STEP 1: Identificazione del Problema Target

```
P = {⟨istanza⟩ | istanza soddisfa condizione C}
```

STEP 2: Selezione del Problema di Partenza (Definizioni pratiche)

3-SAT

```
3-SAT = {⟨ϕ⟩ | ϕ è una formula booleana in 3-CNF soddisfacibile}

Dove ϕ = C₁ ∧ C₂ ∧ ... ∧ Cₘ
Ogni clausola Cᵢ = (l₁ ∨ l₂ ∨ l₃) con esattamente 3 letterali
Letterale = variabile xⱼ o sua negazione ¬xⱼ

Esempio: (x₁ ∨ ¬x₂ ∨ x₃) ∧ (¬x₁ ∨ x₂ ∨ ¬x₃)
```

CLIQUE

```
CLIQUE = {⟨G,k⟩ | G è un grafo non orientato con una clique di dimensione ≥ k}

Clique = sottoinsieme di vertici completamente connessi
In una clique di dimensione k, ogni vertice è collegato a tutti gli altri k-1

Esempio: Grafo con triangolo (clique di dimensione 3)
```

VERTEX-COVER

```
VERTEX-COVER = {⟨G,k⟩ | G ha un vertex cover di dimensione ≤ k}

Vertex Cover = sottoinsieme S ⊆ V tale che ogni arco (u,v) ∈ E
ha almeno un estremo in S (u ∈ S oppure v ∈ S)

Esempio: Per coprire tutti gli archi servono almeno k vertici
```

INDEPENDENT-SET

```
INDEPENDENT-SET = {⟨G,k⟩ | G ha un independent set di dimensione ≥ k}

Independent Set = sottoinsieme S ⊆ V tale che nessuna coppia
di vertici in S è collegata da un arco

Dualità: VERTEX-COVER(k) ⟺ INDEPENDENT-SET(n-k)
```

3-COLOR

```
3-COLOR = {⟨G⟩ | G è un grafo 3-colorabile}

3-colorabile = assegnare colori {1,2,3} ai vertici tale che
vertici adiacenti abbiano colori diversi

Generalizzazione: k-COLOR usa k colori
Caso particolare: 2-COLOR è in P (test bipartizione)
```

HAMILTONIAN-PATH

```
HAMILTONIAN-PATH = {⟨G⟩ | G contiene un cammino hamiltoniano}

Cammino Hamiltoniano = cammino che visita ogni vertice esattamente una volta
(Non deve tornare al vertice di partenza, a differenza del ciclo
hamiltoniano)

Varianti: HAMILTONIAN-CYCLE (deve tornare al punto di partenza)
```

SUBSET-SUM

```
SUBSET-SUM = {⟨S,t⟩ | S = {a₁,...,aₙ} ⊆ ℕ, ∃ T ⊆ S : Σ(T) = t}

Trovare sottoinsieme T di S tale che la somma degli elementi di T
sia esattamente uguale al target t

Esempio: S = {1,2,3,4}, t = 6 → T = {2,4} oppure T = {1,2,3}
```

PARTITION

```
PARTITION = {⟨S⟩ | S = {a₁,...,aₙ} può essere partizionato in due
                   sottoinsiemi con somma uguale}
```

```
Caso speciale di SUBSET-SUM dove t = (Σ(S))/2
Richiede che Σ(S) sia pari e che esista T ⊆ S con Σ(T) = Σ(S)/2

Esempio: S = {1,2,3,4,6} → T₁ = {1,2,3}, T₂ = {4,6} (somma = 8)
```

Tabella decisionale:

```
Grafi + copertura/selezione vertici → VERTEX-COVER, INDEPENDENT-SET
Grafi + sottografi densi → CLIQUE
Grafi + colorazione → 3-COLOR
Grafi + percorsi → HAMILTONIAN-PATH
Logica/soddisfacimento → 3-SAT
Numeri/somme → SUBSET-SUM, PARTITION
```

STEP 4: Template Copy-Paste per Riduzioni

TEMPLATE A: Da 3-SAT

```
Input: Formula ϕ = C₁ ∧ C₂ ∧ ... ∧ Cₘ con variabili x₁,...,xₙ

Costruzione f(ϕ):
1. Gadget variabile per xᵢ: [Forza scelta true/false]
2. Gadget clausola per Cⱼ: [Verifica almeno un letterale vero]
3. Connessioni: [Propaga valori between gadget]
4. Parametro: k = [funzione di n,m]

Claim: ϕ soddisfacibile ⟺ f(ϕ) ha soluzione
```

TEMPLATE B: Da CLIQUE

```
Input: Grafo G = (V,E), intero k

Costruzione f(G,k):
1. Vertici V → [Elementi da selezionare nel nuovo problema]
2. Archi E → [Compatibilità/relazioni nel nuovo problema]
3. Parametro k → [Soglia target]

Claim: G ha clique ≥ k ⟺ f(G,k) ha soluzione
```

TEMPLATE C: Da VERTEX-COVER

```
Input: Grafo G = (V,E), intero k

Costruzione f(G,k):
1. Ogni arco (u,v) → [Vincolo che richiede almeno uno tra u,v]
2. Ogni vertice → [Elemento costoso da selezionare]
3. Budget k → [Limite risorse]

Dualità: VC(k) ⟺ INDEPENDENT-SET(n-k)
Claim: G ha vertex cover ≤ k ⟺ f(G,k) ha soluzione
```

TEMPLATE D: Da INDEPENDENT-SET

```
Input: Grafo G = (V,E), intero k

Costruzione f(G,k):
1. Vertici V → [Elementi che possono essere selezionati]
2. Archi E → [Conflitti: non entrambi i vertici]
3. Parametro k → [Numero minimo da selezionare]

Claim: G ha independent set ≥ k ⟺ f(G,k) ha soluzione
```

TEMPLATE E: Da 3-COLOR

```
Input: Grafo G = (V,E)

Costruzione f(G):
1. Vertici V → [Elementi da classificare in 3 categorie]
2. Archi E → [Vincoli: elementi adiacenti in categorie diverse]
3. 3 colori → [3 stati/tipi possibili]

Claim: G è 3-colorabile ⟺ f(G) ha soluzione
```

TEMPLATE F: Da SUBSET-SUM

```
Input: Numeri a_1,...,a_n, target T

Costruzione f(a_1,...,a_n,T):
1. Numeri a_i → [Elementi con "peso" a_i]
2. Target T → [Obiettivo da raggiungere esattamente]
```

```
3. Padding → [Cifre extra per evitare interferenze]

Claim: ∃ sottoinsieme con somma T ⟺ f(...) ha soluzione
```

STEP 5: Dimostrazione Standard (Copy-Paste)

Struttura Obbligatoria:

```
Lemma: f è riduzione polinomiale da [PROBLEMA-ORIGINE] a [PROBLEMA-TARGET]

(⟹) Supponiamo [INPUT] ∈ [PROBLEMA-ORIGINE]
    Sia S soluzione per [INPUT]
    Costruisco S' = [TRADUZIONE]
    Verifico che S' è soluzione per f([INPUT])

(⟸) Supponiamo f([INPUT]) ∈ [PROBLEMA-TARGET]
    Sia S' soluzione per f([INPUT])
    Costruisco S = [TRADUZIONE INVERSA]
    Verifico che S è soluzione per [INPUT]

Polinomialità:
- Dimensione f([INPUT]) = O([CALCOLA])
- Tempo costruzione = O([CALCOLA])
```

STEP 6: Relazioni tra Problemi del Corso

Dualità Fondamentali:

```
VERTEX-COVER(k) ⟺ INDEPENDENT-SET(n-k)
CLIQUE in G ⟺ INDEPENDENT-SET in Ḡ (grafo complemento)
```

Gerarchia di Difficoltà:

```
P: 2-SAT, 2-COLOR
NP-complete: 3-SAT, 3-COLOR, CLIQUE, VERTEX-COVER, INDEPENDENT-SET
```

STEP 7: Gadget Pre-Costruiti (Copy-Paste)

Da 3-SAT:

```
Gadget Variabile xᵢ:
- Elemento "xᵢ = true"
- Elemento "xᵢ = false"
- Vincolo: esattamente uno dei due

Gadget Clausola (l₁ ∨ l₂ ∨ l₃):
- Richiede almeno uno tra i letterali veri
```

Da 3-COLOR:

```
Gadget Base: Triangolo con 3 vertici
- Forza utilizzo di tutti e 3 i colori
- V₁ (colore 1), V₂ (colore 2), V₃ (colore 3)

Gadget OR: Per clausola (x ∨ y)
- Se x = false e y = false → impossibile colorare
```

STEP 8: Conclusione Standard

```
Poiché [PROBLEMA-ORIGINE] è NP-completo e
[PROBLEMA-ORIGINE] ≤ₚ [PROBLEMA-TARGET],
segue che [PROBLEMA-TARGET] è NP-hard.

Se [PROBLEMA-TARGET] ∈ NP, allora è NP-completo. □
```

Quick Reference: Quando Usare Quale Riduzione

Problemi di selezione ottima: CLIQUE, INDEPENDENT-SET
Problemi di copertura: VERTEX-COVER **Problemi di partizionamento:** 3-COLOR, PARTITION **Problemi logici:** 3-SAT **Problemi numerici:** SUBSET-SUM

Tempo stimato: 15 minuti per esercizio seguendo i template.