

3. Übungsblatt zu Algorithmen II im WS 2021/2022

http://algo2.iti.kit.edu/AlgorithmenII_WS21.php
{sanders, hans-peter.lehmann, daniel.seemaier}@kit.edu

Aufgabe 1 (*Rechnen: Kompression*)

- Bestimmen Sie die *Lempel-Ziv-Faktoren* $f_i = (l_i, p_i)$ des Textes $T_1 = \text{abbbababbbb\$}$.
- Gegeben seien folgende *Lempel-Ziv-Faktoren*. Bestimmen Sie den Text T_2 , aus dem sie entstanden sind.

$(0, \text{a}), (5, 1), (0, \text{b}), (6, 1), (7, 7), (0, \$)$

- Angenommen, jedes Zeichen kann in 1 byte, sowie jeder *Lempel-Ziv-Faktor* in 2 byte repräsentiert werden. Wie viel Prozent des Speicherplatzes von T_2 konnte somit eingespart werden?

Aufgabe 2 (*RMQ in Wavelet Trees*)

Gegeben sei ein Universum von Zahlen \mathcal{U} und ein Feld A von Zahlen aus \mathcal{U} . Geben sie einen Algorithmus an, mit dem sich unter Benutzung eines Wavelet Trees in $\log |\mathcal{U}|$ Zeit $\arg \min_i \{A[i] \mid i \in [a, b]\}$ für Parameter a, b berechnen lässt.

Aufgabe 3 (Rechnen: Suffixarrays und DC3-Algorithmus)

Gegeben sei die Zeichenkette $s = \text{aberakadabera}$.

- Geben Sie den Suffixbaum für s an.
- Geben Sie das Suffixarray für s an.

In der Vorlesung haben Sie einen Linearzeitalgorithmus zur Konstruktion von Suffixarrays kennengelernt. Dieser ist unter dem Namen *DC3-Algorithmus* bekannt. Im Folgenden soll der Algorithmus Schritt für Schritt per Hand ausgeführt werden.

Die Suffixe von s werden zunächst in drei Sequenzen $C^k = \langle s_i \mid (i \bmod 3) = k \rangle$ für $k \in \{0, 1, 2\}$ aufgeteilt. Danach müssen die Sequenzen C^0 und $C^{12} = C^1 \cup C^2$ lexikographisch sortiert werden.

Sortierung von C^{12} :

- Geben Sie die Tripelsequenzen $R^k = \langle s[i..i+2] \mid (i \bmod 3) = k \rangle$ für $k \in \{1, 2\}$ an. Für $i \geq |s|$ gelte $s[i] = \$$ (Auffüllen mit zusätzlichen Abschlusszeichen).
- Bestimmen Sie den *Rang* der Tripel von $R^{12} = R^1 \circ R^2$. Sortieren Sie dazu die Tripel und entfernen mehrfache Vorkommnisse. Die Position eines Tripels in dieser Sortierung gibt seinen Rang an.
- Die berechneten Ränge definieren eine eindeutige Bezeichnung für jedes Tripel in R^{12} . Drücken Sie R^{12} mit Hilfe dieser Ränge aus. Diese Darstellung ergibt die Zeichenkette s^{12} . Muss der DC3-Algorithmus eine Rekursion ausführen?
- Geben Sie das Suffixarray SA^{12} für s^{12} von Hand an (unabhängig, ob der DC3-Algorithmus eine Rekursion durchführt). Vergewissern Sie sich, dass SA^{12} eine Sortierung von C^{12} beschreibt.

Sortierung von C^0 :

- Erstellen Sie eine Zuordnung *rank*, die jedem i mit $s_i \in C^{12}$ den Index von s_i in der sortierten Sequenz C^{12} zuweist. Für alle anderen i sei $\text{rank}(i) = 0$.

Formale Berechnung von *rank* mit Hilfe des Suffixarray SA^{12} nach:

$$\begin{aligned} \text{rank}[3 \cdot (\text{SA}^{12}[i]) + 1] &= i && \text{SA}^{12}[i] < |C^1| \\ \text{rank}[3 \cdot (\text{SA}^{12}[i] - |C^1|) + 2] &= i && \text{SA}^{12}[i] \geq |C^1| \end{aligned}$$

Alle anderen Werte von $\text{rank}[i]$ können gleich 0 gesetzt werden.

- Erstellen Sie Tupel $(s[i], \text{rank}[i+1])$ f.a. $s_i \in C^0$ und sortieren diese lexikographisch. Vergewissern Sie sich, dass diese Sortierung einer Sortierung von C^0 entspricht.

Nachdem C^0 und C^{12} sortiert worden sind, kann das Suffixarray von s bestimmt werden:

- Führen Sie eine Mischen-Operation auf C^0 und C^{12} aus. Die resultierende Sequenz wird mit C bezeichnet. Es gelten folgende Sortierkriterien:

$$s_i \leq s_j \iff \begin{cases} (s[i], \text{rank}[i+1]) \leq (s[j], \text{rank}[j+1]) & s_j \in C^1 \\ (s[i..i+1], \text{rank}[i+2]) \leq (s[j..j+1], \text{rank}[j+2]) & s_j \in C^2 \end{cases}$$

Vergewissern Sie sich, dass C lexikographisch sortiert ist und damit das Suffixarray induziert.

Notation:

- Alle Indizes fangen bei 0 an – analog zu Kapitel 9.3.6, auf dem die Aufgabe basiert.
- $s[i \dots j]$: Zeichen an Stelle i (bis j) in s (z.B. $s[1..3] = \text{ber}$)
- s_i : Suffix von s ab Stelle i (z.B. $s_2 = \text{erakadabera}$)

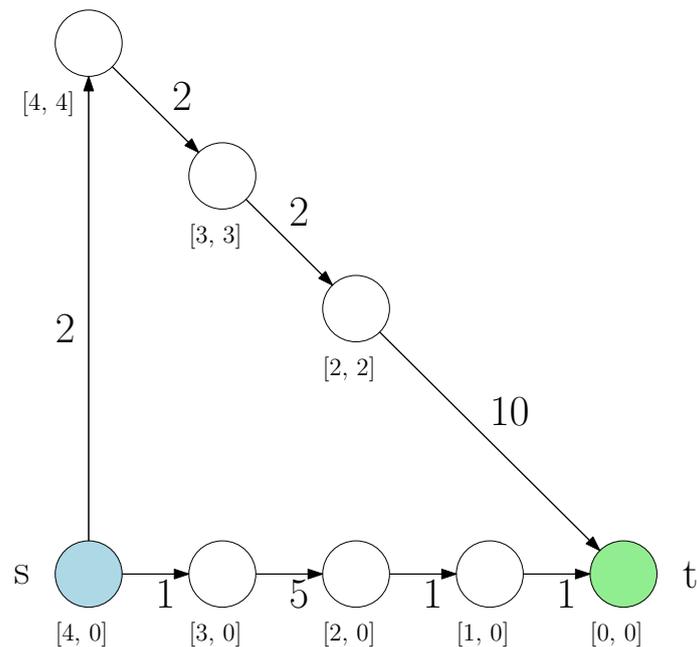
Aufgabe 4 (Rechnen: A* Suche)

Gegeben sei der unten abgebildete Graph. An den Kanten sind Kosten für die Nutzung der Verbindung eingetragen und die Knoten tragen Ortskoordinaten.

- a) Ergänzen Sie den gegebenen Graphen um Knotenpotentiale für eine A* Suche von s nach t . Verwenden Sie Knoten t als Landmarke und die Manhattan-Distanz ($\hat{=}$ Einsnorm $\|\cdot\|_1$) als Abschätzung für die Entfernung zum Ziel.

Hinweis: $\|\cdot\|_1 : \|(x_1, y_1), (x_2, y_2)\|_1 = y_2 - y_1 + x_2 - x_1$.

- b) Tragen Sie die reduzierten Kantengewichte in den Graphen ein.
- c) Wieviele `deleteMin` Operationen führt die A* Suche auf dem Graphen aus? Wieviele eine normale Suche mit Dijkstra's Algorithmus?



Aufgabe 5 (Kleinaufgaben: A* Suche)

- a) Sei $\text{pot}(\cdot)$ eine gültige Potentialfunktion für die A* Suche nach Knoten t in Graph $G(V, E)$. Überprüfen Sie, ob

$$\text{pot}^c = \text{pot} + c, \quad c = \text{const.}$$

ebenfalls eine gültige Potentialfunktion darstellt.

- b) Kann es vorkommen, dass eine A* Suche mehr Knoten absucht als eine Suche mit Dijkstra's Algorithmus für die gleiche Anfrage? Begründen Sie warum nicht oder geben Sie ein Beispiel an.

Aufgabe 6 (*Analyse: Kürzeste Wege (Wiederholung)*)

- a) Betrachten Sie eine Suche mit bidirektionalem Dijkstra. Geben Sie eine Familie von Graphen an, bei der ein ausgezeichnete Knoten u eine Anzahl an **decreaseKey** Operationen erfährt, die linear in der Länge des kürzesten Weges für jede mögliche Anfrage ist.
- b) Bei manchen Algorithmen kann es sinnvoll sein, unterschiedliche Potentialfunktionen zu kombinieren. Zeigen Sie in diesem Zusammenhang: Sind π_1 und π_2 gültige Potentialfunktionen, so ist auch $\pi = \frac{\pi_1 + \pi_2}{2}$ eine gültige Potentialfunktion.
- c) Bei der Durchführung von *Dijkstras Algorithmus* können kürzeste Wege gespeichert werden, indem Vorgängerknoten gespeichert werden. Diese werden immer dann geändert, wenn eine bessere vorläufige Distanz gefunden wird. Dieses kann auch auf einem Graphen durchgeführt werden, der negative Kantengewichte enthält. Das Stopkriterium von Dijkstras Algorithmus muss dafür durch ein schwächeres Kriterium ersetzt werden: Es wird gestoppt, sobald keine Verbesserung mehr gefunden werden kann. Zeigen Sie: Wenn auf diese Art ein Kreis entsteht, so ist dieser negativ.
- d) (*) Dijkstras Algorithmus ist ein Spezialfall des allgemeinen *Labeling Algorithmus*. Der allgemeine Labeling Algorithmus wählt eine beliebige Kante aus, die das Label des Zielknotens verbessert. Geben Sie einen Graphen sowie eine Reihenfolge der Kanten an, sodass bei positiven Kantengewichten eine exponentielle Zahl von Schritten ausgeführt wird.

Hinweis: Achtung, schwierige Knobelaufgabe!

Ausgabe: 22.11.2021

Abgabe: keine Abgabe, keine Korrektur