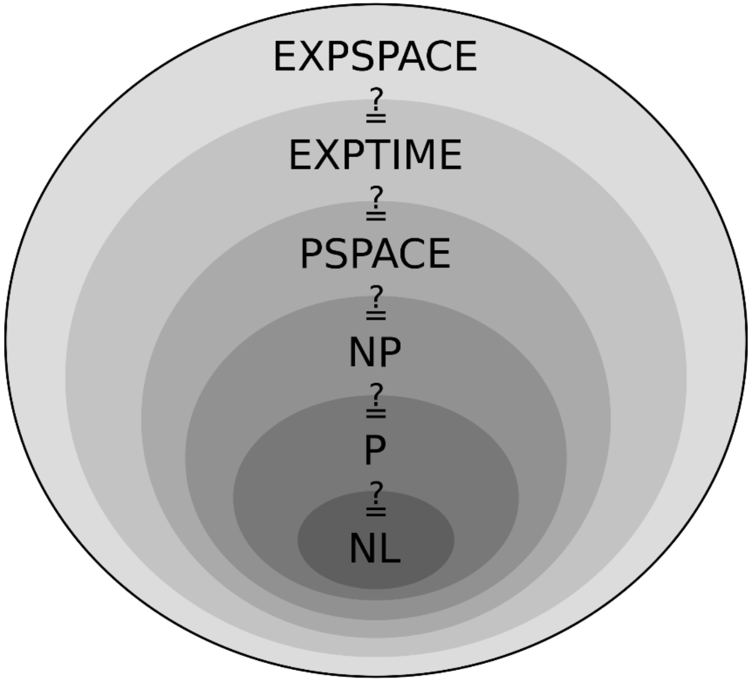
Lecture 10 (Adv)

PSPACE: A Class of Problems Beyond NP



THE UNIVERSITY OF SYDNEY

Geography Game

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

Example: 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?

9.1 PSPACE

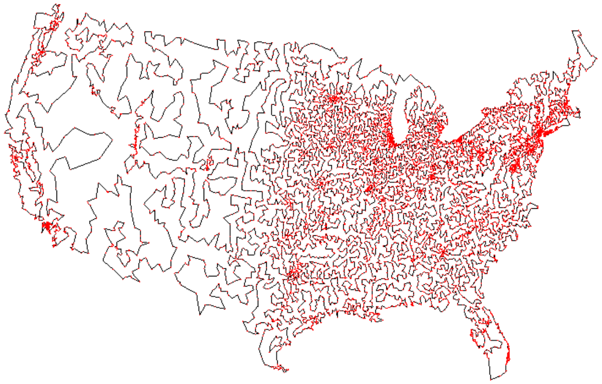
PSPACE

P: Decision problems solvable in polynomial time.

PSPACE: Decision problems solvable in polynomial space.

Example: Euclidean TSP $\in$ PSPACE

(largest solved instance: 85,900)



Observation: P $\subseteq$ PSPACE.

Since poly-time algorithm can consume only polynomial space

PSPACE

Note that there are algorithms that might need exponential time but only polynomial space:

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

Space: $O(\log_2 2^n) \Rightarrow O(n)$ space)

Theorem: 3-SAT is in PSPACE.

Proof:

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

Important corollary: NP $\subseteq$ PSPACE.

Proof: Consider arbitrary problem Y in NP.

- Since $Y \leq_P$ 3-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. ▪

Two types of problems

- Quantified SAT problems
- Planning problems

9.3 Quantified Satisfiability

Quantified Satisfiability (QSAT)

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

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

assume n is odd

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

SAT: $\exists x_1\ \exists x_2\ \exists x_3\ \exists x_4 \ldots\ \exists x_{n-1}\ \exists x_n\ \Phi(x_1, \ldots, x_n)$

Quantified Satisfiability (QSAT)

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

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

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?

Example:

$$(x_1 \lor x_2) \land (x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3)$$

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

$$(x_1 \lor x_2) \land (\overline{x_2} \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3)$$

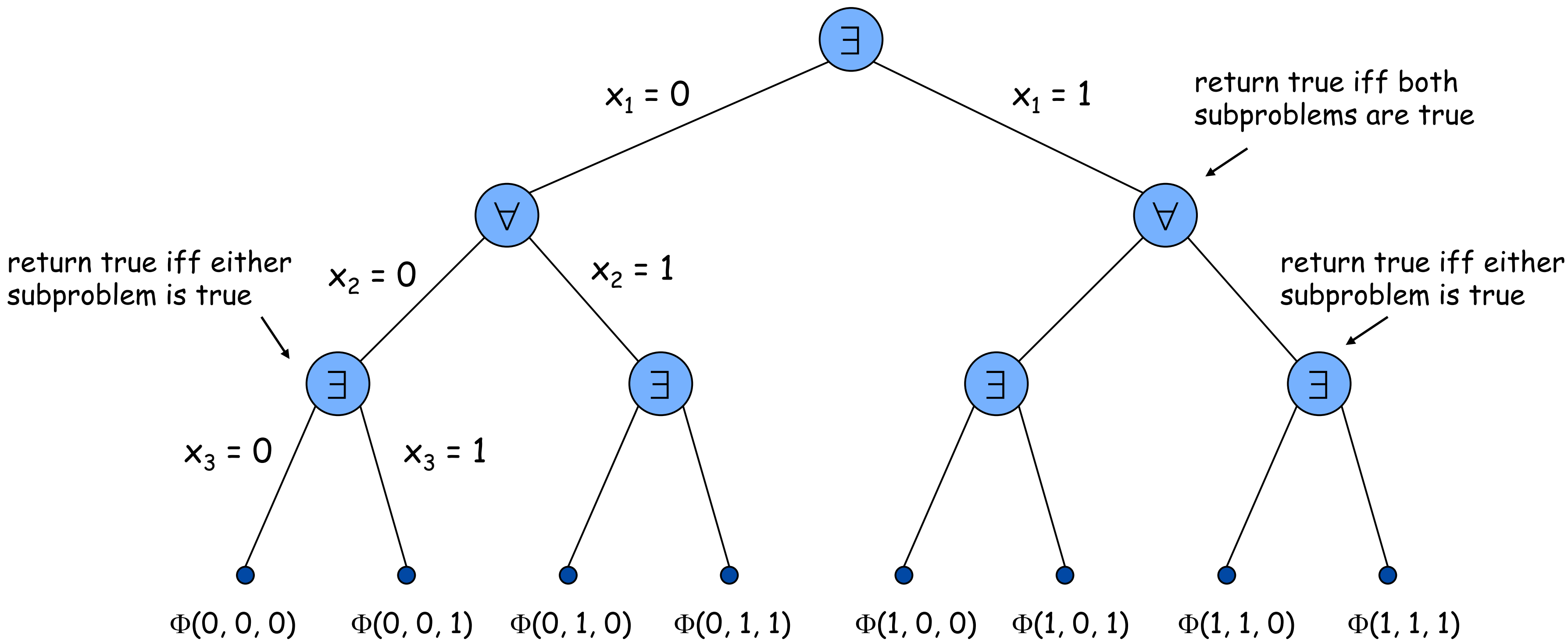
– No. If Amy sets x_1 false; Bob sets x_2 false; Amy loses;
if Amy sets x_1 true; Bob sets x_2 true; Amy loses.

QSAT is in PSPACE

Theorem: QSAT $\in$ PSPACE.

Proof: Recursively try all possibilities.

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

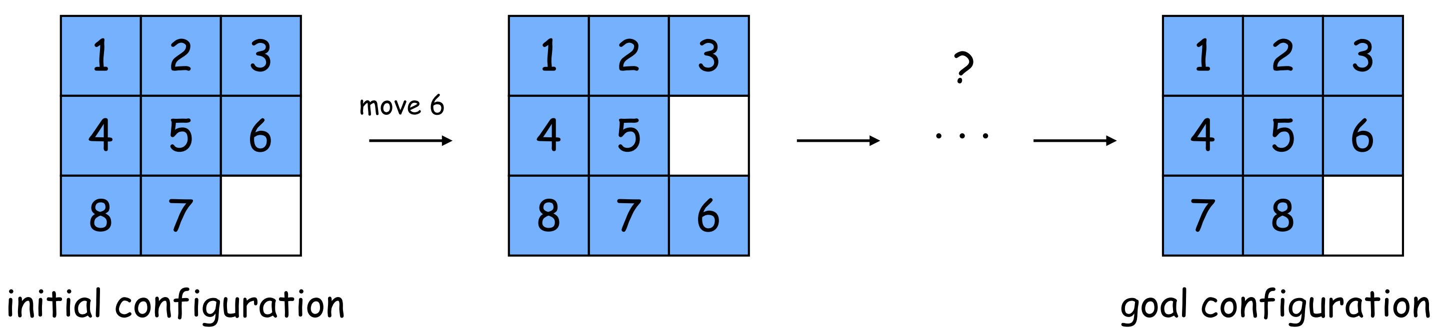


9.4 Planning Problem

15-Puzzle

8-puzzle, 15-puzzle. [Sam Loyd 1870s]

- 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 $C = \{ C_1, \ldots, C_n \}$.

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

- To invoke operator O_i, must satisfy certain prerequisite 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?

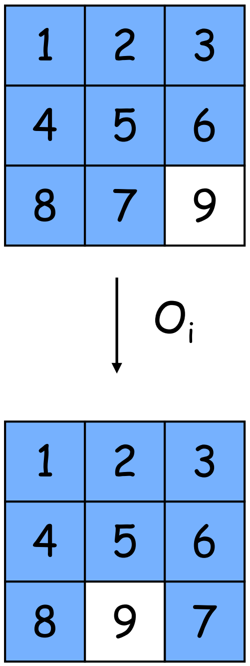
Conditions: C_{ij}, $1 \leq i, j \leq 9$. ⟵ C_{ij} means tile i is in square j

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

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

Operators.

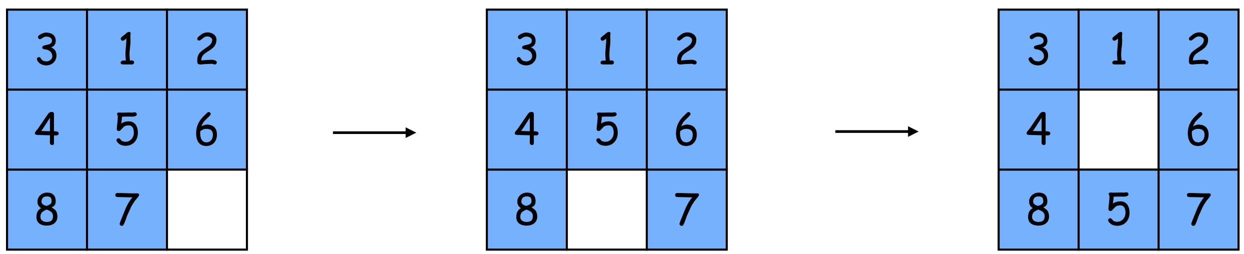
- Precondition to apply $O_i = \{C_{11}, C_{22}, \ldots, C_{66}, C_{78}, C_{87}, C_{99}\}$.
- After invoking O_i, conditions C_{79} and C_{97} become true.
- After invoking O_i, conditions C_{78} and C_{99} become false.



Diversion: Why is 8-Puzzle Unsolvable?

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

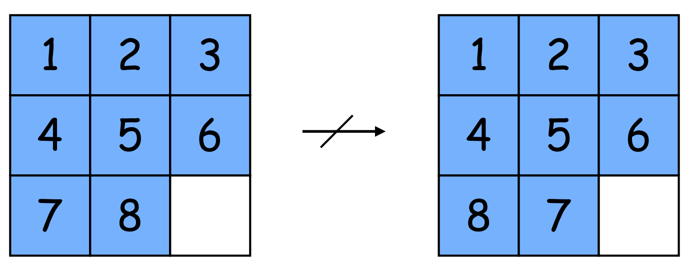
8-puzzle invariant. Any legal move preserves the parity of the number of pairs of pieces in reverse order (inversions).



3 inversions
1-3, 2-3, 7-8

3 inversions
1-3, 2-3, 7-8

5 inversions
1-3, 2-3, 7-8, 5-8, 5-6



0 inversions

1 inversion: 7-8

Planning Problem: Binary Counter

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

Conditions: $C_1, \ldots, C_n$ ← C_i corresponds to bit i = 1

Initial state: $c_0 = \varnothing$ ← all 0s

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

Operators: $O_1, \ldots, O_n$

- To invoke operator O_i, must satisfy $C_1, \ldots, C_{i-1}$ ← 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 $C_1, \ldots, C_{i-1}$ become false ← set i-1 least significant bits to 0

Solution: $\{C_1\} \Rightarrow \{C_2\} \Rightarrow \{C_1, C_2\} \Rightarrow \{C_3\} \Rightarrow \{C_3, C_1\} \Rightarrow \ldots$

Observation: There exists an instance for which the shortest solution has length $2^n - 1$ steps.

Planning Problem: In Exponential Space

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?

Theorem: PLANNING is in EXPTIME.

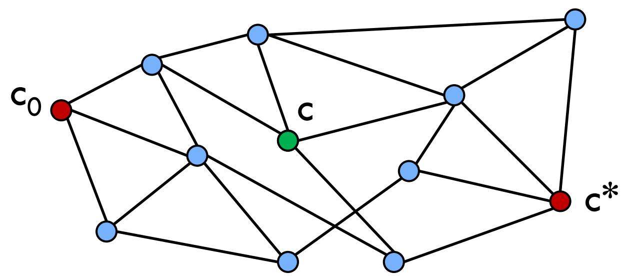
Proof: Run BFS to find path from c_0 to c^* in configuration graph. ▪

Corollary: Configuration graph can have 2^n nodes, and shortest path can be of length at most $2^n - 1$.

Space-efficient algorithm?

What do we know?

– Configuration graph has exponential size.
– There exists a path that "only" uses $2^n - 1$ edges.

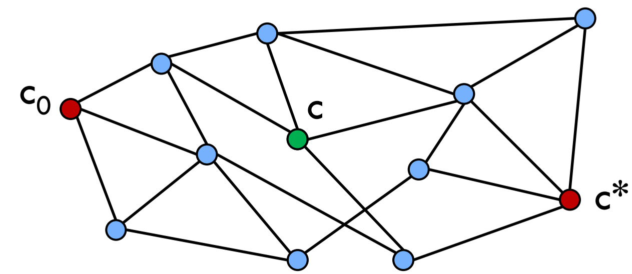


hasPath(c_0,c^*,L) –
Is there a path from c_0 to c^* using $\leq L=2^n$ steps?

Guess a mid-point c on the shortest path.
hasPath(c_0,c^*,L) =
hasPath(c_0,c,L/2) + hasPath(c,c^*,L/2)

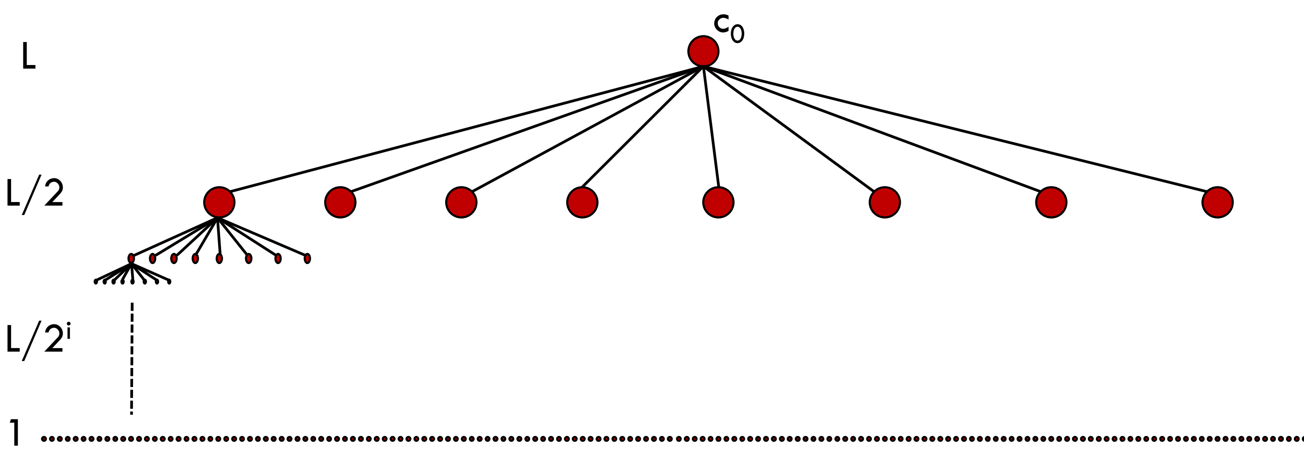
How can we find such a path using only polynomial space?

Space-efficient algorithm?



Guess a mid-point c on the shortest path.

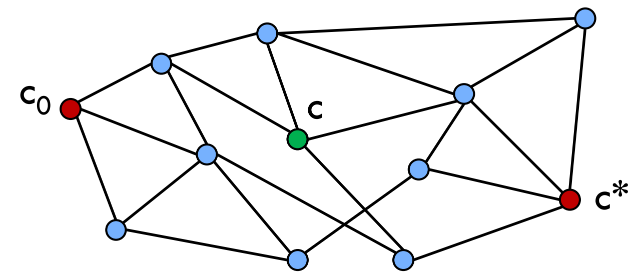
$\text{hasPath}(c_0,c^*,L) = \text{hasPath}(c_0,c,L/2) + \text{hasPath}(c,c^*,L/2)$



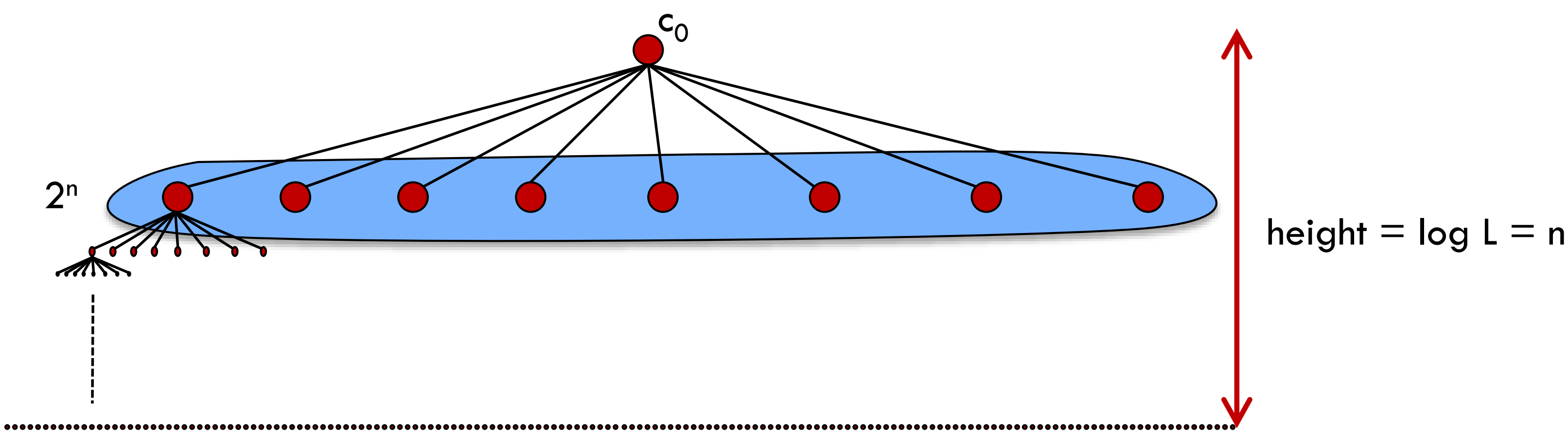
All possible mid-points c between c_0 and c^*

Continue recursively"

Space-efficient algorithm?



Guess a mid-point c on the shortest path.
hasPath(c_0,c^*,L) = hasPath(c_0,c,L/2) + hasPath(c,c^*,L/2)



Total time: $O^*(2^n)^n = O^*(2^{n^2})$

Total space: ~ the height of the tree + polynomial overhead = poly(n)
[by traversing the tree as in QSAT]

Planning Problem: In Polynomial Space

Theorem: PLANNING is in PSPACE.

Proof:

- 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(c1, c2, L) {
   if (L ≤ 1) return correct answer

                                    enumerate using binary counter
   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
}
```

9.5 PSPACE-Complete

PSPACE-Complete

PSPACE: Decision problems solvable in polynomial space.

PSPACE-complete: Problem Y is PSPACE-complete if

(i) Y is in PSPACE and

(ii) for every problem X in PSPACE, $X \leq_P Y$.

Theorem: QSAT is PSPACE-complete. (without proof)

Theorem: PSPACE $\subseteq$ EXPTIME.

Proof: Previous algorithm solves QSAT in exponential time, and QSAT is PSPACE-complete. ▪

Summary. $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$.

it is known that $P \neq EXPTIME$, but unknown which inclusion is strict; conjectured that all are

PSPACE-Complete Problems

More PSPACE-complete problems.

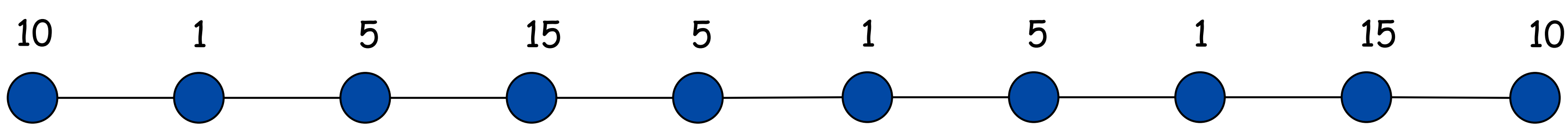
- Competitive facility location.
- Natural generalizations of games.
  - Othello, Hex, Geography, Rush-Hour, Instant Insanity
  - Shanghai, go-moku, Sokoban
- Is it possible to move and rotate complicated object with attachments through an irregularly shaped corridor?
- Is a deadlock state possible within a system of communicating processors?

Competitive Facility Location

Input: Graph 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 Facility Location

Theorem: COMPETITIVE-FACILITY is PSPACE-complete.

Proof:

- To solve in poly-space, use recursion like QSAT, but at each step there are up to n choices instead of 2 (same as PLANNING).
- To show that it's complete, we show that QSAT polynomial reduces to it. Given an instance of QSAT, we construct an instance of COMPETITIVE-FACILITY such that player 2 can force a win iff QSAT formula is true.

Competitive Facility Location

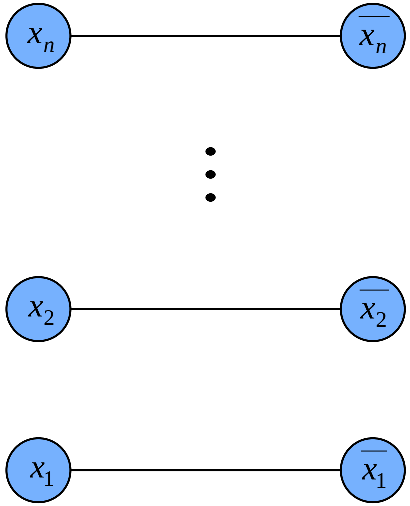
assume n is odd

Construction: Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ of QSAT.

- Include a node for each literal and its negation and connect them.
  - at most one of x_i and its negation can be chosen

A player can pick x_i or $\overline{x}_i$, but not both can be picked.

How can we force the order $x_1, x_2, \ldots, x_n$?

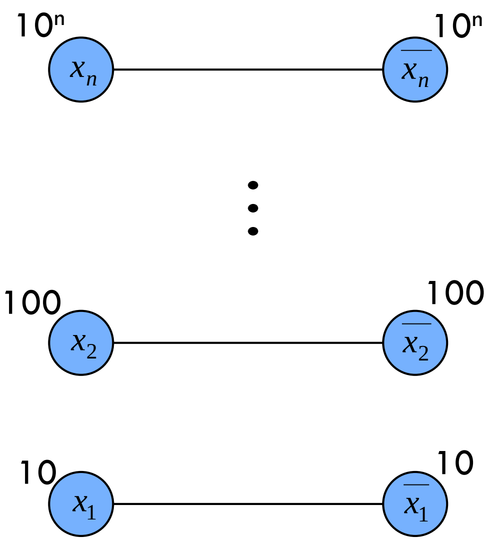


Competitive Facility Location

assume n is odd

Construction: Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ of QSAT.

- 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 $B = c^{n-1} + c^{n-3} + \ldots + c^4 + c^2 + 1$.
  - ensures variables are selected in order $x_n, x_{n-1}, \ldots, x_1$.

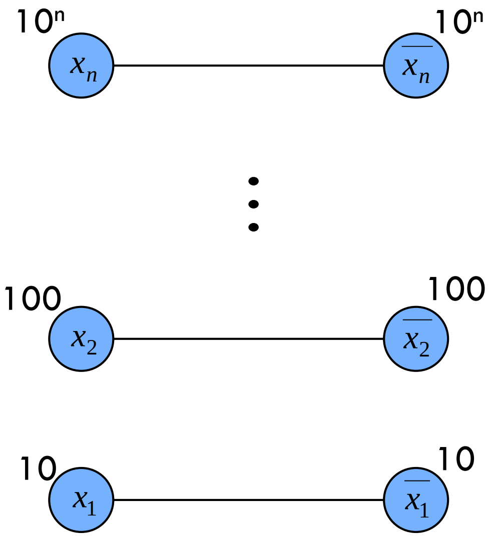


Competitive Facility Location

assume n is odd

Construction: Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ of QSAT.

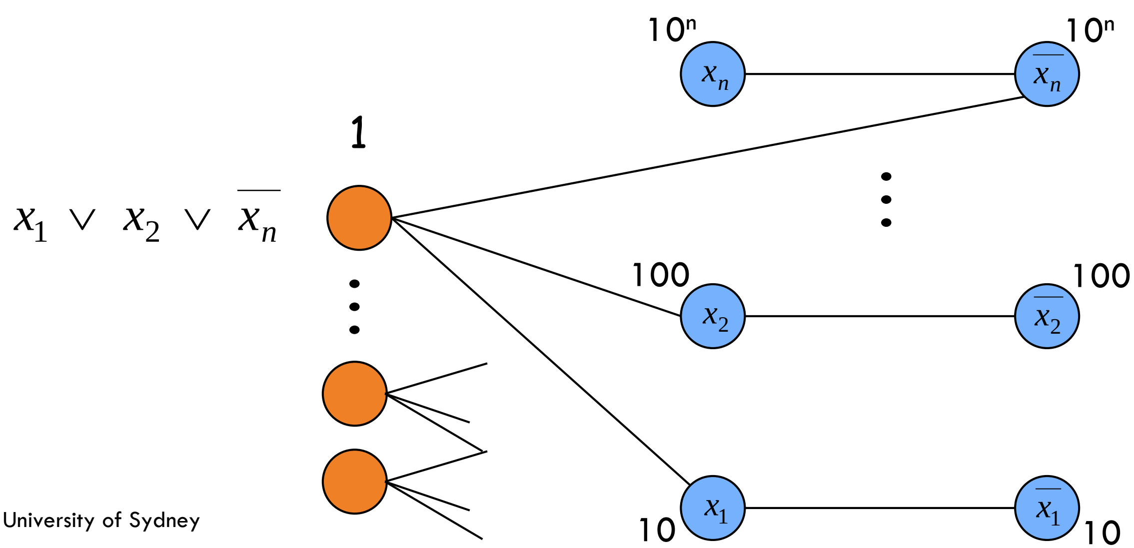
- Include a node for each literal and its negation and connect them.
  - at most one of x_i and its negation can be chosen
- Choose $c \geq k+2$, and put weight c^i on literal x_i and its negation; set $B = c^{n-1} + c^{n-3} + \ldots + c^4 + c^2 + 1$.
  - ensures variables are selected in order $x_n,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



Competitive Facility Location

Construction: Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ of QSAT.

- 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. ▪



Summary

PSPACE: Decision problems solvable in polynomial space.

$$P \subseteq NP \subseteq PSPACE$$

PSPACE-complete: Problem Y is PSPACE-complete if

(i) Y is in PSPACE and

(ii) for every problem X in PSPACE, $X \leq_P Y$.

Theorem: QSAT is PSPACE-complete.

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$