Algorithmische Grundlagen und Vermittlung der Informatik
Lehrstuhl 11 (Algorithm Engineering)
Prof. Dr. Johannes Fischer

technische universität dortmund

Text Indexing and Information Retrieval

Übungsblatt 2

Besprechung: 22.10.2018

Aufgabe 1 (Theorie)

Vervollständigen Sie die Details zu dem $O(n \log n)$-Algorithmus für Suffix-Arrays, der in der Vorlesung skizziert wurde (Abschnitt 3.5.1 im Skript). Außer der dort referenzierten Darstellung können Sie auch die Darstellung aus dem Originalpaper (Manber, Udi, and Gene Myers. "Suffix arrays: a new method for on-line string searches." *SIAM Journal on Computing* 22.5 (1993): 935-948.) oder aus einem Übersichtsartikel (Puglisi, Simon J., William F. Smyth, and Andrew H. Turpin. "A taxonomy of suffix array construction algorithms." *ACM Computing Surveys (CSUR)* 39.1 (2007): 4, Abschnitt 3.1.) als Quelle heranziehen.

Aufgabe 2 (Praxis)

a) Implementieren Sie einen Algorithmus, der das Suffix Array naiv berechnet, d.h. nutzen Sie einen beliebigen Standard-Sortieralgorithmus (z.B. aus einer Java- oder C++-Bibliothek). Der Platzbedarf soll allerdings *linear* sein!

b) Implementieren Sie den Algorithmus aus Aufgabe 1.

c) Vergleichen Sie die Laufzeiten der beiden Implementierungen. Hierzu können Sie die Texte von `http://pizzachili.dcc.uchile.cl/texts.html` nutzen.

Aufgabe 3 (Theorie)

a) Entwerfen Sie einen Linearzeitalgorithmus, der für einen Text T das *kürzeste* Teilwort findet, das nur einmal in T vorkommt.

b) Entwerfen Sie einen Linearzeit-Algorithmus, der für 2 Texte T_1 und T_2 das längste Teilwort findet, das sowohl in T_1 als auch in T_2 vorkommt.

Aufgabe 4 (Theorie)

Entwerfen Sie einen Text-Index linearer Größe, der für ein Muster $P_{1...m}$ ein Array $V[1, m]$ ausgibt, so dass $V[i]$ das längste Präfix von $P_{i...m}$ angibt, das in T vorkommt. Die erwartete Laufzeit soll $O(m)$ sein. Hinweis: Suffixbäume mit entsprechender Zusatzinformation!