

## 9. PSPACE

- ▶ *PSPACE complexity class*
- ▶ *quantified satisfiability*
- ▶ *planning problem*
- ▶ *PSPACE-complete*

Lecture slides by Kevin Wayne

Copyright © 2005 Pearson-Addison Wesley

Copyright © 2013 Kevin Wayne

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

Last updated on Sep 8, 2013 6:54 AM

## Geography game

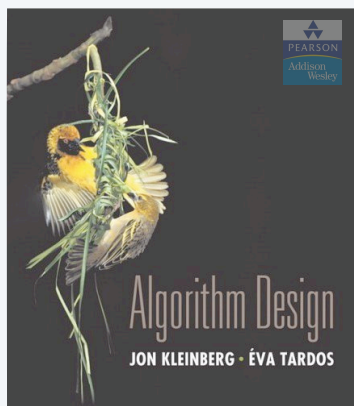
**Geography.** Alice names capital city  $c$  of country she is in. Bob names a capital city  $c'$  that starts with the letter on which  $c$  ends. Alice and Bob repeat this game until one player is unable to continue. Does Alice have a forced win?

**Ex.** Budapest → Tokyo → Ottawa → Ankara → Amsterdam → Moscow → Washington → Nairobi → ...

**Geography on graphs.** Given a directed graph  $G = (V, E)$  and a start node  $s$ , two players alternate turns by following, if possible, an edge out of the current node to an unvisited node. Can first player guarantee to make the last legal move?

**Remark.** Some problems (especially involving 2-player games and AI) defy classification according to  $NP$ ,  $EXPTIME$ ,  $NP$ , and  $NP$ -Complete.

2



## 9. PSPACE

- ▶ *PSPACE complexity class*
- ▶ *quantified satisfiability*
- ▶ *planning problem*
- ▶ *PSPACE-complete*

SECTION 9.1

## PSPACE

**P.** Decision problems solvable in polynomial **time**.

**PSPACE.** Decision problems solvable in polynomial **space**.

**Observation.**  $P \subseteq PSPACE$ .

↑  
poly-time algorithm  
can consume  
only polynomial space

4

## PSPACE

**Binary counter.** Count from 0 to  $2^n - 1$  in binary.

**Algorithm.** Use  $n$  bit odometer.

**Claim.**  $3\text{-SAT} \in PSPACE$ .

**Pf.**

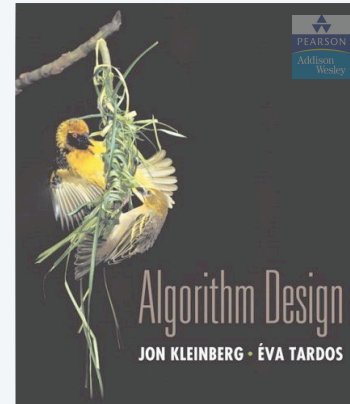
- Enumerate all  $2^n$  possible truth assignments using counter.
- Check each assignment to see if it satisfies all clauses. ▀

**Theorem.**  $NP \subseteq PSPACE$ .

**Pf.** Consider arbitrary problem  $Y \in NP$ .

- Since  $Y \leq_p 3\text{-SAT}$ , there exists algorithm that solves  $Y$  in poly-time plus polynomial number of calls to 3-SAT black box.
- Can implement black box in poly-space. ▀

5



SECTION 9.3

## 9. PSPACE

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

## Quantified satisfiability

**QSAT.** Let  $\Phi(x_1, \dots, x_n)$  be a boolean CNF formula. Is the following propositional formula true?

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

↑  
assume n is odd

**Intuition.** Amy picks truth value for  $x_1$ , then Bob for  $x_2$ , then Amy for  $x_3$ , and so on. Can Amy satisfy  $\Phi$  no matter what Bob does?

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

**Yes.** Amy sets  $x_1$  true; Bob sets  $x_2$ ; Amy sets  $x_3$  to be same as  $x_2$ .

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

**No.** If Amy sets  $x_1$  false; Bob sets  $x_2$  false; Amy loses;

**No.** if Amy sets  $x_1$  true; Bob sets  $x_2$  true; Amy loses.

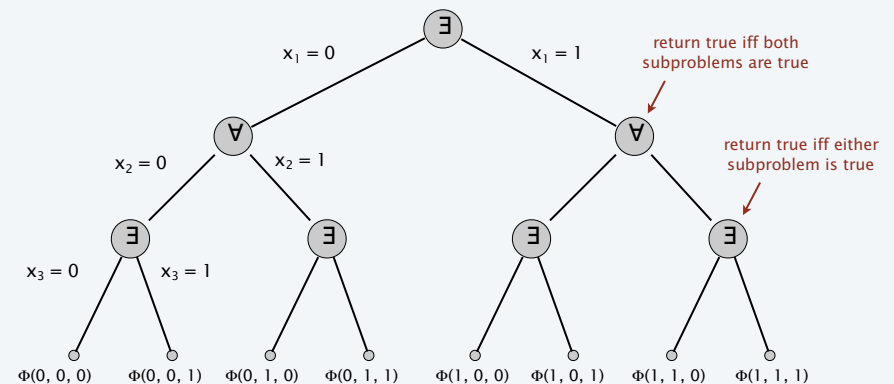
7

## Quantified satisfiability is in PSPACE

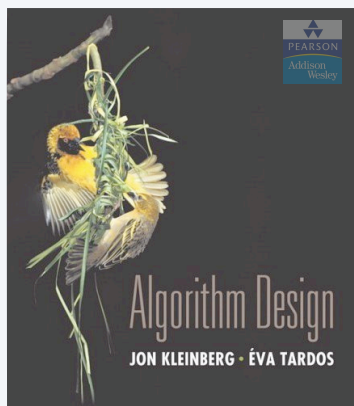
**Theorem.**  $Q\text{-SAT} \in PSPACE$ .

**Pf.** Recursively try all possibilities.

- Only need one bit of information from each subproblem.
- Amount of space is proportional to depth of function call stack.



8



## SECTION 9.4

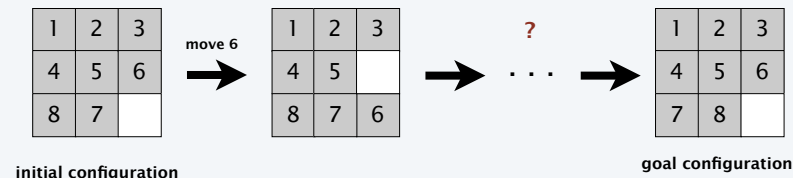
## 9. PSPACE

- ▶ *PSPACE complexity class*
- ▶ *quantified satisfiability*
- ▶ *planning problem*
- ▶ *PSPACE-complete*

## 15-puzzle

8-puzzle, 15-puzzle. [Noyes Chapman 1874]

- Board: 3-by-3 grid of tiles labeled 1–8.
- Legal move: slide neighboring tile into blank (white) square.
- Find sequence of legal moves to transform initial configuration into goal configuration.



## Planning problem

**Conditions.** Set  $C = \{C_1, \dots, C_n\}$ .

**Initial configuration.** Subset  $c_0 \subseteq C$  of conditions initially satisfied.

**Goal configuration.** Subset  $c^* \subseteq C$  of conditions we seek to satisfy.

**Operators.** Set  $O = \{O_1, \dots, O_k\}$ .

- To invoke operator  $O_i$ , must satisfy certain prereq conditions.
- After invoking  $O_i$  certain conditions become true, and certain conditions become false.

**PLANNING.** Is it possible to apply sequence of operators to get from initial configuration to goal configuration?

## Examples.

- 15-puzzle.
- Rubik's cube.
- Logistical operations to move people, equipment, and materials.

## Planning problem: 8-puzzle

**Planning example.** Can we solve the 8-puzzle?

**Conditions.**  $C_{ij}, 1 \leq i, j \leq 9$ .  $\leftarrow C_{ij}$  means tile  $i$  is in square  $j$

**Initial state.**  $c_0 = \{C_{11}, C_{22}, \dots, C_{66}, C_{78}, C_{87}, C_{99}\}$ .

**Goal state.**  $c^* = \{C_{11}, C_{22}, \dots, C_{66}, C_{77}, C_{88}, C_{99}\}$ .

## Operators.

- Precondition to apply  $O_i = \{C_{11}, C_{22}, \dots, C_{66}, C_{78}, C_{87}, C_{99}\}$ .
- After invoking  $O_i$ , conditions  $C_{79}$  and  $C_{97}$  become *true*.
- After invoking  $O_i$ , conditions  $C_{78}$  and  $C_{99}$  become *false*.

**Solution.** No solution to 8-puzzle or 15-puzzle!

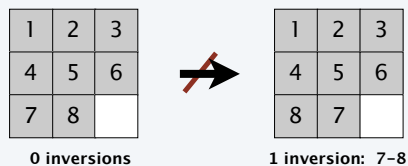
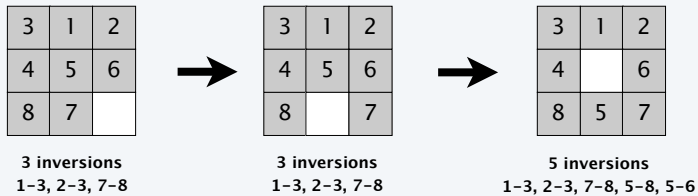
1	2	3
4	5	6
8	7	9



1	2	3
4	5	6
8	9	7

## Diversion: Why is 8-puzzle unsolvable?

**8-puzzle invariant.** Any legal move preserves the parity of the number of pairs of pieces in reverse order (inversions).



13

## Planning problem: binary counter

**Planning example.** Can we increment an  $n$ -bit counter from the all-zeroes state to the all-ones state?

**Conditions.**  $C_1, \dots, C_n$ .  $\leftarrow C_i$  corresponds to bit  $i = 1$

**Initial state.**  $c_0 = \phi$ .  $\leftarrow$  all 0s

**Goal state.**  $c^* = \{C_1, \dots, C_n\}$ .  $\leftarrow$  all 1s

**Operators.**  $O_1, \dots, O_n$ .

- To invoke operator  $O_i$ , must satisfy  $C_1, \dots, C_{i-1}$ .  $\leftarrow i-1$  least significant bits are 1
- After invoking  $O_i$ , condition  $C_i$  becomes true.  $\leftarrow$  set bit  $i$  to 1
- After invoking  $O_i$ , conditions  $C_1, \dots, C_{i-1}$  become false.  $\leftarrow$  set  $i-1$  least significant bits to 0

**Solution.**  $\{\} \Rightarrow \{C_1\} \Rightarrow \{C_2\} \Rightarrow \{C_1, C_2\} \Rightarrow \{C_3\} \Rightarrow \{C_3, C_1\} \Rightarrow \dots$

**Observation.** Any solution requires at least  $2^n - 1$  steps.

14

## Planning problem is in EXPTIME

**Configuration graph  $G$ .**

- Include node for each of  $2^n$  possible configurations.
- Include an edge from configuration  $c'$  to configuration  $c''$  if one of the operators can convert from  $c'$  to  $c''$ .

**PLANNING.** Is there a path from  $c_0$  to  $c^*$  in configuration graph?

**Claim.** PLANNING  $\in$  EXPTIME.

**Pf.** Run BFS to find path from  $c_0$  to  $c^*$  in configuration graph. ■

**Note.** Configuration graph can have  $2^n$  nodes, and shortest path can be of length  $= 2^n - 1$ .

↑  
binary counter

15

## Planning problem is in PSPACE

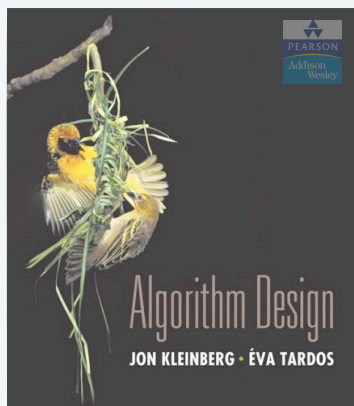
**Theorem.** PLANNING is in PSPACE.

**Pf.**

- Suppose there is a path from  $c_1$  to  $c_2$  of length  $L$ .
- Path from  $c_1$  to midpoint and from  $c_2$  to midpoint are each  $\leq L/2$ .
- Enumerate all possible midpoints.
- Apply recursively. Depth of recursion  $= \log_2 L$ . ■

```
boolean hasPath( $c_1, c_2, L$ ) {
    if ( $L \leq 1$ ) return correct answer
    // enumerate using binary counter
    foreach configuration  $c'$  {
        boolean  $x = \text{hasPath}(c_1, c', L/2)$ 
        boolean  $y = \text{hasPath}(c_2, c', L/2)$ 
        if ( $x$  and  $y$ ) return true
    }
    return false
}
```

16



## SECTION 9.5

## 9. PSPACE

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

## PSPACE-complete

**PSPACE.** Decision problems solvable in polynomial space.

**PSPACE-Complete.** Problem  $Y \in \text{PSPACE-complete}$  if (i)  $Y \in \text{PSPACE}$  and (ii) for every problem  $X \in \text{PSPACE}$ ,  $X \leq_p Y$ .

**Theorem.** [Stockmeyer-Meyer 1973]  $\text{QSAT} \in \text{PSPACE-complete}$ .

**Theorem.**  $\text{PSPACE} \subseteq \text{EXPTIME}$ .

**Pf.** Previous algorithm solves QSAT in exponential time; and QSAT is *PSPACE-complete*. ■

**Summary.**  $P \subseteq NP \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$ .

$\uparrow \quad \uparrow \quad \uparrow$   
 it is known that  $P \neq \text{EXPTIME}$ ,  
 but unknown which inclusion is strict;  
 conjectured that all are

18

## PSPACE-complete problems

**More PSPACE-complete problems.**

- **Competitive facility location.**
  - Natural generalizations of games.
    - Othello, Hex, Geography, Rush-Hour, Instant Insanity
    - Shanghai, go-moku, Sokoban
- Given a memory restricted Turing machine, does it terminate in at most  $k$  steps?
- Do two regular expressions describe different languages?
- Is it possible to move and rotate complicated object with attachments through an irregularly shaped corridor?
- Is a deadlock state possible within a system of communicating processors?

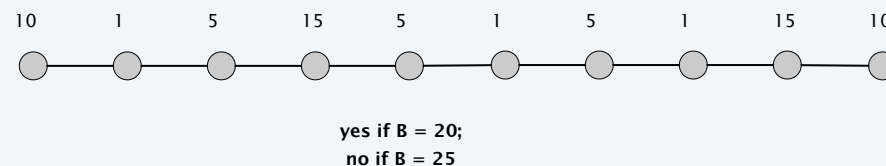
19

## Competitive facility location

**Input.** Graph  $G = (V, E)$  with positive edge weights, and target  $B$ .

**Game.** Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors has been selected.

**Competitive facility location.** Can second player guarantee at least  $B$  units of profit?



20

## Competitive facility location

**Claim.** COMPETITIVE-FACILITY-LOCATION  $\in PSPACE$ -complete.

**Pf.**

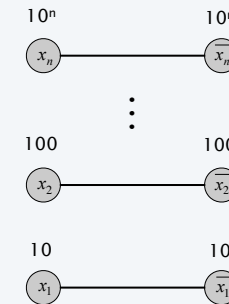
- To solve in poly-space, use recursion like Q-SAT, but at each step there are up to  $n$  choices instead of 2.
- To show that it's complete, we show that Q-SAT polynomial reduces to it. Given an instance of Q-SAT, we construct an instance of COMPETITIVE-FACILITY-LOCATION so that player 2 can force a win iff Q-SAT formula is *true*.

21

## Competitive facility location

**Construction.** Given instance  $\Phi(x_1, \dots, x_n) = C_1 \wedge C_1 \wedge \dots C_k$  of Q-SAT. ← assume  $n$  is odd

- Include a node for each literal and its negation and connect them. (at most one of  $x_i$  and its negation can be chosen)
- Choose  $c \geq k + 2$ , and put weight  $c^i$  on literal  $x^i$  and its negation; set  $B = c^{n-1} + c^{n-3} + \dots + c^4 + c^2 + 1$ . (ensures variables are selected in order  $x_n, x_{n-1}, \dots, x_1$ )
- As is, player 2 will lose by 1 unit:  $c^{n-1} + c^{n-3} + \dots + c^4 + c^2$ .

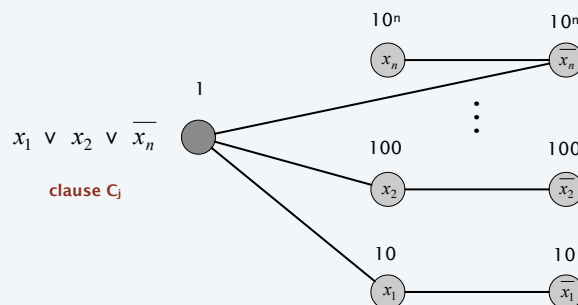


22

## Competitive facility location

**Construction.** Given instance  $\Phi(x_1, \dots, x_n) = C_1 \wedge C_1 \wedge \dots C_k$  of Q-SAT.

- Give player 2 one last move on which she can try to win.
- For each clause  $C_j$ , add node with value 1 and an edge to each of its literals.
- Player 2 can make last move iff truth assignment defined alternately by the players failed to satisfy some clause. ■



23