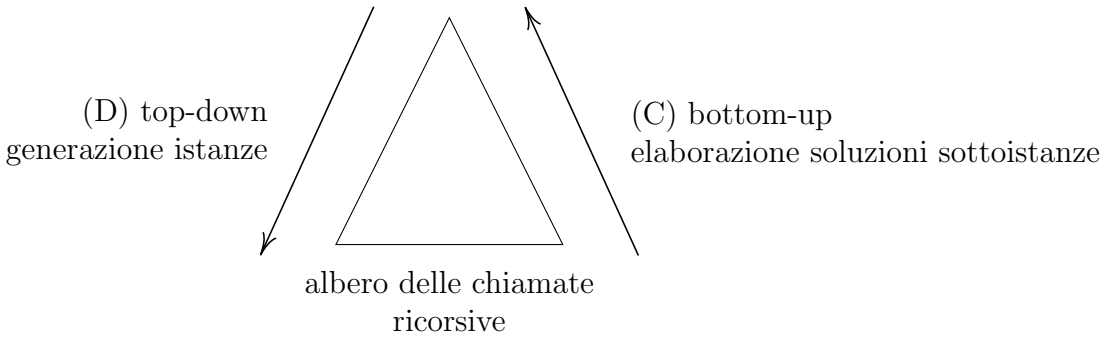
7 Programmazione Dinamica

7.1 Critica al Divide & Conquer (D&C)



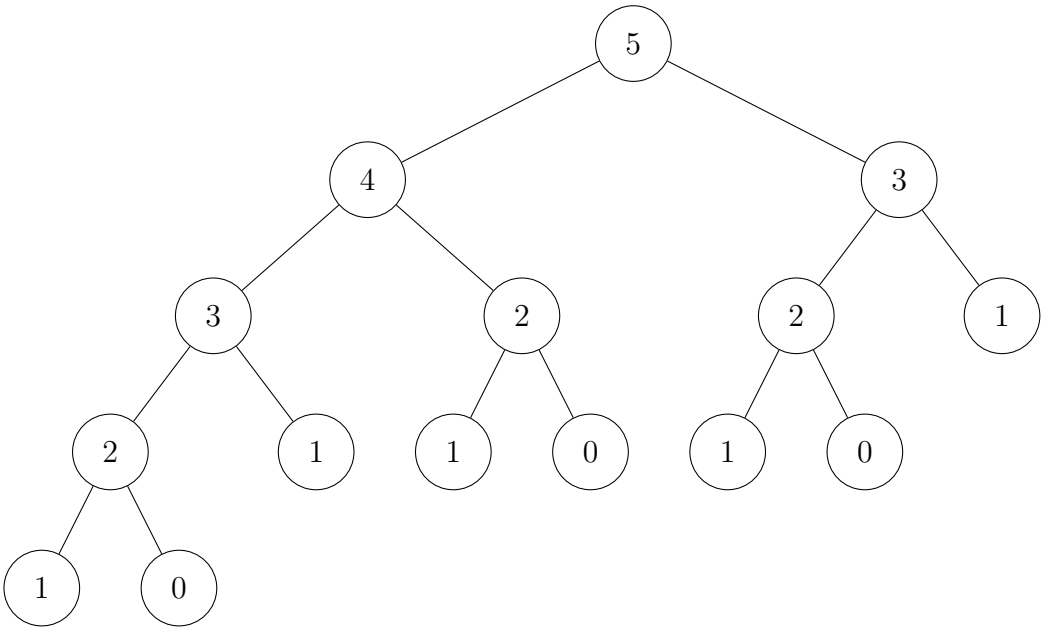
Il processo di soluzione non ha memoria, quindi le soluzioni di sottoistanze vanno ricalcolate.

Esempio Vediamo uno "spreco" usando D&C: la sequenza di Fibonacci.

$$F(n) = \begin{cases} 1 & \text{se } n = 0, 1 \\ F(n-1) + F(n-2) & \text{se } n \geq 2 \end{cases}$$

REC-FIB(n)
1 **if** $(n = 0)$ **or** $(n = 1)$
2 **return** 1
3 **return** REC-FIB($n - 1$) + REC-FIB($n - 2$)

Ad esempio, con $n = 5$



Vengono ricalcolate $F(3)$ e $F(2)$.

Complessità

$$T(n) = \begin{cases} 0 & \text{se } n = 0, 1 \qquad \text{(il "return" costa 0)} \\ T(n-1) + T(n-2) + 1 & \text{se } n \geq 2 \qquad \text{(il "+" costa 1)} \end{cases}$$

$$\begin{aligned} T(n) &\geq T(n-1) + T(n-2) + 1 \\ &\geq 2T(n-2) + 1 \\ &\geq 2(2T(n-2-2)+1) + 1 \\ &= 2^2 T(n-2-2) + 2 + 1 \\ &\geq 2^i T(n - 2 \cdot i) + \sum_{j=0}^{i-1} 2^j \end{aligned}$$

$$i_0 \to i = \left\lfloor \frac{n}{2} \right\rfloor$$

se n è pari: $2^{\frac{n}{2}} T(n - 2\frac{n}{2}) = 2^{\frac{n}{2}} T(0)$

se n è dispari: $\left\lfloor \frac{n}{2} \right\rfloor = \frac{n-1}{2}$

$$2^{\frac{n-1}{2}} T(n - 2\frac{n-1}{2}) = 2^{\frac{n-1}{2}} T(1)$$

Otteniamo

$$T(n) \geq \sum_{j=0}^{\lfloor \frac{n}{2} \rfloor - 1} 2^j = \Theta(2^{\frac{n}{2}})$$

In verità,

$$T(n) = \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

Vediamo ora una versione iterativa:

IT-FIB(n)
1 **if** $(n = 0)$ **or** $(n = 1)$
2 **return** 1
3 $F[0] = 1$
4 $F[1] = 1$
5 **for** $i = 2$ **to** n
6 $F[i] = F[i-1] + F[i-2]$
7 **return** $F[n]$

Complessità $\Theta(n)$

La programmazione dinamica salta la fase top-down.

7.2 Memoizzazione

È un ibrido tra il D&C e la programmazione dinamica che vuole mantenere la fase top-down pur cercando di ricordare le soluzioni ai sottoproblemi.

Def Un algoritmo **memoizzato** è costituito da due subroutine distinte:

1) **routine di inizializzazione**: risolve direttamente i casi base e inizializza una struttura dati che contiene le soluzioni ai casi base e gli elementi per tutte le sottoistanze da calcolare, inizializzate ad un valore di default

2) **routine ricorsiva**: esegue il codice D&C preceduto da un test sulla struttura dati per verificare se la soluzione è già stata calcolata e memorizzata. Se sì, si ritorna, altrimenti la si calcola ricorsivamente e la si memorizza nella struttura.

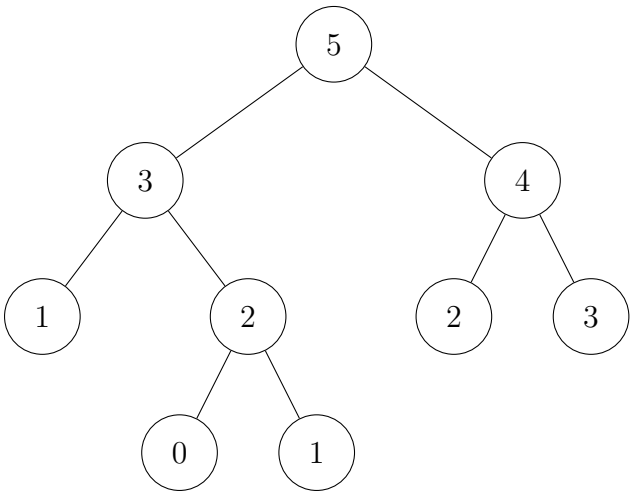
Esempio Riprendiamo l'esempio di prima sulla sequenza di Fibonacci.

```
INIT-FIB(n)
1  if (n = 0) or (n = 1)
2        return 1
3  F[0] = 1
4  F[1] = 1
5  for i = 2 to n
6        F[i] = 0
7  return REC-FIB(n)
```

Complessità $\Theta(n)$

```
REC-FIB(i)
1  if F[i] = 0
2        F[i] = REC-FIB(i − 2) + REC-FIB(i − 1)
3  return F[i]
```

Riprendiamo l'esempio di prima, con $n = 5$



Questa volta, $F(3)$ e $F(2)$ non vengono ricalcolate.
Abbiamo n foglie e $n - 1$ nodi interni (n parte da 0).

7.3 Problemi di Ottimizzazione

$I =$ insieme delle istanze
$S =$ insieme delle soluzioni
$\Pi \subseteq I \times S$
$\forall\ i \in I,\ S(i) = \{s \in S : (i, s) \in \Pi\} =$ insieme delle soluzioni ammissibili
funzione di costo $c\colon S \to \mathbb{R}$
Determinare, data $i \in I,\ s^* \in S(i) : c(s^*) = \min(/\max)\{c(s) : s \in S(i)\}$

Problema della raggiungibilità su un grafo orientato

$I = \{\langle G = (V, E),\ u,\ v\rangle\ :\ V \subseteq \mathbb{N},\ V \text{ finito},\ E \subseteq V \times V,\ u, v \in V\}$
$S = \{\langle v_1, v_2, \ldots, v_k\rangle\ :\ k \geq 1,\ v_i \in \mathbb{N} \quad \forall\ 1 \leq i \leq k\} \cup \{\varepsilon\} \qquad (\varepsilon = \text{cammino vuoto})$

$$\Big(i = \langle G = (V, E),\ u,\ v\rangle,\ s\Big) \in \Pi \iff \begin{cases} S = \varepsilon, \exists \text{ un cammino tra } u \text{ e } v \text{ in } G \\ S = \langle v_1, v_2, \ldots, v_k\rangle,\ v_1 = u,\ v_k = v, \\ \qquad (v_i, v_{i+1}) \in E \quad \forall\ 1 \leq i \leq k \end{cases}$$

$c(\langle v_1, v_2, \ldots, v_k\rangle) = k - 1$
$c(\varepsilon) = +\infty$

Caratteristiche Un problema di ottimizzazione, per essere risolto con la programmazione dinamica, deve avere le seguenti caratteristiche:

- struttura ricorsiva;
- esistenza di sottoistanze ripetute;
- spazio di sottoproblemi "piccolo".

Paradigma Generale

1. Caratterizza la struttura di una soluzione ottima s^* in funzione di soluzione ottime $s_1^*, s_2^*, \ldots, s_k^*$ di sottoistanze di taglia inferiore.
2. Determina una relazione di ricorrenza del tipo $c(s^*) = f(c(s_1^*), \ldots, c(s_k^*))$.
3. Calcola $c(s^*)$ impostando il calcolo in maniera bottom-up (oppure memoizzando).
4. Mantiene informazioni strutturali aggiuntive che permettono di ricostruire s^*.

7.4 Problemi su Stringhe

Def Dato un alfabeto finito Σ, una **stringa**

$$X = \langle x_1, x_2, \ldots, x_m \rangle, \quad x_i \in \Sigma \quad \forall\ 1 \leq i \leq m$$

è una concetazione finita di simboli in Σ.

$m = |X|$ = lunghezza di X
Σ^* = insieme di tutte le stringhe di lunghezza finita costruibili su Σ
ε = stringa vuota

Data una stringa X, il **prefisso** di X è

$$X_i = \langle x_1, x_2, \ldots, x_i \rangle, \quad 1 \leq i \leq m$$

Data una stringa X, il **suffisso** di X è

$$X^i = \langle x_i, x_{i+1}, \ldots, x_m \rangle, \quad 1 \leq i \leq m$$

Per convenzione $X_0 = X^{m+1} = \varepsilon$

Def Data una stringa X, la **sottostringa** di X è

$$X_{i\ldots j} = \langle x_i, x_{i+1}, \ldots, x_j \rangle, \quad 1 \leq i \leq j \leq m$$

Per convenzione $X_{i\ldots j} = \varepsilon \quad$ se $i > j$

possibili sottostringhe di una stringa con m caratteri:

$$\underset{\substack{\uparrow \\ i \neq j}}{\binom{m}{2}} + \underset{\substack{\uparrow \\ i = j}}{m} + \underset{\substack{\uparrow \\ \varepsilon}}{1} = \frac{m(m+1)}{2} = \Theta(m^2)$$

Lo spazio delle sottostringhe "non è troppo grande".

Def Data una stringa

$$X = \langle x_1, x_2, \ldots, x_m \rangle \in \Sigma^*$$

e

$$Z = \langle z_1, z_2, \ldots, z_k \rangle \in \Sigma^*$$

si dice che Z è **sottosequenza** di X se $\exists$ una successione crescente di indici

$$1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq m \ : \ z_j = x_{ij} \quad \forall\ 1 \leq j \leq k$$

Esempio

$X = \langle a, b, c, b, b, d \rangle$
$Z_1 = \langle a, b, c \rangle = X_{1\ldots3}$
$Z_2 = \langle a, c, b \rangle \qquad i_1 = 1, \quad i_2 = 3, \quad i_3 = 4 \text{ o } 5$
$Z_3 = \langle b, b \rangle = X_{4\ldots5} \qquad i_1 = 2, \quad i_2 = 5$

possibili sottosequenze di una stringa con m caratteri:

$$\sum_{k=0}^{m} \underbrace{\binom{m}{k}}_{\substack{\uparrow \\ \text{stringhe lunghe } k \\ \text{prese da un insieme} \\ \text{di } m \text{ elementi}}} = 2^m$$

7.5 Longest Common Subsequence (LCS)

7.5.1 Problema di Ottimizzazione

Date due stringhe X, Y determina Z tale che:

1) Z è sottosequenza di X e Y;

2) Z è la più lunga tra tutte le sottosequenze comuni.

Esempio

$X = \langle a, b, c, b, b, d \rangle$
$Y = \langle a, d, c, c, b, d \rangle$
$Z = \langle a, c, b, d \rangle$ è una LCS (in questo caso è l'unica)
$i_1 = 1, \quad i_2 = 3, \quad i_3 = 4$ o $5, \quad i_4 = 6$
$j_1 = 1, \quad j_2 = 3$ o $4, \quad j_3 = 5, \quad j_4 = 6$

Risolvo il problema:

$|X| = m$
$|Y| = n$

L'approccio "brute force" ha complessità $\Omega(2^m \cdot 2^n)$.

Devo cercare di individuare una proprietà di sottostruttura, cioè la LCS deve "nascondere" al suo interno LCS di qualche stringa più piccola di X e Y.

$X = \langle b, c, f, a \rangle$
$Y = \langle c, f, d, a \rangle$
$Z = LCS(X, Y) = \langle Z', a \rangle \qquad$ con $Z' = LCS(X_3, Y_3)$

$X = \langle X', a \rangle$
$Y = \langle Y', b \rangle$
Z o non termina con a, o non termina con b
$Z = LCS(X', Y)$ o $LCS(X, Y')$
$S = \{LCS(X_i, Y_j) : 0 \leq i \leq m,\ 0 \leq j \leq n\}, \quad |S| = (m{+}1)(n{+}1)$
$\uparrow \quad \uparrow$
$\varepsilon \quad \varepsilon$

7.5.2 Proprietà di Sottostruttura Ottima

Dati i prefissi

$X_i = \langle x_1, x_2, \ldots, x_i \rangle$
$Y_i = \langle y_1, y_2, \ldots, y_j \rangle$
Sia $Z = \langle z_1, z_2, \ldots, z_k \rangle = LCS(X_i, Yj)$

0. caso base: o $i = 0$ o $j = 0$
   $\Rightarrow Z = \varepsilon$

1. $i, k > 0$
   se $x_i = y_j$ allora

   (a) $z_k = x_i (= y_j)$
   (b) $Z_{k-1} = LCS(X_{i-1}, Y_{j-1})$

2. $i, j > 0$
   se $x_i \neq y_j$ allora
   Z è la stringa di lunghezza massima tra $LCS(X_i, Y_{j-1})$ e $LCS(X_{i-1}, Y_j)$

Dimostrazione

0. banale

1. $x_i = y_j$

   $Z = LCS(X_i, Y_j) = \langle z_1, z_2, \ldots, z_k \rangle = \langle x_{i_1}, x_{i_2}, \ldots, x_{i_k} \rangle = \langle y_{j_1}, y_{j_2}, \ldots, y_{j_k} \rangle$
   $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq i, \qquad 1 \leq j_1 \leq j_2 \leq \cdots \leq j_k \leq j$

   (a) Ragioniamo per assurdo

   $z_k = x_{i_k} = y_{j_k}$
   $z_k \neq (x_i = y_j)$
   $\Rightarrow i_k < i, \quad j_k < j$
   $Z' = \langle Z, x_i \rangle$
   $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq i_{k+1} = i, \qquad 1 \leq j_1 \leq j_2 \leq \cdots \leq j_k \leq j_{k+1} = j$

   (b) Devo dimostrare che

   $Z_{k-1} = LCS(X_{i-1}, Y_{j-1})$
   $Z_{k-1} = \langle x_{i_1}, x_{i_2}, \ldots, x_{i_{k-1}} \rangle = \langle y_{j_1}, y_{j_2}, \ldots, y_{j_{k.1}} \rangle$
   $i_{k-1} \leq i - 1 < i$
   $Z_{k-1} = CS(X_{i-1}, Y_{j-1})$

Ora dimostro che

$$Z_{k-1} = LCS(X_{i-1}, Y_{j-1})$$

Suppongo per assurdo che

$$\begin{aligned} &Z_{k-1} \neq LCS(X_{i-1}, Y_{j-1}) \\ &\Rightarrow \exists\ Z' \text{ con } |Z'| \geq k \\ &\Rightarrow \text{creo } Z'' = \langle\ Z'\ ,\ x_i(= y_j)\rangle \\ &\qquad\qquad\qquad\quad \uparrow \quad \uparrow \\ &\qquad\qquad\qquad\quad \geq k \quad 1 \Rightarrow \geq k+1 \end{aligned}$$

2. (come esercizio)