Rod - Cut

Il problema presenta una sottostruttura ottima questo perché la soluzione è data dalla somma del risultato dei sottoproblemi

soluzione ottima

$$r_n = \begin{cases} 0 & \text{se } n = 0 \\ \max\limits_{1 \le i \le n} \{r_i + r_{n-i}\} & \text{se } n \ge 1 \end{cases}$$

Usiamo un array per memorizzare i valori

Facendo un albero di ricorsione, creiamo un albero con archi pesati e si calcola il cammino con costo maggiore

SOLUZIONE RICORSIVA

```
MAX-ROD-CUT(P, n)        → array che memorizza i valori per i tagli
IF n == 0
RETURN 0
q ← -∞
FOR i ← 1 TO n DO
 IF  q < P[i] + MAX-CUT-ROD(P, n-i)
     q ← P[i] + MAX-CUT-ROD(P, n-i)
RETURN q
```

SOLUZIONE CON PROGRAMMAZIONE DINAMICA

CON L'APPROCCIO TOP-DOWN AVREMO SEMPRE BISOGNO DELLA RICORSIONE, PERCIÒ USIAMO UN APPROCCIO BOTTOM-UP

```
MAX-ROD-CUT(P, n)
 r ← NEW ARRAY [n+1]
 r[0] ← 0
 FOR i ← 1 TO n
  r[i] ← -∞
 FOR i ← 1 TO n DO
   FOR j ← 1 TO i DO
   IF r[i] < P[j] + r[i-j]
   r[i] ← P[j] + r[i-j]
 RETURN r[n]
```

Nell'ultima locazione dell'array avremo una soluzione ottima per il nostro sottoproblema

Per sapere le DIMENSIONE dei vari tagli inseriamo un array ausiliario S che salva la dimensione del primo pezzo usato per l'assegnazione di r[i] così facendo sapremo trovare a ritroso i tagli effettuati, andando a sottrarre di volta in volta il primo taglio effettuato