Esercizi di programmazione

Set 3.

Di Giuseppe Martinelli

Matr: 7093926

Esercizio N 3.2 Codici di Huffman

Svolgimento:

L'esercizio verte sulla funzione *buildTree* la quale prende in input un alfabeto di lettere e la loro probabilità. La funzione verte il suo funzionamento sulla classe *Node* la quale crea un'astrazione del concetto di nodo in un albero. Infatti, un oggetto di tipo *Node* contiene al suo interno:

- Una lettera che contraddistingue quel nodo
- Un valore che corrisponde alla probabilità di quella lettera
- Un puntatore al figlio destro e sinistro di quel nodo

La funzione quindi utilizzando i metodi forniti da questa classe costruisce l'albero trie che è alla base dei codici di Huffman, dove ogni lettera e la relativa frequenza costituiscono le foglie dell'albero. La funzione inizialmente converte tutti i nodi dell'alfabeto fornito in input in istanze della classe *Node* e successivamente costruisce l'albero trie scegliendo iterativamente i due nodi dell'alfabeto che hanno le frequenze minori, li unisce formando un nodo padre avente come valore la somma delle frequenze, come figlio destro e sinistro i due nodi e come lettera la concatenazione delle due lettere. Dopo questa operazione il nodo padre viene reinserito nell'alfabeto ed il ciclo riprende finché l'alfabeto non contiene un unico nodo che sarà la radice.

Una volta costruito l'albero, la funzione invoca il metodo *buildCode* il quale prende in input la radice dell'albero , un trie ed infine un dizionario. La funzione è ricorsiva e consiste sostanzialmente in una BFS dove scendendo in profondità l'albero si costruisce un trie per ogni nodo foglia (cioè per ogni lettera). Il caso base della funzione è quello in cui il figlio destro è assente (poteva essere anche il figlio sinistro poiché per costruzione i nodi hanno o grado 2 oppure grado 0), in questo caso si prende il trie costruito nelle chiamate ricorsive precedenti e lo si assegna al dizionario per quel rispettivo nodo e viene returnato. Se il vuoto ha grado 2, allora la funzione prima esplora la parte destra dell'albero aggiungendo al trie corrente il bit 1 ed una volta arrivato al caso base la funzione esplora la parte sinistra usando il dizionario costruito nella parte destra. Questo dizionario alla fine della esecuzione conterrà un codice sicuramente prefix-free (in quanto successivamente controllato dal metodo *isPrefixFree*)

Il codice contiene anche un metodo per la decodifica di messaggi dato un codice prefix-free in input

Esercizio N 3.3 Compressione dei digrammi

Svolgimento:

Per lo svolgimento dell'esercizio, viene riutilizzato il codice scritto nell'esercizio precedente per la creazione di codici di Huffman dato un alfabeto e la frequenza delle lettere. Le frequenze dei digrammi sono state prese dal link seguente (https://gist.github.com/lydell/c439049abac2c9226e53) mentre le frequenze delle singole lettere sono state prese da Wikipedia.

Il codice prima di tutto carica le frequenze che sono presenti in due file diversi (freqDigrammi.json e freqSingoleLettere.json) e vengono normalizzate attraverso la funzione *normalizeFreqData* la quale ha un valore booleano che controlla il comportamento della funzione. Questo valore è stato introdotto poiché le frequenze sono in due formati diversi (il primo file contiene una lista di liste mentre il secondo solo una lista). Da qui si costruisce un dizionario contenente la probabilità delle lettere del primo codice C1 contenente tutti i digrammi.

Per calcolare la probabilità delle lettere del codice C2 si invoca invece la funzione *buildProbabilityC2* la quale crea un dizionario contenente come chiavi digrammi di C1 ed invece per valore la probabilità delle due lettere moltiplicata tra loro.

Per la costruzione dei codici è stato riutilizzato il codice nell'esercizio precedente.

Una volta che i due codici sono stati costruiti, il programma calcola la lunghezza media di ciascuno di essi invocando la funzione *getLunghezzaMediaCodici* dove viene eseguito il calcolo della sommatoria presente nelle dispense, in particolare ci si riferisce alla formula $L(C) = \sum_i p_i|c_i|$ dove p è la probabilità del digramma e c è la lunghezza del codice corrispondente al digramma.

Applicata ad entrambi i codici precedenti, il programma calcola il delta sottraendo la lunghezza media del secondo codice con quella del primo.

Infine, si è utilizzato il primo capitolo di Moby Dick per testare i due codici calcolandosi la differenza tra i due testi.

Set 3.

giovedì 2 giugno 2022 16:07

2.3 La capacità del piccione viaggiatore

Un comandante che assedia un forte usa dei piccioni viaggiatori per comunicare con gli alleati. Ogni piccione porta una lettera (8 bit). Viene liberato un piccione ogni 5 minuti. Ogni piccione impiega tre minuti per raggiungere la destinazione. Si calcoli la capacità in bit/ora di questo collegamento, nei seguenti due casi.

(a) I piccioni raggiungono tutti la destinazione.

(b) I nemici abbattono una frazione α dei piccioni, e sostituiscono ogni piccione abbattuto con uno che porta una lettera scelta a caso. Nei calcoli, non è necessario espandere la funzione entropia binaria $B(\lambda) = H(\lambda, 1-\lambda)$.

Traccia. Per il caso (b), si usi il risultato sui canali simmetrici dimostrato in classe. Può risultare utile tenere presente la seguente uguaglianza, che vale per una generica distribuzione $\mathbf{p} = (p_1, \ldots, p_n)$:

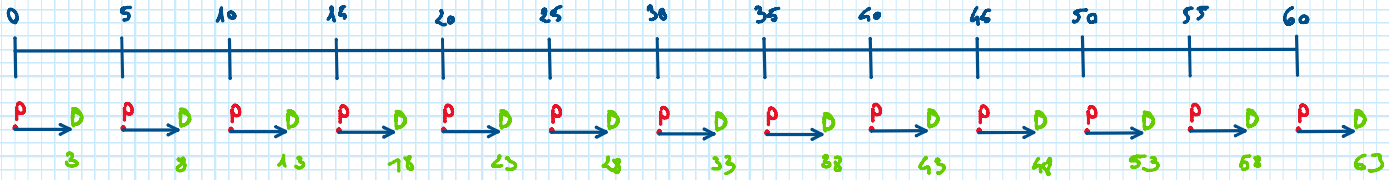
$$H(p_1, \ldots, p_n) = H(p_1) + (1-p_1) H\left(\frac{p_2}{1-p_1}, \ldots, \frac{p_n}{1-p_1}\right).$$

Per la dimostrazione di quest'ultima, sia X una v.a. distribuita come $\mathbf{p}$ e si definisca la v.a. ausiliaria $Y = 1$ se $X = x_1$, $Y = 0$ altrimenti: si espanda $H(X,Y)$ usando la chain rule.

a) Per rispondere a questo quesito basta dividere:

$$60 / 5 = 12$$

Il risultato ottenuto nel punto a) è verificabile qui, infatti se schematizziamo otteniamo che:



Lo schema seguente indica con P le partenze dei piccioni, mentre con D l'arrivo alla destinazione con relativa ora di arrivo. Se si guarda lo schema, è possibile notare come in un'ora ci siano 13 piccioni che effettivamente partono, ma i piccioni che arrivano entro l'orario stabilito sono 12. Quindi, la capacità in bit/ora di questo collegamento considerando che ogni piccione porta 8 bit è:

$$12 \cdot 8 = 96 \text{ bit/ora}$$

b) Supponiamo che un piccione venga abbattuto e sostituito dall'attaccante con uno diverso. Il nuovo piccione porterà una nuova lettera che sarà:

- Uguale a quella del piccione abbattuto con probabilità 1/26

oppure

- Diversa, con probabilità 25/26

Adesso se costruiamo una matrice M dove per ogni riga poniamo la lettera originale e per ogni colonna poniamo la lettera intercettata e per ogni elemento poniamo la probabilità otteniamo che:

$$M = \begin{array}{c} a \\ b \\ c \\ \vdots \\ z \end{array} \begin{bmatrix} 1/26 & \cdots & 1/26 \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ 1/26 & & 1/26 \end{bmatrix}$$

Da qui è possibile calcolare la capacità C in questo modo:

Da qui è possibile calcolare la capacità C in questo modo:

$$C = \log|Y| - H(r)$$

dove $|Y|$ è la cardinalità dell'alfabeto di arrivo ed r è una qualsiasi riga della matrice H. Sapendo che:

$$H(p_1, p_2, \dots, p_{26}) = H(p_1) + (1-p_1) \cdot H\left(\frac{p_2}{1-p_1}, \dots, \frac{p_{26}}{1-p_1}\right)$$

è possibile calcolare $H\left(\frac{p_2}{1-p_1}, \dots, \frac{p_{26}}{1-p_1}\right)$ con la formula vista sopra per 26 volte. Altrimenti è possibile definire due variabili aleatorie X e Y dove X rappresenta il valore della lettera pescata dal piccione dopo lo scambio e Y restituisce valore 1 se $X = X_1$ (cioè se dopo lo scambio la lettera è la stessa inviata dal mittente), altrimenti $Y = 0$.

Utilizzando le seguenti variabili aleatorie possiamo considerare:

$$H(p_1, \dots, p_{26}) \text{ come } H(p, 1-p)$$

$$\Downarrow$$

$$H(1/26, 25/26)$$

Quindi:

$$C = \log|Y| - H(r) = \log|Y| - H(p, 1-p)$$

Inoltre sappiamo che:

$$H(p, 1-p) = -(p \log p + (1-p)\log(1-p))$$

e nel nostro caso:

$$H(p, 1-p) = H(1/26, 25/26) = -\left(1/26 \log 1/26 + 25/26 \log 25/26\right)$$

Quindi:

$$H(p, 1-p) = 0.24$$

Ora è possibile calcolare la capacità C

$$C = \log 26 - 0,24 = 4,46$$

Questa capacità corrisponde al caso in cui tutti i piccioni vengono abbattuti e sostituiti. Ma poiché solo una frazione d viene abbattuta allora:

$$C = 12\left((1-d) \cdot 8 + d \cdot 4,46\right)$$