

# Breve introduzione alla NP-completezza



# Problemi decisionali

- Un problema decisionale è un problema che ammette una risposta di tipo booleano – sì oppure no
- Esempio 1 - Circuito Hamiltoniano (HC): dato un grafo  $G=(V,E)$ , esiste un ciclo in  $G$  che passa una e una sola volta per ciascun nodo in  $V$ ?
- Esempio 2 – Short Path (SP): dati un grafo  $G=(V,E)$ , una coppia di nodi  $u,v \in V$  e un intero  $k$ , esiste in  $G$  un percorso da  $u$  a  $v$  di lunghezza  $\leq k$ ?
- Esempio 3 - Satisfiability (SAT): dati un insieme  $X$  di variabili booleane e una funzione  $f: \{vero, falso\}^X \rightarrow \{vero, falso\}$ , esiste una assegnazione di verità alle variabili in  $X$   $\alpha: X \rightarrow \{vero, falso\}$  tale che  $f(\alpha(X)) = vero$ ?

# Istanze di problemi decisionali

- L'insieme delle istanze di un problema decisionale è la descrizione dell'insieme dei dati di quel problema
- Esempio 1 - Circuito Hamiltoniano (HC): dato un grafo  $G=(V,E)$ , esiste un ciclo in  $G$  che passa una e una sola volta per ciascun nodo in  $V$ ?
  - l'insieme delle istanze di HC è  $I_{HC} = \{ \langle G=(V,E) \rangle : G \text{ è un grafo non orientato} \}$
- Esempio 2 – Short Path (SP): dati un grafo  $G=(V,E)$ , una coppia di nodi  $u,v \in V$  e un intero  $k$ , esiste in  $G$  un percorso da  $u$  a  $v$  di lunghezza  $\leq k$ ?
  - l'insieme delle istanze di SP è  $I_{SP} = \{ \langle G=(V,E), u, v, k \rangle : G \text{ è un grafo} \wedge u, v \in V \wedge k \in \mathbb{N} \}$
- Esempio 1 - Satisfiability (SAT): dati un insieme  $X$  di variabili booleane e una funzione  $f: \{vero, falso\}^X \rightarrow \{vero, falso\}$ , esiste una assegnazione di verità alle variabili in  $X$   $\alpha: X \rightarrow \{vero, falso\}$  tale che  $f(\alpha(X)) = vero$ ?
  - l'insieme delle istanze di SAT è  $I_{SAT} = \{ \langle X, f \rangle : f: \{vero, falso\}^X \rightarrow \{vero, falso\} \}$
- Un'istanza di un problema decisionale  $\Gamma$  è un elemento di  $I_\Gamma$

# Problemi decisionali in P

- Consideriamo il problema SP e il seguente algoritmo che lo *decide*
- **Algoritmo A-SP**
  - **input:**  $G=(V,E)$ ,  $u,v \in V$ ,  $k \in \mathbb{N}$
  - **fase 1:** mediante l'algoritmo di Dijkstra (ad esempio), calcola il percorso  $p$  di lunghezza minima fra  $u$  e  $v$  (se un tale percorso esiste)
  - **fase 2:** verifica se la lunghezza di  $p$  è  $\leq k$  e in caso affermativo rispondi sì, altrimenti rispondi no
- Il numero di "passi" eseguiti da **A-SP** è proporzionale a  $|V|^2$  - ossia è  $O(|V|^2)$
- e quindi il problema SP appartiene alla classe **P**

**la classe dei problemi di decisione che sono decisi da un algoritmo che opera in un numero di "passi" polinomiale nella dimensione dell'istanza**

- OSSERVAZIONE: l'algoritmo A-SP è di tipo costruttivo, ossia, per decidere se esiste in  $G$  un percorso fra  $u$  e  $v$  di lunghezza  $\leq k$  prova a costruire un siffatto percorso. Comunque, per rispondere al quesito di un problema di decisione l'algoritmo potrebbe seguire strade diverse da quella costruttiva – ad esempio, potrebbe dimostrare se qualche proprietà connessa al quesito è vera
  - ad esempio, per decidere se un grafo è planare è sufficiente verificare se esso soddisfa la formula di Eulero, senza dover descriverne un disegno sul piano

# Problemi decisionali in NP

- Consideriamo il problema HC: per questo problema non è stato fino ad ora possibile progettare un algoritmo che lo decida in tempo polinomiale
  - ossia, utilizzando un numero di "passi" proporzionale alla dimensione della sua istanza
  - dove la dimensione dell'istanza  $\langle G=(V,E) \rangle$  è, sostanzialmente, il numero di bit necessari a descrivere  $G$
- Però, potremmo chiedere consiglio ai nostri amici
- Potremmo, cioè, pensare ad un algoritmo della forma seguente:
- **Algoritmo A1-HC**
  - **input:**  $G=(V,E)$
  - **fase 1:** chiedi ad un amico bravo (diciamo, un *genio*) se  $G$  contiene un ciclo che passa una e una sola volta per ciascun nodo
  - **fase 2:** se l'amico dice di sì allora la risposta è sì, altrimenti la risposta è no
- Bello, intuitivo, facile, ma questo ragionamento ha una pecca...
- Come faccio a fidarmi della risposta del mio amico (seppur bravo)?

# Problemi decisionali in NP

- Eh, no! Non posso fidarmi della risposta del mio amico (seppur bravo)
- Devo almeno chiedergli una prova che quel che dice è vero!
- Una prova della quale non mi fiderò, ma che andrò a verificare!
- Modifichiamo, allora, l'algoritmo **A1-HC** nel modo seguente:
- **Algoritmo NA-HC**
  - **input:**  $G=(V,E)$
  - **fase 1:** chiedi ad un amico bravo (un *genio*!) un ciclo  $c$  che passa una e una sola volta per ciascun nodo di  $G$ , se un tale ciclo esiste
  - **fase 2:** verifica se  $c$  passa davvero una e una sola volta per ciascun nodo di  $G$  e, in tal caso, la risposta è sì
- Ora, verificare "se  $c$  passa davvero una e una sola volta per ciascun nodo di  $G$ " richiede tempo polinomiale nella dimensione di  $G$
- Perciò, se disponessimo di un genio capace di suggerirci il ciclo  $c$  giusto
- in tempo polinomiale riusciremmo a rispondere sì correttamente

# Problemi decisionali in NP

## ➤ Algoritmo NA-HC

- **input:**  $G=(V,E)$
  - **fase 1:** chiedi ad un amico bravo (un *genio!*) un ciclo  $c$  che passa una e una sola volta per ciascun nodo di  $G$ , se un tale ciclo esiste
  - **fase 2:** verifica se  $c$  passa davvero una e una sola volta per ciascun nodo di  $G$  e, in tal caso, la risposta è sì
- OSSERVAZIONE 1: esattamente come nel caso dei problemi in  $P$ , non è detto che il genio ci debba suggerire una prova costruttiva del fatto che la nostra istanza ha risposta sì
- potrebbe fornirci, genericamente, una dimostrazione del fatto che la nostra istanza ha risposta sì
- La cosa importante è che per ogni istanza che ha risposta sì il genio può fornirci una prova di questo fatto
- e che tale prova può essere verificata in tempo polinomiale nella dimensione dell'input

# Problemi decisionali in NP

## ■ Algoritmo NA-HC

- **input:**  $G=(V,E)$
  - **fase 1:** chiedi ad un amico bravo (un *genio*!) un ciclo  $c$  che passa una e una sola volta per ciascun nodo di  $G$ , se un tale ciclo esiste
  - **fase 2:** verifica se  $c$  passa davvero una e una sola volta per ciascun nodo di  $G$  e, in tal caso, la risposta è sì
- OSSERVAZIONE 2: "per ogni istanza che ha risposta sì il genio può fornirci una prova di questo fatto e tale prova può essere verificata in tempo polinomiale nella dimensione dell'input". Bene! Ma se l'istanza ha risposta no?!
- Riflettiamo: se il nostro genio ci comunica un ciclo che non passa per tutti i nodi di  $G$ , o che passa più volte per lo stesso nodo, cosa possiamo concludere?
  - Riflettiamo: per poter concludere che  $G$  non contiene un ciclo hamiltoniano è necessario che **nessun** ciclo in  $G$  passi una e una sola volta per ciascun nodo
    - non è certo sufficiente il singolo ciclo  $c$  mostratoci dal genio!
  - Perciò, possiamo concludere soltanto che il nostro genio non ha saputo trovare un ciclo hamiltoniano in  $G$ 
    - ma non sappiamo se non l'ha trovato perchè non esiste o perchè lui non è abbastanza geniale!



# La classe NP

- **La classe NP è la classe dei problemi verificabili in tempo polinomiale**
- ossia, un problema è in NP se
  - esiste un genio (del quale non mi fido) tale che
  - per ogni istanza del problema che ha risposta sì
  - il genio mi sa suggerire una prova che quell'istanza ha risposta sì
  - e quella prova io posso verificarla in tempo polinomiale nella dimensione dell'istanza
- Ma, in definitiva, il genio possiamo evitare di tirarlo in ballo:
- ossia, **un problema è in NP se**
  - **per ogni istanza del problema che ha risposta sì**
  - **esiste una prova che quell'istanza ha risposta sì**
  - **e quella prova io posso verificarla in tempo polinomiale nella dimensione dell'istanza**

# P è contenuto in NP

- Ri-pensiamo al problema **SP**
- L'algoritmo **A-SP** è stato progettato proprio per decidere SP
- Dunque, se l'esecuzione dell'algoritmo **A-SP** con input una qualche istanza  $\langle G=(V,E),u,v,k \rangle$  risponde sì, è detta esecuzione stessa una prova che  $\langle G=(V,E),u,v,k \rangle$  ha risposta sì
- perciò, verificare la prova significa, sostanzialmente, ... ripetere l'esecuzione di **A-SP** con input  $\langle G=(V,E),u,v,k \rangle$ 
  - ossia, in qualche modo, prova e verifica coincidono
- e poiché l'esecuzione di **A-SP** con input  $\langle G=(V,E),u,v,k \rangle$  termina entro un numero di "passi" polinomiale nella dimensione di  $\langle G=(V,E),u,v,k \rangle$
- questo prova che  $SP \in NP$ !
- E siccome lo stesso ragionamento lo possiamo ripetere per ogni problema appartenente a P, possiamo concludere che

$$P \subseteq NP$$

# La congettura fondamentale

- Dunque,  $P \subseteq NP$
- Ma, sino ad ora non si è riusciti a dimostrare se si tratta di una relazione di inclusione stretta oppure di una uguaglianza
- anche se si *congettura* che sia  $P \neq NP$
- e siccome NP contiene moltissimi problemi di notevole rilevanza applicativa che non si riesce a collocare in P

sulla soluzione della congettura  $P \neq NP$

che è la **congettura fondamentale della teoria della complessità computazionale**  
è stato posto un premio di un milione di dollari

- Per provare a individuare i problemi separatori fra P e NP
  - ossia, i problemi appartenenti a NP-P
- sono stati introdotti i concetti di riduzione polinomiale e di NP-completezza

# Riduzioni polinomiali

- Un problema  $\Gamma$  è riducibile polinomialmente a un problema  $\Delta$  se
  - esiste un algoritmo  $A$  che, presa in input una istanza  $x$  di  $\Gamma$ , in un numero di "passi" polinomiale nella lunghezza di  $x$  calcola una istanza  $y$  di  $\Delta$  in modo tale che
  - $x$  ha risposta sì (per  $\Gamma$ ) se e soltanto se  $y$  ha risposta sì (per  $\Delta$ )
- se  $\Gamma$  è riducibile polinomialmente a  $\Delta$  scriviamo  $\Gamma \leq \Delta$
- In pratica: se so ridurre polinomialmente  $\Gamma$  a un problema  $\Delta$  che so decidere allora posso combinare la riduzione con l'algoritmo che decide  $\Delta$  ed ottenere un algoritmo che decide  $\Gamma$  e non impiega "troppi più passi" di quelli impiegati per decidere  $\Delta$

# P e la riducibilità polinomiale

- Se so ridurre polinomialmente  $\Gamma$  a un problema  $\Delta$  che so decidere allora posso combinare la riduzione con l'algoritmo che decide  $\Delta$  ed ottenere un algoritmo che decide  $\Gamma$  e non impiega "troppi più passi" di quelli impiegati per decidere  $\Delta$
- In particolare, vale il seguente

**TEOREMA (chiusura di P rispetto alla riducibilità polinomiale):**

**Siano  $\Gamma$  e  $\Delta$  due problemi decisionali; se  $\Gamma \leq \Delta$  e  $\Delta \in P$  allora  $\Gamma \in P$**

# Il problema SAT

- Consideriamo il seguente problema decisionale

Satisfiability (SAT): dati un insieme  $X$  di variabili booleane e una funzione  $f: \{vero, falso\}^X \rightarrow \{vero, falso\}$ , esiste una assegnazione di verità alle variabili in  $X$   $\alpha: X \rightarrow \{vero, falso\}$  tale che  $f(\alpha(X)) = vero$  ?

- Nel 1971 è stato dimostrato il seguente

**TEOREMA (di Cook-Levin):**

**per ogni problema di decisione  $\Gamma \in NP$  vale che  $\Gamma \leq SAT$**

- Il teorema di Cook-Levin e il teorema di chiusura di  $P$  rispetto alla riducibilità polinomiale mostrano che

Se esistesse un algoritmo polinomiale per decidere SAT allora, per ogni  $\Gamma \in NP$ , esisterebbe un algoritmo polinomiale per decidere  $\Gamma$

- ossia,

**Se esistesse un algoritmo polinomiale per decidere SAT allora sarebbe  $P=NP$**

# Il problema SAT

- Consideriamo il seguente problema decisionale

Satisfiability (SAT): dati un insieme  $X$  di variabili booleane e una funzione  $f: \{vero, falso\}^X \rightarrow \{vero, falso\}$ , esiste una assegnazione di verità alle variabili in  $X$   $\alpha: X \rightarrow \{vero, falso\}$  tale che  $f(\alpha(X)) = vero$  ?

- SAT  $\in$  NP: infatti se  $\langle X, f \rangle$  è un'istanza sì di SAT allora una prova di ciò è una assegnazione di verità alle variabili in  $X$  (che può essere verificata in tempo proporzionale alla dimensioni di  $f$ )
- Inoltre, come abbiamo visto

**Se esistesse un algoritmo polinomiale per decidere SAT allora sarebbe  $P=NP$**

- e, dunque, possiamo affermare che

**Se  $P \neq NP$  allora SAT  $\in$  NP - P**

- SAT è un possibile problema separatore fra P e NP

# I problemi NP-completi

- Esistono in NP numerosi problemi con le stesse caratteristiche di SAT: sono i problemi NP-completi

- Un problema  $\Delta$  è NP-completo se**

**1)  $\Delta \in \text{NP}$**

**2) per ogni problema  $\Gamma \in \text{NP}$  vale che  $\Gamma \leq \Delta$**

- Dunque, SAT è NP-completo
- Dunque, come SAT, i problemi NP-completi sono i possibili separatori fra P e NP
- Per dimostrare che un problema  $\Delta \in \text{NP}$  è NP-completo si usa il seguente

**TEOREMA: se  $\Gamma$  è NP-completo e  $\Gamma \leq \Delta$  (con  $\Delta \in \text{NP}$ ), allora  $\Delta$  è NP-completo**

- ossia, per dimostrare che un problema  $\Delta \in \text{NP}$  è NP-completo è sufficiente scegliere un problema  $\Gamma$  che già sappiamo essere NP-completo e mostrare che  $\Gamma \leq \Delta$
- E di problemi NP-completi noti ce ne sono a bizzeffe
  - HC, CLIQUE, VERTEX COVER, DOMINATING SET, TRAVELLING SALESMAN PROBLEM, ...
  - solo per citarne alcuni su grafi