Immagine che contiene testo, Carattere, schermata, bianco e nero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, lettera

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, documento

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Immagine che contiene testo, carta, Carattere, bianco e nero

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco e nero, documento

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, algebra

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, bianco, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, bianco e nero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, documento

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, bianco, schermata

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, ricevuta, bianco

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, ricevuta

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, bianco e nero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, diagramma

Descrizione generata automaticamente

Un verificatore per PebbleDestruction può essere implementato come segue:

V = "Su input ⟨G,p,m⟩, dove G è un grafo, p è una funzione che assegna ciottoli ai vertici, e m è una sequenza di mosse:

1. Verifica che G sia un grafo non orientato valido.
2. Verifica che p(v) ≥ 0 per ogni vertice v.
3. Per ogni mossa in m: a. Verifica che il vertice di partenza abbia almeno 2 ciottoli. b. Verifica che il vertice di arrivo sia adiacente al vertice di partenza. c. Rimuovi 2 ciottoli dal vertice di partenza e aggiungi 1 ciottolo al vertice di arrivo.
4. Alla fine, verifica che tutti i vertici tranne uno abbiano 0 ciottoli.
5. Se tutte le verifiche sono passate, accetta. Altrimenti, rifiuta."

Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamenteQuesto verificatore opera in tempo polinomiale rispetto alla dimensione dell'input e verifica correttamente se una data sequenza di mosse risolve il problema PebbleDestruction.

1. Per Constrained-4-Color:

a) Dimostriamo che è in NP: Un verificatore V per Constrained-4-Color opera come segue: V = "Su input ⟨G, f1, ..., fn, c1, ..., cn⟩:

1. Verifica che ci ∈ {1, 2, 3, 4} e ci ≠ fi per ogni i.
2. Per ogni arco (u, v) in G, verifica che cu ≠ cv.
3. Se tutte le verifiche passano, accetta. Altrimenti, rifiuta."

Questo verificatore opera in tempo polinomiale, quindi Constrained-4-Color è in NP.

b) Dimostriamo che è NP-hard: Riduciamo 3-Color a Constrained-4-Color. Data un'istanza G di 3-Color, costruiamo un'istanza G' di Constrained-4-Color come segue:

* G' ha gli stessi vertici e archi di G
* Per ogni vertice v in G', fissiamo fv = 4

Questa riduzione è corretta perché:

* Se G è 3-colorabile, allora G' è colorabile con 3 colori, rispettando il vincolo fv = 4
* Se G' è colorabile rispettando i vincoli, allora G è 3-colorabile

La riduzione è chiaramente calcolabile in tempo polinomiale.

Quindi, Constrained-4-Color è NP-hard.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteCombinando (a) e (b), concludiamo che Constrained-4-Color è NP-completo.

Ridurremo il problema del Circuito Hamiltoniano a PebbleDestruction.

Data un'istanza G = (V, E) del Circuito Hamiltoniano, costruiamo un'istanza di PebbleDestruction come segue:

* Il grafo è G' = G
* Per ogni vertice v ∈ V, p(v) = 2

Questa riduzione è corretta perché:

* Se G ha un circuito Hamiltoniano, allora esiste una sequenza di mosse in G' che rimuove tutti i ciottoli tranne uno: Segui il circuito Hamiltoniano, rimuovendo 2 ciottoli da ogni vertice e aggiungendone 1 al successivo.
* Se esiste una soluzione per PebbleDestruction in G', allora G ha un circuito Hamiltoniano: La sequenza di mosse in G' corrisponde a un cammino che visita ogni vertice esattamente una volta, formando un circuito Hamiltoniano.

La riduzione è chiaramente calcolabile in tempo polinomiale. Quindi, PebbleDestruction è NP-hard.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, lettera

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

(a) Dimostrazione che List-Coloring è in NP:

Un problema è in NP se esiste un verificatore polinomiale.

Verificatore per List-Coloring: Input: ⟨G, L, c⟩, dove G è un grafo, L è la lista dei colori ammissibili per ogni vertice, e c è una colorazione proposta

1. Per ogni vertice v di G, verifica che c(v) ∈ L(v)
2. Per ogni arco (u, v) di G, verifica che c(u) ≠ c(v)
3. Se entrambe le condizioni sono soddisfatte, accetta. Altrimenti, rifiuta.

Questo verificatore opera in tempo polinomiale, quindi List-Coloring è in NP.

(b) Dimostrazione che List-Coloring è NP-hard:

Useremo una riduzione da 3-Color, che sappiamo essere NP-hard.

Sia G = (V, E) un'istanza di 3-Color. Costruiamo un'istanza (G', L) di List-Coloring:

* G' = G
* Per ogni vertice v ∈ V, L(v) = {1, 2, 3}

Chiaramente, G ha una 3-colorazione se e solo se (G', L) ha una colorazione valida per List-Coloring.

Questa riduzione è polinomiale e preserva le soluzioni, quindi List-Coloring è NP-hard.

Immagine che contiene testo, Carattere, bianco, ricevuta

Descrizione generata automaticamentePoiché List-Coloring è sia in NP che NP-hard, è NP-completo.

(a) Dimostrazione che HAM375 è un problema NP:

Un problema è in NP se esiste un verificatore polinomiale.

Verificatore per HAM375: Input: ⟨G, C⟩, dove G è un grafo e C è un ciclo proposto

1. Verifica che C contenga esattamente n-375 vertici di G
2. Verifica che ogni vertice in C appaia esattamente una volta
3. Verifica che per ogni coppia di vertici consecutivi in C, esista un arco in G che li collega
4. Se tutte le condizioni sono soddisfatte, accetta. Altrimenti, rifiuta.

Questo verificatore opera in tempo polinomiale, quindi HAM375 è in NP.

(b) Dimostrazione che HAM375 è NP-hard:

Useremo una riduzione dal problema del circuito Hamiltoniano (HAM), che sappiamo essere NP-hard.

Sia G = (V, E) un'istanza di HAM con |V| = n. Costruiamo un'istanza G' di HAM375:

1. Inizia con G' = G
2. Aggiungi 375 nuovi vertici v\_1, ..., v\_375 a G'
3. Per ogni v\_i, aggiungi archi da v\_i a tutti i vertici di G

Dimostrazione della correttezza:

* Se G ha un circuito Hamiltoniano, allora G' ha un ciclo che attraversa tutti i vertici tranne i 375 nuovi (cioè il circuito Hamiltoniano originale).
* Se G' ha un ciclo che attraversa n vertici (tutti tranne 375), questo ciclo deve includere tutti i vertici di G, formando un circuito Hamiltoniano in G.

Questa riduzione è polinomiale e preserva le soluzioni, quindi HAM375 è NP-hard.

Poiché HAM375 è sia in NP che NP-hard, è NP-completo.Immagine che contiene testo, schermata, diagramma, linea

Descrizione generata automaticamente

(a) Dimostrazione che EQUITABLE-3-COLOR è in NP:

Un problema è in NP se esiste un verificatore polinomiale.

Verificatore per EQUITABLE-3-COLOR: Input: ⟨G, c⟩, dove G è un grafo e c è una colorazione proposta

1. Verifica che c usi solo 3 colori
2. Per ogni arco (u, v) di G, verifica che c(u) ≠ c(v)
3. Conta il numero di vertici ni di ogni colore i
4. Verifica che |ni - nj| ≤ 1 per ogni coppia di colori i, j
5. Se tutte le condizioni sono soddisfatte, accetta. Altrimenti, rifiuta.

Questo verificatore opera in tempo polinomiale, quindi EQUITABLE-3-COLOR è in NP.

(b) Dimostrazione che EQUITABLE-3-COLOR è NP-hard:

Useremo una riduzione da 3-COLOR, che sappiamo essere NP-hard.

Sia G = (V, E) un'istanza di 3-COLOR. Costruiamo un'istanza G' di EQUITABLE-3-COLOR:

1. Inizia con G' = G
2. Aggiungi 3k nuovi vertici isolati a G', dove k = |V|
3. Dividi questi 3k vertici in tre gruppi di k vertici ciascuno

Dimostrazione della correttezza:

* Se G ha una 3-colorazione, possiamo estenderla a G' colorando ciascun gruppo di k vertici aggiuntivi con uno dei tre colori. Questo produce una 3-colorazione equa di G'.
* Se G' ha una 3-colorazione equa, la restrizione di questa colorazione a G è una 3-colorazione valida di G.

Questa riduzione è polinomiale e preserva le soluzioni, quindi EQUITABLE-3-COLOR è NP-hard.

Poiché EQUITABLE-3-COLOR è sia in NP che NP-hard, è NP-completo.

Immagine che contiene testo, cartone animato

Descrizione generata automaticamente

(a) Dimostrazione che COMMITTEE è un problema NP:

1. Verificatore: data un'istanza ⟨D1,...,Dn,I⟩ e un certificato C (sottoinsieme di docenti):
   * Verifica che |C| ≤ 4
   * Per ogni dipartimento Di, verifica che esiste almeno un docente in C
   * Per ogni coppia di docenti in C, verifica che non si detestano Questo verificatore opera in tempo polinomiale.
2. Il certificato ha dimensione polinomiale rispetto all'input. Quindi, COMMITTEE è in NP.

(b) Dimostrazione che COMMITTEE è NP-hard: Riduzione dal problema 3SAT (noto essere NP-completo).

Data una formula 3SAT φ con variabili x1,...,xn e clausole C1,...,Cm:

1. Crea un dipartimento Di per ogni variabile xi
2. Crea un dipartimento Ej per ogni clausola Cj
3. Per ogni xi, crea due docenti: xi e ¬xi
4. Per ogni Cj = (l1 ∨ l2 ∨ l3), crea tre docenti: l1j, l2j, l3j
5. I docenti si detestano se:
   * Rappresentano una variabile e la sua negazione
   * Rappresentano letterali diversi della stessa clausola
   * Rappresentano lo stesso letterale in clausole diverse

Questa costruzione garantisce che:

* φ è soddisfacibile se e solo se esiste una commissione valida
* La riduzione è calcolabile in tempo polinomiale

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamentePoiché COMMITTEE è in NP e è NP-hard, è NP-completo.

(a) Un certificato per PIVOT è una coalizione C ⊆ {1, ..., n-1} tale che:

- Σ\_{j∈C} W[j] < (1/2) Σ\_{j=1}^n W[j]

- Σ\_{j∈C} W[j] + W[n] > (1/2) Σ\_{j=1}^n W[j]

Dato ⟨n,W⟩ e un certificato C, possiamo verificare in tempo polinomiale che:

1. C ⊆ {1, ..., n-1} (scorrendo C)

2. Le due disuguaglianze sono soddisfatte (calcolando le somme)

Quindi PIVOT ∈ NP.

(b) Riduciamo SET-PARTITION ≤\_p PIVOT.

Data un'istanza ⟨S⟩ di SET-PARTITION, con S = {x\_1, ..., x\_m}, costruiamo:

- n = m+1

- W[i] = x\_i per i = 1, ..., m

- W[n] = (1/2) Σ\_{x∈S} x

Allora:

- Se ⟨S⟩ ∈ SET-PARTITION, esiste S\_1 ⊆ S tale che

Σ\_{x∈S\_1} x = (1/2) Σ\_{x∈S} x = W[n].

Quindi C = {i | x\_i ∈ S\_1} dimostra che m+1 è un pivot.

- Se ⟨n,W⟩ ∈ PIVOT, esiste C ⊆ {1, ..., m} che soddisfa le disuguaglianze.

Ponendo S\_1 = {x\_i | i ∈ C}, le disuguaglianze implicano che

Σ\_{x∈S\_1} x = (1/2) Σ\_{x∈S} x, quindi ⟨S⟩ ∈ SET-PARTITION.

La riduzione è in tempo polinomiale, quindi PIVOT è NP-hard.

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

(a) Per dimostrare che ALIBABA è un problema NP, forniamo un verificatore polinomiale V:

V = "Su input 〈〈N, P, W, M, L〉, B〉:

1. Controlla se B è un sottoinsieme di {1, ..., N}. Se no, rifiuta.
2. Calcola ∑\_{j∈B} W[j] e verifica se è ≤ M. Se no, rifiuta.
3. Calcola ∑\_{j∈B} P[j] e verifica se è > L. Se no, rifiuta.
4. Accetta."

(b) Dimostrazione che ALIBABA è NP-hard:

Riduzione da SUBSET-SUM a ALIBABA: Data un'istanza ⟨S,t⟩ di SUBSET-SUM, costruiamo un'istanza di ALIBABA:

* N = |S|
* W[i] = S[i] per ogni i
* P[i] = S[i] per ogni i
* M = t
* L = t

Questa riduzione è corretta perché:

* Esiste S' ⊆ S tale che Σ[x∈S'] x = t se e solo se esiste B ⊆ {1, ..., N} tale che Σ[i∈B] W[i] ≤ M e Σ[i∈B] P[i] ≥ L

La riduzione è calcolabile in tempo polinomiale.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamentePoiché ALIBABA è in NP e è NP-hard, è NP-completo.

1. a) DIET è in NP. Ecco un verificatore polinomiale:

V = "Su input 〈〈S₁,...,S\_n,k〉,T〉:

1. Verifica che T sia un sottoinsieme di {1,...,n} con |T|=k. Se no, rifiuta.
2. Verifica che ogni elemento di {1,...,n} compaia in qualche S\_i con i∈T. Se no, rifiuta.
3. Accetta."

Chiaramente V impiega tempo polinomiale.

(b) DIET è NP-hard per riduzione da VERTEX-COVER. f(〈G,k〉) = 〈S₁,...,S\_n,k〉 dove n è il numero di vertici di G e S\_i = {j | (i,j) è un arco di G}.

Se 〈G,k〉∈VERTEX-COVER allora esiste un vertex cover C di dimensione k, quindi prendendo gli insiemi S\_i con i∈C si ottiene una dieta valida, perché ogni elemento (vertice) compare in qualche S\_i (è coperto da C). Viceversa, se f(〈G,k〉)∈DIET allora esiste una dieta valida di k insiemi S\_i, e i vertici corrispondenti formano un vertex cover per G. Quindi f è una riduzione polinomiale corretta e DIET è NP-hard.

Essendo DIET sia in NP che NP-hard, è NP-completo.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

(a) SUPPLY è in NP. Un verificatore polinomiale è: V = "Su input 〈〈S₁,...,S\_n,k〉,T〉:

1. Verifica che T sia una fornitura valida di dimensione k. Se no, rifiuta.
2. Per ogni i = 1,...,n, verifica che ∑\_{j∈T} S\_i[j] ≤ {1,...,n}. Se no, rifiuta.
3. Accetta." Chiaramente V impiega tempo polinomiale.

(b) SUPPLY è NP-hard per riduzione da VERTEX-COVER. Sia f(〈G,k〉) = 〈S₁,...,S\_n,k〉 dove:

* n è il numero di vertici di G
* S\_i[j] = 1 se (i,j) è un arco di G, S\_i[j] = 0 altrimenti Allora: Se 〈G,k〉 ∈ VERTEX-COVER, esiste un vertex cover C di dimensione k, e prendendo la fornitura T = C abbiamo che ∑\_{j∈T} S\_i[j] = 1 per ogni vertice i ∈ {1,...,n}, perciò f(〈G,k〉) ∈ SUPPLY. Viceversa, se f(〈G,k〉) ∈ SUPPLY, esiste una fornitura valida T di dimensione k tale che ogni vertice i ha una adiacenza con un vertice in T. Quindi T è un vertex cover per G e 〈G,k〉 ∈ VERTEX-COVER.

Quindi f riduce VERTEX-COVER a SUPPLY in tempo polinomiale, e SUPPLY è NP-hard. Essendo SUPPLY sia NP che NP-hard, è NP-completo.