Aufgabe 1: LaTeX-Dokument

Teilnahme-ID: ?????

Bearbeiter/-in dieser Aufgabe:
Vor- und Nachname

21. April 2025

Inhaltsverzeichnis

Anleitung: Trage oben in den Zeilen 8 bis 10 die Aufgabennummer, die Teilnahme-ID und die/den Bearbeiterin/Bearbeiter dieser Aufgabe mit Vor- und Nachnamen ein. Vergiss nicht, auch den Aufgabennamen anzupassen (statt „LaTeX-Dokument“)!

Dann kannst du dieses Dokument mit deiner LaTeX-Umgebung übersetzen. Die Texte, die hier bereits stehen, geben ein paar Hinweise zur Einsendung. Du solltest sie aber in deiner Einsendung wieder entfernen!

1 Lösungsidee

1.1 Huffman Codierung

TODO Quelle vlt: https://people.eng.unimelb.edu.au/ammoffat/abstracts/compsurv19moffat.pdf
David Huffman hat 1952 eine Methode veröffentlicht, die heute als *Huffman Codierung* bekannt ist. Er stellte eine Methode vor, um eine optimale Präfixfreie Codierung für einen bestimmten Text zu finden. Dabei geht man von einem Alphabet A mit n verschiedenen Zeichen $a_1, \ddot{,} a_n \in A$ aus, gemeinsam mit der Häufigkeitsverteilung der Zeichen $p_i = \frac{\text{Häufigkeit von } a_i \text{ im zu codierenden Text}}{\text{Länge des Textes}}$. Wir nehmen an, dass das Alphabet so sortiert ist, dass $p_i \leq p_{i+1}$ für alle $0 \leq i < n-1$ gilt. Der Text soll mit einem Ausgabealphabet O codiert werden, das aus r verschiedenen Zeichen $o_1, \ldots, o_r \in O$ besteht. In der Bwinf-Aufgabenstellung besteht dieses Ausgabealphabet O aus den r verschiedenen Perlen (die aber alle den gleichen Durchmesser haben). Um den ursprünglichen Text zu codieren, müssen wir nun jedem Buchstaben a_i ein *Codewort* w_i zuordnen, dass aus einer Kette an Buchstaben aus dem Ausgabealphabet besteht $w_i = o_j o_k o_l \ldots$.
Diese Codierung soll *Präfixfrei* sein, also kein Codewort soll Teil eines anderen Codeworts sein. Dadurch kann der Text als Aneinanderreihung von Codewörtern (ohne „Komma“ dazwischen) übertragen und eindeutig decodiert werden.
Aufgrund von dieser Eigenschaft können wir den Code als einen Baum darstellen, der n Blätter hat und bei dem jeder Knoten höchstens r Kinder hat.

Definition 1. *Wir nennen einen solchen Baum mit n blättern und höchstens r Kindern* ***valide****, da er eine mögliche Codetabelle darstellt.*

Jedes Blatt eines solchen Baums repräsentiert ein Codewort w, dass eindeutig durch den Pfad von der Wurzel des Baums zu diesem Blatt definiert ist. Der Pfad wird durch die Kanten des Baums definiert, die mit den Perlen beschriftet sind. Da alle Perlen gleich groß sind, also die Länge aller Buchstaben des Ausgabealphabets gleich ist, ist die Beschriftung der Kanten beliebig. Es ist lediglich wichtig, dass alle Kanten eines Knotens mit unterschiedlichen Buchstaben beschriftet sind. Die Länge des Codeworts $|w|$ entspricht der Anzahl der Kanten auf dem Pfad von der Wurzel zu diesem Blatt.
Da wir eine optimale Codetabelle (also eine möglichst kurze Perlenkette) erstellen wollen, müssen wir die „Kosten“ eines Baums definieren. Diese Kosten eines Baums hängen natürlich auch davon ab, welches Codewort welchem Buchstaben zugeordnet wird. FÜr diese Definition gehen wir davon aus, dass wir eine solche Zuordnung $w_i \to a_i$ haben.

Definition 2. *Die Kosten eines Baums T ist definiert als das Produkt der Länge jedes Codeworts w_i mit der Antrittswahrscheinlichkeit p_i des codierten Buchstabens a_i:*

$$cost(T) = \sum_{i \leq n}^{i=0} |w_i| \cdot p_i$$

Da die Kosten eines Baumes proportional zu der Länge der resultierenden Perlenkette sind, kann die Aufgabenstellung darauf reduziert werden, den validen Baum T mit den minimalen Kosten zu finden.

Die Zuordnung der Codewörter zu den Buchstaben des Alphabets ist für jeden Baum eindeutig, da die Gesamtlänge des codierten Textes minimiert werden soll.

Jede Codetabelle, bei der r Perlen mit dem selben Durchmesser verwendet werden dürfen, kann als Baum, bei dem jeder Knoten höchstens r Kinder hat, dargestellt werden. ...

Im Aufgabenteil b) haben die Perlen aber unterschiedliche, ganzzahlige Durchmesser $c_1 \leq c_2 \leq \cdots \leq c_r = C$. Um die Eigenschaft zu erhalten, dass die Länge des Pfads von der Wurzel des Baums zu einem Blatt der Länge des Codes entspricht, müssen die Kanten des Baums der Länge der Perlen entsprechen.
...
TODO Beispiel

Definition 3. *$d_1, d_2, \ldots, d_C$ bezeichnet die Anzahl der Perlen mit den Durchmessern $1, 2, \ldots, C$.*

TODO Beispiel

Definition 4. *Voller Baum (alle Knoten haben entweder 0 oder r Kinder)*

Einführung ...
Labeling ist trivial (und kann ab jetzt weggelassen werden), weil...

Definition 5. *Wir bennen die Blätter des Baums nach aufsteigender Tiefe: $depth(v_1) \leq depth(v_2) \leq \cdots \leq depth(v_n)$*

Definition 6. *Wir nennen einen Baum Ebene-i-Baum, wenn alle internen Knoten auf einer Ebene $\leq i$ liegen.*

Definition 7. *Die Signatur einer Ebene i des Baums T ist das $C+1$-Tupel*

$$sig_i(T) = (m; l_1, l2, \ldots, l_C)$$

wobei $m = |\{v \in T | vist\ ein\ Blatt, tiefe(v) \leq i\}|$ die Anzahl der Blätter von T mit einer Tiefe von höchstens i ist und

$$l_k = |\{u \in T | tiefe(u) = i + k\}, k \in \{1, \ldots, C\}$$

die Anzahl der Knoten auf Ebene $i + k$ ist.

Nun definieren mit die Expand-Operation, die einen Ebene-$i+1$-Baum aus einem Ebene-i-Baum erzeugt.

Definition 8. *Sei T ein Ebene-i-Baum mit der Signatur $sig_T = (m; l_1, l_2, \ldots, l_C)$. Die Expand Operation wandelt $0 \leq q \leq l_1$ Blätter auf Ebene $i+1$ in interne Knoten um, indem r Kinder an jedes dieser Blätter angehängt werden. Die Signatur des resultierenden Ebene-$i+1$-Baums ist*

$$sig_{T'} = (m + l_1, l_2, \ldots, l_C) + q * (-1, d_1, d_2, \ldots, d_C)$$

(TODO jeder volle Binärbaum der höhe h kann durch h Expand Operationen erzeugt werden?)

Um einen optimalen Code zu finden, definieren wir die Kosten eines Baumes.

Definition 9. *Sei T ein Ebene-i-Baum mit der Signatur $sig_T = (m; l_1, l_2, \ldots, l_C)$. Falls $m \leq n$, dann sind die Kosten von T gleich ...*

Definition 10. *Die Kosten eines Baumes T mit $\geq n$ Blätter sind*

$$cost(T) = \sum_{i=1}^{n} v_i * p_i$$

(TODO diese Definition geht vom optimalen labeling aus)

2 Umsetzung

Hier wird kurz erläutert, wie die Lösungsidee im Programm tatsächlich umgesetzt wurde. Hier können auch Implementierungsdetails erwähnt werden. TODO erweiterung: die Kosten der Perlen sind keine ganzen Zahlen mehr, sondern reelle Zahlen.

3 Beispiele

Genügend Beispiele einbinden! Die Beispiele von der BwInf-Webseite sollten hier diskutiert werden, aber auch eigene Beispiele sind sehr gut – besonders wenn sie Spezialfälle abdecken. Aber bitte nicht 30 Seiten Programmausgabe hier einfügen!

4 Quellcode

Unwichtige Teile des Programms sollen hier nicht abgedruckt werden. Dieser Teil sollte nicht mehr als 2–3 Seiten umfassen, maximal 10.