Lezione16 – problemi e codifiche

Lezione del 07/05/2024

Dai Linguaggi ai Problemi

- Le teorie della calcolabiltà e della complessità sono fondate sul concetto di appartenenza di una parola ad un insieme di parole: un concetto
  - semplice
  - elegante
  - formale
  - rigoroso
- Tuttavia, nella vita reale, non ti capita spesso di domandarti "ma questa parola apparterrà forse a questo insieme?"
- Nella vita reale, piuttosto, ti capita di dover trovare le soluzioni ad istanze di problemi
- E, allora, queste teorie sarebbe bello trasferirle nel mondo dei problemi
- Ma il concetto "trovare la soluzione ad una istanza di un problema" è, senza dubbio, più arbitrario
  - se vogliamo, più evanescente
- meno rigoroso di quello di appartenenza di una parola ad un insieme di parole

Dai Linguaggi ai Problemi

- E, allora, questo concetto di "trovare la soluzione ad una istanza di un problema" dobbiamo renderlo meno arbitrario
- ossia, più rigoroso!
- Dobbiamo formalizzarlo
  - e questo comporterà la gestione di numerose… questioncine
- Allora, cominciamo: come possiamo schematizzare un "problema"?
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - cercare gli oggetti che soddisfino certi vincoli
  - e, sulla base degli oggetti trovati, fornire un qualche tipo di risposta
- E nella dispensa 7, al paragrafo 7.1, trovate un po' di esempi

Formalizzare Problemi

- **ESEMPIO: dato un numero intero… [segue richesta relativa ai divisori del numero]**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - **dati un insieme di oggetti conosciuti** – l'insieme dei dati che costituisce una ***istanza*** del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - cercare gli oggetti che soddisfino certi vincoli
  - e, sulla base degli oggetti trovati, fornire un qualche tipo di risposta
- **Dati un insieme di oggetti conosciuti:** dobbiamo descrivere le istanze del problema, ossia in cosa consiste ciascuna istanza del problema
- descriviamo le istanze del problema definendo un insieme $\mathfrak{I}$ - l'**insieme delle istanze**
  - un elemento di $\mathfrak{I}$ corrisponde ad una istanza del problema
  - nell'**ESEMPIO**: $\mathfrak{I} = \mathbb{N}$

Formalizzare Problemi

- **ESEMPIO: dato un numero intero… [segue richesta relativa ai divisori del numero]**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - **all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili**
  - cercare gli oggetti che soddisfino certi vincoli
  - e, sulla base degli oggetti trovati, fornire un qualche tipo di risposta
- **all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili:** dobbiamo descrivere cosa ci viene richiesto di cercare – contenitori? Pere? Rettangoli? Numeri? Cosa?!
- descriviamo le soluzioni possibili per una istanza x del problema definendo un insieme **S**(x)
  - **S**(x) descrive tutti gli oggetti che dobbiamo testare per verificare se soddisfano i vincoli del problema
  - nell'**ESEMPIO**: **S**(x) = { $y \in \mathbb{N} : y \leq x$ }

Formalizzare Problemi

- **ESEMPIO: dato un numero intero… [segue richiesta relativa ai divisori del numero]**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - **cercare gli oggetti che soddisfino certi "vincoli"**
  - e, sulla base degli oggetti trovati, fornire un qualche tipo di risposta
- **cercare gli oggetti che soddisfino certi "vincoli":** dobbiamo descrivere quali oggetti, all'interno delle soluzioni possibili, soddisfano la richiesta del problema
- descriviamo le soluzioni possibili associate ad una istanza x che soddisfano i vincoli del problema definendo un insieme $\eta(S(x))$ di **soluzioni effettive** per l'istanza x
  - $\eta(S(x))$ è l'insieme che contiene tutti gli oggetti che sono soluzioni possibili per x e che soddisfano i vincoli del problema
  - nell'**ESEMPIO**: poiché il problema si pone qualche domanda circa i divisori di un dato numero x, $\eta(S(x)) = \{ y \in S(x) : y \text{ è un divisore di } x \}$

Formalizzare Problemi

- **ESEMPIO: dato un numero intero… [segue richiesta relativa ai divisori del numero]**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - cercare gli oggetti che soddisfino certi "vincoli"
  - e, **sulla base degli oggetti trovati, fornire un qualche tipo di risposta**
- **sulla base degli oggetti trovati, fornire un qualche tipo di risposta :** dipendentemente dalla domanda posta dal problema, dobbiamo rispondere fornendo quanto ci viene richiesto
- descriviamo la risposta al problema definendo una funzione ρ che associa all'insieme delle soluzioni effettive per l'istanza x una risposta scelta nell'insieme **R** delle risposte
- E, per chiarire questa questione, dobbiamo entrare nel dettaglio di **[segue richiesta relativa ai divisori del numero]**

Formalizzare Problemi

- **ESEMPIO 1: dato un numero intero n, elencare tutti i divisori di n**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - cercare gli oggetti che soddisfino certi "vincoli"
  - e, **sulla base degli oggetti trovati, fornire un qualche tipo di risposta**
- **sulla base degli oggetti trovati, fornire un qualche tipo di risposta :** dipendentemente dalla domanda posta dal problema, dobbiamo rispondere fornendo quanto ci viene richiesto
- In questo caso, **R** = $2^{\mathbb{N}}$
  - ossia, la risposta ad una istanza del problema è un sottoinsieme di $\mathbb{N}$
- e, per ogni istanza **n** del problema, $\rho(\eta(\mathbf{S}(\mathbf{n}))) = \eta(\mathbf{S}(\mathbf{n}))$

Formalizzare Problemi

- **ESEMPIO 2: dato un numero intero n, verificare se n è primo**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - cercare gli oggetti che soddisfino certi "vincoli"
  - e, **sulla base degli oggetti trovati, fornire un qualche tipo di risposta**
- **sulla base degli oggetti trovati, fornire un qualche tipo di risposta :** dipendentemente dalla domanda posta dal problema, dobbiamo rispondere fornendo quanto ci viene richiesto
- In questo caso, **R** = { vero, falso}
- e, per ogni istanza **n** del problema, $\rho(\eta(S(n))) = [\ \eta(S(n)) = \{1,n\}\ ]$

Formalizzare Problemi

- **ESEMPIO 3: dato un numero intero n, calcolare un divisore d non banale di n (ossia, d > 1 e d < n)**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - cercare gli oggetti che soddisfino certi "vincoli"
  - e, **sulla base degli oggetti trovati, fornire un qualche tipo di risposta**
- **sulla base degli oggetti trovati, fornire un qualche tipo di risposta :** dipendentemente dalla domanda posta dal problema, dobbiamo rispondere fornendo quanto ci viene richiesto
- In questo caso, **R** = $\mathbb{N}$
- e, per ogni istanza n del problema, $\rho(\eta(\mathbf{S}(\mathbf{n})))$ è un qualunque elemento di $\eta(\mathbf{S}(\mathbf{n}))$ che sia diverso da 1 e da **n**
  - **ATTENZIONE:** d potrebbe non esistere! In questo caso, **… secondo voi, che si fa?**

Formalizzare Problemi

- **ESEMPIO 4: dato un numero intero n, calcolare il più grande divisore non banale d di n (ossia, d > 1 e d < n)**
- Di qualunque problema stiamo parlando, la struttura di un problema è sostanzialmente la seguente
  - dati un insieme di oggetti conosciuti – l'insieme dei dati che costituisce un'istanza del problema
  - all'interno di un secondo insieme di oggetti – l'insieme delle soluzioni possibili
  - cercare gli oggetti che soddisfino certi "vincoli"
  - e, **sulla base degli oggetti trovati, fornire un qualche tipo di risposta**
- **sulla base degli oggetti trovati, fornire un qualche tipo di risposta :** dipendentemente dalla domanda posta dal problema, dobbiamo rispondere fornendo quanto ci viene richiesto
- In questo caso, **R** = $\mathbb{N}$
- e, per ogni istanza n del problema, $\rho(\eta(S(n)))$ è il più grande elemento di $\eta(S(n))$ che sia diverso da 1 e da **n**
  - **ATTENZIONE:** d potrebbe non esistere! In questo caso, **… secondo voi, che si fa?**

Tipi di problemi

- ESEMPIO 4: dato un numero intero **n**, calcolare il più grande divisore non banale d di **n** (ossia, d > 1 e d < **n**)
  - è un *problema di ottimizzazione*, in quanto alle soluzioni effettive è associata una misura e viene richiesto di trovare una soluzione effettiva di misura massima (come in questo caso), oppure minima
- ESEMPIO 3: dato un numero intero **n**, calcolare un divisore non banale d di **n** (ossia, d > 1 e d < **n**)
  - è un *problema di ricerca*, in quanto viene richiesto di trovare (e mostrare) una qualunque soluzione effettiva
  - sono i problemi con i quali abbiamo maggiore confidenza
- ESEMPIO 1: dato un numero intero **n**, elencare tutti i divisori di **n**
  - è un *problema di enumerazione*, in quanto ci viene richiesto di elencare tutte le soluzioni effettive
- ESEMPIO 2: dato un numero intero **n**, verificare se **n** è primo
  - è un *problema di decisione* (o *decisionale*), in quanto ci viene richiesto di decidere se l'istanza possiede una certa proprietà

Problemi e macchine

- Naturalmente, i due diversi tipi di macchine di Turing risolvono diversi tipi di problemi
  - Trasduttori per i problemi di ricerca, di enumerazione, e di ottimizzazione
  - Riconoscitori per i problemi di decisione
- La Teoria della Complessità si occupa, per lo più, di decidere dell'appartenenza di parole ad insiemi di parole
  - come abbiamo studiato sino ad ora
- utilizzando riconoscitori,

- Sembra naturale estendere quanto studiato nella dispensa 6 ai problemi decisionali:
- per questo ci occuperemo, d'ora in avanti di soli problemi decisionali

Problemi decisionali

- Abbiamo visto che un problema, in generale, può essere descritto da una quintupla $\langle \mathfrak{I}, S, \eta, \rho, R \rangle$, dove
  - η è il sottoinsieme di S che specifica quali, fra le soluzioni possibili, sono le soluzioni effettive per una data istanza $x \in \mathfrak{I}$
  - ρ è la funzione che associa all'insieme delle soluzioni effettive $\eta(S(x))$ una risposta (elemento di R) all'istanza x del problema
- Nel caso di problemi decisionali, sappiamo che R = { vero, falso}
- questo significa che, in effetti, **ρ è un predicato**
  - ossia, una funzione booleana
  - o, per dirla semplice, una proposizione logica il cui valore di verità dipende da qualche incognita
- Allora, possiamo riassumere η, ρ, R in un unico predicato π: **π(x,S(x))=vero se e soltanto se l'insieme delle soluzioni possibili per x soddisfa i vincoli del problema**
- E, quindi, un problema decisionale è descritto da una tripla $\langle \mathfrak{I}, S, \pi \rangle$

Problemi decisionali: esempi

- Un problema decisionale è descritto da una tripla $\langle \mathfrak{I}, \mathbf{S}, \pi \rangle$
- **Esempio 1:** dati un grafo non orientato G, una coppia di nodi s e t, e un intero k, decidere se esiste in G un percorso da s a t di lunghezza = k
  - $\mathfrak{I} = \{ \langle G, s, t, k \rangle : G$ è un grafo non orientato $\wedge$ s,t sono due nodi di G $\wedge$ k $\in \mathbb{N} \}$
  - $\mathbf{S}(G, s, t, k) = \{ \langle u_0, u_1, \ldots, u_k \rangle :$ per i=0, … ,k, u_i è un nodo del grafo $\}$
  - $\pi(G, s, t, k, \mathbf{S}(G,, s, t, k)) = \exists \langle u_0, u_1, \ldots, u_k \rangle \in \mathbf{S}(G, s, t, k) : s=u_0 \wedge t=u_k \wedge$ $\forall$ i=0, … ,k-1, [(u_i, u_{i+1}) è un arco del grafo]
- **Esempio 2:** dato un insieme X di variabili booleane ed un predicato f, definito sulle variabili in X e contenente i soli operatori $\wedge$, $\vee$ e $\neg$, decidere se esiste una assegnazione a di valori in {vero, falso} alle variabili in X tali che f(a(X))=vero
  - $\mathfrak{I} = \{ \langle X,f \rangle : X$ è un insieme di variabili booleane $\wedge$ f e un predicato su X$\}$
  - $\mathbf{S}(X,f) = \{ a: X \rightarrow \{$vero, falso$\} \}$ (S è l'insieme delle assegnazioni di verità alle variabili in X)
  - $\pi(X,f, \mathbf{S}(X,f)) = \exists a \in \mathbf{S}(X,f) : f(a(X)) =$ vero
- **Nota bene:** ciascun problema decisionale può essere descritto da diverse triple $\langle \mathfrak{I}, \mathbf{S}, \pi \rangle$!

Da Problema a Linguaggio

- A questo punto, formalizzato il concetto di problema decisionale,
- siamo *quasi* pronti ad estendere quanto abbiamo studiato sulla complessità dei linguaggi alla complessità dei problemi decisionali
- E, visto che la complessità dei linguaggi è studiata utilizzando la Macchina di Turing
- Utilizzeremo la Macchina di Turing anche per studiare la complessità dei problemi decisionali
- Ma per utilizzare una macchina di Turing per risolvere un problema decisionale
  - anzi, per *deciderlo*
- abbiamo bisogno di trasformare le *istanze* di quel problema in *parole*
  - sennò, cosa scriviamo sul nastro?
- Ossia, occorre **codificare** opportunamente le istanze di un problema decisionale
  - e questa è una questione **parecchio** delicata

Codifica

- Nel paragrafo 7.4 viene introdotta la questione delle codifiche attraverso un esempio: l' **Esempio 2** che abbiamo visto poc'anzi
- **Esempio 2:** dato un insieme X di variabili booleane ed un predicato f, definito sulle variabili in X e contenente i soli operatori $\wedge$, $\vee$ e $\neg$, decidere se esiste una assegnazione a di valori in {vero, falso} alle variabili in X tali che f(a(X))=vero
  - $\mathfrak{I}$ = { ⟨ X,f ⟩ : X è un insieme di variabili booleane ∧ f e un predicato su X}
- Di questo problema viene considerato un caso particolare: 3SAT
  - la funzione f è in una forma particolare: $f = c_1 \wedge c_2 \ldots \wedge c_m$
  - e ciascuna c_j prende il nome di clausola ed è l'or (∨) di tre letterali
  - dove un letterale è una variabile o una variabile negata – tipo $x_1 \vee \neg x_2 \vee x_3$
- Come codificare gli elementi di $\mathfrak{I}$?
- Abbiamo due possibilità:
  - 1) codifichiamo la struttura di f
  - 2) codifichiamo "il significato" di f

Codifica

- **Esempio 2:** dato un insieme X di variabili booleane ed un predicato f, definito sulle variabili in X e contenente i soli operatori $\wedge$, $\vee$ e $\neg$, decidere se esiste una assegnazione a di valori in {vero, falso} alle variabili in X tali che f(a(X))=vero
  - $\mathfrak{I}$ = { ⟨ X,f ⟩ : X è un insieme di variabili booleane ∧ f e un predicato su X }
  - dove $f = c_1 \wedge c_2 \ldots \wedge c_m$ e ciascuna c_j è del tipo $x_1 \vee \neg x_2 \vee x_3$
- CODIFICA χ_1 : codifichiamo la struttura di f
  - rappresentiamo ciascun elemento di $X = \{x_1, x_2, , x_n\}$ con n=|X| bit:

    x_i ha l'i-esimo bit 1 e tutti gli altri bit 0
  - rappresentiamo un letterale in una clausola mediante la rappresentazione della variabile corrispondente al letterale preceduta da 0 se il letterale è la variabile non negata, preceduta da 1 se se il letterale è la variabile negata
  - gli v in una clausola sono rappresentati da '2'
  - gli ∧ fra due clausole sono rappresentati da '3'
  - premettiamo alla codifica di f tanti '4' quanti gli elementi di X – ossia, |X| '4'
- Ad esempio, se $X = \{x_1, x_2, x_3\}$ e $f = c_1 \wedge c_2$ con $c_1 = \mathbf{x_1} \vee x_2 \vee x_3$ e $c_2 = x_1 \vee \mathbf{\neg x_2} \vee \neg x_3$ rappresentiamo ⟨ X,f ⟩ come

  444 **0 100** 2 0 010 2 0 001 3 0 100 2 **1 010** 2 1 001

Codifica

- **Esempio 2:** dato un insieme X di variabili booleane ed un predicato f, definito sulle variabili in X e contenente i soli operatori $\wedge$, $\vee$ e $\neg$, decidere se esiste una assegnazione a di valori in {vero, falso} alle variabili in X tali che f(a(X))=vero
  - $\mathfrak{I}$ = { ⟨ X,f ⟩ : X è un insieme di variabili booleane ∧ f e un predicato su X }
  - la funzione f è in una forma particolare: $f = c_1 \wedge c_2 \ldots \wedge c_m$ e ciascuna c_j è l'or ($\vee$) di tre letterali dove un letterale è una variabile o una variabile negata – tipo $x_1 \vee \neg x_2 \vee x_3$
- CODIFICA χ_2 : codifichiamo "il significato" di f – codifichiamo f in ***forma esplicita***
  - qualunque funzione è completamente descritta descrivendo i valori che essa assume in **tutti** i punti del suo insieme di esistenza
  - naturalmente, se una funzione è definita su $\mathbb{N}$ non possiamo descrivere il valore che essa assume per ogni $n \in \mathbb{N}$: in numeri naturali sono infiniti!
  - invece, la f della nostra istanza ⟨ X,f ⟩ di 3SAT è definita su $\{vero, falso\}^{|X|}$
  - quindi, poiché X è un insieme finito, l'insieme di esistenza di f è finito
  - allora, possiamo codificare f in forma esplicita
  - mediante la sua **tavola di verità**

Codifica

- CODIFICA χ_2 : codifichiamo f in ***forma esplicita*** mediante la sua tavola di verità
  - esempio: se X = {x_1, x_2, x_3} e f = $c_1 \wedge c_2$ con $c_1 = x_1 \vee x_2 \vee x_3$ e $c_2 = x_1 \vee \neg x_2 \vee \neg x_3$

x_1	x_2	x_3	f
vero	vero	vero	vero
vero	vero	falso	vero
vero	falso	vero	vero
vero	falso	falso	vero
falso	vero	vero	falso
falso	vero	falso	vero
falso	falso	vero	vero
falso	falso	falso	falso

- codificando vero con '1' e falso con '0', e scrivendo le righe della tavola una di seguito all'altra, separate da '2'
  - esempio: 1111 2 1101 2 1011 2 1001 2 0110 2 0101 2 0011 2 0000 2

Codifica e soluzione

- **Esempio 2:** dato un insieme X di variabili booleane ed un predicato f, definito sulle variabili in X e contenente i soli operatori $\wedge$, $\vee$ e $\neg$, decidere se esiste una assegnazione a di valori in {vero, falso} alle variabili in X tali che f(a(X))=vero
  - $\mathfrak{I}$ = { ⟨ X,f ⟩ : X è un insieme di variabili booleane ∧ f e un predicato su X }
  - la funzione f è in una forma particolare: $f = c_1 \wedge c_2 \ldots \wedge c_m$ e ciascuna c_j è l'or ($\vee$) di tre letterali dove un letterale è una variabile o una variabile negata – tipo $x_1 \vee \neg x_2 \vee x_3$
- SOLUZIONE: data ⟨ X,f ⟩ istanza di 3SAT, per decidere se f è *soddisfacibile*,
  - ossia, se esiste una assegnazione **a** di valori in {vero, falso} alle variabili in X tali che f(**a**(X))=vero
- consideriamo il seguente algoritmo:
  - 1) calcola n = |X|;
  - 2) per ogni assegnazione di verità **a** all'insieme delle n variabili in X : verifica se f (**a**(X)) = vero e, in tal caso termina nello stato di accettazione q_A;
  - 3) se non ha mai terminato in q_A al passo 2, termina nello stato di rigetto q_R.
- Vediamo ora il precedente algoritmo implementato utilizzando entrambe le codifiche.

Codifica e soluzione

- Se ⟨ X,f ⟩ è codificata secondo la **CODIFICA χ_1** :
  - Utilizziamo una macchina di Turing T_1 a due nastri e che opera in due fasi:
  - all'inizio della computazione, **χ_1(X,f)** è scritta sul primo nastro, il secondo nastro è vuoto
  - Fase 1: utilizzando i '4' iniziali della codifica di ⟨ X,f ⟩ (che rappresentano il numero |X| di elementi di X), scrive sul secondo nastro tutte le parole binarie di lunghezza |X|, separate le une dalle altre da un '5': ciascuna parola binaria corrisponde ad una assegnazione di verità agli elementi di X
    - ad esempio, se |X|=3, 010 corrisponde a: $a(x_1)$=falso, $a(x_2)$=vero, $a(x_3)$=falso
  - Fase 2: per ogni assegnazione di verità **a** scritta sul secondo nastro, utilizzando la codifica di f scritta sul primo nastro, verifica se **a** soddisfa f: se ciò accade, accetta e termina
  - se ha terminato la fase 2 senza accettare, rigetta
- Bene, ma quanto è dtime (T_1, **χ_1(X,f)**)?
  - **Fase 1**: se n = |X| , la fase 1 richiede almeno 2^n passi – tante sono le assegnazioni possibili!
  - $|\chi_1(X,f)| < n + [3(n+1) +3]\ (2n)^3 < n^4 + 7n\ (8n^3) < 57\ n^4$
- E, quindi, **dtime (T_1, χ_1(X,f)) > 2^n > $2^{\sqrt[4]{|\chi_1(X,f)|}/57}$**

Codifica e soluzione

- Se ⟨ X,f ⟩ è codificata secondo la **CODIFICA χ_2** :
  - esempio: 1111 2 1101 2 1011 2 1001 2 0110 2 0101 2 0011 2 0000 2
  - Utilizziamo una macchina di Turing T_2 ad un solo nastro:
  - all'inizio della computazione, **χ_2 (X,f)** è scritta sul nastro
  - T_2 scandisce l'input: poiché il carattere ('0' o '1') a sinistra di un '2' è il valore assunto da f quando alle sue variabili sono assegnati i valori a sinistra di quel carattere, se trova un '1' a sinistra di un '2' allora accetta e termina
  - poiché **χ_2 (X,f)** contiene in sé tutte le possibili assegnazioni di verità alle variabili in f, se T_2 ha terminato scansione dell'input senza accettare, rigetta
- Bene, ma quanto è dtime (T_2, **χ_2 (X,f)**)?
- Questa volta è facilissimo: T_2 deve solo scandire una volta l'input!
- E, quindi, **dtime (T_2, χ_2 (X,f)) = $|\chi_2$ (X,f)|**

Codifica e soluzione

- Riassumiamo
- Se ⟨ X,f ⟩ è codificata secondo la **CODIFICA χ_1**, implementiamo l'algoritmo mediante una macchina T_1 tale che **dtime (T_1, χ_1 (X,f)) > $2^{\beta(n)}$**
  - con $\beta(n) \approx \sqrt[4]{|\chi_1(X,f)|}$
  - ossia, l'algoritmo che decide 3SAT impiega tempo esponenziale nella lunghezza della CODIFICA χ_1
- Se ⟨ X,f ⟩ è codificata secondo la **CODIFICA χ_2**, implementiamo l'algoritmo mediante una macchina T_2 tale che **dtime (T_2, χ_2 (X,f)) = $|\chi_2(X,f)|$**
  - ossia, *lo stesso algoritmo* che decide 3SAT impiega tempo lineare nella lunghezza della CODIFICA χ_2
- Ora, ricordando che un linguaggio è nella classe P se esiste una macchina di Turing deterministica che lo decide in tempo polinomiale, possiamo concludere che il linguaggio associato a 3SAT appartiene a P?
- Osservate che *T_1 e T_2 implementano lo stesso algoritmo*
  - ma operano su due codifiche differenti!

Codifica e soluzione

- Osservate che T_1 e T_2 *implementano lo stesso algoritmo*
  - ma operano su due codifiche differenti!
- Dunque, la caratteristica *essere un algoritmo polinomiale* dipende dal modo in cui è codificato l'input?
  - Sì e no, in effetti…
- Perché la complessità di un algoritmo è espressa in termini di lunghezza dell'input
  - e, quindi, da come viene codificato il suo input!
- E noi, la codifica dell'input possiamo renderla lunga quanto ci pare
  - ad esempio, aggiungendoci un sacco di caratteri privi di significato
- Possiamo prendere, ad esempio, la CODIFICA χ_1 e aggiungervi, alla fine, $2^{|X|}$ '5' – e così otterremmo
  - χ_3 **(X,f)** = 444 0 100 2 0 010 2 0 001 3 0 100 2 1 010 2 1 001 55555555 (adesso $|\chi_3(X,f)| > 2^n$)
  - da cui deriveremmo una macchina T_3 per 3SAT tale che **dtime** $(T_3, \chi_3(X,f)) \in O(|\chi_3(X,f)|)$
- "ma questa codifica è **irragionevolmente** lunga!", direte voi…

Codifica e soluzione

- Possiamo prendere, ad esempio, la CODIFICA χ_1 e aggiungervi, alla fine, $2^{|X|}$ '5'
- E così otterremmo
  - χ_3 (X,f) = 444 0 100 2 0 010 2 0 001 3 0 100 2 1 010 2 1 001 55555555
  - da cui deriveremmo una macchina T_3 per 3SAT tale che **dtime** (T_3, χ_3(X,f)) $\in$ O(| χ_3(X,f)|)
- "ma questa codifica è irragionevolmente lunga!", direte voi…
- E infatti, rispondo io!
- Ripensiamo alle codifiche χ_1 e χ_2 :
  - la codifica χ_1 rappresenta di ⟨ X,f ⟩ solo l'informazione *strettamente necessaria*, ossia, la struttura di f
  - la codifica χ_2 rappresenta, invece, ⟨ X,f ⟩ in forma estesa - in effetti, χ_2 **contiene la soluzione del problema** così che per trovare la soluzione è sufficiente leggere la codifica
  - ma questo significa che calcolare la codifica χ_2 ha richiesto un sacco di tempo!
  - Ossia, detto altrimenti, **il tempo impiegato dalla computazione $T_1(\chi_1(X,f))$ lo dobbiamo impiegare noi per calcolare χ_2 (X,f) se vogliamo utilizzare quest'ultima codifica**…

Codifiche (ir)ragionevoli

- Possiamo prendere, ad esempio, la CODIFICA χ_1 e aggiungervi, alla fine, $2^{|X|}$ '5'
- "ma questa codifica è irragionevolmente lunga!", direte voi…
- Ripensiamo a χ_1 e χ_2 : la codifica χ_2 è molto più lunga della codifica χ_1
- in effetti, χ_2 è esponenzialmente più lunga di χ_1

- Informalmente, **una codifica χ per un problema Γ è irragionevole se esiste un'altra codifica χ'**
  tale che le parole in cui χ codifica le istanze di Γ sono "più che polinomialmente" più lunghe delle parole in cui χ' codifica le istanze di Γ
- Questo significa che **esiste una funzione "più che polinomiale" F tale che , per qualche istanza x di Γ, $|\chi(x)| \geq F(|\chi'(x)|)$**
  - F: $\mathbb{N} \rightarrow \mathbb{N}$ è "più che polinomiale" se, per ogni $k \in \mathbb{N}$, $F(n) \in \Omega(n^k)$
- Ossia, informalmente, il rapporto fra $|\chi(x)|$ e $|\chi'(x)|$ è più grande di qualsiasi polinomio!
  - Quel che accadeva a χ_1 e χ_2 : perciò, χ_2 è una codifica irragionevole di 3SAT

Codifiche ragionevoli

- Informalmente, una codifica χ per un problema Γ è **irragionevole** se **esiste un'altra codifica χ'** tale che le parole in cui χ codifica le istanze di Γ sono "più che polinomialmente" più lunghe delle parole parole in cui χ' codifica le istanze di Γ
- E, quindi, **una codifica χ per un problema Γ è ragionevole se**,
  - **<u>comunque</u> si scelga un'altra codifica χ' per Γ**,
  - esistono tre interi k, h_1 e h_2 tali che, **per ogni istanza x di Γ** ,
  - $| \chi(x)| \leq h_1 | \chi'(x)|^k + h_2$
- Questo significa che
  - se χ è una codifica ragionevole per Γ ,
  - comunque scegliamo un'altra codifica χ' per Γ ,
  - può succedere che le parole risultanti dalla codifica χ' siano più corte delle parole risultanti dalla codifica χ
  - ma esiste un polinomio p tale che, qualunque sia l'istanza x di Γ , $| \chi (x) |$ non è più grande di $p(| \chi'(x) |)$

Complessità di problemi e codifica

- Alla luce di quanto abbiamo detto sino ad ora, dovrebbe essere chiaro che
- Possiamo estendere ai problemi quello che abbiamo studiato relativamente alla complessità di linguaggi
- **a patto però di utilizzare codifiche ragionevoli per codificare le istanze dei problemi**
- perché quando si utilizzano codifiche irragionevoli
- non ha più senso parlare della complessità di un problema
  - perché potremmo aver trasferito nella complessità della codifica la complessità di risoluzione del problema
  - esattamente come abbiamo discusso nel caso della codifica χ_2 del problema 3SAT
- Per questo, **d'ora in poi, faremo riferimento sempre a codifiche ragionevoli**

- E con questo abbiamo terminato il paragrafo 7.4