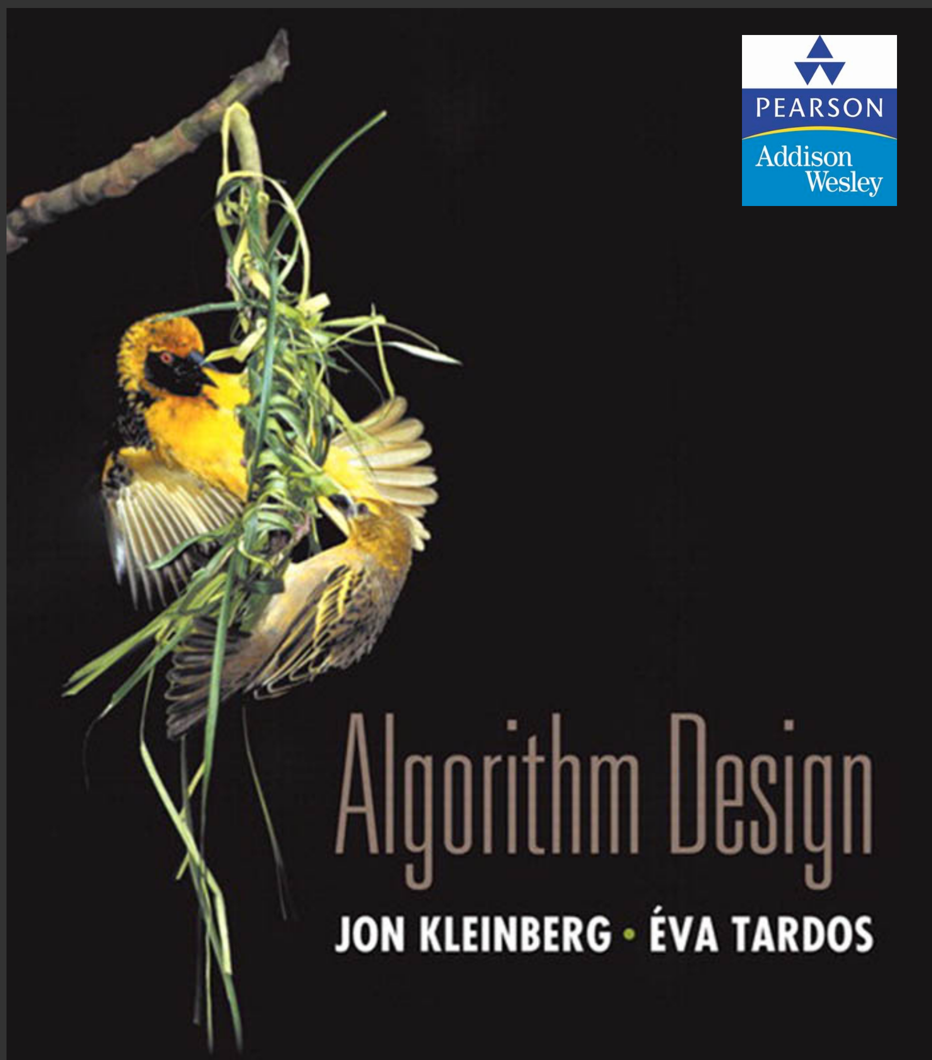


Traduzione e adattamento di Vincenzo Bonifaci
Original lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

9. PSPACE

- *classe di complessità PSPACE*
- *soddisfacibilità quantificata*
- *problema della pianificazione*
- *PSPACE-completezza*

Last updated on 19/02/2020 14:24

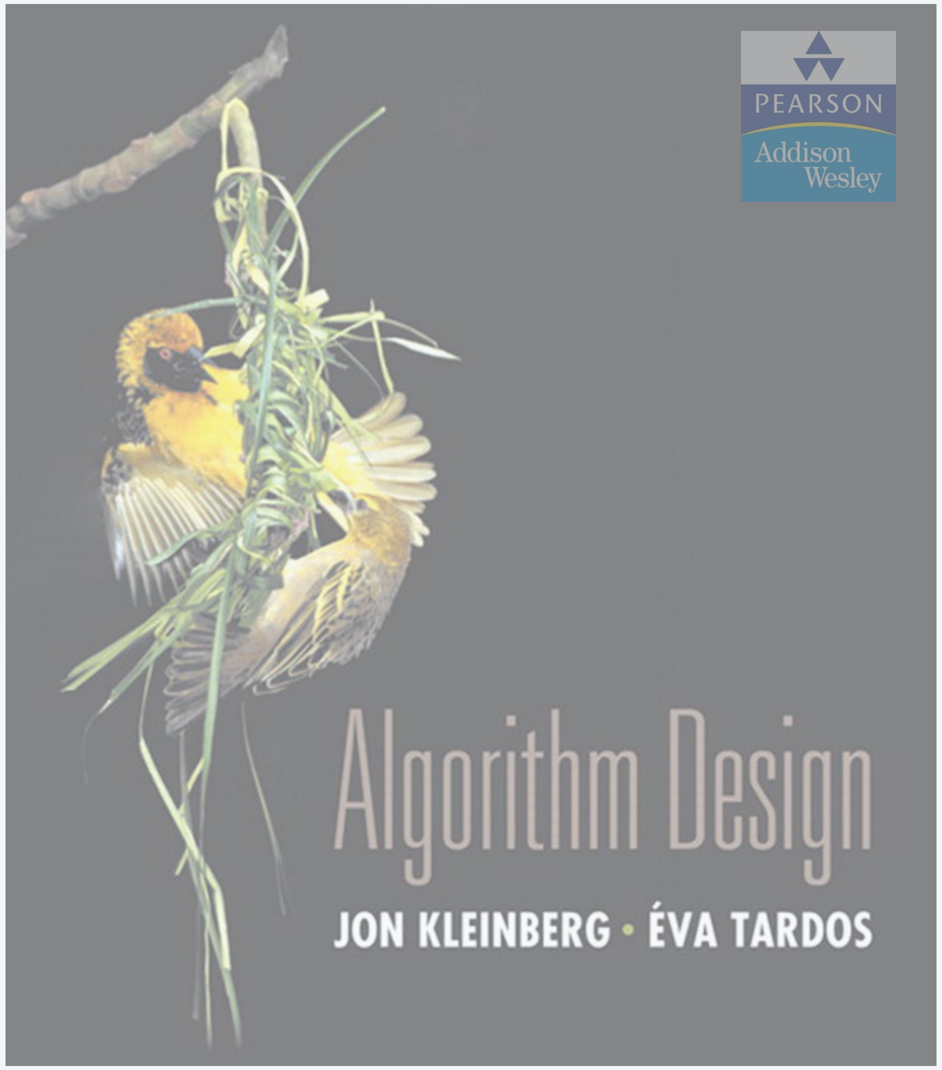
Gioco geografia

Geografia. Alice nomina una città capitale c del paese in cui si trova. Bob nomina una capitale c' con la lettera con cui finisce c. Alice e Bob ripetono questo gioco finché qualche giocatore non può più continuare.
Esiste Alice una strategia vincente per Alice?

Es. Budapest → Tokyo → Ottawa → Ankara → Amsterdam → Moscow → Washington → Nairobi → ...

Geografia su grafi. Dato un digrafo $G = (V, E)$ ed un nodo iniziale s, due giocatori si alternano seguendo, se possibile, un arco uscente dal nodo corrente verso un nodo non visitato. Il primo giocatore ha una strategia per arrivare a compiere l'ultima mossa ammissibile?

Nota. Alcuni problemi (in particolari quelli di giochi a 2 giocatori e di AI) evadono la classificazione in termini di **P**, **EXPTIME**, **NP**, e **NP**-completi.



9. PSPACE

- *classe di complessità PSPACE*
- *quantified satisfiability*
- *planning problem*
- *PSPACE-complete*

PSPACE

P. Problemi di decisione risolvibili in tempo polinomiale.

PSPACE. Problemi di decisione risolvibili in spazio polinomiale.

Osservazione. **P** $\subseteq$ **PSPACE**.

un algoritmo tempo-polinomiale
può consumare solo una quantità
polinomiale di spazio

PSPACE

Contatore binario. Conta da 0 a $2^n - 1$ in binario.

Algoritmo. Usa un contatore ad n bit.

Prop. 3-SAT $\in$ **PSPACE.**

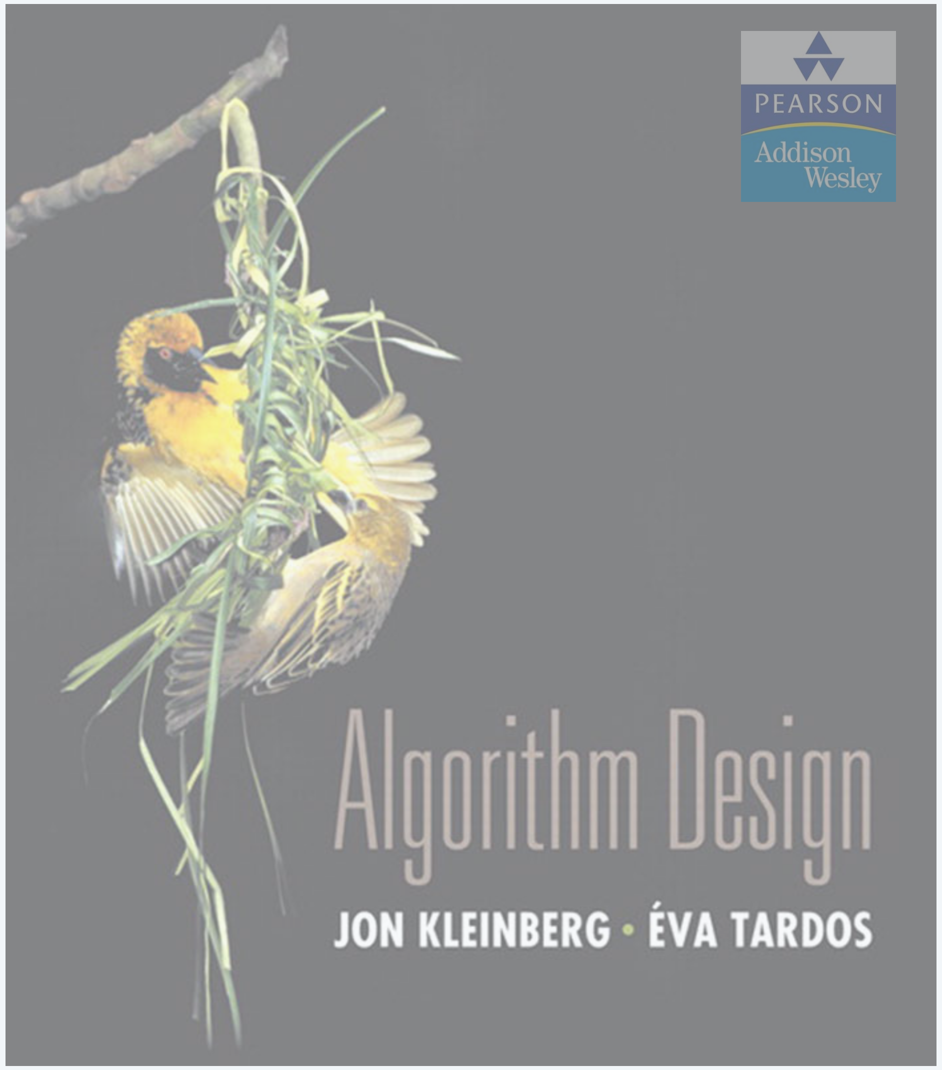
Dim.

- Enumerate tutte le 2^n possibili assegnazioni di verità tramite contatore.
- Controlla ogni assegnazione per vedere se soddisfa tutte le clausole. ▪

Teorema. **NP** $\subseteq$ **PSPACE.**

Dim. Considera un qualunque problema $Y \in$ **NP**.

- Poiché $Y \leq_P$ 3-SAT, esiste un algoritmo che risolve Y in tempo polinomiale più un numero polinomiale di chiamate ad una scatola nera per 3-SAT.
- La scatola nera può essere implementata in spazio polinomiale. ▪



9. PSPACE

- *PSPACE complexity class*
- *soddisfacibilità quantificata*
- *planning problem*
- *PSPACE-complete*

Soddisfacibilità quantificata

QSAT. Sia $\Phi(x_1, \ldots, x_n)$ una formula booleana in CNF. È vera la seguente formula proposizionale?

$$\exists\, x_1 \;\forall\, x_2 \;\exists\, x_3 \;\forall\, x_4 \;\ldots\; \forall\, x_{n\text{-}1} \;\exists\, x_n \;\Phi(x_1, \ldots, x_n)$$

assumiamo n sia dispari

Intuizione. Amy sceglie un valore di verità per x_1, poi Bob per x_2, poi Amy per x_3, e così via. Amy può soddisfare Φ a prescindere dalle scelte di Bob?

Es. $(x_1 \vee x_2) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$

Sì. Amy setta x_1 a true; Bob setta x_2; Amy setta x_3 uguale a x_2.

Es. $(x_1 \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$

No. Se Amy setta x_1 a false; Bob setta x_2 a false; Amy perde;

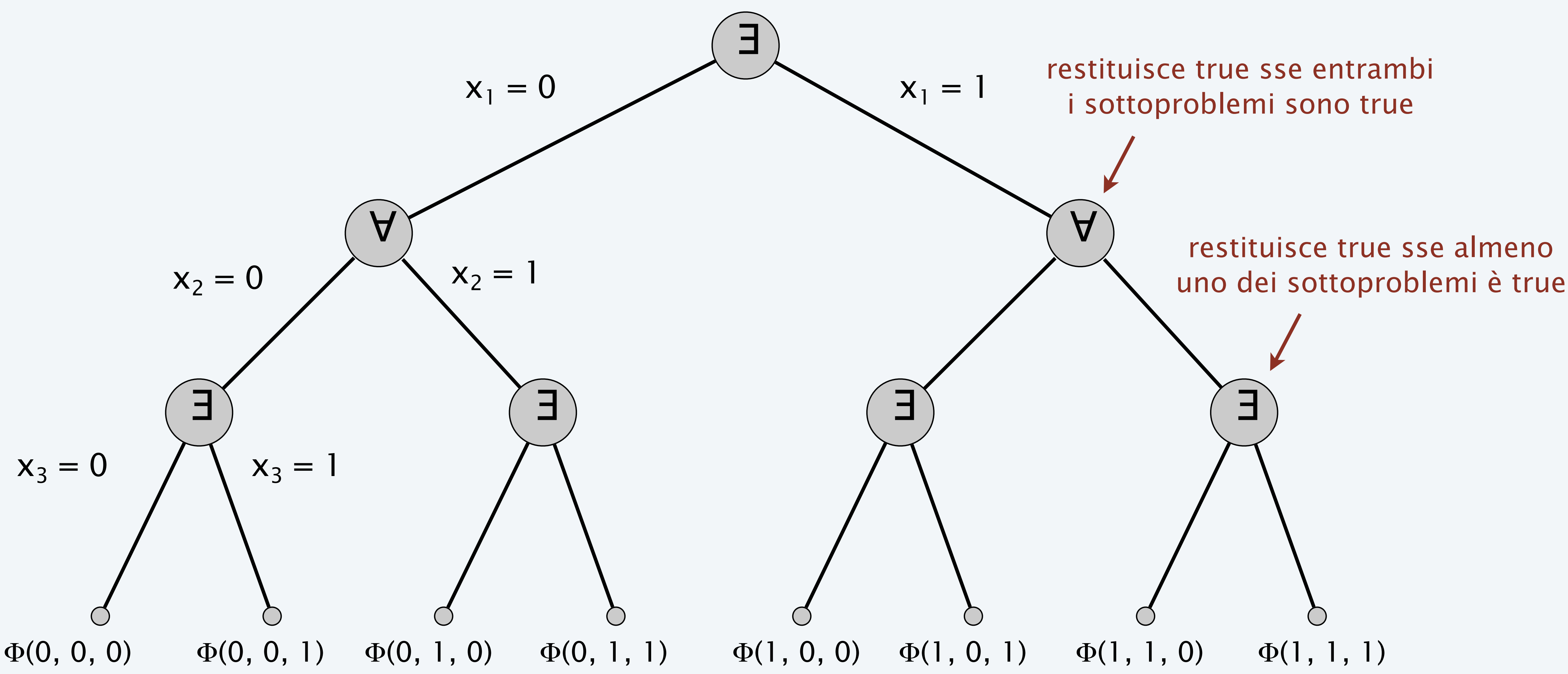
se Amy setta x_1 a true; Bob setta x_2 a true; Amy perde.

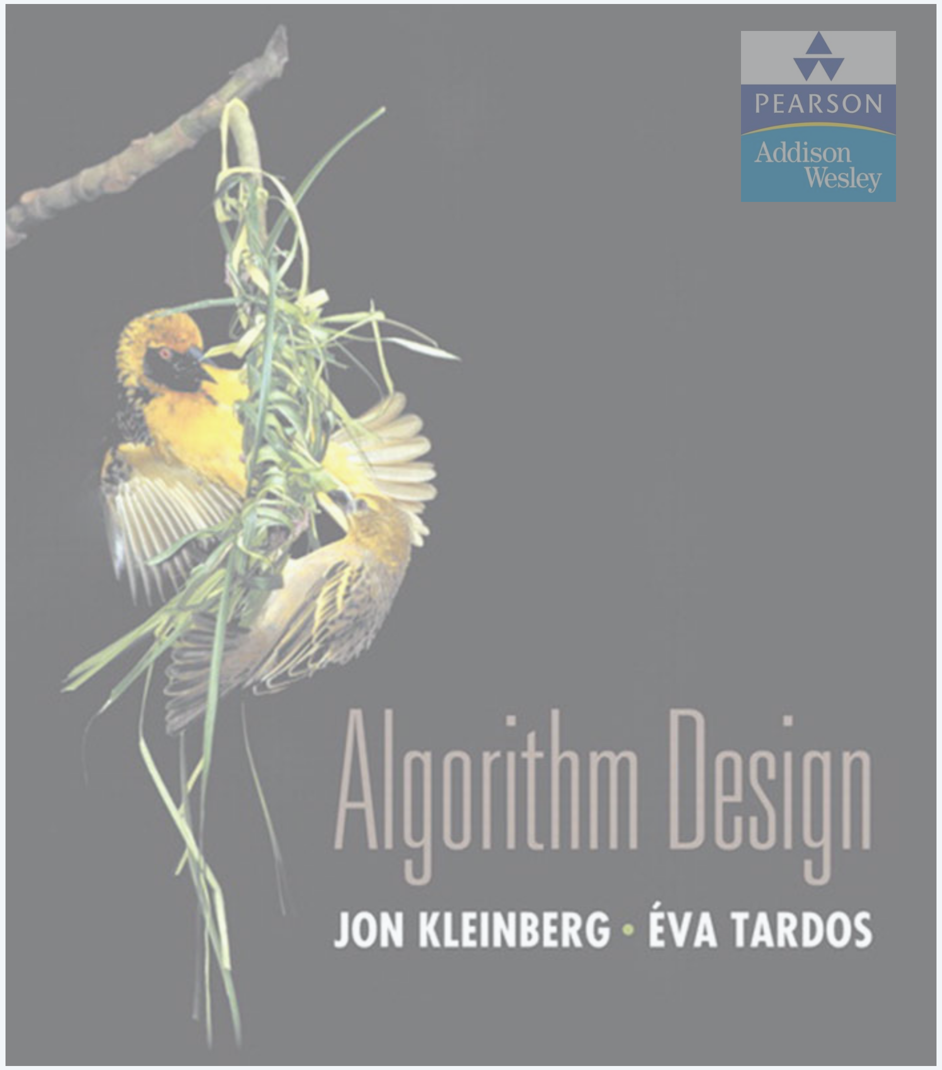
Quantified satisfiability è in PSPACE

Teorema. Q-SAT ∈ **PSPACE.**

Dim. Prova ricorsivamente tutte le possibilità.

- Serve solo un bit di informazione per ogni sottoproblema.
- Lo spazio usato è proporzionale alla profondità della pila di attivazione delle invocazioni di funzione.





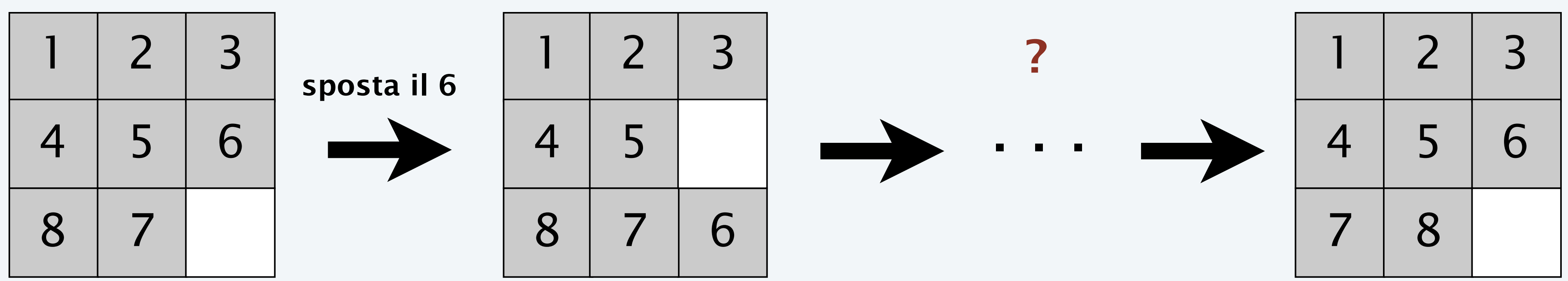
9. PSPACE

- PSPACE complexity class
- quantified satisfiability
- problema della pianificazione
- PSPACE-complete

Gioco del 15

Gioco dell'8, gioco del 15. [Noyes Chapman 1874]

- Tabellone: griglia 3-per-3 di piastrelle etichettate 1–8.
- Mossa lecita: scorrere una piastrella adiacente verso il riquadro vuoto.
- Trova una sequenza di mosse lecite che trasformi la configurazione iniziale nella configurazione obiettivo.



configurazione iniziale

configurazione obiettivo

Problema della pianificazione

Condizioni. Insieme $C = \{ C_1, \ldots, C_n \}$.

Configurazione iniziale. Sottoinsieme $c_0 \subseteq C$ di condizioni inizialmente vere.

Configurazione obiettivo. Sottoinsieme $c^* \subseteq C$ di condizioni da soddisfare.

Operatori. Insieme $O = \{ O_1, \ldots, O_k \}$.

- L'operatore O_i può essere invocato solo sotto certe condizioni.
- Dopo aver applicato O_i alcune condizioni divengono vere, e alcune condizioni divengono false.

PLANNING. È possibile applicare una sequenza di operatori per passare dalla configurazione iniziale alla configurazione obiettivo?

Esempi.

- Gioco del 15.
- Cubo di Rubik.
- Operazioni di logistica per movimentare persone, equipaggiamento, e materiali.

Problema della pianificazione: il gioco dell'8

Esempio di pianificazione. Possiamo risolvere l'istanza del gioco dell'8?

Condizioni. $C_{ij}, 1 \le i, j \le 9$. ← C_{ij} significa che la piastrella i è nel riquadro j

1	2	3
4	5	6
8	7	9

Stato iniziale. $c_0 = \{ C_{11}, C_{22}, \ldots, C_{66}, C_{78}, C_{87}, C_{99} \}$.

Stato obiettivo. $c^* = \{ C_{11}, C_{22}, \ldots, C_{66}, C_{77}, C_{88}, C_{99} \}$.

$\downarrow O_i$

1	2	3
4	5	6
8	9	7

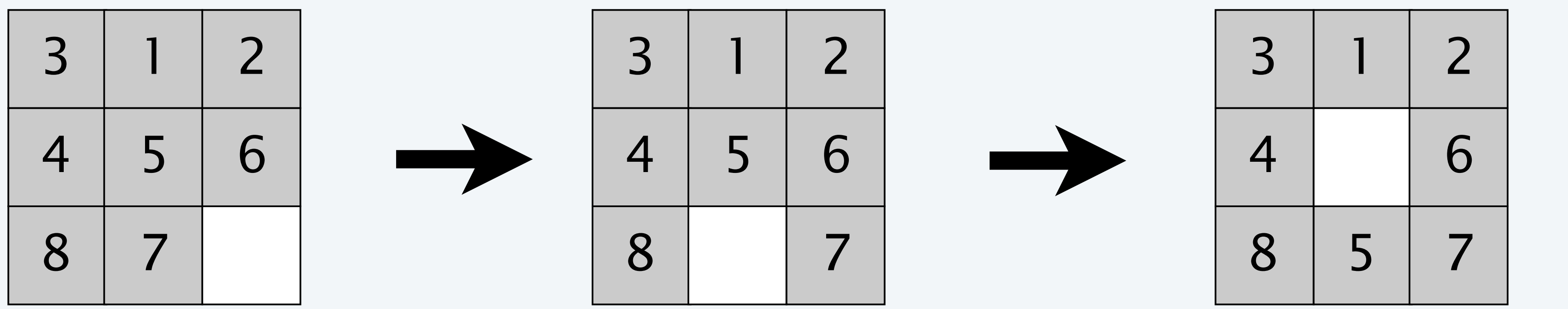
Operatori.

- Precondizioni per $O_i = \{ C_{11}, C_{22}, \ldots, C_{66}, C_{78}, C_{87}, C_{99} \}$.
- Dopo aver invocato O_i, condizioni C_{79} e C_{97} diventano *vere*.
- Dopo aver invocato O_i, condizioni C_{78} e C_{99} diventano *false*.

Soluzione. Quella particolare istanza del gioco dell'8 non ha soluzione!

Digressione: Perché quell'istanza del gioco dell'8 è irrisolvibile?

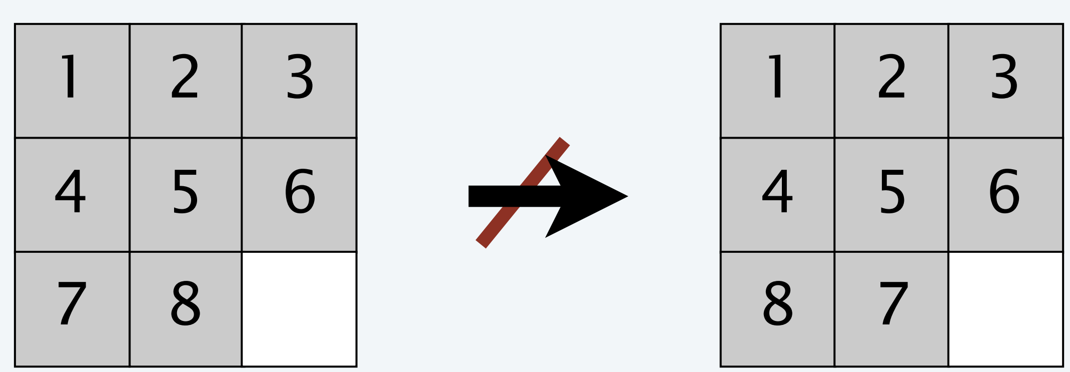
Invariante del gioco dell'8. Qualunque mossa lecita preserva la parità del numero di coppie di pezzi in ordine inverso (numero di inversioni).



3 inversioni
1–3, 2–3, 7–8

3 inversioni
1–3, 2–3, 7–8

5 inversioni
1–3, 2–3, 7–8, 5–8, 5–6



0 inversioni

1 inversione: 7–8

Problema della pianificazione: contatore binario

Esempio di pianificazione. Possiamo incrementare un contatore di n-bit dalla configurazione di tutti 0 alla configurazione di tutti 1?

Condizioni. $C_1, \ldots, C_n$. ⟵ C_i corrisponde al bit i = 1

Stato iniziale. $c_0 = \phi$. ⟵ tutti 0

Stato obiettivo. $c^* = \{C_1, \ldots, C_n\}$. ⟵ tutti 1

Operatori. $O_1, \ldots, O_n$.

gli i–1 bit meno significativi valgono 1

- Per invocare l'operatore O_i, devono valere $C_1, \ldots, C_{i-1}$.
- Dopo aver invocato O_i, la condizione C_i diventa vera. ⟵ imposta il bit i a 1
- Dopo aver invocato O_i, le condizioni $C_1, \ldots, C_{i-1}$ diventano false. ⟵ imposta gli i-1 bit meno significativi a 0

Soluzione. $\{\} \Rightarrow \{C_1\} \Rightarrow \{C_2\} \Rightarrow \{C_1, C_2\} \Rightarrow \{C_3\} \Rightarrow \{C_3, C_1\} \Rightarrow \ldots$

Osservazione. Ogni soluzione richiede almeno $2^n - 1$ passi.

Il problema della pianificazione è in EXPTIME

Grafo delle configurazioni G.

- Un nodo per ognuna delle 2^n configurazioni possibili.
- Un arco da una configurazione c' ad una configurazione c'' se uno degli operatori può trasformare c' in c''.

PLANNING. Esiste un cammino da c_0 a c^* nel grafo delle configurazioni?

Prop. PLANNING $\in$ **EXPTIME**.

Dim. Eseguiamo una BFS per trovare il cammino da c_0 a c^* nel grafo delle configurazioni. ▪

Nota. Il grafo delle configurazioni può avere 2^n nodi, e il cammino minimo può avere lunghezza = $2^n - 1$.

contatore binario

Il problema della pianificazione è in PSPACE

Teorema. PLANNING $\in$ **PSPACE.**

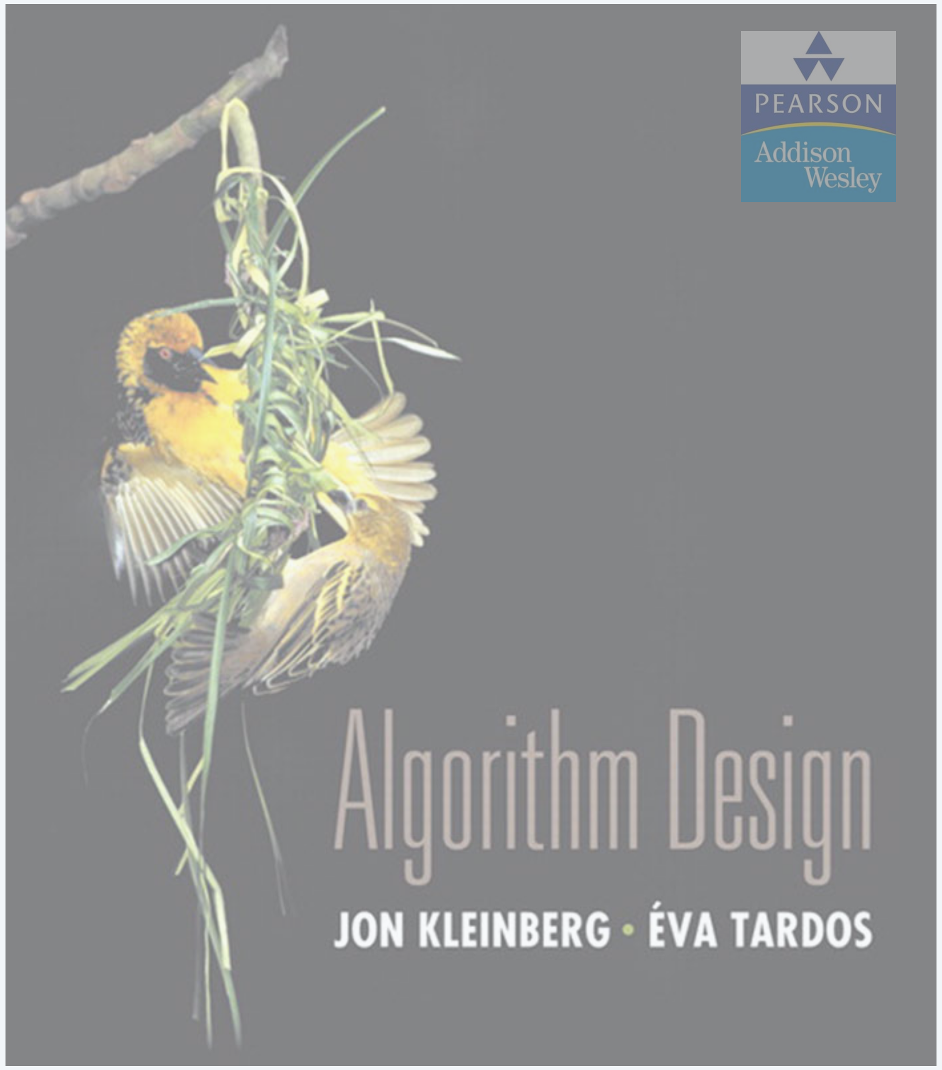
Dim. Supponiamo ci sia un cammino da c_1 a c_2 di lunghezza L.

- Il cammino da c_1 al nodo mediano e da c_2 al nodo mediano sono lunghi al più $\leq L/2$.
- Enumera tutti i possibili nodi mediani.
- Applica ricorsivamente. Profondità della ricorsione $= \log_2 L$. ▪

```
boolean hasPath(c1, c2, L) {
   if (L ≤ 1) return correct answer

               enumerate tramite contatore binario

   foreach configuration c′ {
      boolean x = hasPath(c1, c′, L/2)
      boolean y = hasPath(c2, c′, L/2)
      if (x and y) return true
   }
   return false
}
```



9. PSPACE

- *PSPACE complexity class*
- *quantified satisfiability*
- *planning problem*
- *PSPACE-completezza*

Problemi PSPACE-completi

PSPACE. Problemi di decisioni risolubili in spazio polinomiale.

PSPACE-completi. Problema $Y \in$ **PSPACE**-completi se (i) $Y \in$ **PSPACE** e (ii) per ogni problema $X \in$ **PSPACE**, $X \leq_P Y$.

Teorema. [Stockmeyer–Meyer 1973] QSAT $\in$ **PSPACE**-completi.

Teorema. **PSPACE** $\subseteq$ **EXPTIME**.

Dim. L'algoritmo precedente risolve QSAT in tempo esponenziale; e QSAT è **PSPACE**-*complete*. ▪

Riepilogo. **P** $\subseteq$ **NP** $\subseteq$ **PSPACE** $\subseteq$ **EXPTIME**.

è noto che P ≠ EXPTIME,
ma non è noto se l'inclusione sia stretta;
si congettura che lo siano tutte

Problemi PSPACE-completi

Altri problemi PSPACE-completi.

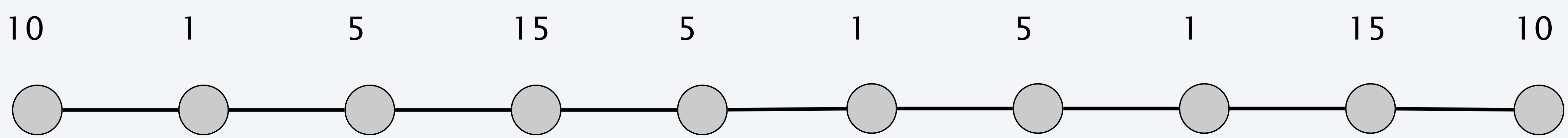
- Locazione competitiva di impianti [Competitive facility location].
- Generalizzazioni naturali di alcuni giochi.
  - Othello, Hex, Geography, Rush-Hour, Instant Insanity
  - Shanghai, go-moku, Sokoban
- Data una macchina di Turing a memoria limitata, essa termina in al più k passi?
- Date due espressioni regolari, esse descrivono linguaggi diversi?
- È possibile spostare e ruotare un oggetto complicato attraverso un corridoio di forma irregolare?
- È possibile raggiungere uno stato di deadlock all'interno di un sistema di processori comunicanti?

Locazione competitiva di impianti

Input. Grafo $G = (V, E)$ con pesi positive sugli archi, e un bersaglio B.

Gioco. Due giocatori in competizione si alternano nello scegliere nodi. Non è permesso scegliere un nodo se uno dei suoi vicini è stato selezionato.

Locazione competitiva di impianti. Il secondo giocatore può garantirsi un profitto di almeno B unità?



sì se B = 20;
no se B = 25

Locazione competitiva di impianti

Prop. COMPETITIVE-FACILITY-LOCATION $\in$ **PSPACE**-completi.

Dim.

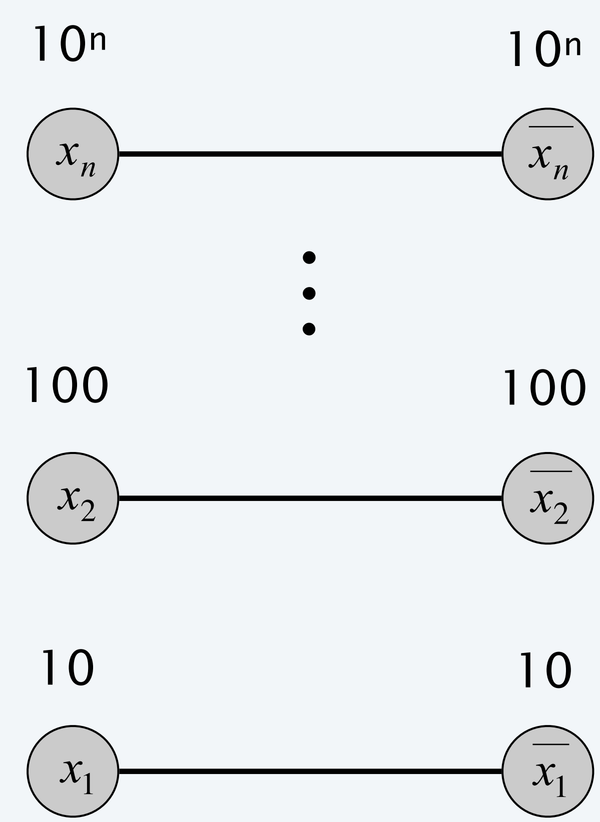
- Per risolverlo in spazio polinomiale, usiamo la ricorsione come in Q-SAT, ma ad ogni passo ci sono fino a n scelte anziché 2.

- Per mostrare che il problema è completo, mostriamo che Q-SAT è riducibile in tempo polinomiale ad esso. Data un'istanza di Q-SAT, costruiamo un'istanza di COMPETITIVE-FACILITY-LOCATION tale che il giocatore 2 ha una strategia vincente sse iff la formula Q-SAT è *vera*.

Locazione competitiva di impianti

Costruzione. Data un'istanza $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ di Q-SAT. ← n è assunto dispari

- Aggiungi un nodo per ogni letterale e il suo negato e collegali.
  (al più uno tra x_i e il suo negato possono essere scelti)
- Scegli $c \geq k + 2$, e metti un peso c^i sul letterale x^i e il suo negato;
  poni $B = c^{n-1} + c^{n-3} + \ldots + c^4 + c^2 + 1$.
  (assicura che le variabili siano selezionate in ordine $x_n, x_{n-1}, \ldots, x_1$)
- Così com'è, il giocatore 2 perderà per 1 punto: $c^{n-1} + c^{n-3} + \ldots + c^4 + c^2$.



Locazione competitiva di impianti

Costruzione. Data un'istanza $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ di Q-SAT.

- Dai al giocatore 2 un'ultima mossa in cui può cercare di vincere.
- Per ogni clausola C_j, aggiungi un nodo di valore 1 ed un arco verso ognuno dei suoi letterali.
- Il giocatore 2 può fare l'ultima mossa sse l'assegnazione di verità definita dai due giocatori non soddisfa qualche clausola. ▪