a given shift once it finds a mismatch or matches the entire pattern.) Thus, for randomly chosen strings, the naive algorithm is quite efficient.

Suggested solution

Let X be a random variable whose value is the number of comparisons made. We can decompose X as $X = \sum_{s=0}^{n-m} X_s$ where X_s is a random variable whose value is the number of comparisons made when checking for an occurrence at shift s. We can argue that $X_s = \frac{1-d^{-m}}{1-d^{-1}}$ directly, or explicitly decompose it into indicator random variables, which we do here. X_s can be expressed as $X_s = \sum_{i=1}^{m} X_{s,i}$ where $X_{s,i}$ is an indicator random variable for the event that the i-th iteration of the loop is reached when checking for an occurrence of the pattern at shift s. The event $P[i] = T[s+i]$ occurs independently and with probability $1/d$ for all $i \in [m]$, and we proceed to the next iteration iff $P[i] = T[s+i]$. Thus, the probability that the i-th iteration is reached is $(1/d)^{i-1}$ and hence $\mathrm{E}[X_{s,i}] = (1/d)^{i-1}$ for $s = 0, 1, \ldots, n-m, i = 1, 2, \ldots, m$.

We now get

$$\begin{aligned}
\mathrm{E}[X_s] &= \sum_{i=1}^{m} \mathrm{E}[X_{s,i}] \\
&= \sum_{i=1}^{m} \left(\frac{1}{d}\right)^{i-1} \\
&= \sum_{i=0}^{m-1} \left(\frac{1}{d}\right)^{i} \\
&= \frac{(1/d)^m - 1}{1/d - 1} \\
&= \frac{1-d^{-m}}{1-d^{-1}} \leq \frac{1}{1-d^{-1}} \leq 2
\end{aligned}$$

where the fourth equality follows from the general formula for the sum of a truncated geometric series (see e.g. Rosen p. 176). Now,

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{s=0}^{n-m} \mathrm{E}[X_s] \\
&\leq \sum_{s=0}^{n-m} 2 \\
&= 2(n-m+1)
\end{aligned}$$

as desired.

CLRS, 32.1-4

Exercise

Suppose we allow the pattern P to contain occurrences of a gap character $\diamond$ that can match an arbitrary string of characters (even one of zero length). For example,

Example omitted. See CRLS page 962.

The gap character may occur an arbitrary number of times in the pattern but not at all in the text. Give a polynomial-time algorithm to determine whether such a pattern P occurs in a given text T, and analyze the running time of your algorithm.

Suggested solution

We observe that an occurrence of such a pattern P in a text T corresponds to a sequence of matches of the substrings of P between the gap characters.

Formally, let $P_1, P_2, \ldots, P_k$ be substrings of P not containing $\diamond$ such that $P = P_1 \diamond P_2 \diamond \cdots \diamond P_k$ and let T be the text.

If $k = 0$ then return true immediately. Otherwise, find the first occurrence of P_1 in T. If no such occurrence exists, then return false. Otherwise, let s_1 be the shift of the first occurrence of P_1 in T and recursively find $P_2, P_3, \ldots, P_k$ in $T[s_1 + |P_1| + 1 :]$ and return the result.

The algorithm degenerates to the naive string matching algorithm if P does not contain any gap characters and otherwise the runtime only improves as we are simply solving a sequence of non-overlapping subproblems of smaller size.

CLRS, 32.2-1

Exercise

Working modulo 11, how many spurious hits does the Rabin-Karp matcher encounter in the text 3141592653589793 when looking for the pattern $P = 26$?

Suggested solution

We have $26 \mod 11 = 4$. Below we list the result of $x \mod 11$ for every length 2 substring of the text.

- $31 \mod 11 = 9$. No hit.
- $14 \mod 11 = 3$. No hit.
- $41 \mod 11 = 8$. No hit.
- $15 \mod 11 = 4$. Spurious hit.
- $59 \mod 11 = 4$. Spurious hit.
- $92 \mod 11 = 4$. Spurious hit.
- $26 \mod 11 = 4$. Hit.
- $65 \mod 11 = 10$. No hit.
- $53 \mod 11 = 9$. No hit.
- $35 \mod 11 = 2$. No hit.
- $58 \mod 11 = 3$. No hit.
- $89 \mod 11 = 1$. No hit.
- $97 \mod 11 = 9$. No hit.
- $79 \mod 11 = 2$. No hit.
- $93 \mod 11 = 5$. No hit.

3 of the hits are spurious.

CLRS, 32.2-4

Exercise

Alice has a copy of a long n-bit file $A = (a_{n-1}, A_{n-2}, ..., A_0)$, and Bob similarly has an n-bit file $B = (b_{n-1}, b_{n-2}, ..., b_0)$. Alice and Bob wish to know if their files are identical. To avoid transmitting all of A or B, they use the following fast probabilistic check. Together, they select a prime $q > 1000n$ and randomly select an integer x from $\{0, 1, ..., q-1\}$. Letting

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \mod q \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i \mod q,$$

Alice evaluates $A(x)$ and Bob evaluates $B(x)$. Prove that if $A \neq B$, there is at most one chance in 1000 that $A(x) = B(x)$, whereas if the two files are the same, $A(x)$ is necessarily the same as $B(x)$. (Hint: See Exercise 31.4-4.)

Suggested solution

We start by proving the claim from Exercise 31.4-4 that a polynomial of degree t has at most t distinct zeros modulo q. We assume that it has already been proven that a polynomial f of degree t can be written as $f(x) = (x-a)g(x)$ whenever a is a zero of f.

We prove the claim by induction on t. For $t = 1$, the polynomial f has the form $f(x) = ax + b$ for some constants a and b. The zeros of f are the solutions to the modular congruence $ax \equiv -b \mod q$, which has a unique solution since q is prime. Suppose that the claim holds for $t > 1$ and let f be a polynomial of degree $t+1$. If f has no zeros then we are done and otherwise let a be a zero of f. Then $f(x) = (x-a)g(x)$ for some polynomial g of degree t. By the induction hypothesis, g has at most t zeros modulo q. Any zero of f is either a or a zero of g and thus f has at most $t+1$ zeros modulo q.

We now prove the claim of this exercise. If $A(x) \equiv B(x) \mod q$ then

$$\sum_{i=0}^{n-1} (a_i - b_i) x^i \equiv 0 \mod q.$$

The polynomial $f(x) = \sum_{i=0}^{n-1} (a_i - b_i) x^i$ has degree at most $n-1$ and thus has at most $n-1$ zeros modulo q. Thus, the probability that the randomly chosen x is a zero of f is at most $\frac{n-1}{q} < \frac{1}{1000}$. Furthermore, if $A = B$ then clearly $A(x) = B(x)$ (without the modulus) and thus also $A(x) \equiv B(x) \mod q$.

CLRS, 32.3-1

Exercise

Draw a state-transition diagram for the string-matching automaton for the pattern $P = aabab$ over the alphabet $\Sigma = \{a, b\}$ and illustrate its operation on the text string *aaababaabaababaab*.

Suggested solution

See figure 1 for a state-transition diagram of the string-matching automaton. We demonstrated the operation of this automaton in class. It operates just like any other DFA.

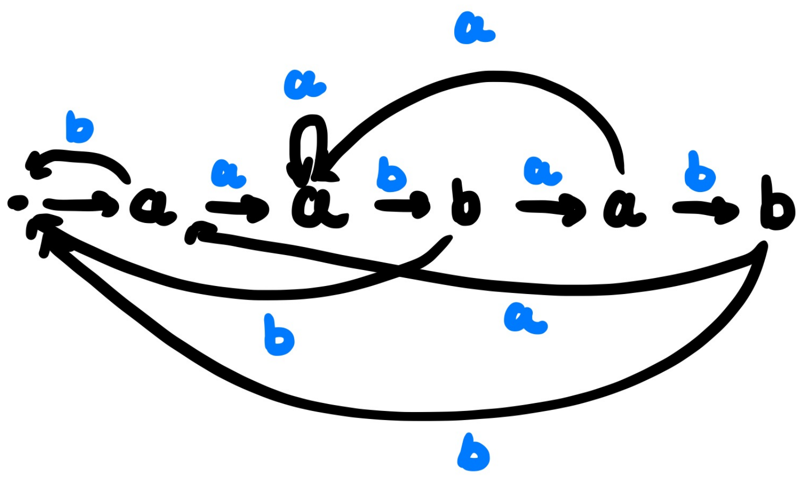

Figure 1