Lezione 13

Sommario della Lezione

Programmazione Dinamica

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**:

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore
- **Calcolare la soluzione globale al problema in modo ricorsivo,**

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore
- **Calcolare la soluzione globale al problema in modo ricorsivo,** facendo precedere ciascuna chiamata ricorsiva con un controllo per verificare se la soluzione al relativo sottoproblema è stata già calcolata.

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore
- **Calcolare la soluzione globale al problema in modo ricorsivo,** facendo precedere ciascuna chiamata ricorsiva con un controllo per verificare se la soluzione al relativo sottoproblema è stata già calcolata.

Gli algoritmi di Programmazione Dinamica hanno bisogno di memorizzare le soluzioni ai sottoproblemi intermedi.

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore
- **Calcolare la soluzione globale al problema in modo ricorsivo,** facendo precedere ciascuna chiamata ricorsiva con un controllo per verificare se la soluzione al relativo sottoproblema è stata già calcolata.

Gli algoritmi di Programmazione Dinamica hanno bisogno di memorizzare le soluzioni ai sottoproblemi intermedi. Spesso ciò viene effettuato memorizzandole in tabelle.

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore
- **Calcolare la soluzione globale al problema in modo ricorsivo,** facendo precedere ciascuna chiamata ricorsiva con un controllo per verificare se la soluzione al relativo sottoproblema è stata già calcolata.

Gli algoritmi di Programmazione Dinamica hanno bisogno di memorizzare le soluzioni ai sottoproblemi intermedi. Spesso ciò viene effettuato memorizzandole in tabelle.
Il secondo passo può essere sostituito in algoritmi iterativi con il seguente:

- **Calcolare le soluzioni ai sottoproblemi in maniera "bottom-up"**:

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore
- **Calcolare la soluzione globale al problema in modo ricorsivo,** facendo precedere ciascuna chiamata ricorsiva con un controllo per verificare se la soluzione al relativo sottoproblema è stata già calcolata.

Gli algoritmi di Programmazione Dinamica hanno bisogno di memorizzare le soluzioni ai sottoproblemi intermedi. Spesso ciò viene effettuato memorizzandole in tabelle.
Il secondo passo può essere sostituito in algoritmi iterativi con il seguente:

- **Calcolare le soluzioni ai sottoproblemi in maniera "bottom-up"**:scrivere un algoritmo che parta con i casi base della ricorrenza e proceda via via risolvendo problemi di taglia sempre maggiore,

Il progetto di algoritmi basati sulla Programmazione Dinamica prevede due passi separati:

- **Formulare il problema in termini ricorsivi**: ovvero scrivere una espressione per la soluzione all'intero problema che sia una combinazione di soluzioni a sottoproblemi di taglia minore
- **Calcolare la soluzione globale al problema in modo ricorsivo,** facendo precedere ciascuna chiamata ricorsiva con un controllo per verificare se la soluzione al relativo sottoproblema è stata già calcolata.

Gli algoritmi di Programmazione Dinamica hanno bisogno di memorizzare le soluzioni ai sottoproblemi intermedi. Spesso ciò viene effettuato memorizzandole in tabelle.
Il secondo passo può essere sostituito in algoritmi iterativi con il seguente:

- **Calcolare le soluzioni ai sottoproblemi in maniera "bottom-up"**:scrivere un algoritmo che parta con i casi base della ricorrenza e proceda via via risolvendo problemi di taglia sempre maggiore, considerandoli nell'ordine corretto

Input: Un valore monetario V,

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$,

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$.

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$. In altri termini, la moneta generica i vale $v[i]$ Euro.

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$. In altri termini, la moneta generica i vale $v[i]$ Euro.

Output: Il minimo numero di monete il cui valore totale sia esattamente pari a V.

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$. In altri termini, la moneta generica i vale $v[i]$ Euro.

Output: Il minimo numero di monete il cui valore totale sia esattamente pari a V. (Assumiamo di avere a disposizione un numero illimitato di monete di valore $v[i]$, per ogni i)

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$. In altri termini, la moneta generica i vale $v[i]$ Euro.

Output: Il minimo numero di monete il cui valore totale sia esattamente pari a V. (Assumiamo di avere a disposizione un numero illimitato di monete di valore $v[i]$, per ogni i)

In altri termini, indicato con $a_i \geq 0$ il numero di monete di valore $v[i]$ che usiamo

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$. In altri termini, la moneta generica i vale $v[i]$ Euro.

Output: Il minimo numero di monete il cui valore totale sia esattamente pari a V. (Assumiamo di avere a disposizione un numero illimitato di monete di valore $v[i]$, per ogni i)

In altri termini, indicato con $a_i \geq 0$ il numero di monete di valore $v[i]$ che usiamo (che può anche essere pari a zero, nel senso che la moneta i-esima non viene utilizzata),

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$. In altri termini, la moneta generica i vale $v[i]$ Euro.

Output: Il minimo numero di monete il cui valore totale sia esattamente pari a V. (Assumiamo di avere a disposizione un numero illimitato di monete di valore $v[i]$, per ogni i)

In altri termini, indicato con $a_i \geq 0$ il numero di monete di valore $v[i]$ che usiamo (che può anche essere pari a zero, nel senso che la moneta i-esima non viene utilizzata), vogliamo *minimizzare* il numero totale di monete usate, pari a

$$a_1 + a_2 + \ldots + a_n$$

Input: Un valore monetario V, un insieme di monete che denoteremo con l'insieme $\{1, \ldots, n\}$, i cui valori (ad es., in Euro)) sono contenuti nel vettore $v = v[1] \ldots v[n]$, con $v[1] > v[2] > \ldots > v[n] = 1$. In altri termini, la moneta generica i vale $v[i]$ Euro.

Output: Il minimo numero di monete il cui valore totale sia esattamente pari a V. (Assumiamo di avere a disposizione un numero illimitato di monete di valore $v[i]$, per ogni i)

In altri termini, indicato con $a_i \geq 0$ il numero di monete di valore $v[i]$ che usiamo (che può anche essere pari a zero, nel senso che la moneta i-esima non viene utilizzata), vogliamo *minimizzare* il numero totale di monete usate, pari a

$$a_1 + a_2 + \ldots + a_n$$

sotto la condizione che

$$\sum_{i=1}^{n} a_i v[i] = V$$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro,

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro,

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro.

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili.

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$,

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$, di valore totale $\sum_{i=1}^{4} a_i v[i] = 26$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$, di valore totale $\sum_{i=1}^{4} a_i v[i] = 26$

2. $a_1 = 2$, $a_2 = 0$ $a_3 = 3$, $a_4 = 0$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$, di valore totale $\sum_{i=1}^{4} a_i v[i] = 26$

2. $a_1 = 2$, $a_2 = 0$ $a_3 = 3$, $a_4 = 0 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 5$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$, di valore totale $\sum_{i=1}^{4} a_i v[i] = 26$

2. $a_1 = 2$, $a_2 = 0$ $a_3 = 3$, $a_4 = 0 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 5$ di valore totale $\sum_{i=1}^{4} a_i v[i] = 2 \cdot 10 + 3 \cdot 2 = 26$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$, di valore totale $\sum_{i=1}^{4} a_i v[i] = 26$

2. $a_1 = 2$, $a_2 = 0$ $a_3 = 3$, $a_4 = 0 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 5$ di valore totale $\sum_{i=1}^{4} a_i v[i] = 2 \cdot 10 + 3 \cdot 2 = 26$

3. $a_1 = 2$ $a_2 = 1$ $a_3 = 0$, $a_4 = 1$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$, di valore totale $\sum_{i=1}^{4} a_i v[i] = 26$

2. $a_1 = 2$, $a_2 = 0$ $a_3 = 3$, $a_4 = 0 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 5$ di valore totale $\sum_{i=1}^{4} a_i v[i] = 2 \cdot 10 + 3 \cdot 2 = 26$

3. $a_1 = 2$ $a_2 = 1$ $a_3 = 0$, $a_4 = 1 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 4$

Vediamo un esempio. Sia $V = 26$ (Euro), l'insieme delle monete pari a $\{1, 2, 3, 4\}$ ed il vettore dei valori delle monete dato da

$$v[1] = 10, v[2] = 5, v[3] = 2, v[4] = 1$$

ovvero abbiamo a disposizione monete di valore pari a 10 Euro, monete di 5 Euro, monete di 2 Euro e monete del valore di 1 Euro. Vogliamo "cambiare un assegno" di 26 Euro, usando il minor numero di monete possibili. Vi sono diverse possibili soluzioni::

1. $a_1 = a_2 = a_3 = 0$, $a_4 = 26 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 26$, di valore totale $\sum_{i=1}^{4} a_i v[i] = 26$

2. $a_1 = 2$, $a_2 = 0$ $a_3 = 3$, $a_4 = 0 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 5$ di valore totale $\sum_{i=1}^{4} a_i v[i] = 2 \cdot 10 + 3 \cdot 2 = 26$

3. $a_1 = 2$ $a_2 = 1$ $a_3 = 0$, $a_4 = 1 \quad \Rightarrow$ il numero totale di monete usato è $a_1 + a_2 + a_3 + a_4 = 4$ di valore totale
$\sum_{i=1}^{4} a_i v[i] = 2 \cdot 10 + 1 \cdot 5 + 1 \cdot 1 = 26$

Passo 1 di PD: **Formulare il problema ricorsivamente**.

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

- Domanda: E quali sono i sottoproblemi del problema di partenza (che chiede di esprimere il valore V usando il minor numero di monete, ognuna delle quali di un possibile valore $v[1] > v[2] > \ldots > v[n] = 1$)?

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

- Domanda: E quali sono i sottoproblemi del problema di partenza (che chiede di esprimere il valore V usando il minor numero di monete, ognuna delle quali di un possibile valore $v[1] > v[2] > \ldots > v[n] = 1$)?
- Risposta: Sono tutti i sottoproblemi che si ottengono qualora si voglia esprimere un **qualsiasi** valore $0 \leq j \leq V$

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

- Domanda: E quali sono i sottoproblemi del problema di partenza (che chiede di esprimere il valore V usando il minor numero di monete, ognuna delle quali di un possibile valore $v[1] > v[2] > \ldots > v[n] = 1$)?
- Risposta: Sono tutti i sottoproblemi che si ottengono qualora si voglia esprimere un **qualsiasi** valore $0 \leq j \leq V$ usando il minor numero di monete, ognuna delle quali di un possibile valore $v[i] > v[i+1] > \ldots > v[n] = 1$, $i \geq 1$

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

- Domanda: E quali sono i sottoproblemi del problema di partenza (che chiede di esprimere il valore V usando il minor numero di monete, ognuna delle quali di un possibile valore $v[1] > v[2] > \ldots > v[n] = 1$)?
- Risposta: Sono tutti i sottoproblemi che si ottengono qualora si voglia esprimere un **qualsiasi** valore $0 \leq j \leq V$ usando il minor numero di monete, ognuna delle quali di un possibile valore $v[i] > v[i+1] > \ldots > v[n] = 1$, $i \geq 1$

In altri termini, un generico **sottoproblema** è individuato dal generico **valore** j, per $j = 0, \ldots, V$

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

- Domanda: E quali sono i sottoproblemi del problema di partenza (che chiede di esprimere il valore V usando il minor numero di monete, ognuna delle quali di un possibile valore $v[1] > v[2] > \ldots > v[n] = 1$)?
- Risposta: Sono tutti i sottoproblemi che si ottengono qualora si voglia esprimere un **qualsiasi** valore $0 \leq j \leq V$ usando il minor numero di monete, ognuna delle quali di un possibile valore $v[i] > v[i+1] > \ldots > v[n] = 1$, $i \geq 1$

In altri termini, un generico **sottoproblema** è individuato dal generico **valore** j, per $j = 0, \ldots, V$ e dal **sottoinsieme** di monete $\{i, \ldots, n\}$, ovvero dal sottovettore dei loro valori $v[i] \ldots v[n]$.

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

- Domanda: E quali sono i sottoproblemi del problema di partenza (che chiede di esprimere il valore V usando il minor numero di monete, ognuna delle quali di un possibile valore $v[1] > v[2] > \ldots > v[n] = 1$)?
- Risposta: Sono tutti i sottoproblemi che si ottengono qualora si voglia esprimere un **qualsiasi** valore $0 \leq j \leq V$ usando il minor numero di monete, ognuna delle quali di un possibile valore $v[i] > v[i+1] > \ldots > v[n] = 1$, $i \geq 1$

In altri termini, un generico **sottoproblema** è individuato dal generico **valore** j, per $j = 0, \ldots, V$ e dal **sottoinsieme** di monete $\{i, \ldots, n\}$, ovvero dal sottovettore dei loro valori $v[i] \ldots v[n]$.
Denotiamo con $C(i, j)$ il minimo numero di monete necessario per esprimere la somma $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$,

Passo 1 di PD: **Formulare il problema ricorsivamente**. Ovvero, scrivere la soluzione all'intero problema come una combinazione di soluzioni a sottoproblemi di taglia minore

- Domanda: E quali sono i sottoproblemi del problema di partenza (che chiede di esprimere il valore V usando il minor numero di monete, ognuna delle quali di un possibile valore $v[1] > v[2] > \ldots > v[n] = 1$)?
- Risposta: Sono tutti i sottoproblemi che si ottengono qualora si voglia esprimere un **qualsiasi** valore $0 \leq j \leq V$ usando il minor numero di monete, ognuna delle quali di un possibile valore $v[i] > v[i+1] > \ldots > v[n] = 1$, $i \geq 1$

In altri termini, un generico **sottoproblema** è individuato dal generico **valore** j, per $j = 0, \ldots, V$ e dal **sottoinsieme** di monete $\{i, \ldots, n\}$, ovvero dal sottovettore dei loro valori $v[i] \ldots v[n]$.
Denotiamo con $C(i, j)$ il minimo numero di monete necessario per esprimere la somma $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, (noi siamo interessati a $C(1, V)$)

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$.

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$ con le monete di valore $v[i], \ldots, v[3]$, con $1 \leq i \leq 3$.

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$ con le monete di valore $v[i], \ldots, v[3]$, con $1 \leq i \leq 3$.

	j												
	0	1	2	3	4	5	6	7	8	9	10	11	12

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$ con le monete di valore $v[i], \ldots, v[3]$, con $1 \leq i \leq 3$.

	j												
	0	1	2	3	4	5	6	7	8	9	10	11	12
3	0	1	2	3	4	5	6	7	8	9	10	11	12

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$ con le monete di valore $v[i], \ldots, v[3]$, con $1 \leq i \leq 3$.

	j												
	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	1	2	3	4	5	1	2	3	4	5	6	2
3	0	1	2	3	4	5	6	7	8	9	10	11	12

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$ con le monete di valore $v[i], \ldots, v[3]$, con $1 \leq i \leq 3$.

		j												
		0	1	2	3	4	5	6	7	8	9	10	11	12
i	1	0	1	2	3	4	5	1	2	3	4	1	2	2
	2	0	1	2	3	4	5	1	2	3	4	5	6	2
	3	0	1	2	3	4	5	6	7	8	9	10	11	12

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$ con le monete di valore $v[i], \ldots, v[3]$, con $1 \leq i \leq 3$.

		j												
		0	1	2	3	4	5	6	7	8	9	10	11	12
i	1	0	1	2	3	4	5	1	2	3	4	1	2	2
	2	0	1	2	3	4	5	1	2	3	4	5	6	2
	3	0	1	2	3	4	5	6	7	8	9	10	11	12

Ad esempio, $C(2, 8) = 3$, dovendo necessariamente usare una moneta di valore 6 e due monete di valore 1

Esempio: $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$.

Calcoliamo le entrate della matrice in cui l'indice di riga i specifica che sono disponibili le monete di valore $v[i], \ldots, v[3]$. L'indice di colonna j specifica il valore monetario totale che si deve esprimere.

La generica entrata nella riga i e colonna j della matrice indica il minor numero di monete necessario per poter esprimere il valore monetario j, $0 \leq j \leq 12$ con le monete di valore $v[i], \ldots, v[3]$, con $1 \leq i \leq 3$.

		j												
		0	1	2	3	4	5	6	7	8	9	10	11	12
i	1	0	1	2	3	4	5	1	2	3	4	1	2	2
	2	0	1	2	3	4	5	1	2	3	4	5	6	2
	3	0	1	2	3	4	5	6	7	8	9	10	11	12

Ad esempio, $C(2, 8) = 3$, dovendo necessariamente usare una moneta di valore 6 e due monete di valore 1 e $C(1, 12) = 2$

Vediamo come esprimere in maniera ricorsiva i valori $C(i, j)$

Vediamo come esprimere in maniera ricorsiva i valori $C(i, j) =$ **minimo** numero di monete necessario per esprimere valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$.

Vediamo come esprimere in maniera ricorsiva i valori $C(i, j) =$ **minimo** numero di monete necessario per esprimere valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$.

Possono accadere due casi:

- Nella soluzione che ci fornisce il minimo numero di monete $C(i, j)$ per esprimere il valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$,

Vediamo come esprimere in maniera ricorsiva i valori $C(i, j) =$ **minimo** numero di monete necessario per esprimere valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$.

Possono accadere due casi:

- Nella soluzione che ci fornisce il minimo numero di monete $C(i, j)$ per esprimere il valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta i-esima di valore $v[i]$ **non compare**.

Vediamo come esprimere in maniera ricorsiva i valori $C(i, j) =$ **minimo** numero di monete necessario per esprimere valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$.

Possono accadere due casi:

- Nella soluzione che ci fornisce il minimo numero di monete $C(i, j)$ per esprimere il valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta i-esima di valore $v[i]$ **non compare**. Ovvero $C(i, j)$ può essere calcolata tenendo in considerazione solo le monete di valore $v[i+1] > \ldots > v[n]$.

Vediamo come esprimere in maniera ricorsiva i valori $C(i,j) =$ **minimo** numero di monete necessario per esprimere valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$.

Possono accadere due casi:

- Nella soluzione che ci fornisce il minimo numero di monete $C(i,j)$ per esprimere il valore monetario $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta i-esima di valore $v[i]$ **non compare**. Ovvero $C(i,j)$ può essere calcolata tenendo in considerazione solo le monete di valore $v[i+1] > \ldots > v[n]$.

  Detto in altri termini, in questo caso vale che $C(i,j) = C(i+1,j)$.

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$,

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i, j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i, j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i, j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i, j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).
  L'osservazione chiave è che $k = C(i, j - v[i])$.

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i, j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).
  L'osservazione chiave è che $k = C(i, j - v[i])$.
  Se fosse $k > C(i, j - v[i])$,

- Nella soluzione che ci dà il minimo numero di monete $C(i, j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i, j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).
  L'osservazione chiave è che $k = C(i, j - v[i])$.
  Se fosse $k > C(i, j - v[i])$, potremmo sostituire le monete $i_1, \ldots, i_k$ con le $C(i, j - v[i])$ monete che sappiamo, per definizione di $C(\cdot, \cdot)$, avere come somma di valori uguale a $j - v[i]$,

- Nella soluzione che ci dà il minimo numero di monete $C(i,j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i,j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).
  L'osservazione chiave è che $k = C(i, j - v[i])$.
  Se fosse $k > C(i, j - v[i])$, potremmo sostituire le monete $i_1, \ldots, i_k$ con le $C(i, j - v[i])$ monete che sappiamo, per definizione di $C(\cdot,\cdot)$, avere come somma di valori uguale a $j - v[i]$, ed ottenere, usando poi la moneta di valore $v[i]$, un valore totale pari a $(j - v[i]) + v[i] = j$,

- Nella soluzione che ci dà il minimo numero di monete $C(i,j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i,j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).
  L'osservazione chiave è che $k = C(i, j - v[i])$.
  Se fosse $k > C(i, j - v[i])$, potremmo sostituire le monete $i_1, \ldots, i_k$ con le $C(i, j - v[i])$ monete che sappiamo, per definizione di $C(\cdot,\cdot)$, avere come somma di valori uguale a $j - v[i]$, ed ottenere, usando poi la moneta di valore $v[i]$, un valore totale pari a $(j - v[i]) + v[i] = j$, usando un numero di monete pari a $1 + C(i, j - v[i]) < 1 + k = C(i,j)$,

- Nella soluzione che ci dà il minimo numero di monete $C(i,j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i,j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).
  L'osservazione chiave è che $k = C(i, j - v[i])$.
  Se fosse $k > C(i, j - v[i])$, potremmo sostituire le monete $i_1, \ldots, i_k$ con le $C(i, j - v[i])$ monete che sappiamo, per definizione di $C(\cdot, \cdot)$, avere come somma di valori uguale a $j - v[i]$, ed ottenere, usando poi la moneta di valore $v[i]$, un valore totale pari a $(j - v[i]) + v[i] = j$, usando un numero di monete pari a $1 + C(i, j - v[i]) < 1 + k = C(i,j)$, **contro** l'ipotesi che $C(i,j) = 1 + k$ è il **minimo** numero di monete per esprimere la somma $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$.

- Nella soluzione che ci dà il minimo numero di monete $C(i,j)$ per esprimere $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$, la moneta di valore $v[i]$ **appare**.
  $\Rightarrow C(i,j) = 1 + k$, dove l'1 conta la presenza della moneta di valore $v[i]$, e il k conta le monete restanti usate, siamo esse $i_1, \ldots, i_k$
  I valori $v[i_1], \ldots v[i_k]$ delle monete $i_1, \ldots, i_k$ devono **necessariamente** sommare a $j - v[i]$ (visto che poi sommando la moneta di valore $v[i]$, che sappiamo esserci, arriviamo al valore totale j).
  L'osservazione chiave è che $k = C(i, j - v[i])$.
  Se fosse $k > C(i, j - v[i])$, potremmo sostituire le monete $i_1, \ldots, i_k$ con le $C(i, j - v[i])$ monete che sappiamo, per definizione di $C(\cdot, \cdot)$, avere come somma di valori uguale a $j - v[i]$, ed ottenere, usando poi la moneta di valore $v[i]$, un valore totale pari a $(j - v[i]) + v[i] = j$, usando un numero di monete pari a $1 + C(i, j - v[i]) < 1 + k = C(i,j)$, **contro** l'ipotesi che $C(i,j) = 1 + k$ è il **minimo** numero di monete per esprimere la somma $j \leq V$, usando monete di valore $v[i] > v[i+1] > \ldots > v[n]$.

Se invece $k < C(i, j - v[i])$, l'assurdo sarebbe ancora più immediato.

Se invece $k < C(i, j - v[i])$, l'assurdo sarebbe ancora più immediato.

Infatti, avremmo che con le monete $i_1, \ldots, i_k$ (che sono in **numero** inferiore al minimo $C(i, j - v[i])$)

Se invece $k < C(i, j - v[i])$, l'assurdo sarebbe ancora più immediato.

Infatti, avremmo che con le monete $i_1, \ldots, i_k$ (che sono in **numero** inferiore al minimo $C(i, j - v[i])$) riusciremmo ad esprimere il valore $j - v[i]$,

Se invece $k < C(i, j - v[i])$, l'assurdo sarebbe ancora più immediato.

Infatti, avremmo che con le monete $i_1, \ldots, i_k$ (che sono in **numero** inferiore al minimo $C(i, j - v[i])$) riusciremmo ad esprimere il valore $j - v[i]$, **contro** l'ipotesi che $C(i, j - v[i])$ è il **minimo** numero di monete per esprimere la somma $j - v[i] \leq V$, usando monete di valore $v[i] > v[i + 1] > \ldots > v[n]$.

Se invece $k < C(i, j - v[i])$, l'assurdo sarebbe ancora più immediato.

Infatti, avremmo che con le monete $i_1, \ldots, i_k$ (che sono in **numero** inferiore al minimo $C(i, j - v[i])$) riusciremmo ad esprimere il valore $j - v[i]$, **contro** l'ipotesi che $C(i, j - v[i])$ è il **minimo** numero di monete per esprimere la somma $j - v[i] \leq V$, usando monete di valore $v[i] > v[i + 1] > \ldots > v[n]$.

Riassumendo, abbiamo dimostrato che **nel caso** in cui nella soluzione che ci dà il minimo numero di monete per esprimere la somma $j \leq V$, usando monete di valore $v[i] > v[i + 1] > \ldots > v[n]$, la moneta i-esima di valore $v[i]$ **appare**,

Se invece $k < C(i, j - v[i])$, l'assurdo sarebbe ancora più immediato.

Infatti, avremmo che con le monete $i_1, \ldots, i_k$ (che sono in **numero** inferiore al minimo $C(i, j - v[i])$) riusciremmo ad esprimere il valore $j - v[i]$, **contro** l'ipotesi che $C(i, j - v[i])$ è il **minimo** numero di monete per esprimere la somma $j - v[i] \leq V$, usando monete di valore $v[i] > v[i + 1] > \ldots > v[n]$.

Riassumendo, abbiamo dimostrato che **nel caso** in cui nella soluzione che ci dà il minimo numero di monete per esprimere la somma $j \leq V$, usando monete di valore $v[i] > v[i + 1] > \ldots > v[n]$, la moneta i-esima di valore $v[i]$ **appare**, allora vale che

$$C(i, j) = 1 + C(i, j - v[i])$$

Riassumendo ancora di più:

- **Se** la moneta di valore $v[i]$ **non appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i+1,j)$$

Riassumendo ancora di più:

- **Se** la moneta di valore $v[i]$ **non appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i+1,j)$$

- **Se** la moneta di valore $v[i]$ **appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i,j-v[i]) + 1$$

Riassumendo ancora di più:

- **Se** la moneta di valore $v[i]$ **non appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i+1,j)$$

- **Se** la moneta di valore $v[i]$ **appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i,j-v[i]) + 1$$

- Ma noi **non** sappiamo se essa appare o non appare, come procediamo allora?

Riassumendo ancora di più:

- **Se** la moneta di valore $v[i]$ **non appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i+1,j)$$

- **Se** la moneta di valore $v[i]$ **appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i,j-v[i]) + 1$$

- Ma noi **non** sappiamo se essa appare o non appare, come procediamo allora?
- Calcoliamo (ricorsivamente) **entrambe** le soluzioni $C(i-1,j)$ e $C(i,j-v[i])$

Riassumendo ancora di più:

- **Se** la moneta di valore $v[i]$ **non appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i+1,j)$$

- **Se** la moneta di valore $v[i]$ **appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i,j-v[i]) + 1$$

- Ma noi **non** sappiamo se essa appare o non appare, come procediamo allora?
- Calcoliamo (ricorsivamente) **entrambe** le soluzioni $C(i-1,j)$ e $C(i,j-v[i])$ e ci prendiamo la **migliore**,

Riassumendo ancora di più:

- **Se** la moneta di valore $v[i]$ **non appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i+1,j)$$

- **Se** la moneta di valore $v[i]$ **appare** nella soluzione che usa il minor numero di monete, allora

$$C(i,j) = C(i,j-v[i]) + 1$$

- Ma noi **non** sappiamo se essa appare o non appare, come procediamo allora?
- Calcoliamo (ricorsivamente) **entrambe** le soluzioni $C(i-1,j)$ e $C(i,j-v[i])$ e ci prendiamo la **migliore**, ovvero

$$C(i,j) = \min\{C(i+1,j), 1 + C(i,j-v[i])\}$$

Più precisamente...

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \end{cases}$$

Più precisamente...

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i, j - v[i])\} & \text{se } v[i] \leq j \end{cases}$$

Più precisamente...

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i,j-v[i])\} & \text{se } v[i] \leq j \end{cases}$$

con i casi base della ricorrenza pari a:

$$C(n,j) = j, \quad \forall j = 0, \ldots, V.$$

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i, j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i,j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

```
Rec_CambioMonete(v[i]...v[n],j) % fà uso di una tabella T(i,j)
1. IF (i==n) {
2.    RETURN j
```

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i,j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

```
Rec_CambioMonete(v[i]...v[n],j) % fà uso di una tabella T(i,j)
1. IF (i==n) {
2.    RETURN j
3.       } ELSE {
4.          IF (T(i,j) non è definito) {
```

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i,j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

```
Rec_CambioMonete(v[i]...v[n],j) % fà uso di una tabella T(i,j)
1. IF (i==n) {
2.    RETURN j
3.       } ELSE {
4.          IF (T(i,j) non è definito) {
5.             IF (v[i] ≤ j){
```

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i,j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

```
Rec_CambioMonete(v[i]...v[n],j) % fà uso di una tabella T(i,j)
1. IF (i==n) {
2.    RETURN j
3.       } ELSE {
4.          IF (T(i,j) non è definito) {
5.             IF (v[i] ≤ j){
6.             T(i,j) = min{Rec_CambioMonete(v[i+1]...v[n],j),
                          1 + Rec_CambioMonete(v[i]...v[n],j − v[i])}
```

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i,j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

```
Rec_CambioMonete(v[i]...v[n], j) % fà uso di una tabella T(i,j)
1. IF (i==n) {
2.    RETURN j
3.       } ELSE {
4.          IF (T(i,j) non è definito) {
5.             IF (v[i] ≤ j){
6.             T(i,j) = min{Rec_CambioMonete(v[i+1]...v[n], j),
                              1 + Rec_CambioMonete(v[i]...v[n], j − v[i])}
                } ELSE {
7.             T(i,j) = Rec_CambioMonete(v[i+1]...v[n], j)
            }
        }
```

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i,j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

```
Rec_CambioMonete(v[i]...v[n],j) % fà uso di una tabella T(i,j)
1. IF (i==n) {
2.    RETURN j
3.       } ELSE {
4.          IF (T(i,j) non è definito) {
5.             IF (v[i] ≤ j){
6.             T(i,j) = min{Rec_CambioMonete(v[i+1]...v[n],j),
                            1 + Rec_CambioMonete(v[i]...v[n],j − v[i])}
                } ELSE {
7.             T(i,j) = Rec_CambioMonete(v[i+1]...v[n],j)
            }
        }
8. RETURN(T(i,j))
```

Passo 2 di PD: Calcola la soluzione ai distinti sottoproblemi e memorizza ciascuna sottosoluzione nella entrata opportuna di una tabella $T[i,j]$, in modo tale che esse possano essere usate nel seguito, se occorre.

```
Rec_CambioMonete(v[i]...v[n],j) % fà uso di una tabella T(i,j)
1. IF (i==n) {
2.    RETURN j
3.       } ELSE {
4.          IF (T(i,j) non è definito) {
5.             IF (v[i] ≤ j){
6.             T(i,j) = min{Rec_CambioMonete(v[i+1]...v[n],j),
                            1 + Rec_CambioMonete(v[i]...v[n],j − v[i])}
                } ELSE {
7.             T(i,j) = Rec_CambioMonete(v[i+1]...v[n],j)
            }
        }
8. RETURN(T(i,j))
```

nplessità $= \Theta(nV)$

Ricordiamo:

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i, j - v[i])\} & \text{se } v[i] \leq j \end{cases}$$

$$C(n,j) = j, \quad \forall j = 0, \ldots, V.$$

Ricordiamo:

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i, j - v[i])\} & \text{se } v[i] \leq j \end{cases}$$

$$C(n,j) = j, \quad \forall j = 0, \ldots, V.$$

Sull'esempio con $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$ l'algoritmo costruirebbe la matrice seguente, e produrrebbe in output il valore $C(1,12) = 2$

Ricordiamo:

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i, j - v[i])\} & \text{se } v[i] \leq j \end{cases}$$

$$C(n,j) = j, \quad \forall j = 0, \ldots, V.$$

Sull'esempio con $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$ l'algoritmo costruirebbe la matrice seguente, e produrrebbe in output il valore $C(1,12) = 2$

		j												
		0	1	2	3	4	5	6	7	8	9	10	11	12
	1													
i	2													
	3													

Ricordiamo:

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i, j - v[i])\} & \text{se } v[i] \leq j \end{cases}$$

$$C(n,j) = j, \quad \forall j = 0, \ldots, V.$$

Sull'esempio con $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$ l'algoritmo costruirebbe la matrice seguente, e produrrebbe in output il valore $C(1,12) = 2$

		j												
		0	1	2	3	4	5	6	7	8	9	10	11	12
	1													
i	2													
	3	0	1	2	3	4	5	6	7	8	9	10	11	12

Ricordiamo:

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i, j - v[i])\} & \text{se } v[i] \leq j \end{cases}$$

$$C(n,j) = j, \quad \forall j = 0, \ldots, V.$$

Sull'esempio con $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$ l'algoritmo costruirebbe la matrice seguente, e produrrebbe in output il valore $C(1,12) = 2$

		j												
		0	1	2	3	4	5	6	7	8	9	10	11	12
	1													
i	2	0	1	2	3	4	5	1	2	3	4	5	6	2
	3	0	1	2	3	4	5	6	7	8	9	10	11	12

Ricordiamo:

$$C(i,j) = \begin{cases} C(i+1,j) & \text{se } v[i] > j, \\ \min\{C(i+1,j), 1 + C(i, j - v[i])\} & \text{se } v[i] \leq j \end{cases}$$

$$C(n,j) = j, \quad \forall j = 0, \ldots, V.$$

Sull'esempio con $V = 12$, e monete di valore $v[1] = 10$, $v[2] = 6$, $v[3] = 1$ l'algoritmo costruirebbe la matrice seguente, e produrrebbe in output il valore $C(1,12) = 2$

		j												
		0	1	2	3	4	5	6	7	8	9	10	11	12
	1	0	1	2	3	4	5	1	2	3	4	1	2	2
i	2	0	1	2	3	4	5	1	2	3	4	5	6	2
	3	0	1	2	3	4	5	6	7	8	9	10	11	12

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

A ciascuna possibile soluzione è associato un costo

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

A ciascuna possibile soluzione è associato un costo (ad es., il numero di monete per esprimere V).

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

A ciascuna possibile soluzione è associato un costo (ad es., il numero di monete per esprimere V).

Noi cerchiamo una soluzione di minimo costo

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

A ciascuna possibile soluzione è associato un costo (ad es., il numero di monete per esprimere V).

Noi cerchiamo una soluzione di minimo costo (o di massimo costo, se esso rappresenta un "guadagno" per noi).

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

A ciascuna possibile soluzione è associato un costo (ad es., il numero di monete per esprimere V).

Noi cerchiamo una soluzione di minimo costo (o di massimo costo, se esso rappresenta un "guadagno" per noi). I problemi di ottimizzazione sono quindi caratterizzati dal fatto che ogni istanza di input può avere diverse possibili soluzioni,

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

A ciascuna possibile soluzione è associato un costo (ad es., il numero di monete per esprimere V).

Noi cerchiamo una soluzione di minimo costo (o di massimo costo, se esso rappresenta un "guadagno" per noi). I problemi di ottimizzazione sono quindi caratterizzati dal fatto che ogni istanza di input può avere diverse possibili soluzioni, e noi cerchiamo quella che "ottimizza" il costo

Il problema del Cambio di Monete è il primo esempio di **Problema di Ottmizzazione** che abbiamo visto.

Informalmente, un Problema di Ottimizzazione è caratterizzato dal fatto che ad ogni possibile istanza di input (ad es., il valore V ed i valori $v[1], \ldots, v[n]$ delle monete nel problema precedente), è possibile associare più soluzioni (ad es., i diversi modi di esprimere il valore V con le monete di valore $v[1], \ldots, v[n]$).

A ciascuna possibile soluzione è associato un costo (ad es., il numero di monete per esprimere V).

Noi cerchiamo una soluzione di minimo costo (o di massimo costo, se esso rappresenta un "guadagno" per noi). I problemi di ottimizzazione sono quindi caratterizzati dal fatto che ogni istanza di input può avere diverse possibili soluzioni, e noi cerchiamo quella che "ottimizza" il costo (ovvero, lo minimizza o lo massimizza, a seconda dello specifico problema in questione).

Esercizio: massimizzare il punteggio all'esame

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti.

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$.

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$. Il tempo ha disposizione per lo svolgimento dell'esame è T minuti,

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$. Il tempo ha disposizione per lo svolgimento dell'esame è T minuti, dove T potrebbe essere inferiore alla somma dei tempi totali $t[1] + \ldots + t[n]$ necessari per rispondere a tutte le domande.

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$. Il tempo ha disposizione per lo svolgimento dell'esame è T minuti, dove T potrebbe essere inferiore alla somma dei tempi totali $t[1] + \ldots + t[n]$ necessari per rispondere a tutte le domande.
- Vogliamo progettare un algoritmo efficiente che, dati i vettori $p[1..n]$,

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$. Il tempo ha disposizione per lo svolgimento dell'esame è T minuti, dove T potrebbe essere inferiore alla somma dei tempi totali $t[1] + \ldots + t[n]$ necessari per rispondere a tutte le domande.
- Vogliamo progettare un algoritmo efficiente che, dati i vettori $p[1..n]$, $t[1..n]$

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$. Il tempo ha disposizione per lo svolgimento dell'esame è T minuti, dove T potrebbe essere inferiore alla somma dei tempi totali $t[1] + \ldots + t[n]$ necessari per rispondere a tutte le domande.
- Vogliamo progettare un algoritmo efficiente che, dati i vettori $p[1..n]$, $t[1..n]$ ed il valore T,

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$. Il tempo ha disposizione per lo svolgimento dell'esame è T minuti, dove T potrebbe essere inferiore alla somma dei tempi totali $t[1] + \ldots + t[n]$ necessari per rispondere a tutte le domande.
- Vogliamo progettare un algoritmo efficiente che, dati i vettori $p[1..n]$, $t[1..n]$ ed il valore T, restituisce il punteggio **massimo** che si può ottenere rispondendo correttamente ad un opportuno sottoinsieme delle n domande

Esercizio: massimizzare il punteggio all'esame

- L'input al problema consta di n domande $D_1, \ldots, D_n$ che valgono, rispettivamente, $p[1], \ldots, p[n]$ punti. Ogni domanda D_i richiede $t[i]$ minuti per il suo svolgimento, per ogni $i = 1, \ldots, n$. Il tempo ha disposizione per lo svolgimento dell'esame è T minuti, dove T potrebbe essere inferiore alla somma dei tempi totali $t[1] + \ldots + t[n]$ necessari per rispondere a tutte le domande.
- Vogliamo progettare un algoritmo efficiente che, dati i vettori $p[1..n]$, $t[1..n]$ ed il valore T, restituisce il punteggio **massimo** che si può ottenere rispondendo correttamente ad un opportuno sottoinsieme delle n domande **entro il tempo massimo** di T minuti.

Formulazione ricorsiva del problema

Formulazione ricorsiva del problema

Per formulare il problema in termini ricorsivi occorre individuare chi sono i sottoproblemi.

Formulazione ricorsiva del problema

Per formulare il problema in termini ricorsivi occorre individuare chi sono i sottoproblemi. Procederemo nel modo seguente: consideremo i sottoproblemi che corrispondono **solo** alle domande $\{D_1, \ldots, D_i\} \subseteq \{D_1, \ldots, D_i, \ldots, D_n\}$,

Formulazione ricorsiva del problema

Per formulare il problema in termini ricorsivi occorre individuare chi sono i sottoproblemi. Procederemo nel modo seguente: consideremo i sottoproblemi che corrispondono **solo** alle domande $\{D_1, \ldots, D_i\} \subseteq \{D_1, \ldots, D_i, \ldots, D_n\}$, e ad un tempo d'esame $j \leq T$, dove $i = 1, \ldots, n$ e $j = 0, \ldots, T$.

Formulazione ricorsiva del problema

Per formulare il problema in termini ricorsivi occorre individuare chi sono i sottoproblemi. Procederemo nel modo seguente: consideremo i sottoproblemi che corrispondono **solo** alle domande $\{D_1, \ldots, D_i\} \subseteq \{D_1, \ldots, D_i, \ldots, D_n\}$, e ad un tempo d'esame $j \leq T$, dove $i = 1, \ldots, n$ e $j = 0, \ldots, T$.

Sia $P[i, j]$ =punteggio massimo che è possibile ottenere rispondendo correttamente ad un opportuno sottoinsieme delle domande $\{D_1, \ldots, D_i\}$

Formulazione ricorsiva del problema

Per formulare il problema in termini ricorsivi occorre individuare chi sono i sottoproblemi. Procederemo nel modo seguente: consideremo i sottoproblemi che corrispondono **solo** alle domande $\{D_1, \ldots, D_i\} \subseteq \{D_1, \ldots, D_i, \ldots, D_n\}$, e ad un tempo d'esame $j \leq T$, dove $i = 1, \ldots, n$ e $j = 0, \ldots, T$.

Sia $P[i, j]$ =punteggio massimo che è possibile ottenere rispondendo correttamente ad un opportuno sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di j minuti, per $i = 1, \ldots, n$ e $j = 0, \ldots, T$.

Formulazione ricorsiva del problema

Per formulare il problema in termini ricorsivi occorre individuare chi sono i sottoproblemi. Procederemo nel modo seguente: consideremo i sottoproblemi che corrispondono **solo** alle domande $\{D_1, \ldots, D_i\} \subseteq \{D_1, \ldots, D_i, \ldots, D_n\}$, e ad un tempo d'esame $j \leq T$, dove $i = 1, \ldots, n$ e $j = 0, \ldots, T$.

Sia $P[i, j]$ =punteggio massimo che è possibile ottenere rispondendo correttamente ad un opportuno sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di j minuti, per $i = 1, \ldots, n$ e $j = 0, \ldots, T$.
Per ogni $j = 0, \ldots T$ abbiamo

$$P[1, j] = \begin{cases} p[1] & \text{se } j \geq t[1], \\ \end{cases}$$

Formulazione ricorsiva del problema

Per formulare il problema in termini ricorsivi occorre individuare chi sono i sottoproblemi. Procederemo nel modo seguente: consideremo i sottoproblemi che corrispondono **solo** alle domande $\{D_1, \ldots, D_i\} \subseteq \{D_1, \ldots, D_i, \ldots, D_n\}$, e ad un tempo d'esame $j \leq T$, dove $i = 1, \ldots, n$ e $j = 0, \ldots, T$.

Sia $P[i, j]$ =punteggio massimo che è possibile ottenere rispondendo correttamente ad un opportuno sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di j minuti, per $i = 1, \ldots, n$ e $j = 0, \ldots, T$.
Per ogni $j = 0, \ldots T$ abbiamo

$$P[1, j] = \begin{cases} p[1] & \text{se } j \geq t[1], \\ 0 & \text{altrimenti.} \end{cases}$$

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

- decidiamo di **rispondere** alla domanda D_i.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti,

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

• decidiamo di **non rispondere** alla domanda D_i.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

• decidiamo di **non rispondere** alla domanda D_i. In questo caso occorre scegliere la migliore soluzione in $\{D_1, \ldots, D_{i-1}\}$

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

• decidiamo di **non rispondere** alla domanda D_i. In questo caso occorre scegliere la migliore soluzione in $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a j.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

• decidiamo di **non rispondere** alla domanda D_i. In questo caso occorre scegliere la migliore soluzione in $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a j. In questo caso otterremo $P[i-1, j]$ punti.

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

• decidiamo di **non rispondere** alla domanda D_i. In questo caso occorre scegliere la migliore soluzione in $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a j. In questo caso otterremo $P[i-1, j]$ punti. Come facciamo a sapere cosa è meglio?

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

• decidiamo di **non rispondere** alla domanda D_i. In questo caso occorre scegliere la migliore soluzione in $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a j. In questo caso otterremo $P[i-1, j]$ punti. Come facciamo a sapere cosa è meglio?
In generale, per $i = 2, \ldots n$, $j = 0, \ldots T$

$$P[i,j] = \begin{cases} \max\{P[i-1,j], P[i-1,j-t[i]] + p[i]\} & \text{se } j \geq t[i], \\ \end{cases}$$

In generale, consideriamo il sottoinsieme delle domande $\{D_1, \ldots, D_i\}$ avendo a disposizione il tempo massimo di $j \leq T$ minuti. Abbiamo due possibilità :

• decidiamo di **rispondere** alla domanda D_i. Questo ci fa guadagnare $p[i]$ punti, e ci fa spendere $t[i]$ minuti.Per il resto dell'esame, occorre cercare la **migliore** soluzione nell'insieme di domande $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a $j - t[i]$.
Tale migliore soluzione ci farà guadagnare $P[i-1, j-t[i]]$ punti a cui vanno aggiunti i $p[i]$ punti ottenuti dall'aver risposto a D_i. In totale otterremo $P[i-1, j-t[i]] + p[i]$ punti.

• decidiamo di **non rispondere** alla domanda D_i. In questo caso occorre scegliere la migliore soluzione in $\{D_1, \ldots, D_{i-1}\}$ avendo un tempo totale a disposizione pari a j. In questo caso otterremo $P[i-1, j]$ punti. Come facciamo a sapere cosa è meglio?
In generale, per $i = 2, \ldots n$, $j = 0, \ldots T$

$$P[i,j] = \begin{cases} \max\{P[i-1,j], P[i-1,j-t[i]] + p[i]\} & \text{se } j \geq t[i], \\ P[i-1,j] & \text{altrimenti.} \end{cases}$$

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.    IF (j>=t[1]) {
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.   IF (j>=t[1]) {
3.     RETURN p[1]
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.   IF (j>=t[1]) {
3.     RETURN p[1]
4.      } ELSE {
5.       RETURN 0
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.   IF (j>=t[1]) {
3.    RETURN p[1]
4.     } ELSE {
5.      RETURN 0
6.       } ELSE {
7.       IF (P(i,j) non è definito) {
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.   IF (j>=t[1]) {
3.    RETURN p[1]
4.     } ELSE {
5.      RETURN 0
6.       } ELSE {
7.       IF (P(i,j) non è definito) {
8.         IF (t[i]<= j){
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.   IF (j>=t[1]) {
3.    RETURN p[1]
4.     } ELSE {
5.      RETURN 0
6.       } ELSE {
7.       IF (P(i,j) non è definito) {
8.         IF (t[i]<= j){
9.         P(i,j)= max{Esami(p[1]...p[i-1],j-t[i])+p[i],
                        Esami(p[1]...p[i-1],j)}
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.    IF (j>=t[1]) {
3.     RETURN p[1]
4.      } ELSE {
5.       RETURN 0
6.        } ELSE {
7.        IF (P(i,j) non è definito) {
8.          IF (t[i]<= j){
9.          P(i,j)= max{Esami(p[1]...p[i-1],j-t[i])+p[i],
                          Esami(p[1]...p[i-1],j)}
10.         } ELSE {
11.          P(i,j)=Esami(p[1]...p[i-1],j)
          }
      }
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.   IF (j>=t[1]) {
3.    RETURN p[1]
4.     } ELSE {
5.      RETURN 0
6.       } ELSE {
7.       IF (P(i,j) non è definito) {
8.         IF (t[i]<= j){
9.         P(i,j)= max{Esami(p[1]...p[i-1],j-t[i])+p[i],
                        Esami(p[1]...p[i-1],j)}
10.        } ELSE {
11.         P(i,j)=Esami(p[1]...p[i-1],j)
          }
      }
12. RETURN (P(i,j))
```

L'algoritmo

```
Esami(p[1]...p[i], j) % fà uso di una tabella P(i,j)
1. IF (i==1){
2.   IF (j>=t[1]) {
3.    RETURN p[1]
4.     } ELSE {
5.      RETURN 0
6.       } ELSE {
7.       IF (P(i,j) non è definito) {
8.         IF (t[i]<= j){
9.         P(i,j)= max{Esami(p[1]...p[i-1],j-t[i])+p[i],
                        Esami(p[1]...p[i-1],j)}
10.        } ELSE {
11.         P(i,j)=Esami(p[1]...p[i-1],j)
          }
      }
12. RETURN (P(i,j))
```

mplessità : $\Theta(nT)$