Greedy (VIII parte)

Progettazione di Algoritmi a.a. 2023-24

Matricole congrue a 1

Docente: Annalisa De Bonis

1

1

Data Compression

Domanda. Dato un testo che usa 32 simboli (26 lettere, spazi, e simboli di puteggiatura) . Come codifichiamo il testo in bit?

Domanda. Alcuni simboli (e, t, a, o, i, n) sono usati piu` frequentemente di altri. Come usiamo questa informazione per ridurre la lunghezza della codifica?

Domanda. Come decodifichiamo la codifica binaria? In altre parole come capiamo dove finisce la codifica di un simbolo e dove comincia quella del simbolo successive?

Esempio. Codifichiamo a b e come segue:

c(a) = 01

c(b) = 010

c(e) = 1

A cosa corrisponde 0101?

01 01 →aa

010 1 →be

La decodifica non e` unica 👎

2

2

Data Compression

Domanda. Dato un testo che usa 32 simboli (26 lettere, spazi, e simboli di puteggiatura) . Come codifichiamo il testo in bit?

Risposta. Codifichiamo fino a 2^5 simboli differenti usando 5 bit per ciascun simbolo. Codifica a lunghezza fissa.

Vantaggio. Bit decodificabili in modo univoco

Esempio: Nel codice ASCII, i caratteri sono codificati in strighe di 8 bit. Per decodificare 010000100100000010100011 divido la sequenza in blocchi di 8 bit e ottengo 01000010 01000001 0100011 = B A C

Svantaggio. Tutti i simboli hanno codifiche della stessa lunghezza indipendentemente dalla loro frequenza

3

Data Compression

Domanda. Alcuni simboli (e, t, a, o, i, n) sono usati piu` frequentemente di altri. Come usiamo questa informazione per ridurre la lunghezza della codifica?

Risposta. Codifica i simboli piu` frequenti con meno bit e quelli meno frequenti con piu` bit

Vantaggio. Madiamente si risparmia

Svantaggio. Rispetto alla codifica a lunghezza fissa possiamo ridurre la lunghezza della codifica ma ora potremmo non capire dove cominciano e finiscono le codifiche dei singoli simboli.

Domanda. Come decodifichiamo la codifica binaria se usiamo lunghezze variabili? In altre parole come capiamo dove finisce la codifica di un simbolo e dove comincia quella del simbolo successive?

Risposta. Usa un simbolo per separare (come il simbolo speciale pausa nel codice Morse) oppure elimina le ambiguita` assicurandoti che la codifica di un simbolo non sia prefisso della codifica di un altro simbolo.

4

Codici prefissi

Definizione. Un codice prefisso per un insieme S e` una funzione c che mappa ciascun $x \in S$ in una stringa di 0 e 1 in modo tale che per ogni due simboli $x,y \in S$, $x \neq y$, c(x) non e` prefisso di c(y).

Esempio.

c(a) = 11
c(e) = 01
c(k) = 001
c(l) = 10
c(u) = 000

Qual e` la codifica di 1001000001 ?

Per capirlo basta leggere la sequenza da sinistra a destra e non appena si individua "un pezzo" che corrisponde ad una parola codice la si decodifica con il simbolo di S ad essa associato 10 01 000 001 = "leuk"

Supponiamo di conoscere la frequenza dei simboli a,e,k,l,u. in un testo di 1G: f_a=0.4, f_e=0.2, f_k=0.2, f_l=0.1, f_u=0.1

Quanto e` lunga la codifica del testo?

$2*f_a + 2*f_e + 3*f_k + 2*f_l + 3*f_u = 2.3G$

5

Optimal Prefix Codes

Definizione.Il numero medio di bit per lettera (ABL) di un codice prefisso c e` la sommatoria su tutti i simboli della frequenza del simbolo per il numero di bit della sua codifica.

encoding:

$$ABL(c) = \sum_{s \in S} f(s)\ |c(s)|$$

|c(s)| = lunghezza della codifica (binaria) della lettera s
f(s)= frequenza dela lettera s

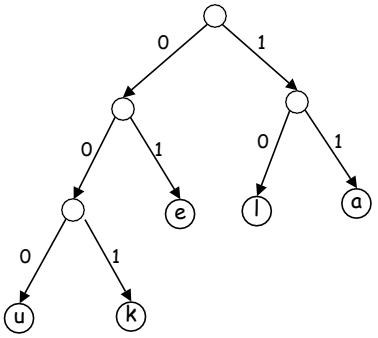
Se usiamo un codice c per codificare simboli con frequenze f(s) allora per codificare una stringa di lunghezza n abbiamo bisogno di n*ABL(c) bit.

Vorremmo un codice prefisso con con ABL quanto piu` piccolo possible.

6

Rappresentazione dei codici prefissi mediante alberi binari

Ex. c(a) = 11
c(e) = 01
c(k) = 001
c(l) = 10
c(u) = 000



Come e` fatto un albero binario di un codice prefisso?
Le foglie sono etichettate con i simboli codificati
Gli archi sono associati a 0 o a 1
Per un certo simbolo x, i valori dei bit associati agli archi lungo il persorso dalla radice alla foglia etichettata con x formanto la codifica di x.

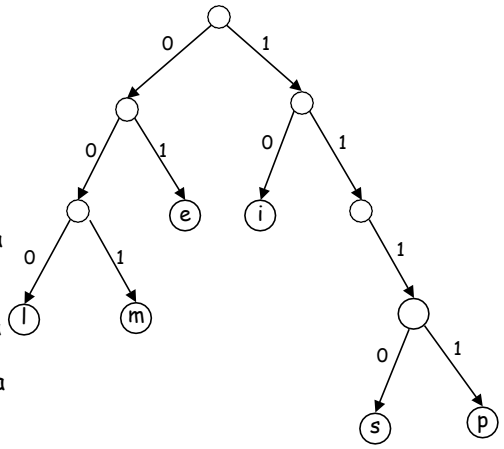
7

7

Rappresentazione dei codici prefissi mediante alberi binari

Cosa rappresenta la sequenza
111010001111101000 ?

Per capirlo basta seguire il percorso radice-foglia ottenuto scandendo man mano i bit della sequenza: si parte dalla radice e ogni volta che si raggiunge un nodo si sceglie l'arco uscente sinistro se il bit e` 0 e quello destro se il bit e` 1. Una volta che si raggiunge una foglia, si decodifica la parte di sequenza scandita con il simbolo associato alla foglia e si riparte dalla radice ricominciando a scandire la sequenza dal bit successivo a quello in cui ci eravamo fermati precedentemente

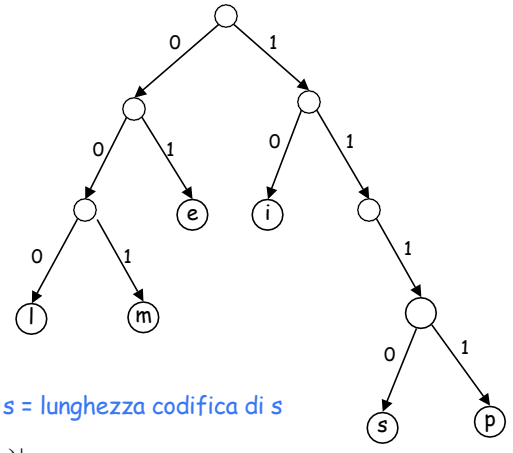


1110 10 001 1111 01 000
s i m p e l

8

8

Rappresentazione dei codici prefissi mediante alberi binari



Profondita` foglia associata a s = lunghezza codifica di s

$$ABL(T) = \sum_{s \in S} f(s)\,|depth(s)|$$

ABL(c) = ABL(T) se T e` l'albero del codice prefisso

9

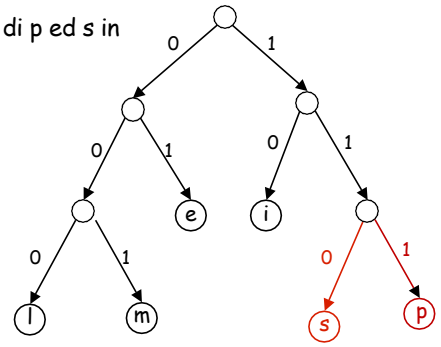
9

Rappresentazione dei codici prefissi mediante alberi binari

Domanda. Come possiamo rendere il codice piu` efficiente?

Risposta. Cambia le codifiche di p ed s in modo che siano piu` corte.

Ora l'albero e` pieno



PROGETTAZIONE DI ALGORITMI a.a. 2023-24
A. DE BONIS

10

10

Rappresentazione dei codici prefissi mediante alberi binari

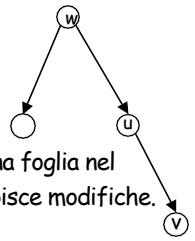
Definizione. Un albero binario e` pieno se ogni nodo o e` una foglia o ha due figli

Affermazione. L'albero binario corrispondente al codice prefisso ottimo e` pieno

Dim. per assurdo

Supponi che T sia un albero binario di un codice prefisso ottimo e che non sia pieno.

- Cio` vuol dire che c'e` un nodo con solo un figlio v
- Caso 1: u e` la radice: cancella u e usa v come radice
- Caso 2: u non e` la radice
  - Sia w il padre di u
  - Cancella u e rendi v figlio di w al posto di v
- In entrambi i casi il numero di bit per codificare ciascuna foglia nel sottoalbero di v e` diminuto. Il resto dell'albero non subisce modifiche.
- Il nuovo albero ha ABL piu` piccolo → contraddizione.



11

11

Codici prefissi ottimi: una falsa partenza

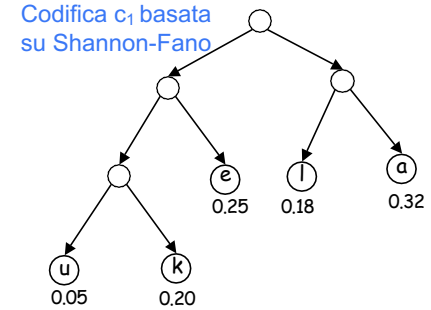
Domanda. Dove si trovano le lettere con alta frequenza nell'albero di un codice prefisso?

Risposta. Vicino alla radice.

Schema Greedy: Crea l'albero in modo top-down, dividi S in due insiemi S_1 e S_2 con frequenze simili. Costruisci ricorsivamente gli alberi per S_1 e d S_2. [Shannon-Fano, 1949]

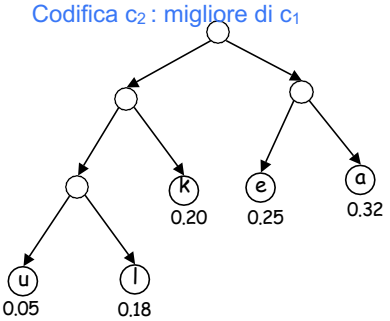
Esempio: f_a=0.32, f_e=0.25, f_k=0.20, f_l=0.18, f_u=0.05

Codifica c_1 basata su Shannon-Fano



$ABL(c_1)=.32 \cdot 2 + .25 \cdot 2 + .20 \cdot 3 + .18 \cdot 2 + .05 \cdot 3 = 2.25$

Codifica c_2 : migliore di c_1



$ABL(c_2)=.32 \cdot 2 + .25 \cdot 2 + .20 \cdot 2 + .18 \cdot 3 + .05 \cdot 3 = 2.23.$

12

12

Codici prefissi ottimi: Codifica di Huffman

Sia T* l'albero binario di un codice prefisso ottimo

Osservazione 1. Le frequenze piu` basse devono essere a livelli piu` bassi dell'albero T*.

Dim. Sia a l'etichetta di u e b quella di v in T*. Se per assurdo depth(u)>depth(v) e f(a)>f(b) allora possiamo scambiare le etichette di u e v ottenendo un albero T con ABL piu` piccolo di quello di T*→ contraddizione al fatto che T* e` ottimo

Osservazione 2. Per n > 1, il livello piu` in basso di T* contiene sempre almeno due foglie che sono figlie dello stesso padre.

Dim. Sia u la foglia a livello piu` basso. Abbiamo dimostrato che T* e` pieno per cui **u ha un fratello v.** Se per assurdo v fosse un nodo interno allora ci sarebbero nodi (discendenti di v) di profondita` maggiore rispetto ad u→ contraddizione al fatto che u si trova al livello piu` basso.

13

Codici prefissi ottimi: Codifica di Huffman

Affermazione. Esiste un albero T* di un codice prefisso ottimo in cui le 2 lettere con frequenza piu` piccola sono assegnate a foglie che sono sul livello piu` basso e che sono figlie dello stesso nodo in T*.

Dim.

Osservazione 2.→ Per n > 1, il livello piu` in basso di T* contiene almeno due foglie a e b che sono figlie dello stesso padre

Osservazione 1→ Tra le foglie sul livello piu` basso ci sono le due associate con i due simboli x e y con frequenza piu` piccola

Siano q e t le foglie etichettate con x e y. Supponiamo che in T* le foglie q e t non siano fratelli. Siccome T* e` pieno q e t devono entrambi avere un fratello. Sia u il fratello di q e v il fratello di t. Possiamo rendere q e t fratelli scambiando l'etichetta di u con quella di t. Questo scambio lascia inalterato lo ABL in quanto i quattro nodi q, t,u,v sono sullo stesso livello.

14

Codici prefissi ottimi: Codifica di Huffman

Schema greedy. [Huffman, 1952] Crea l'albero in modo bottom-up. Crea due foglie per i due simboli y e z con frequenza piu` piccola. Sostituisci y e z con la meta-lettera yz con frequenza pari a f(y)+f(z). Costruisci ricorsivamente l'albero per il codice prefisso per l'insieme dormato dalle lettere diverse da y e z con in piu` l'aggiunta si yz.

15

Algoritmo per la codifica di Huffman

```
Huffman(S) {
   if |S|=2 {
      return tree with root and 2 leaves
   } else {
      let y and z be lowest-frequency letters in S
      S' = S
      remove y and z from S'
      insert new letter ω in S' with f_ω=f_y+f_z
      T' = Huffman(S')
      T = add two children y and z to leaf ω from T'
      return T
   }
}
```

Complessita`:

T(n) = T(n-1) + $O(n)$ per n>2
T(n)=$O(1)$ per n=2
→ $O(n^2)$

O(n) dovuto alla ricerca delle lettere con minima frequenza

Se manteniamo le lettere di S in un Heap in cui le chiavi sono le frequenze delle lettere → T(n) = T(n-1) + $O(\log n)$ per n>2 → T(n)=$O(n \log n)$
+ il tempo per inizializzare (all'esterno) la coda a priorita` = $O(n \log n)$

Totale O(nlogn)

16

Codifica di Huffman: Ottimalita`

Teorema. La codifica di Huffman per S ottiene il minimo ABL tra tutti i codice prefissi.

La dimostrazione e` per induzione.

Prima pero` dimostriamo la seguente affermazione:

Affermazione. Sia S un insieme di lettere sia S'=S-{y,z}U{ω}, dove y se z sono le lettere di frequenza minore in S e ω e` il meta-simbolo con f(ω)=f(z)+f(y). Sia T un albero binario di un codice prefisso in cui y e z sono asssociate a due foglie figlie dello stesso padre. E sia T' l'albero ottenuto cancellando le foglie etichettate con y e z e trasformando il padre in una foglia etichettata con ω. Si ha che ABL(T')=ABL(T)-f_ω

$$\begin{aligned}
ABL(T) &= \sum_{x\in S} f_x \cdot \text{depth}_T(x) \\
&= f_y \cdot \text{depth}_T(y) + f_z \cdot \text{depth}_T(z) + \sum_{x\in S, x\neq y,z} f_x \cdot \text{depth}_T(x) \\
&= (f_y + f_z)\cdot(1+\text{depth}_{T'}(\omega)) + \sum_{x\in S, x\neq y,z} f_x \cdot \text{depth}_T(x) \\
&= f_\omega \cdot (1+\text{depth}_{T'}(\omega)) + \sum_{x\in S, x\neq y,z} f_x \cdot \text{depth}_T(x) \\
&= f_\omega + \sum_{x\in S'} f_x \cdot \text{depth}_{T'}(x) \\
&= f_\omega + ABL(T')
\end{aligned}$$

17

17

Codifica di Huffman: Ottimalita`

Dimostrazione dell'ottimalita` di Huffman (induzione su n=|S|)

Base: per n=2 il codice di Huffman assegna 0 a una lettera e 1 all'altra: non c'e modo di fare meglio

Ipotesi induttiva: Supponiamo che Huffman sia ottimo per n-1.

Sia T' l'albero di Huffman per S' . Per ipotesi induttiva T' e` ottimo per S'=S-{y,z}U{ω} (y se z sono le lettere di frequenza minore in S e ω e` il meta-simbolo con f(ω)=f(z)+f(y)

continua

18

Codifica di Huffman: Ottimalita`

Passo induttivo: Sia T l'albero di Huffman per S

- Supponiamo per assurdo che un altro albero binario Z sia ottimo per S e che ABL(Z)<ABL(T).
- Per l'affermazione nella slide 17, possiamo scegliere Z ottimo in modo che z e y in Z siano associate a due foglie figlie dello stesso padre e situate sul livello piu` basso
- Cancelliamo le foglie etichettate con y e z da Z trasformando il padre in foglia con etichetta ω creando cosi` un nuovo albero Z' che e` un albero di un codice prefisso per S'
- Z' non puo` avere ABL minore di T' in quanto T' per ipotesi induttiva e` ottimo per S'
- Dall'affermazione precedente (slide 16) si ha che

  $ABL(Z')=ABL(Z)-f_{\omega}$, $ABL(T')=ABL(T)-f_{\omega}$.

  perche' Z' e T' sono ottenuti cancellando le foglie etichettate con y e z da Z e T rispettivamente

  Siccome si ha anche, per ipotesi assurda, che ABL(Z) < ABL(T)

  → ABL(Z') < ABL(T')→ contrazione all' ipotesi induttiva