Exercise 6

$ms(k) \rightsquigarrow$ longest substring α of S starting at position k that is a substring of T.

We can find $ms(k)$ by using the suffix tree of T, traversing along $S_{k,n}$ until we get stuck.

$\Rightarrow ms(k)$ is the number of verified characters.

$\rightsquigarrow O(n^2)$

Use suffix links!

$ms(i) = j \Rightarrow S_{i,i+j-1}$ is a substring of T

$\Rightarrow S_{i+1,i+j-1}$ is a substring of T

$\Rightarrow$ There is a path along $S_{i+1,i+j-1}$ in T.

$\rightsquigarrow$ For computing $mS(i+1)$ we want to get to the end point of that path

- If traversal of $S_{i,i+j-1}$ stopped at a node p

  $\Rightarrow$ follow suffix link of $p \rightsquigarrow$ node q that has path labelling $S_{i+1,i+j-1}$.

- If traversal stops on edge, go back to last inner node follow suffix link, choose an outgoing edge and skip verified characters.

$\Rightarrow$ We get to the point where traversal along $S_{i+1,\, i+j-1}$ would lead us. (in constant time)

Running time: Suffix Tree of T : $O(m)$
Calculating $ms(1), \dots, ms(n)$: $O(n)$
(We use each character of S only a constant time)

Problem 17

a) We wanted the LCSubseq of two strings $S \in \Sigma^m$ and $T \in \Sigma^n$.

Choose
- $p(a,a) = 1$
- $p(a,b) = p(a,-) = p(-,b) = 0$, $a \neq b$
- maximisation

The score in the lower right corner denotes the length of the LCSubseq.
Reconstruction: backtracking + recording the matched characters

Runtime:
- Filling the DP-matrix $= O(n \cdot m)$
- Backtracking $= O(n+m)$

b) We have strings $R \in \Sigma^m$, $S \in \Sigma^n$ and a text $T \in \Sigma^\ell$.

Aim: Decide whether any $w \in R \sqcup\!\sqcup S$ is a subsequence of T.

Define $M_{i,j,k}$ which is set to 1 iff a prefix $T_{1,k}$ contains a word from $R_{1,i} \sqcup\!\sqcup S_{1,j}$ as a subsequence, 0 otherwise.

Initialisation:

- $M_{0,0,k} = 1$ for $k \geq 0$
- $M_{i,j,0} = 0$ for $i \neq 0 \vee j \neq 0$

Recurrence:

$$M_{i,j,k} = \begin{cases} M_{i-1,j,k-1} & \text{if } R_i = T_k \wedge S_j \neq T_k \quad \text{(Case A)} \\ M_{i,j-1,k-1} & \text{if } R_i \neq T_k \wedge S_j = T_k \quad \text{(Case B)} \\ M_{i,j,k-1} & \text{if } R_i \neq T_k \wedge S_j \neq T_k \quad \text{(Case C)} \\ M_{i-1,j,k-1} \vee M_{i,j-1,k-1} & \text{if } R_i = S_j = T_k \quad \text{(Case D)} \end{cases}$$

Case A:

If $M_{i-1,j,k-1} = 0 \implies$ no $w \in R_{1,i-1} \sqcup S_{1,j}$ is a subsequence of $T_{1,k-1}$

$\implies$ no $w \in R_{1,i} \sqcup S_{1,j}$ can be subsequence of $T_{1,k}$

Otherwise: $w \in R_{1,i-1} \sqcup S_{1,j}$ is a subsequence of $T_{1,k-1}$

$\implies w \cdot R_i \in R_{1,i} \sqcup S_{1,j}$ is a subsequence of $T_{1,k}$ by pairing up R_i and T_k

Case B: symmetric, swap roles of R and S

Case C: We can pair up neither S_j nor R_i with T_k

Then $w \in R_{1,i} \sqcup S_{1,j}$ can be a subsequence of $T_{1,k}$ iff w is a subsequence of $T_{1,k-1}$.

Case D: Apply Case A and Case B simultaneously.

Runtime:
- Filling the matrix in $O(m \cdot n \cdot \ell)$
- Decision: $O(1)$

Problem 18

We have: • K_1, K_2, two cycles of a minimal cycle cover K
• $w_1 \in K_1$, $w_2 \in K_2$, elements (= nodes = strings) of these cycles

We want to show: $ov(w_1, w_2) < cost(K_1) + cost(K_2)$

Assumption: SCSP-input is substring-free

We know: $ov(w_1, w_2) < \min\{|w_1|, |w_2|\}$ $(*)$

Case 1: $|w_1| < cost(K_1)$ and $|w_2| < cost(K_2)$

$$ov(w_1, w_2) \overset{(*)}{<} \min\{|w_1|, |w_2|\} \leq |w_1| + |w_2| \leq cost(K_1) + cost(K_2)$$

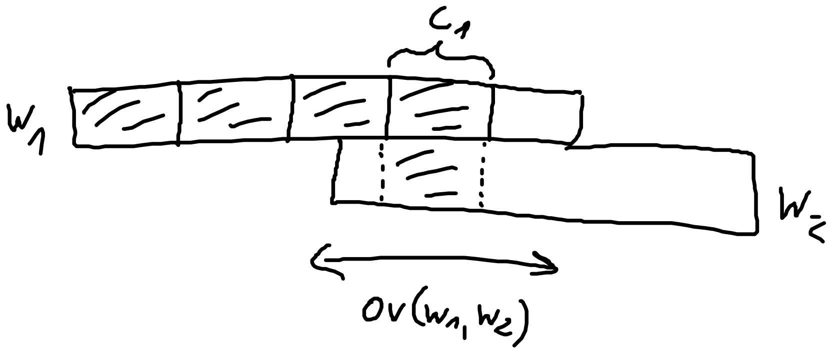
Case 2: w.l.o.g. $|w_1| \geq cost(K_1)$ $(+)$

Towards the contradiction, assume $ov(w_1, w_2) \geq cost(K_1) + cost(K_2)$ (Δ)

By definition of distance graph, every word of a cycle is a sub-word of its cyclic string(and can be characterised by start index, end index and number of full copies of c, the cyclic string.

From $(+)$ follows that w_1 must contain at least 1 full copy of c_1.

Since (Δ), $ov(w_1, w_2) \geq cost(K_1)$ and thus at least 1 full copy of c_1 is contained in the overlap.



Since (Δ), $ov(w_1, w_2) \geq cost(K_2)$ and thus by $(*)$, $|w_2| > cost(K_2)$,

$\Longrightarrow$ At least 1 full of c_2 is contained in w_2.

c_1 and c_2 are primitive (due to the minimality of K), i.e.

$c_i \neq r_i^{k_i}$ for $k_i \geq 2$, r_i any substring, $i \in \{1, 2\}$.

Full copies of c_1 and c_2 are contained in the overlap.

$\Longrightarrow c_1 = c_2$

But then we can construct a common cycle K containing strings from both K_1 and K_2 with costs $|c_1| = |c_2| = cost(K_1) = cost(K_2)$.

↯ minimality of cycle cover