Aufgabe 1: LaTeX-Dokument

Teilnahme-ID: ?????

Bearbeiter/-in dieser Aufgabe:
Vor- und Nachname

22. 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 $a_1, \ldots, a_n \in A$ 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 nach absteigender Häufigkeit sorteiert ist, also $p_i \geq p_{i+1}$ für alle $0 \leq i < n-1$. 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.
Dafür ist aber auch die Zuordnung, welcher Buchstabe von welchem Codewort codiert werden soll, entscheidend. Für diese Aufgabenstellung ist die Zuordnung aber für jeden Baum eindeutig, da wir die Kosten des Baums minimieren wollen. Um das zu erreichen, müssen wir das längste Codewort zu dem Buchstaben zugeordnet, der am seltensten vorkommt und das kürzeste Codewort zu dem Buchstaben zuordnen, der am häufigsten vorkommt. Dafür ordnen wir im Folgenden die Blätter Baumes in aufsteigender Reihenfolge, also $|w_0| \leq \cdots \leq |w_n|$. Da das Alphabet nach absteigender Reihenfolge sortiert ist, also $p_0 \geq \cdots \geq p_n$, liefert die Zuordnung von w_i zu a_i für jeden Baum die geringsten Kosten.

Definition 3. *Wir nennen einen solchen Baum, der die minimalen Kosten (kürzeste Perlenkette) für eine bestimmte Häufigkeitsverteilung hat* ***optimal***

1.1.1 Konstruktion des optimalen Baums

Definition 4. *Wir nennen einen r-när Baum, bei dem jeder Knoten maximal r Kinder hat* ***vollständig****, wenn jeder Knoten genau r Kinder hat.*

Lemma 1. *Ein optimaler binärer Huffman Baum ist vollständig. TODO braucht man das??*

Beweis. Ein optimaler binärer Huffman Baum ist im allgemeinen vollständig, da in einem Optimalen Baum kein Knoten mit nur einem Kind existieren kann. Würde man diesen Knoten zu einem Blatt machen hätte man einen Baum mit der gleichen Anzahl an Blättern aber geringeren Kosten. □

Das gilt im allgemeinen Fall bei Bäumen mit $r > 2$ Kindern aber nicht.
Für die allgemein bekannte Konstruktion (TODO Quelle) eines binären Huffman Baums werden wiederholt zwei Teilbäume zu einem zusammengefasst, bis schließlich nur noch einer übrig ist. Im allgemeinen Fall müssen immer r Teilbäume zu einem zusammengefasst werden, was aber nicht für jede Anzahl an Buchstaben n im Alphabet immer funktioniert.

Beispiel 1. *Für $n = 4$ werden im ersten Schritt drei Knoten zu einem zusammengefasst. Im nächsten sind nur noch zwei Knoten übrig. Würde man diese Knoten nun zu einem fertigen Baum kombinieren wäre dieser für bestimmte Häufigkeitsverteilungen (z.B. $p_0 = p_1 = p_2 = p_3$) nicht optimal.*

Daher muss das Alphabet zuerst mit so vielen Platzhalterbuchstaben, deren Antreffwahrscheinlichkeit 0 ist, aufgefüllt werden, dass gilt: $n \mod (n-1) = 1$.
Jetzt können wir mit der Konstruktion beginnen:

1. Erstelle für jeden Buchstaben o_i im Alphabet (inklusive Platzhalter) einen Teilbaum (der aus einem Wurzelknoten besteht) und beschriften diesen mit seiner Antreffwahrscheinlichkeit p_i.

2. Wähle die r Teilbäume mit den kleinsten Antreffwahrscheinlichkeiten. (Diese Wahl ist nicht eindeutig)

3. Beschrifte einen neuen Knoten mit der Summe der Antreffwahrscheinlichkeiten der r gewählten Teilbäume und erstelle einen neuen Teilbeum mit dem neuen Knoten als Wurzel und den gewählten Teilbäumen als Kinder.

4. Wieder hole Schritt 2 und 3, bis nur noch ein Baum übrig ist.

TODO Bilder
TODO QUelle für allgemeine Konstruktion

1.1.2 Beweis der optimalität

Der Standartbeweis für die Optimalität von Binären Huffman Bäumen kann leicht verallgemeinert werden, daher Dazu müssen wir zuerst das „Sibling-Lemma“ auf $r-$när Bäume verallgemeinern:

Lemma 2. *Seien $l_0, l_1, \dots l_{r-1}$ die r Buchstaben mit der geringsten Antreffwahrscheinlichkeit. Sie sind in einem Huffman Baum „Geschwister“, also Kinder des gleichen Knotens Y und liegen auf der untersten Ebene des Baums.*

Beweis. Die Buchstaben $l_0, l_1, \dots l_{r-1}$ sind sicher Geschwister, da sie als erstes ausgewählt werden.
Wenn $l_0, l_1, \dots l_{r-1}$ nicht auf der unteresten Ebene des Baums liegen würden, dann müsste es einen Internen Knoten X geben, dessen Antreffwahrscheinlichkeit kleiner ist als die des Knotens Y, dessen Kinder $l_0, l_1, \dots l_{r-1}$ sind. Dafür müsste aber mindestens einer der Kinder von X eine geringere Antreffwahrscheinlichkeit haben als einer der Kinder $l_0, l_1, \dots l_{r-1}$ von Y. Das wiederspricht aber unserer Anfänglichen Annahme, das $l_0, l_1, \dots l_{r-1}$ die r Buchstaben mit der geringsten Antreffwahrscheinlichkeit sind. Daher können wir sagen, dass $l_0, l_1, \dots l_{r-1}$ immer auf der untersten Ebene des Baums liegen. □

Damit können wir nun mithilfe von Induktion über die Größe n des Alphabets den Beweis durchführen:

Satz 1. *Die oben beschrieben Konstruktion für $r-$näre Huffman Bäume liefert einen optimalen Baum und damit eine bestmögliche Codetabelle.*

Beweis. Induktionsanfang: Für $n \leq r$ ist ein Huffman Baum offensichtlich optimal.
Induktionshypothese: Alle Huffman Bäume mit $< n$ Blättern sind optimal.
Induktionsschritt: Nun müssen wir zeigen, ein Huffman Baum T mit n Blättern optimal ist. Seien $l_0, l_1, \dots l_{r-1}$ wieder die r Buchstaben mit der geringsten Antreffwahrscheinlichkeit. Sei Y der Knoten in T, dessen Kinder $l_0, l_1, \dots l_{r-1}$ sind. Sei T' nun ein Huffman Baum identisch zu T, bei dem Y zu einem Blattknoten konvertiert wurde (mit der gleichen Antreffwahrscheinlichkeit wie der Knoten Y in T). Aufgrund der Induktionshypothese ist T' optimal, da er $n - (r-1) < n$ Blätter hat.
Wenn T nicht optimal wäre, gäbe es einen Baum T_1 mit geringeren Kosten als T. Aus T_1 könnte man wieder einen Baum T_1' mit $n - (r-1)$ Blättern erstellen, in dem man Y zu einem Knoten konvertiert. Wenn T_1 geringere Kosten als T hätte, hätte auch T_1' geringere Kosten als T', woraus man schließen kann, dass T optimal sein muss. □

Im Aufgabenteil b) haben die Perlen aber unterschiedliche, ganzzahlige Durchmesser $c_1 \leq c_2 \leq \dots \leq c_r = C$. 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 5. *$d_1, d_2, \dots, d_C$ bezeichnet die Anzahl der Perlen mit den Durchmessern $1, 2, \dots, C$.*

TODO Beispiel

Definition 6. *Voller Baum (alle Knoten haben entweder 0 oder r Kinder)*

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

Definition 7. *Wir bennen die Blätter des Baums nach aufsteigender Tiefe: $depth(v_1) \leq depth(v_2) \leq \dots \leq depth(v_n)$*

Definition 8. *Wir nennen einen Baum Ebene-i-Baum, wenn alle internen Knoten auf einer Ebene $\leq i$ liegen.*

Definition 9. *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 10. *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 11. *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 12. *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.