

## LEZIONE 8

### 8.1 ELEMENTI DELLA STRATEGIA GREEDY

Un algoritmo greedy opera facendo ad ogni "busto" la scelta che sembra essere migliore al momento.

Questa strategia non sempre produce una soluzione ottima.

### COME SVILUPPARE UN ALGORITMO GREEDY

1. Definire le sottostrutture ottime del problema.
2. Sviluppare una soluzione ricorsiva.
3. Ricorrere che facendo vere scelte greedy, si risolve un solo sottoproblema da risolvere.
4. Dimostrare che è sempre sicuro fare le scelte greedy.
5. Sviluppare un algoritmo ricorsivo che riempie tutte le scelte greedy.
6. Convertire l'algoritmo ricorsivo in un algoritmo iterativo

OSS: Questi passi costituiscono lo progetto di base di un algoritmo greedy.

Alterevolmente,

1. Formulare il problema di ottimizzazione con uno grafo in cui si fa una scelta e si riconduce con un sotto problema.
2. Dimostrare che esiste sempre una soluzione ottimale che fa la scelta greedy, così che le scelte greedy si faccia.
3. Dimostrare la sottostruzione ottimale del problema mostrando che, avendo fatto la scelta greedy, quello che rimane è un sotto problema tale che contiene una soluzione ottimale al sotto problema con le scelte greedy, si ottiene una soluzione ottimale al problema originario.

La strategia greedy non funziona sempre. Ma ci sono due ragioni per cui ciò può accadere:

- le proprietà di scelta greedy;
- le proprietà di sottostruzione ottimale.

## 8.2 PROPRIETÀ DI SCELTA GREEDY

Una delle proprietà giuste delle proprietà di scelta greedy se possono assenblare le scelte successive ottime globali, facendo scelte che sono localmente ottime.

OSS: La scelta fatta da un algoritmo greedy può dipendere dalle scelte precedenti, ma non può dipendere in alcun modo delle scelte successive o dei sottoproblemi successivi.

La programmazione dinamica risolve i sottoproblemi prima di fare le proprie scelte (approccio bottom-up).

Fu invece

Le strategie greedy fanno le loro proprie scelte prima di risolvere qualche sottoproblema (approccio top-down).

COME SI DEMOSTRA CHE UN PROBLEMA HA LA PROPRIETÀ DI SCELTA GREEDY?

In generale, si cerca di trovare la soluzione ottima (globale) ad un sotto problema e si mostra come uno di questi sostituisce le scelte greedy o qualche altra scelta, ottenendo così una soluzione simile, ma non peggiore.

COME SI DEMONSTRÀ LA PROPRIETÀ DI SOTTOSTRUTTURA OTTIMA QUANDO VOGLIAMO APPLICARE UNA STRATEGIA GREEDY?

È più semplice che nel caso generale. In questo caso, avendo risolto al sotto problema avevamo fatto le scelte greedy per il problema originario. Bisogna solo solo fare vedere che una soluzione ottima al sotto problema corrisponde alle scelte greedy già fatte, genera una soluzione ottima al problema originario.

### 8.3 DEMONSTRAZIONE DELLA SCELTA GREEDY PER IL PROBLEMA DELLA SELEZIONE DI ATTIVITÀ

## ACTIVITY SELECTION PROBLEM

input: un insieme  $S = \{a_1, \dots, a_n\}$  di attività, per ogni  $i$ ,  $a_i = [s_i, f_i]$  ( $0 \leq s_i < f_i < +\infty$ ).

goal: selezionare un sottinsieme  $A \subseteq S$  tale che

$A$  abbia cardinalità massima, e le attività in  $A$  siano compatibili, ovvero

$$\forall a_i, a_j \in A, a_i \neq a_j, [s_i, f_i] \cap [s_j, f_j] = \emptyset.$$

Po' si assume che le attività siano ordinate per finish-time crescente, ovvero

$$f_1 \leq f_2 \leq \dots \leq f_n.$$

THM: Se  $S_k$  è un sottoproblema non vuoto, e se qui l'attività di  $S_k$  con il minimo (primo) finish-time.

Allora, qui è basata su un sottinsieme di  $S_k$  con dimensione minima di attività compatibili.

Proof:

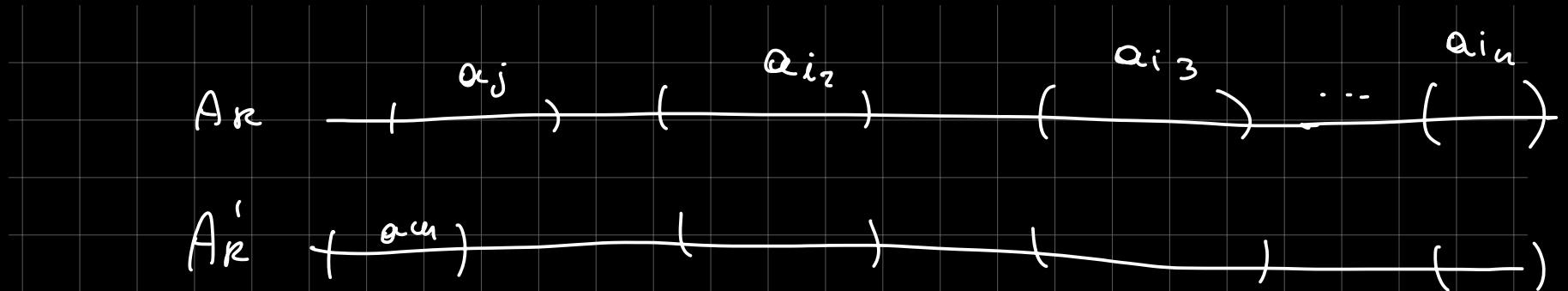
Sia  $A_k$  una soluzione ottima al problema  $S_k$ , ovvero che  $A_k$  sia un sottowspazio di ottimale competibili di  $S_k$  di cardinalità  $\ell$  minima.

Sia  $a_j$  l'ottimale di  $A_k$  con il valore frutto-brutto.

Se  $a_j = a_m$ , abbiamo frutto perché  $a_m \in A_k$  soluzione ottima di  $S_k$ .

Se  $a_j \neq a_m$ , si  $A'_k := (A_k - \{a_j\}) \cup \{a_m\}$ .

- $|A'_k| = |A_k|$  la cardinalità è uguale
- Le ottimale su  $A'_k$  sono competibili, perché le ottimale su  $(A_k - \{a_j\})$  sono competibili visto che le ottimale su  $A_k$  sono competibili. Inoltre,  $a_m = [s_m, f_m]$   
 $a_j = [s_j, f_j]$   
e  $f_m \leq f_j$ , ovvero  $a_m$  frutta poca



$\Rightarrow A'_k$  è soluzione ottenuta a  $S_k$ ,  $\alpha_m \in A'_k$ .  $\square$

#### 8.4 DISTRIBUZIONE DELLA SELEZIONE GREEDY PER L'ALGORITMO DI HUFFMAN

Riassumeremo il problema di progettare un codice binario su cui ogni carattere sia rappresentato da un'unica stringa binaria di lunghezza variabile.

L'idea dei codici a lunghezza variabile è assegnare codici corti a caratteri frequenti e codici lunghi a caratteri rari.

In un codice prefix-free nessuna codifica è prefissa di un'altra codifica.

Un prefix-free codice può essere rappresentato da un albero binario  $T$  le cui foglie sono i caratteri: si interpreta la codifica di un carattere come il cammino dalla radice alle foglie (caminata):

$$\begin{aligned} "0" &\leftrightarrow \text{via al nodo figlio sx} \\ "1" &\leftrightarrow \text{via al nodo figlio dx} \end{aligned}$$

L'albero  $T$  ha  $|C|$  foglie ( $C$  è l'alfabeto dei caratteri) ed esternamente  $|C|-1$  nodi interni.

$c.\text{freq}$ : frequenza del carattere  $c \in C$ ;

$d_T(c)$ : profondità (depth), ovvero la lunghezza di un cammino dalla radice alle foglie  $c$  in  $T$ .

Il # bit necessari a codificare un testo usando un tale albero  $T$  è

$$B(T) = \sum_{c \in C} c.\text{freq} \cdot d_T(c)$$

(costo dell'albero)

L'algoritmo di Huffman costruisce un albero di codifica a lunghezza variabile prefix-free di costo minimo utilizzando una strategia greedy.

Lemma: Siano  $x, y \in C$  con le frequenze minime. Allora, esiste un codice prefix-free ottimo su cui le codifiche di  $x$  e  $y$  hanno la stessa lunghezza (minima) e differiscono per un solo bit (l'ultimo).

Proof:  $T$  è l'albero che rappresenta il codice.

Siano  $a, b \in C$  foglie sorelle di  $T$  di minima profondità.

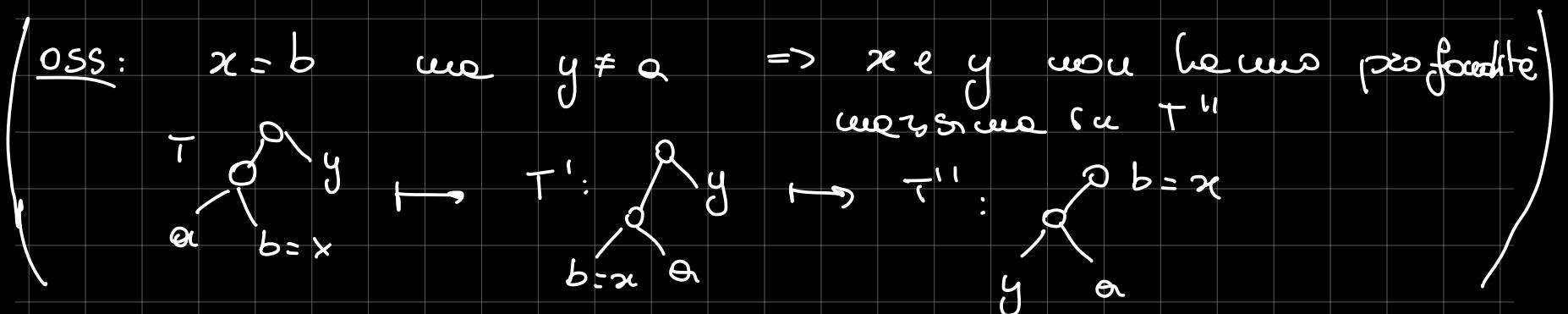
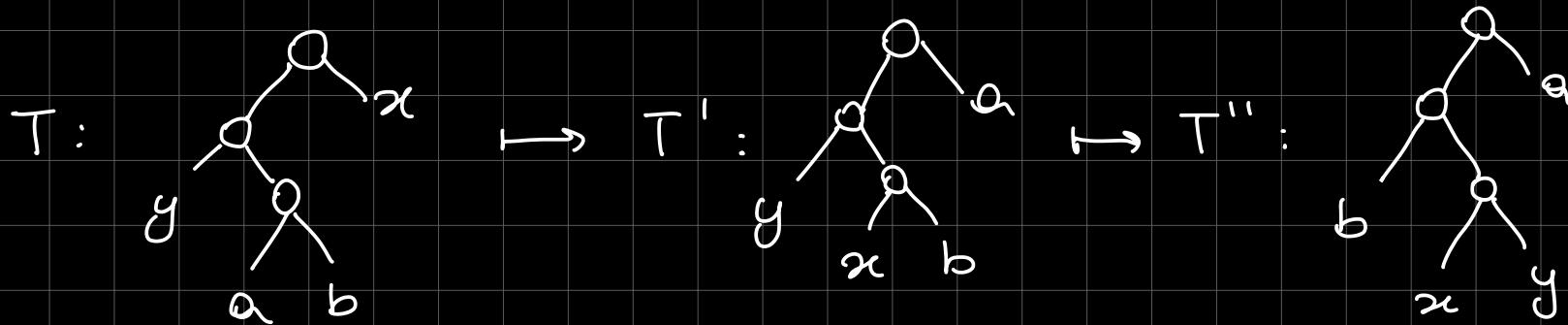
Wlog (without loss of generality), possiamo assumere  $a.freq \leq b.freq$  e  $x.freq \leq y.freq$ .

Allora che  $x.freq \leq a.freq$  e  $y.freq \leq b.freq$ .

Può accadere che  $x.freq = a.freq$  o  $y.freq = b.freq$ .

Quoltre, se  $x.freq = b.freq \Rightarrow x.freq = a.freq = b.freq = y.freq$   
 e il lemma è falso che lo sarebbe vero. Per ciò  
 $x.freq < b.freq \Rightarrow \boxed{x \neq b}$ .

Nell'albero  $T$  sceglieremo le posizioni di  $a$  e  $x$   
 ottenendo  $T'$ ; su  $T'$  sceglieremo le posizioni di  $b$  e  $y$   
 ottenendo  $T''$ .



$B(T)$  è corretto (ottenuto).  $B(T'')$  è corretto?

$$\begin{aligned}
 B(\tau) - B(\tau') &= \sum_{c \in C} c \cdot \text{freq} \cdot d_\tau(c) - \left( \sum_{c \in C} c \cdot \text{freq} \cdot d_{\tau'}(c) \right) = \\
 &= \underbrace{\sum_{c \in C \setminus \{a, x\}} c \cdot \text{freq} (d_\tau(c) - d_{\tau'}(c))}_{\text{"}} + \\
 &\quad + x \cdot \text{freq} (d_\tau(x) - d_{\tau'}(x)) + a \cdot \text{freq} (d_\tau(a) - d_{\tau'}(a)) \\
 &= x \cdot \text{freq} (d_\tau(x) - d_{\tau'}(x)) + a \cdot \text{freq} (d_\tau(a) - d_{\tau'}(x)) = \\
 &\quad \underbrace{(a \cdot \text{freq} - x \cdot \text{freq})}_{\text{V/0}} \underbrace{(d_\tau(a) - d_{\tau'}(x))}_{\text{V/0}} \geq 0
 \end{aligned}$$

$$\Rightarrow B(\tau) \geq B(\tau')$$

Analogamente,  $B(\tau') \geq B(\tau'')$

$\Rightarrow B(\tau'') \leq B(\tau') \leq B(\tau)$ , wie  $B(\tau)$  ist stabil

quindi  $B(\bar{T}') = B(T)$ .

Anche  $\bar{T}'$  è ottenuto dal contesto con le foglie di profondità massima, ovvero  $\text{height}$  più lunghe. □

Esercizio: Siano  $x, y \in C$  di frequenze diverse.

Sia  $C' = (C \setminus \{x, y\}) \cup \{z\}$ . Definiamo c.freq in  $C'$  uguale a c.freq in  $C$  e  $z.\text{freq} = x.\text{freq} + y.\text{freq}$ .

Sia  $T'$  un albero che rappresenta il codice prefix-free ottenuto per  $C'$ . Si  $\bar{T}'$  ottenuto rappresenta le foglie con un nodo interno genitore di  $x$  e  $y$ .

Allora,  $T$  è la rappresentazione di un codice prefix-free ottenuto per  $C$ .

Proof:  $\forall c \in C \setminus \{x, y\} : d_{\bar{T}}(c) = d_{T'}(c) \Rightarrow$

$$c.freq \cdot d_T(c) = c.freq \cdot d_{T'}(c)$$

Poiché  $d_T(x) = d_T(y) = d_T(z) + 1 \Rightarrow$

$$\underline{x.freq \cdot d_i(x)} + \underline{y.freq \cdot d_i(y)} = (\underline{x.freq} + \underline{y.freq})(d_{T'}(z) + 1)$$

Allora,  $B(T) = \sum_{c \in C \setminus \{x, y\}} c.freq d_T(c) + \underbrace{(x.freq + y.freq)(d_{T'}(z) + 1)}_{(z.freq)(d_{T'}(z) + 1)}$

$$B(\bar{T}) = B(T') + z.freq = B(T') + (x.freq + y.freq)$$

$$B(T') = B(\bar{T}) - (x.freq + y.freq)$$

Se  $T$  non fosse ottimo per  $C \Rightarrow \exists T''$  per  $C$ :

$$B(T'') < B(\bar{T}).$$

$\oplus$

$T''$  ha  $x$  e  $y$  come foglie di profondità minore (w.l.o.g., per le cui foglie).

$T'''$  è ottenuto da  $T''$  inserendo  $x$  e  $y$  e il loro  
germone comune con una foglia a tale cl<sup>o</sup>  
 $z \cdot freq = x \cdot freq + y \cdot freq$

$$B(T''') = B(T'') - (x \cdot freq + y \cdot freq) < B(T') - (x \cdot freq + y \cdot freq) = B(T')$$

che controddice l'<sup>o</sup>spostos per cui  
 $T'$  è ottimo per C.  $\square$