

9. PSPACE

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

Geography game

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

Q. Does Alice have a forced win?

Ex. Budapest \rightarrow Tokyo \rightarrow Ottawa \rightarrow Ankara \rightarrow Amsterdam \rightarrow Moscow \rightarrow
Washington \rightarrow etc.

Geography game

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

Q. Does Alice have a forced win?

Ex. Budapest \rightarrow Tokyo \rightarrow Ottawa \rightarrow Ankara \rightarrow Amsterdam \rightarrow Moscow \rightarrow
Washington \rightarrow etc.

Geography on graphs. Given a digraph $G = (V, E)$ and a start node s , two players alternate turns by following, if possible, an edge leaving 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 \mathcal{P} , EXPTIME, \mathcal{NP} , and \mathcal{NP} -complete.

PSPACE complexity class

PSPACE

\mathcal{P} . Decision problems solvable in polynomial *time*.

PSPACE. Decision problems solvable in polynomial *space*.

PSPACE

\mathcal{P} . Decision problems solvable in polynomial *time*.

PSPACE. Decision problems solvable in polynomial *space*.

Observation. $\mathcal{P} \subseteq \text{PSPACE}$.

- Both are asymptotic growth rate
- poly-time algorithm can consume only polynomial space

PSPACE: facts

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

Algorithm. Use n -bit “odometer”.

- exponential time, polynomial space
- space can be *reused*

PSPACE: facts

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

Algorithm. Use n -bit “odometer”.

- exponential time, polynomial space
- space can be *reused*

Claim. 3-SAT \in PSPACE.

Pf.

- Enumerate all 2^n possible truth assignments using single n -bit counter.
- Check each assignment to see if it satisfies all clauses, reuse space.

PSPACE: facts

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

Algorithm. Use n -bit “odometer”.

- exponential time, polynomial space
- space can be *reused*

Claim. 3-SAT \in PSPACE.

Pf.

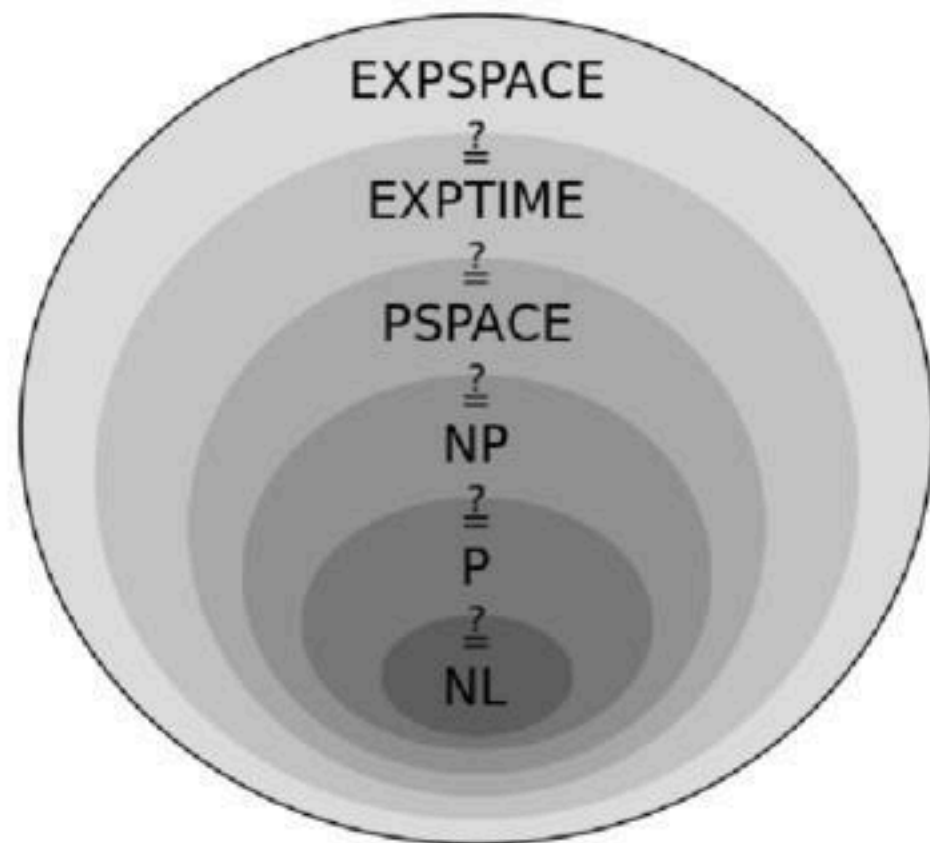
- Enumerate all 2^n possible truth assignments using single n -bit counter.
- Check each assignment to see if it satisfies all clauses, reuse space.

Theorem. $\mathcal{NP} \subseteq$ PSPACE.

Pf. Consider arbitrary problem $Y \in \mathcal{NP}$.

- Since 3-SAT is \mathcal{NP} -complete, $Y \leq_P$ 3-SAT,
- Can implement black box in poly-space.

Various classes of problems



Quantified satisfiability

Quantified satisfiability

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

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

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

Quantified satisfiability

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

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

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

Ex. $(x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$

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

Quantified satisfiability

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

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

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

Ex. $(x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$

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

Ex. $(x_1 \vee x_2) \wedge (\bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$

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

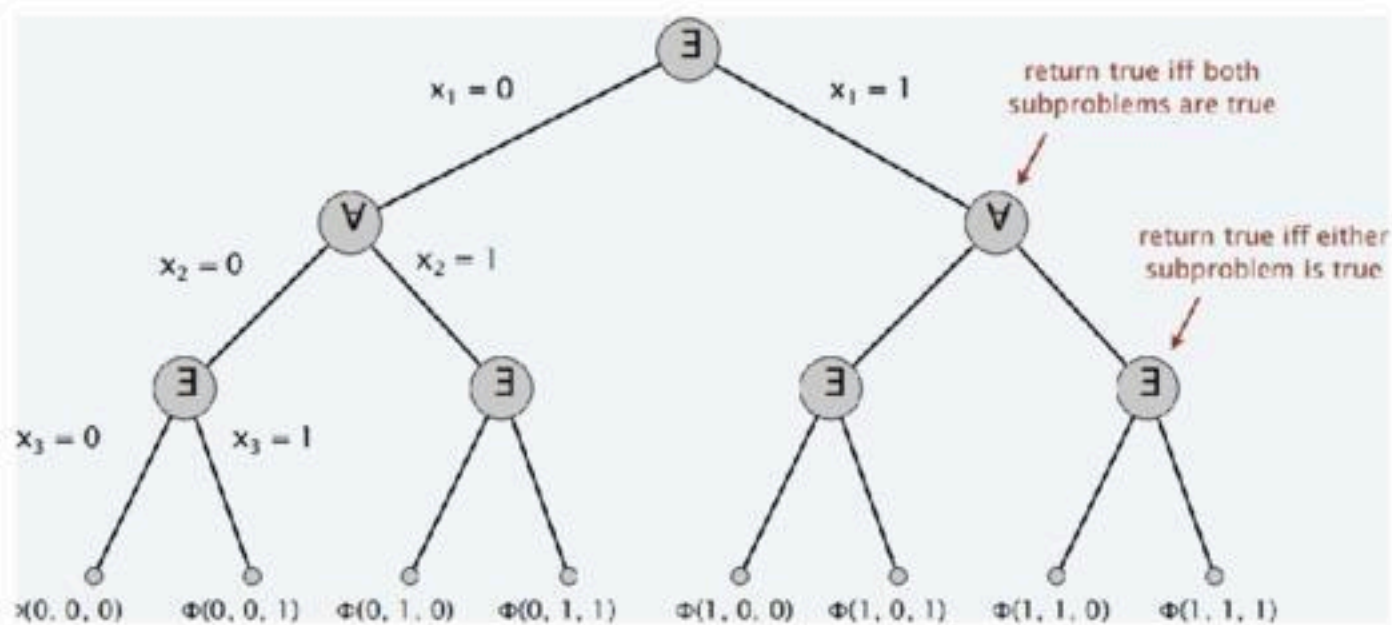
No. if Amy sets x_1 true; Bob sets x_2 true; Amy loses.

Q-SAT \in PSPACE

Theorem. Q-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.

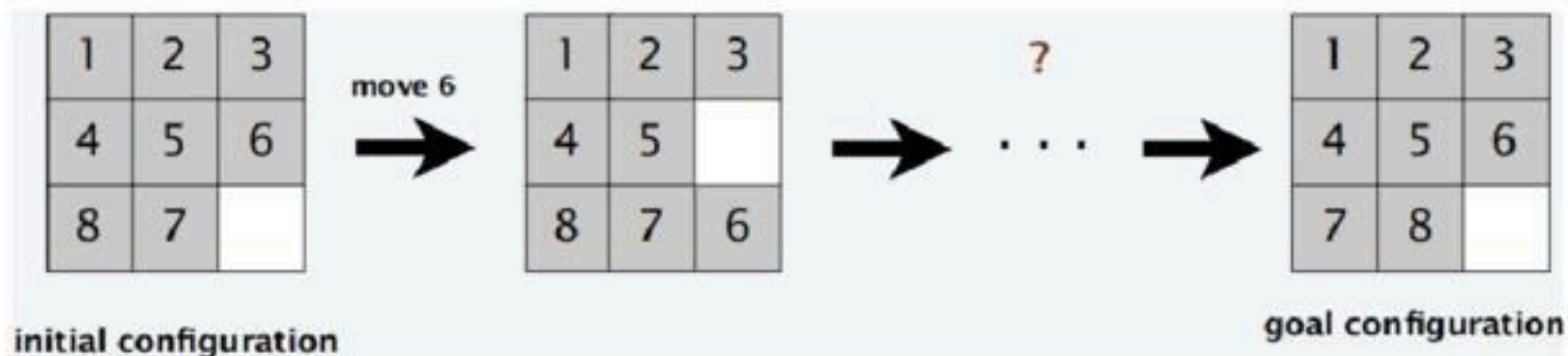


Planning problem

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 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?

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. $C_{ij}, 1 \leq i, j \leq 9$: 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.

1	2	3
4	5	6
8	7	9

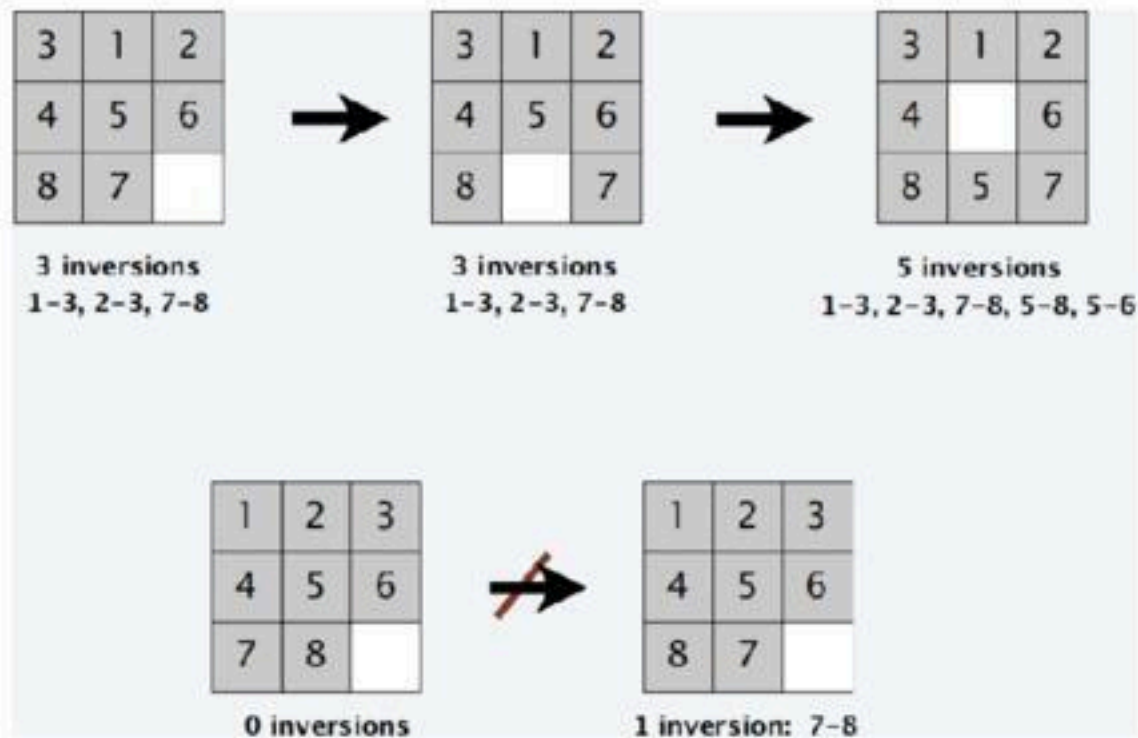


1	2	3
4	5	6
8	9	7

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

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**).



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 : C_i corresponds to bit- $i = 1$.

Initial state. $c_0 = \emptyset$: all 0s.

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

Operators. O_1, \dots, O_n .

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

PLANNING \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?

PLANNING \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).

PLANNING \in PSPACE

Theorem. PLANNING \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 = $\log_2 L$.

hasPath(c_1, c_2, L)

1. if ($L \leq 1$) return correct answer;
2. foreach configuration c'
 1. $x = \text{hasPath}(c_1, c', L/2)$;
 2. $y = \text{hasPath}(c_2, c', L/2)$;
 3. if (x and y) return true;
3. return false;

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 $X \in \text{PSPACE}$, $X \leq_P Y$.

- “hardest” PSPACE-complete problem

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 $X \in \text{PSPACE}$, $X \leq_P Y$.

- “hardest” PSPACE-complete problem

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.

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 $X \in \text{PSPACE}$, $X \leq_P Y$.

- “hardest” PSPACE-complete problem

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. $\mathcal{P} \subseteq \mathcal{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$.

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

PSPACE-complete problems

More PSPACE-complete problems.

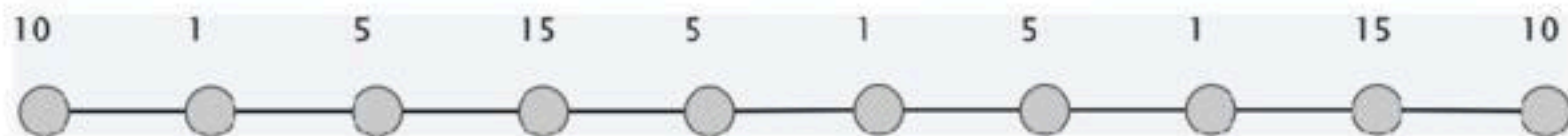
- **Competitive facility location.**
- Natural generalizations of games.
- Given a memory restricted Turing machine, does it terminate in at most k steps?
- Do two regular expressions describe different languages?
- Is it possible to move and rotate complicated object with attachments through an irregularly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

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?



- yes if $B = 20$; no if $B = 25$

COMPETITIVE-LOC \in PSPACE-complete

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

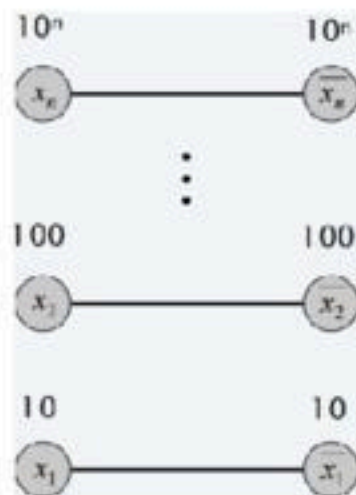
Pf.

- 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.

COMPETITIVE-LOC \in PSPACE-complete

Construction. Given instance $\Phi(x_1, \dots, x_n) = C_1 \wedge C_2 \wedge \dots \wedge C_k$ of Q-SAT.

- 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: $c^{n-1} + c^{n-3} + \dots + c^4 + c^2$.



COMPETITIVE-LOC \in PSPACE-complete

Construction. Given instance $\Phi(x_1, \dots, x_n) = C_1 \wedge C_2 \wedge \dots \wedge C_k$ 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.

