CSCI 550: Advanced Algorithms
1. Space Complexity

- PSPACE
- quantified satisfiability
- planning problem
- PSPACE-completeness

Geography game

Geography.
- Alice names the capital city c of the country she's 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?

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 leaving the current node to an unvisited node. Is the first player guaranteed to make the last legal move?

Remark. Some problems (especially involving 2-player games and AI) defy classification according to **NP**, **EXP**, or **NP**-complete.

CSCI 550: Advanced Algorithms
1. Space Complexity

- PSPACE
- quantified satisfiability
- planning problem
- PSPACE-completeness

PSPACE

P. Decision problems solvable in polynomial time.

PSPACE. Decision problems solvable in polynomial space.

Observation. **P** ⊆ **PSPACE**.

poly-time algorithms
can consume
only polynomial space

PSPACE

Binary counter. Count from 0 to $2^n - 1$ in binary.
Algorithm. Use an n bit "odometer".

Claim. 3-SAT ∈ **PSPACE**.
Pf.

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

Theorem. **NP** ⊆ **PSPACE**.
Pf. Consider an arbitrary problem $Y \in$ **NP**.

- Since $Y \leq_P$ 3-SAT, there exists an algorithm that solves Y in poly-time, plus a polynomial number of calls to the 3-SAT "black box".
- We can implement the "black box" in poly-space. ▪

CSCI 550: Advanced Algorithms
1. Space Complexity

- PSPACE
- *quantified satisfiability*
- planning problem
- PSPACE-completeness

Quantified satisfiability

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

Intuition. Amy picks a 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?

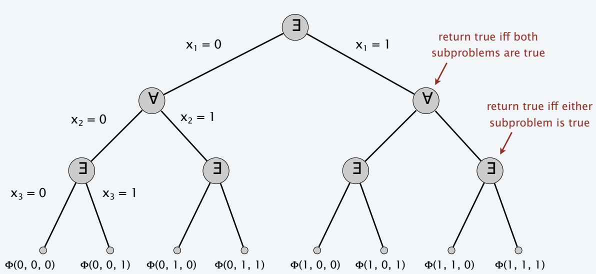
Ex. $(x_1 \lor x_2) \land (x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3)$
Yes. Amy sets x_1 true; Bob sets x_2; Amy sets x_3 to be the same as x_2.

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

Quantified satisfiability

Theorem. Q-SAT ∈ **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.

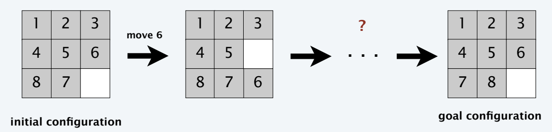


CSCI 550: Advanced Algorithms
1. Space Complexity

n-puzzles

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

- Board: 3-by-3 grid of tiles labeled 1–8.
- Legal move: slide neighbouring tile into blank (white) square.
- Target: find a sequence of legal moves to transform an initial configuration into the 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 some sequence of operators to get from an initial configuration to the goal configuration?

Examples.

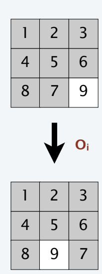
- 8-puzzle, 15-puzzle.
- Rubik's cube.
- Logistical operations to move people, equipment, and materials.

Planning problem: 8-puzzle

Planning example. Can we solve the 8-puzzle?

Conditions. C_{ij}, $1 \le i, j \le 9$. ← 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*.



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. $c_0 = \phi$. ← 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. $\{\ \} \Rightarrow \{C_1\} \Rightarrow \{C_2\} \Rightarrow \{C_1, C_2\} \Rightarrow \{C_3\} \Rightarrow \{C_3, C_1\} \Rightarrow \ldots$

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

Planning problem

Configuration graph G.

- Include a node for each of the 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 the configuration graph?

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

Pf. Run BFS to find path from c_0 to c^* in the configuration graph. ▪

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

↑ binary counter

Planning problem

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 (c1, c2, L)
   IF (L ≤ 1) RETURN true

                     enumerate using binary counter
   FOREACH configuration c′ {
      x = HASPATH (c1, c′, L/2)
      y = HASPATH (c2, c′, L/2)
      IF (x and y) RETURN true
   RETURN false
```

CSCI 550: Advanced Algorithms
1. Space Complexity

- PSPACE
- quantified satisfiability
- planning problem
- **PSPACE-completeness**

PSPACE-complete

PSPACE. Decision problems solvable in polynomial space.

PSPACE-complete. Problem $Y \in$ **PSPACE**-complete if (i) $Y \in$ **PSPACE** and (ii) for every problem $X \in$ **PSPACE**, $X \leq_P Y$.

Theorem. [Stockmeyer–Meyer, 1973] QSAT $\in$ **PSPACE**-complete.

Theorem. **PSPACE** $\subseteq$ **EXPTIME**.
Pf. 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 ≠ EXPTIME,
but unknown which inclusion is strict;
conjectured that all are

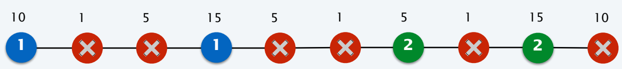
PSPACE-complete problems

More PSPACE-complete problems.
- Competitive facility location.
- Natural generalizations of games.
  - Geography, Hex, Othello, Rush-Hour, Instant Insanity
  - Gomoku, Sokoban, Shanghai
- 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 a 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. A player may not select a node if any of its neighbours has previously been selected.

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



yes if B = 20;
no if B = 25

Competitive facility location

Claim. COMPETITIVE-FACILITY-LOCATION ∈ **PSPACE**-complete.

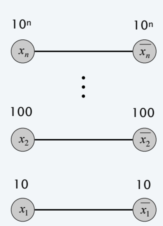
Pf.

- To solve in poly-space, use recursion like Q-SAT, but at each step allow for up to n choices instead of 2.
- To show that it's **PSPACE**-complete, we show that Q-SAT polynomial-time 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 the corresponding Q-SAT formula is *true*.

Competitive facility location

Construction. Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_2 \wedge \ldots\ C_k$ 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 $B = c^{n-1} + c^{n-3} + \ldots + c^4 + c^2 + 1$.
  (Ensures variables are selected in order $x_n, x_{n-1}, \ldots, x_1$)
- As is, player 2 will lose by 1 unit: $c^{n-1} + c^{n-3} + \ldots + c^4 + c^2$.



Competitive facility location

Construction. Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_2 \wedge \ldots C_k$ of Q-SAT: ← assume n is odd

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