DM582 Solutions

Mads Anker Nielsen
madsn20@student.sdu.dk

May 2, 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 9

CLRS Exercise 32.3-2

Exercise

Draw a state-transition diagram for the string-matching automaton for the pattern *ababbabbababbababbabb* over the alphabet $\Sigma = \{a, b\}$.

Suggested solution

See Figure 1.

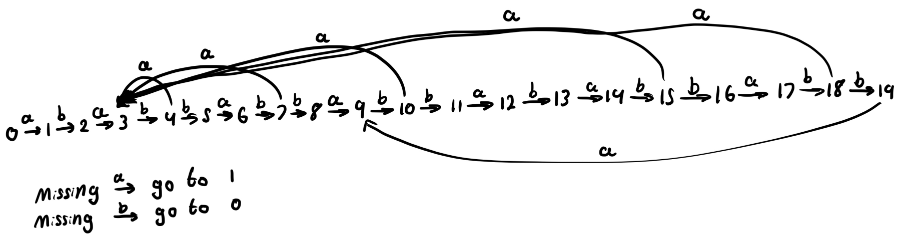

Figure 1

CLRS Exercise 32.4-1

Exercise

Compute the prefix function for the pattern $ababbabbabbababbabb$.

Suggested solution

$\pi[i]$ is the largest $k < i$ such that $P[:k]$ is a suffix of $P[:i]$. It is simple to compute by brute force or by using the algorithm `COMPUTE-PREFIX-FUNCTION` from CLRS. We get the following values:

- $\pi[1] = 0$, since the only $k < i$ is 0.
- $\pi[2] = 0$, since $P[:1] = a$ is not a suffix of $P[:2] = ab$.
- $\pi[3] = 1$, since $P[:1] = a$ is a suffix of $P[:3] = aba$.
- $\pi[4] = 2$, since $P[:2] = ab$ is a suffix of $P[:4] = abab$.
- $\pi[5] = 0$ $(\dots)$
- $\pi[6] = 1$
- $\pi[7] = 2$
- $\pi[8] = 0$
- $\pi[9] = 1$
- $\pi[10] = 2$
- $\pi[11] = 0$
- $\pi[12] = 1$
- $\pi[13] = 2$
- $\pi[14] = 3$
- $\pi[15] = 4$
- $\pi[16] = 5$
- $\pi[17] = 6$
- $\pi[18] = 7$
- $\pi[19] = 8$

CLRS Exercise 32.4-3

Exercise

Explain how to determine the occurrences of pattern P the text T by examining the π function for the string PT (the string of length $m+n$ that is the concatenation of P and T).

Suggested solution

We start by noticing that if $\pi[s+m] = m$, then $PT[:m] = P$ is a (proper) suffix of $PT[:s+m]$ by the definition of π. Thus, P occurs in PT with shift s if $\pi[s+m] = m$. However, this condition is not necessary. Consider e.g. the following example $P = ab$ and $T = abab$. Then $\pi[6] = 4$ and yet P occurs with shift $6-2$ in PT.

Instead, we use the function π^* from Lemma 32.5 of page 980. We can compute π^* only by examining π. Lemma 32.5 states that $\pi^*[q] = \{k | k < q \text{ and } P[:k] \text{ suffix of } P[:q]\}$. Thus, $PT[:m] = P$ is a suffix of $PT[:q]$ (and hence occurs with shift $q-m$) for some $q > m$ iff $m \in \pi^*[q]$. We should discard shifts less than m to avoid conting occurences that overlap with P.

CLRS Exercise 32.4-5

Exercise

Use a potential function to show that the running time of `KMP-MATCHER` is $\theta(n)$. (*Algorithm shown below*).

```
KMP-MATCHER(T, P, n, m)
1  π = COMPUTE-PREFIX-FUNCTION(P, m)
2  q = 0                          // number of characters matched
3  for i = 1 to n                 // scan the text from left to right
4      while q > 0 and P[q + 1] ≠ T[i]
5          q = π[q]               // next character does not match
6      if P[q + 1] == T[i]
7          q = q + 1              // next character matches
8      if q == m                  // is all of P matched?
9          print "Pattern occurs with shift" i − m
10         q = π[q]               // look for the next match

COMPUTE-PREFIX-FUNCTION(P, m)
1  let π[1 : m] be a new array
2  π[1] = 0
3  k = 0
4  for q = 2 to m
5      while k > 0 and P[k + 1] ≠ P[q]
6          k = π[k]
7      if P[k + 1] == P[q]
8          k = k + 1
9      π[q] = k
10 return π
```

Suggested solution

We show that each iteration of the for loop takes around constant time, from which is follows that the total running time is $\theta(n)$.

Let $\Phi_i = q$ to be the potential at the beginning of the i-th iteration of the for loop. All operations done outside the while loop on line 4-5 takes constant time, so we may suppose that one unit of potential is enough to pay for all these operations. Suppose the while loop on line 4 is executed k times in the ith iteration. The amortized cost of the ith iteration is then

$$\hat{c}_i = 1 + k + \Phi_i - \Phi_{i-1}.$$

We observe that, by definition, $\pi[q] < q$ for all q. Thus, the potential decreases by at least 1 in each iteration of the while loop. The potential only increases by 1 from the operation $q = q + 1$ on line 6, so we have $\Phi_i - \Phi_{i-1} \leq -k + 1$ implying

$$\hat{c}_i \leq 2 \in O(1)$$

which is what we wanted to show.

CLRS Exercise 32.4-6

Exercise

Show how to improve KMP-MATCHER by replacing the occurrence of in line 5 by (but not line 10) by π', where π is defined recursively for $q = 1, 2, \ldots, m-1$ by the equation

$$\pi'[q] = \begin{cases} 0 & \text{if } \pi[q] = 0, \\ \pi'[\pi[q]] & \text{if } \pi[q] \neq 0 \text{ and } P[\pi[q]+1] = P[q+1], \\ \pi[q] & \text{if } \pi[q] \neq 0 \text{ and } P[\pi[q]+1] \neq P[q+1]. \end{cases}$$

Explain why the modified algorithm is correct, and explain in what sense this change constitutes an improvement.

Suggested solution

Suppose line 5 is being executed in the unmodified algorithm. If $\pi[q] \neq 0$ and $P[\pi[q]+1] = P[q+1]$, then the loop will execute again since $P[q+1] \neq T[i]$ and thus $P[\pi[q]+1] \neq T[i]$. Thus, assigning $q = \pi[\pi[q]]$ would not change the behavior of the algorithm in this case. The idea behind π' is to take advantage of this to save some iterations of the while loop.

Formally, we argue that assigning $q = \pi'[q]$ eventually results the variable q eventually (after the while loop) taking the same value as if we had assigned $q = \pi[q]$.

We argue by induction on q. For $q = 1$ we have $\pi'[1] = 0 = \pi[1]$ and the loop terminates immediately. Let $q > 1$. In cases 1 and 3 of the definition of π', we have $\pi'[q] = \pi[q] < q$, so the claim holds by the induction hypothesis. In case 2, we have argued that assigning $q = \pi[\pi[q]]$ does not change the value that q eventually takes. Since, $\pi[q] < q$, assigning $q = \pi'[\pi[q]]$ eventually results in q taking the same value as if we had assigned $q = \pi[\pi[q]]$ by the induction hypothesis.

CLRS Exercise 32.4-7

Exercise

Give a linear-time algorithm to determine whether a text T is a cyclic rotation of another string T'. For example, `braze` and `zebra` are cyclic rotations of each other.

Suggested solution

First check if T and T' have the same length and return false immediately if not. The $|T'|$-length substring of $T'T'$ are exactly the cyclic rotations of T', so T is a cyclic rotation of T' if and only if T is a substring of $T'T'$. We can construct $T'T'$ and determine whether T occurs as a substring in linear time by using KMP.

CLRS Exercise 32.4-8

Exercise

Give an $O(m|\Sigma|)$-time algorithm for computing the transition function δ for the string-matching automaton corresponding to a given pattern P. (Hint: Prove that $\delta(q, a) = \delta(\pi[q], a)$ if $q = m$ or $P[q + 1] \neq a$.)

Suggested solution

We start by recalling that, by the definition of δ, $\delta(q, a) = q+1$ if $P[q+1] = a$ and $q < m$. Otherwise, either $q = m$ or $P[q+1] \neq a$. Suppose the claim from the hint holds. Then we can compute $\delta(q, a)$ by setting $\delta(q, a) = \delta(\pi[q], a)$ if $q = m$ or $P[q + 1] \neq a$ and $\delta(q, a) = q + 1$ otherwise. Since $\pi[q] < q$, if we compute $\delta(q, a)$ in order of increasing q, then for each $a \in \Sigma$ and $q \in \{0, 1, \ldots, m\}$, we can compute $\delta(q, a)$ in constant time: Check if $P[q + 1] = a$ and $q < m$ and set $\delta(q, a) = q + 1$ and otherwise assigning to $\delta(q, a)$ the previously computed $\delta(\pi[q], a)$ (this is standard dynamic programming). Thus, computing δ via this approach takes $O(m|\Sigma|)$ time.

Note: the following proof is quite notation heavy and does not provide much intuition as to why this claim holds. Drawing the state transition diagram and arguing on basis of that, the argument is intuitively more clear (but not so nice to write down formally).

We have left to prove the claim from the hint. Recall that $\delta(q, a) = \sigma(P[: q]a)$ where $\sigma(X)$ is the largest k such that $P[: k]$ is a suffix of X.

Since $q = m$ or $P[q + 1] \neq a$, we have $\sigma(P[: q]a) \leq q$. Let $k = \sigma(P[: q]a) \leq q$. We start by showing $\sigma(P[: \pi[q]]a) \geq k$. Indeed, $P[: k - 1]$ is prefix which is a proper suffix of $P[: q]$ and $P[: \pi[q]]$ is the longest prefix which is a proper suffix of $P[: q]$. Thus, $P[k - 1]a = P[: k]$ is a suffix of $P[: \pi[q]]a$ and hence $k = \sigma(P[: q]a) \leq \sigma(P[: \pi[q]]a)$.

We also have $\sigma(P[: \pi[: q]]a) \leq \sigma(P[: q]a)$ since $P[: \pi[q]]a$ is a suffix of $P[: q]a$.

In conclusion, $\sigma(P[: \pi[: q]]a) = \sigma(P[: q]a)$ as desired.