P and NP Complexity Classes

Algorithms

Motivation

- Design of an algorithm is an important task of a Computer Scientist
  - Need to show that an algorithm is correct
  - Need to show how an algorithm behaves
  - **Need to know whether an efficient algorithm <u>can</u> exist**

Motivation

- Complexity Theory
  - Presents classes of algorithms and suspects differences between them
  - Needed to formulate modern cryptography

Class P

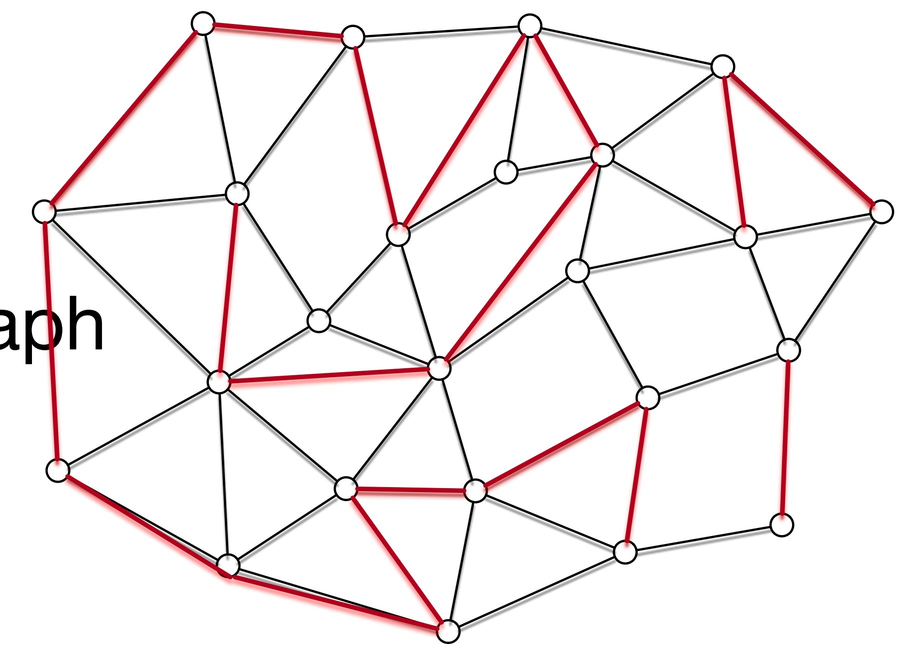
- Most algorithms considered in class are polynomial time bound

$$\exists k \in \mathbb{N} \qquad \textbf{runtime(n)} \in O(n^k)$$

- In reality, algorithms with runtimes in $\Omega(n^k), k > 2$ are useless in many circumstances
- Class P: set of problems for which there exists a polynomially bound

Class NP

- Algorithms in P can be "efficiently" calculated
- Sometimes, problems are hard to solve but easy to verify
- Example: Finding a path of length *n* in a graph
- NP: Class of problems for which a solution can be solved in polynomial time
- Alternative Formulation: Can be solved by a non-deterministic algorithm that is polynomially bound
    - The algorithm "guesses" a solution and then verifies it



A Conjecture and a Theorem

- Conjecture: $\mathscr{P} \neq \mathscr{N}\mathscr{P}$
- Theorem: There exists problems $p \in \mathscr{N}\mathscr{P}$ such that
    - $p \in \mathscr{P} \implies \mathscr{P} = \mathscr{N}\mathscr{P}$
  - These problems are called <u>NP-complete</u>
  - The existence of NP-complete problems is evidence that the conjecture might be true

A Conjecture and a Theorem

- There are very similar problems that are in different classes
  - Graphs
    - $\mathscr{P}$ Eulerian tour: A cycle that visits every edge at least once (though it usually visits vertices several times):
    - $\mathscr{NPC}$ Hamiltonian cycle: A simple cycle that visits every vertex exactly once (but usually does not visit every edge)

A Conjecture and a Theorem

- There are very similar problems that are in different classes
  - Graphs:
    - $\mathscr{P}$ Shortest Path between two vertices
    - $\mathscr{NPC}$ Existence of a path of certain length

A Conjecture and a Theorem

- There are very similar problems that are in different classes
  - Satisfiability
    - Given a boolean formula in conjunctive normal form, find a variable assignment that makes the formula true.

$$(\neg a \vee b)(a \vee \neg b \vee c)(a \vee c)(\neg a \vee b \vee \neg c)$$

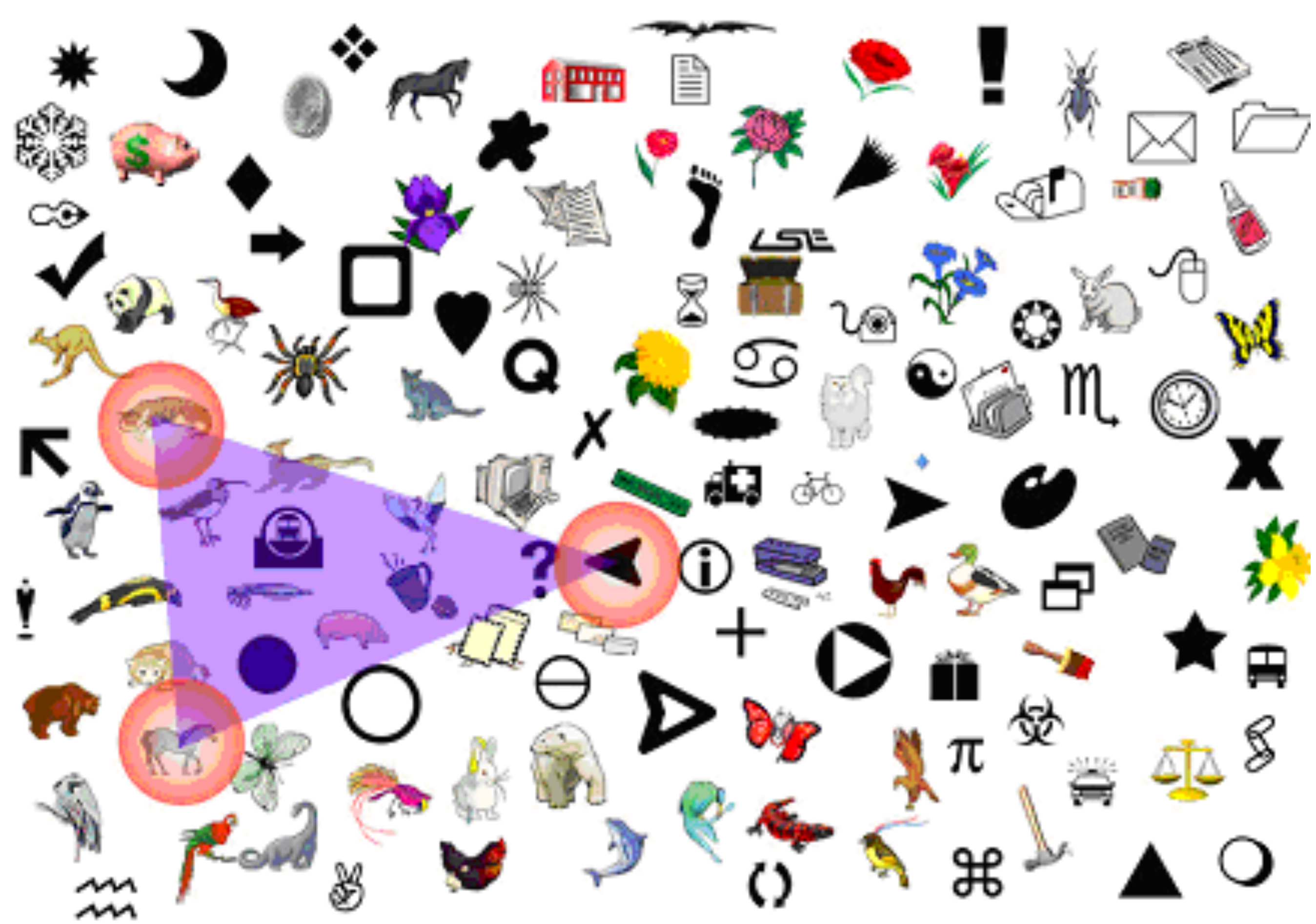
$$a = 1,\ b = 1,\ c = 1$$

A Conjecture and a Theorem

- There are very similar problems that are in different classes
  - Satisfiability
    - Many problems can be formulated in terms of satisfiability

Human-Computer Authentication

- Click in the triangle defined by the secret icons



Human-Computer Authentication

- Safety question
  - How many interactions do need to be observed in order to find the secret?
  - One problem: people usually click on the center of the triangle
  - Greater problem: Evaluation can be formulated as a satisfiability problem
    - Good (non-P) algorithms exists to solve them usually fast
    - Can be shown that few interactions need to be observed in order to deduce the secret

A Conjecture and a Theorem

- There are very similar problems that are in different classes
  - Satisfiability

$\mathscr{P}$
  - 2-CNF Satisfiability
    - Decide satisfiability if the or-clauses can have one or two variables
  - 3-CNF Satisfiability

$\mathscr{NPC}$
    - Decide satisfiability if the or-clause can have three (or less) variables

Existence vs. Solution

- Decision problem: Answer is yes or no
- Optimization problem: Find a feasible solution

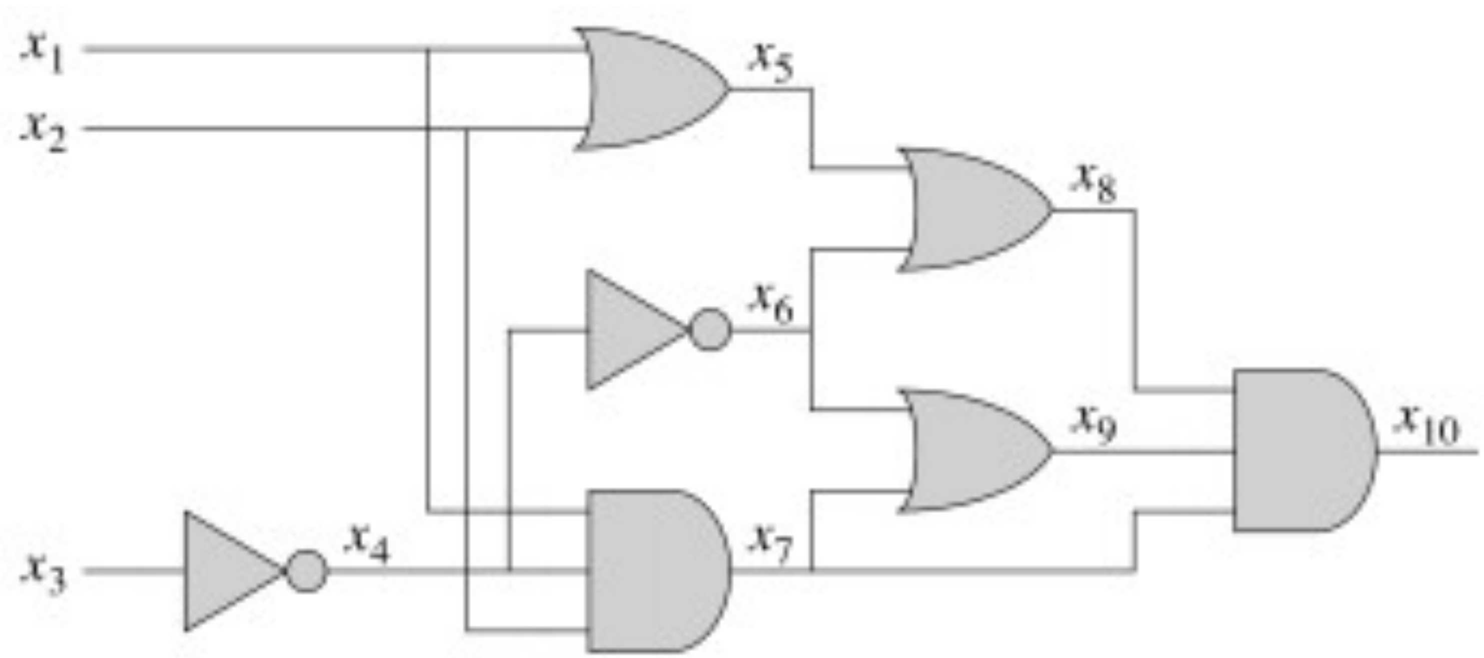
- Can change optimization problems into decision problem
    - Example: Finding longest path in a graph
    - Decision Problem: is there a path of length l
    - Solve the decision problem repeatedly in order to find the maximum length of a path
    - Once found the maximum length, remove edges one by one to see whether the maximum length has changed
    - So, solving the decision problem allows you to find a solution

Encodings

- We look at the run-time in dependence on the size of the problem
- But the size of the problem can change if we use a different encoding
- Example: Knapsack
  - Rectangle is of size $n \times m$
  - Can encode numbers in unary
    - Then dynamic programming is in P
  - Can encode numbers in binary
    - Then dynamic programming is not in P
    - If *l* is the number of digits, then rectangle is of size $2^l \times 2^l$

A first problem in NPC

- A boolean circuit consists of the normal gates and has no cycles (input feeding back)
- Decision problem: Is a circuit satisfiable, i.e. is there a selection of inputs such that the circuit output is 1



A first problem in NPC

- Given a description of the circuit, we can check in polynomial time whether a given assignment satisfies the circuit

A first problem in NPC

- Assume a decision problem in NP.
- Thus, there exists an algorithm A that verifies a solution y for problem x in polynomial time
- The idea is to translate this into a circuit

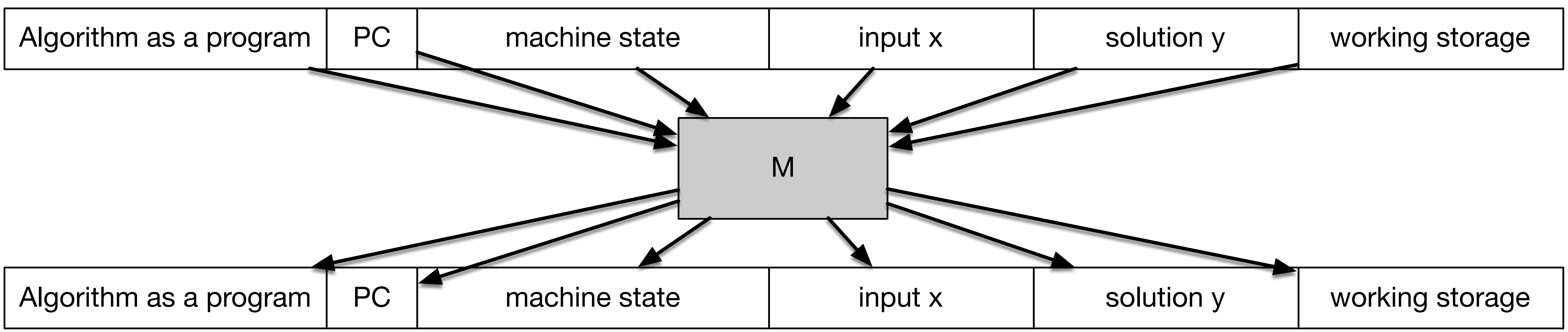
A first problem in NPC

- The algorithm is executed on a computer with program counter PC, machine state, and working storage
- The computer has as input the algorithm (program), the problem x and the solution

Algorithm as a program	PC	machine state	input x	solution y	working storage

A first problem in NPC

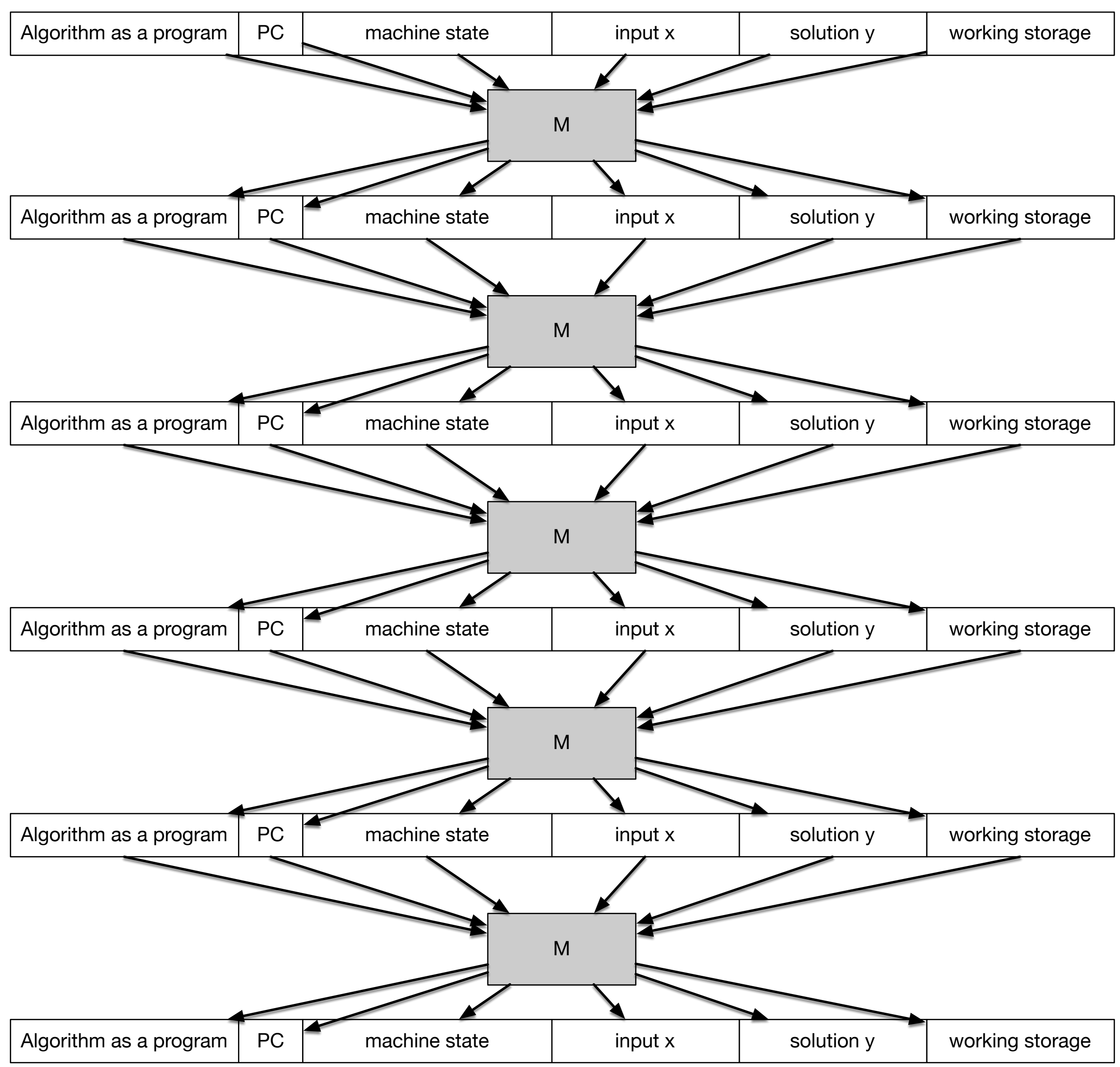
- The computer then executes a first step. This is emulated by a Boolean circuit.



- The circuit M is independent of the instruction executed and the input

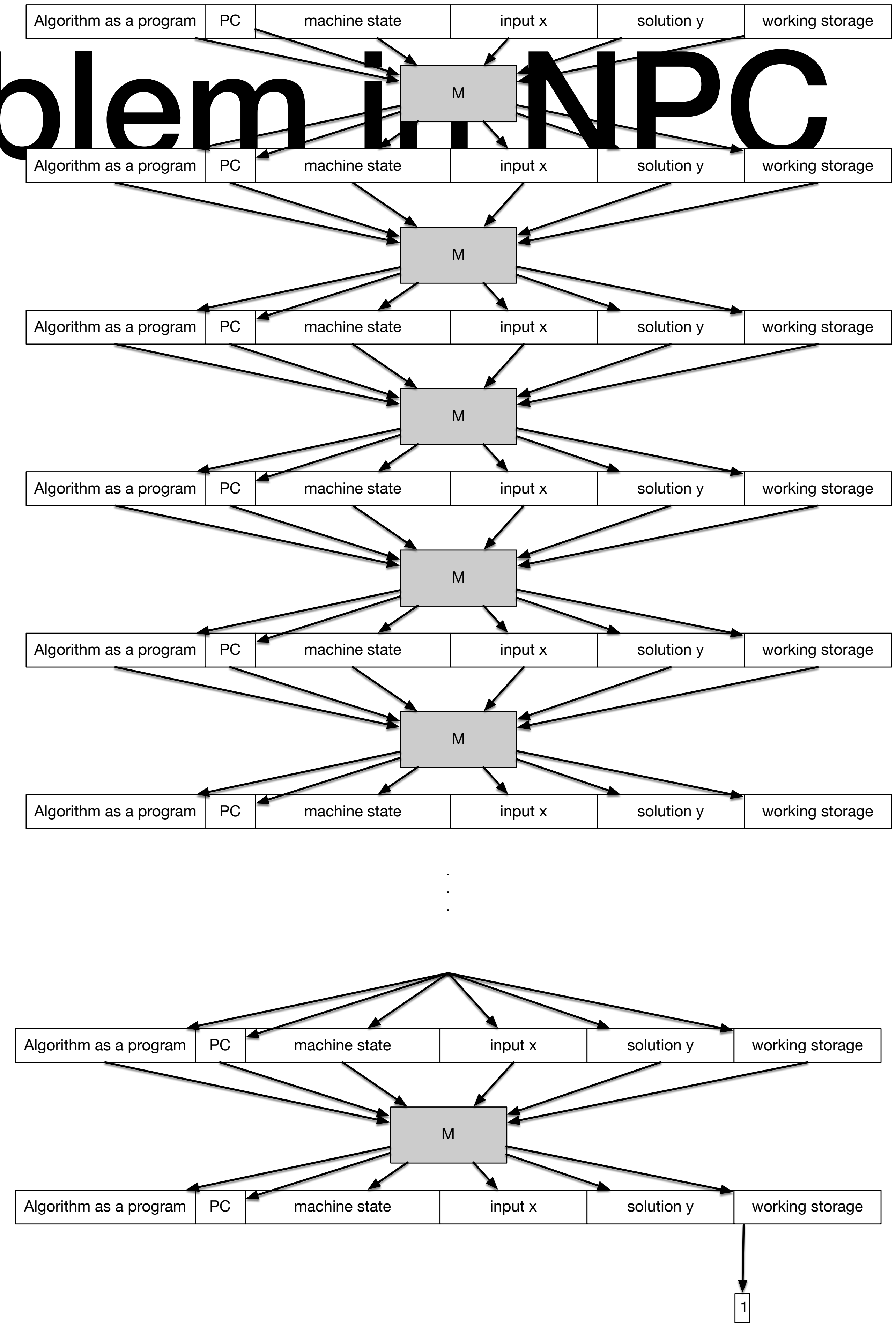
A first problem in NPC

- We repeat this over and over



A first problem in NPC

- Eventually, the first bit in working storage will contain the answer, 0 for failure and 1 for solution worked



A first problem in NPC

- What is the size of the circuit?
  - Since the algorithm runs in worst case time

$$f(n) \in \mathbb{R}[n]$$

  - The working storage is smaller than $f(n)$
  - The aux. machine state is smaller than $f(n)$
  - Program, PC, x, y are smaller than $f(n)$
- First row is smaller than $f(n)^6$

A first problem in NPC

- The number of rows is smaller than $f(n)$
- Total size of the circuit is smaller than $f(n)^7$
  - which is still a polynomial

A first problem in NPC

- Given a problem in NP
  - Can construct the circuit in polynomial time
    - Details are skipped over
- Circuit has input y (the guessed solution)
- And outputs whether for a given x the solution y is indeed a solution
- If circuit-satisfiability is in P:
  - Can decide whether a given x has a solution in polynomial time
- Therefore, the original problem would be in P