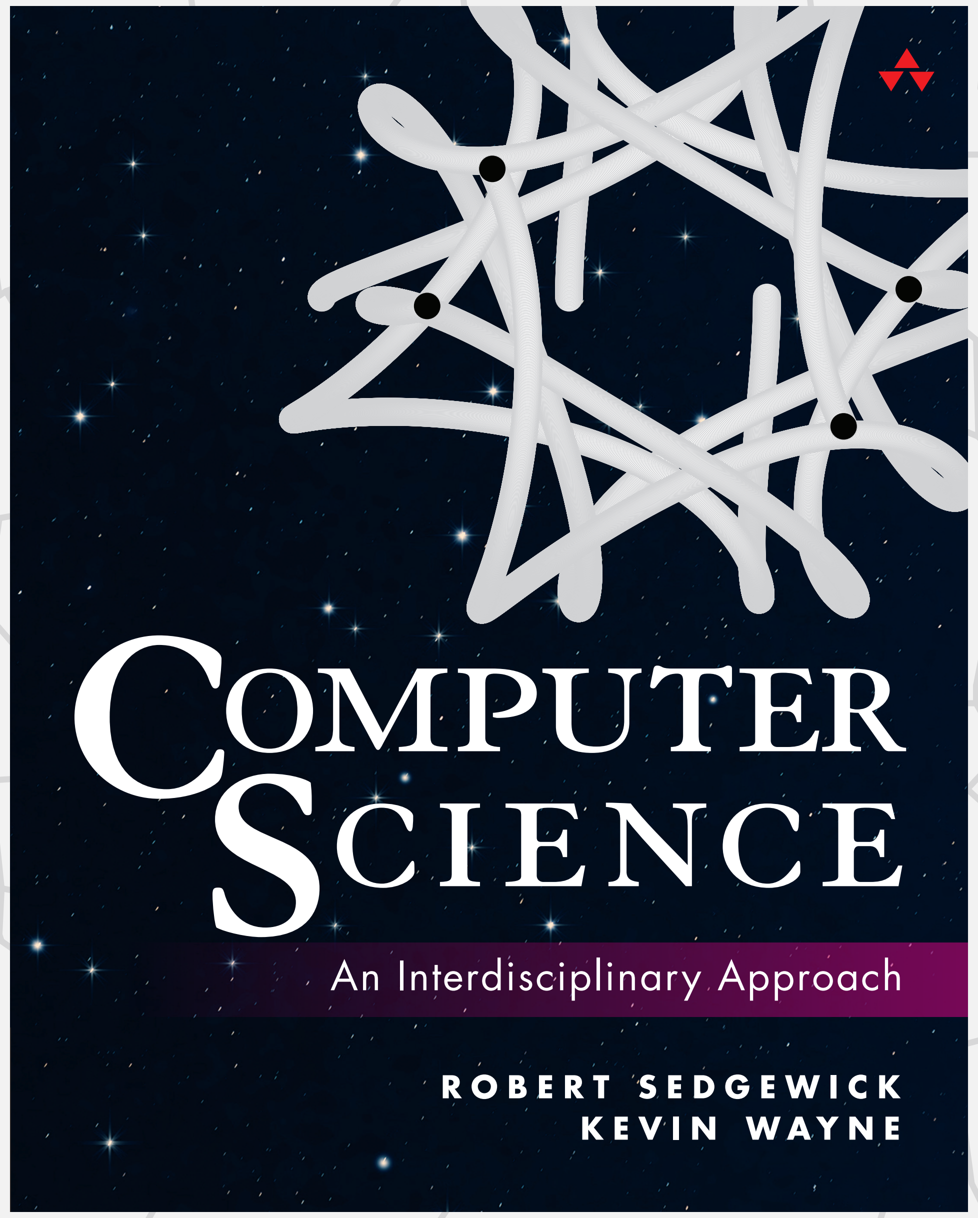
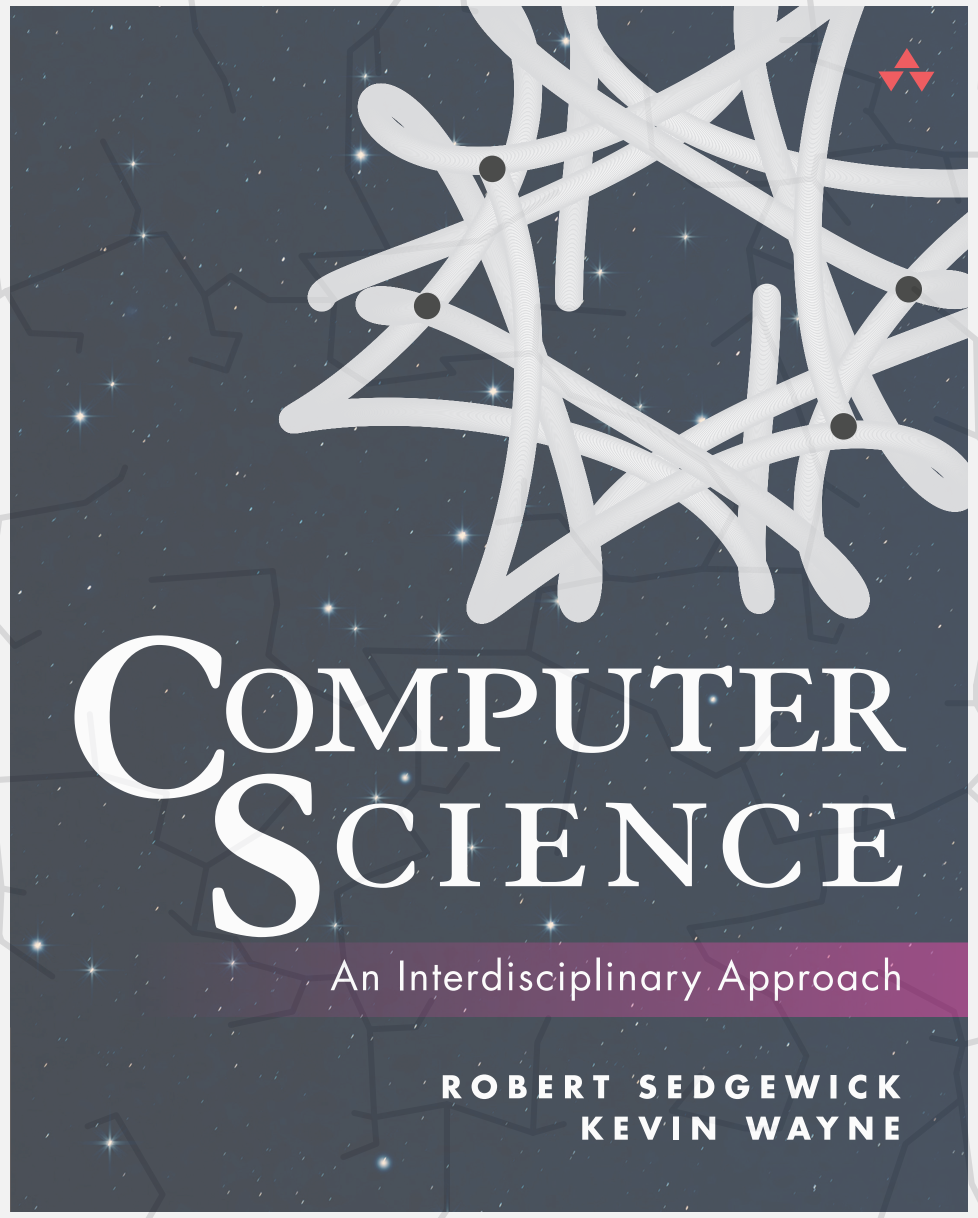


http://introcs.cs.princeton.edu

ALL QUESTIONS THEORY

- *REs, DFAs, and NFAs*
- *Universality and computability*
- *P, NP, NP-complete*



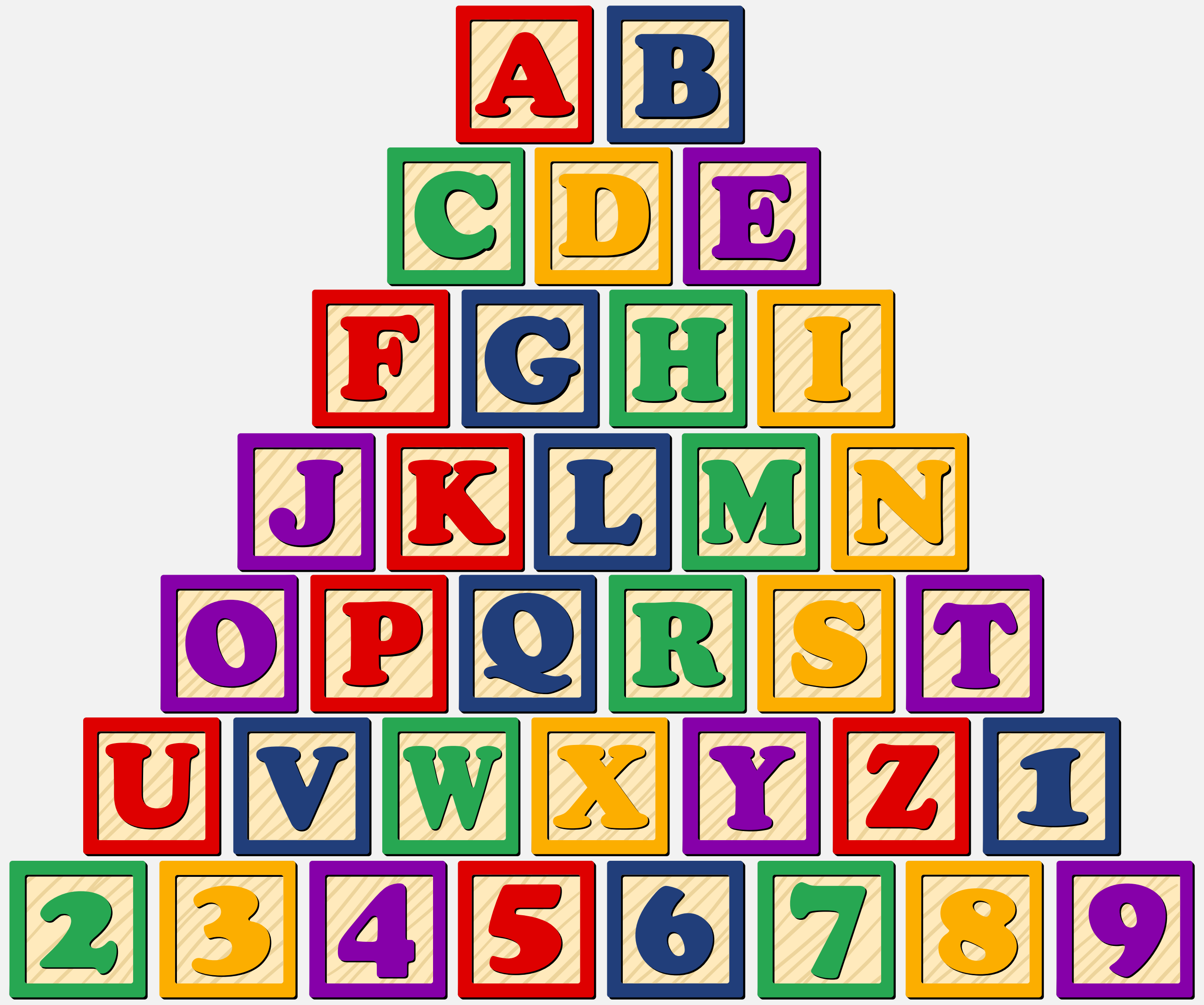
http://introcs.cs.princeton.edu

ALL QUESTIONS THEORY

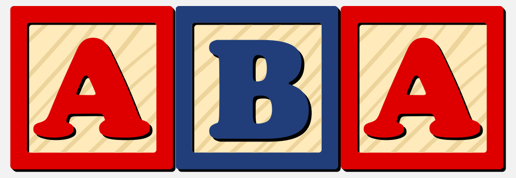
- *REs, DFAs, and NFAs*
- *Universality and computability*
- *P, NP, NP-complete*

Definitions

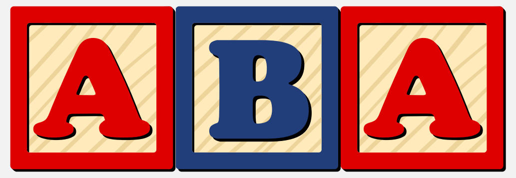
Alphabet. Finite set of symbols.

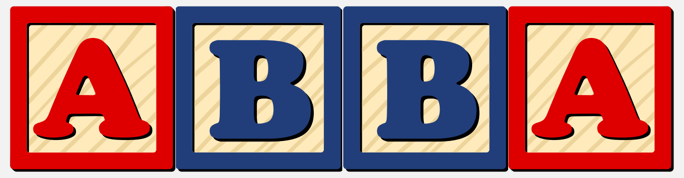


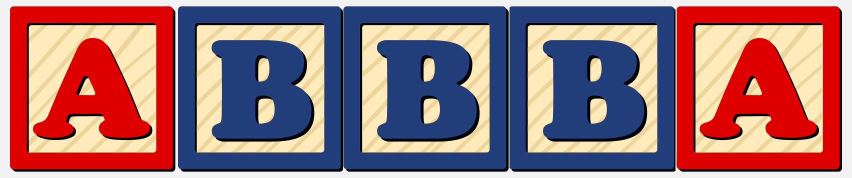
String. Sequence of alphabet symbols.



Formal language. Set of strings.

{ 

, 

, 

, … }

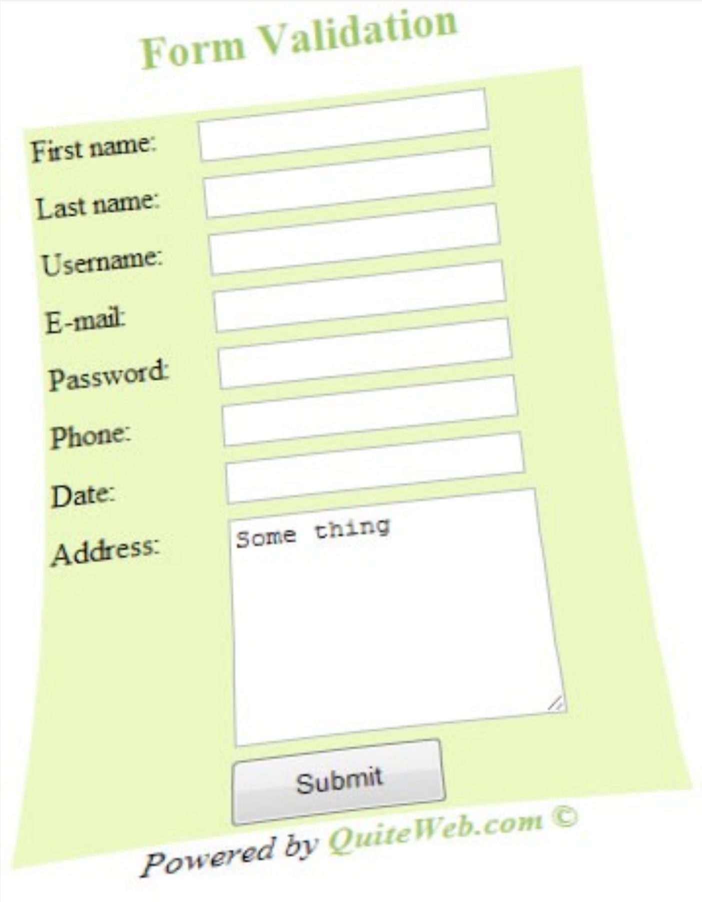
Big ideas

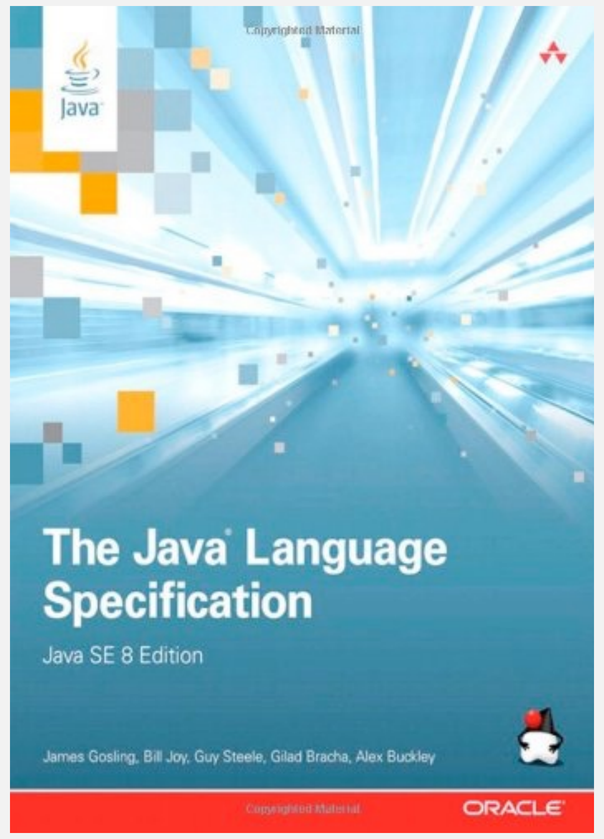
Regular expression. Concise notation for specifying a formal language.

Relevance. Widely used in practice to

- Validate web forms.
- Express legal queries.
- Specify biological sequences.
- Define elements of a programming language.
- Enforce programming style guidelines.
- ....

```
<module name="LocalVariableName">
    <property name  = "format"
              value = "^[a-z][a-zA-Z0-9]*$"/>
</module>
```





Big ideas

Regular expression. Concise notation for specifying a formal language.

operation	order	example RE	matches	does not match
concatenation	3	AABAAB	AABAAB	*every other string*
union	4	AA | BAAB	AA BAAB	*every other string*
closure	2	AB*A	AA ABBBBBBBBA	AB ABABA
parentheses	1	A(A|B)AAB	AAAAB ABAAB	*every other string*
		(AB)*A	A ABABABABABA	AA ABBA

Spring 2015, Question 3

For each English-language description at right, design an RE that describes the formal language.

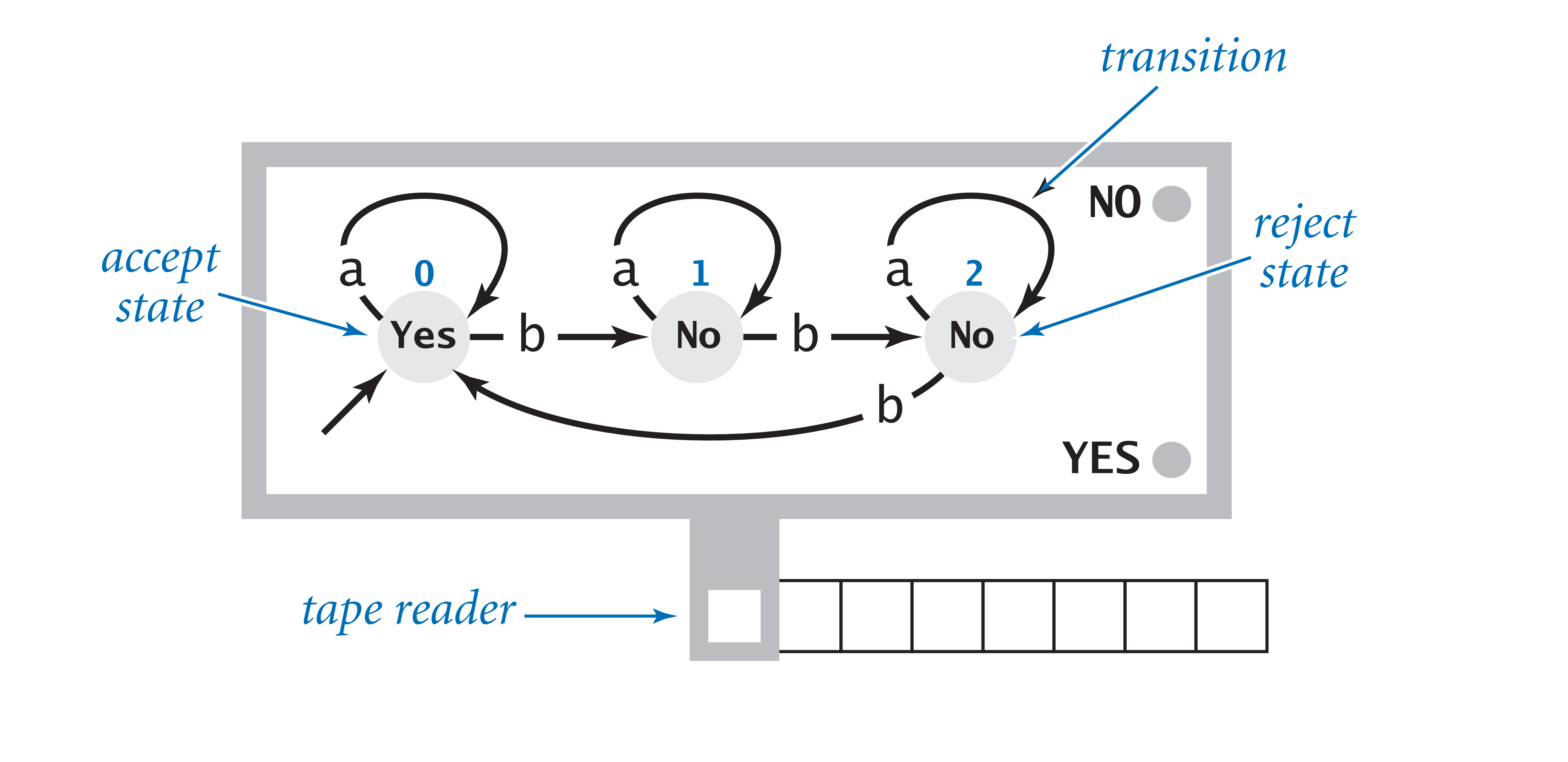
______________________________ Binary strings beginning and ending with the same bit

______________________________ Binary strings containing exactly two 0s that are not adjacent and no other 0s

______________________________ Binary numbers divisible by 8

Big ideas

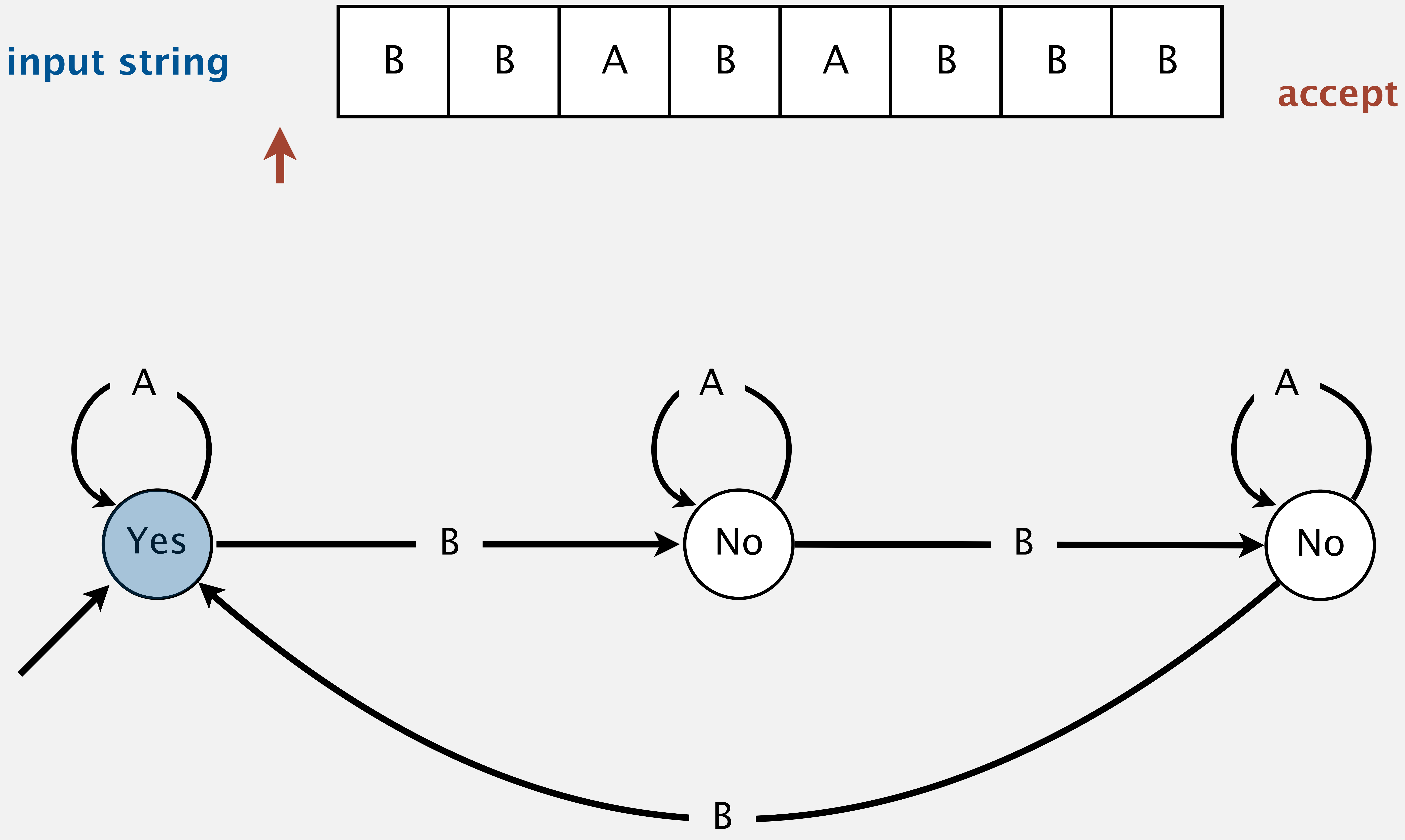
DFA. A simple machine that recognizes a language. For each input symbol and each state, there is exactly one possible state transition.



Kleene's theorem. REs, DFAs, and NFAs are equivalent models—they all characterize the regular languages.

Relevance. DFAs and NFAs solve the recognition problem for REs.

Tracing a DFA



Spring 2009, Question 9

For each DFA at right, select the best-matching RE at left.

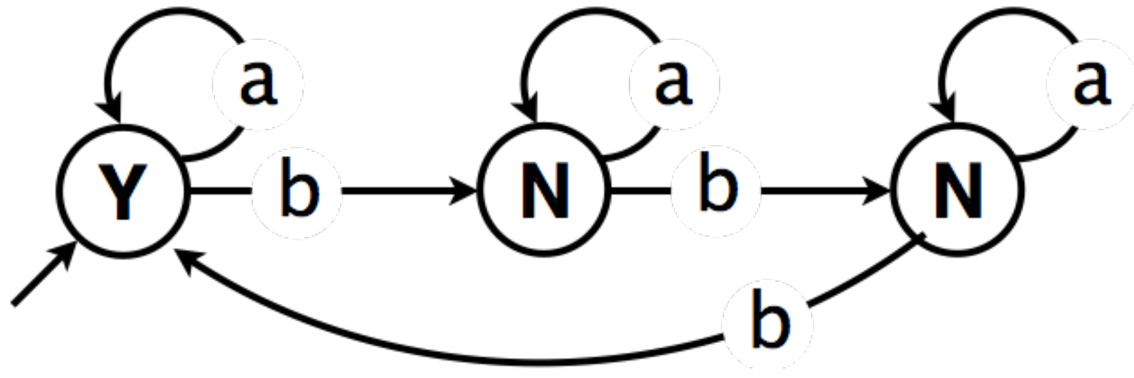
A. `a*|(a*ba*ba*ba*)*`

B. `(ab)*`

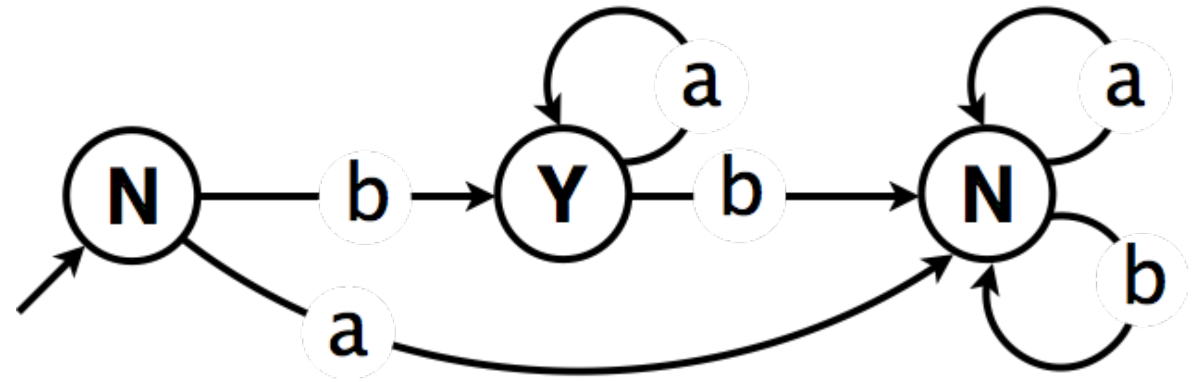
C. `(a*b)+`

D. `ba*`

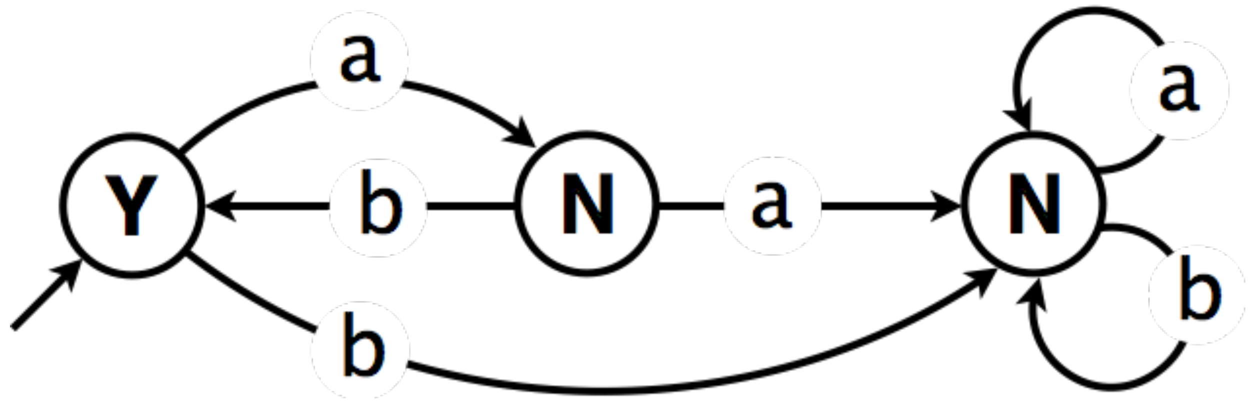
____

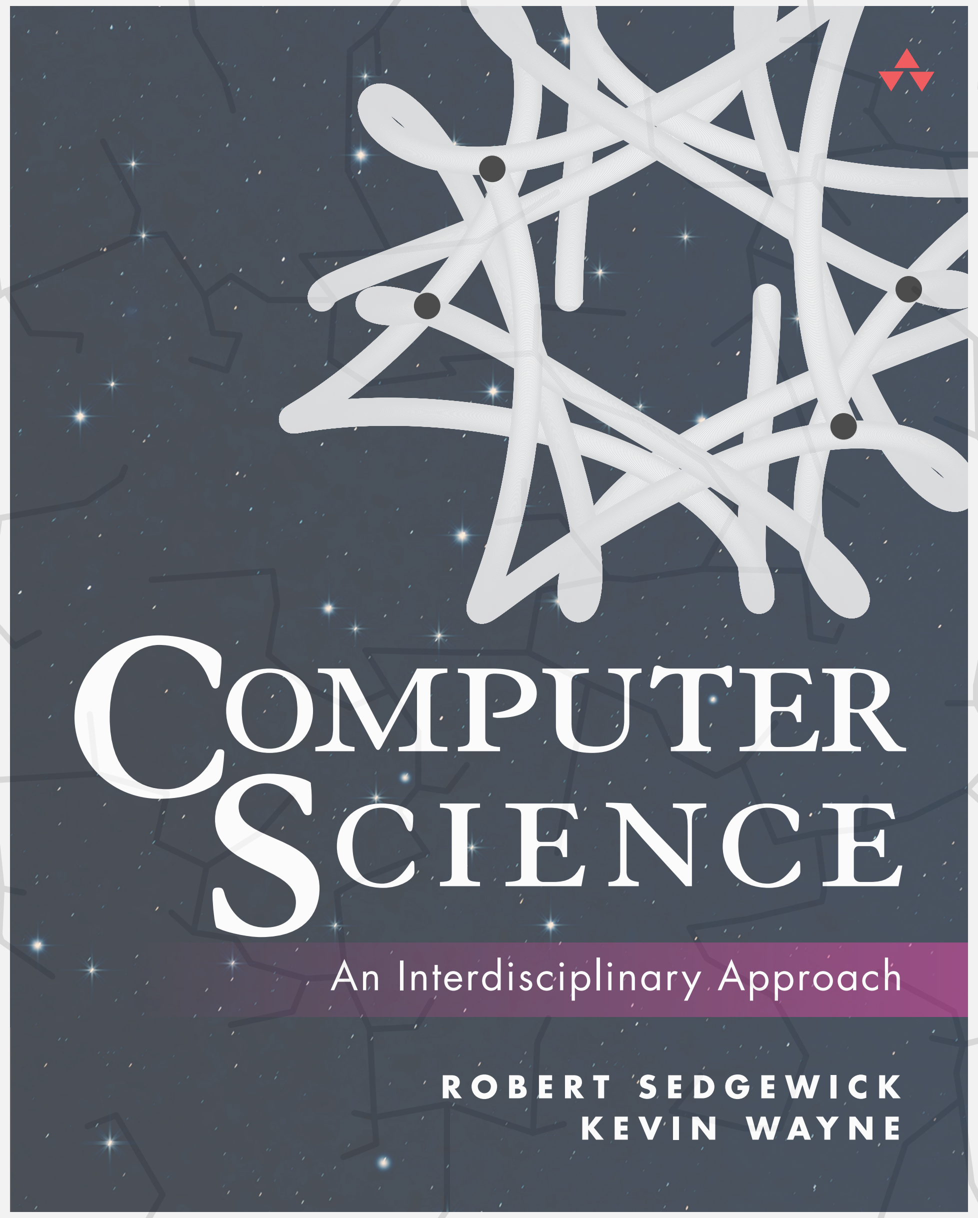


____



____





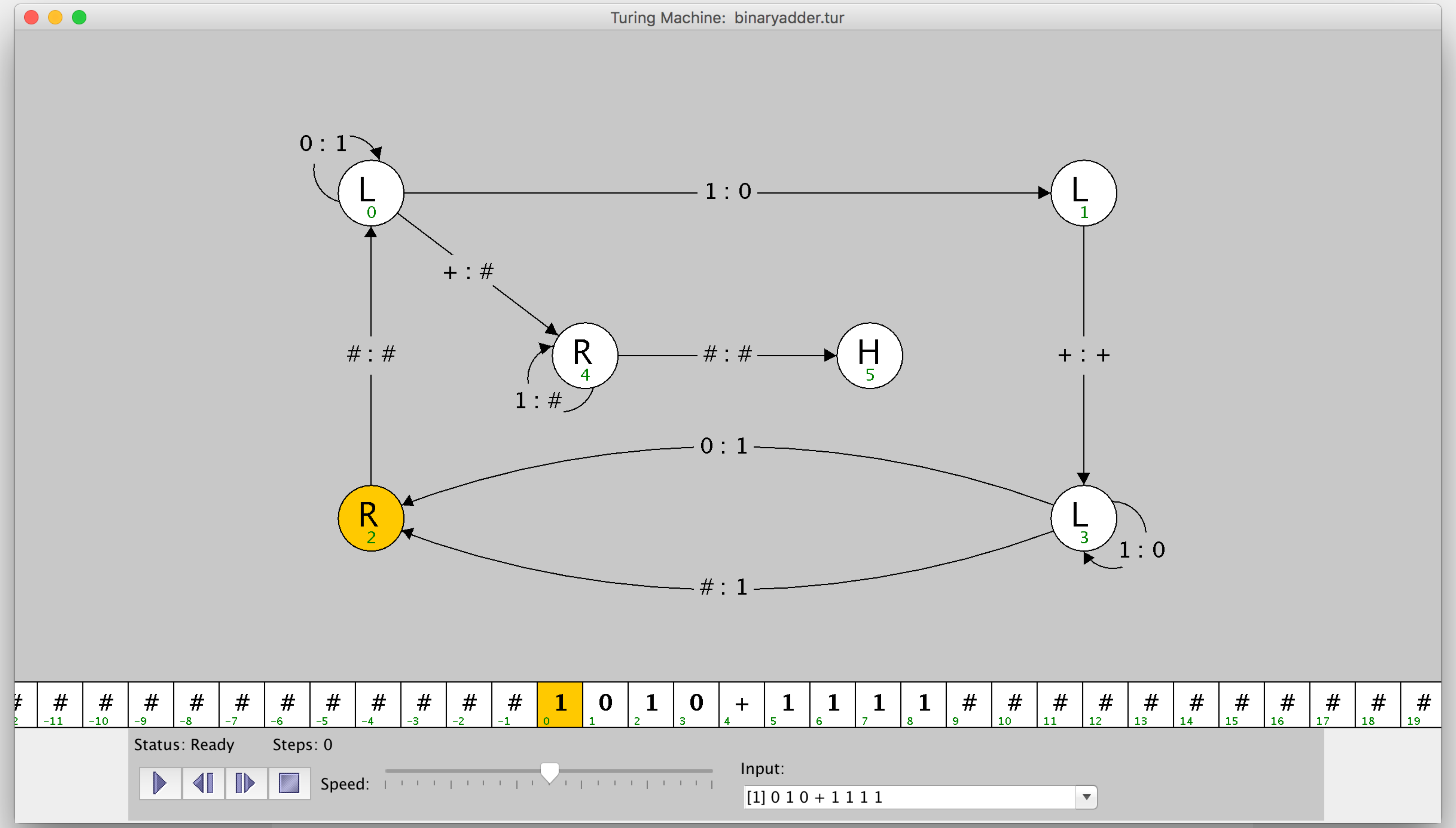
http://introcs.cs.princeton.edu

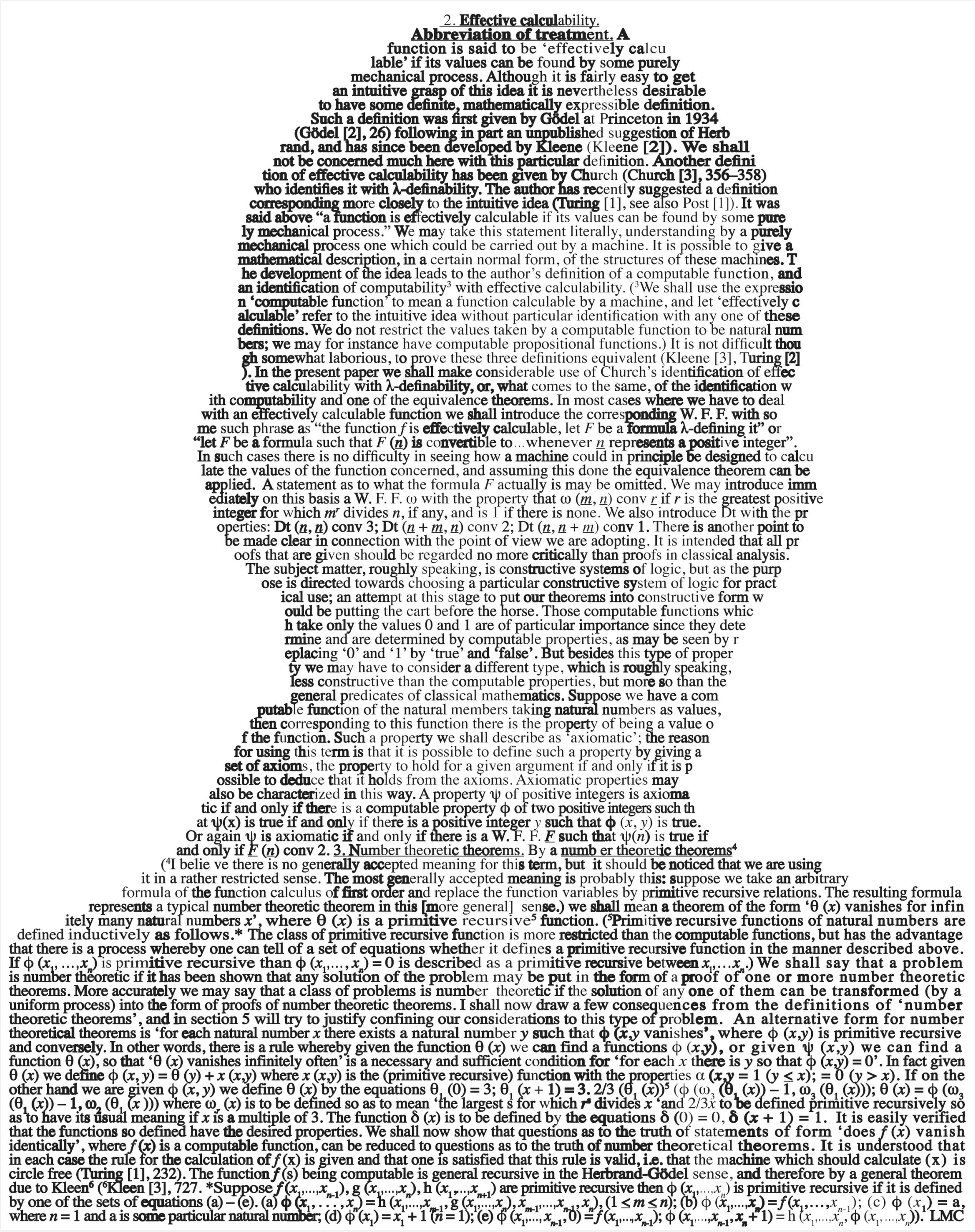
ALL QUESTIONS THEORY

- REs, DFAs, and NFAs
- *Universality and computability*
- P, NP, NP-complete

Big ideas

Turing machine. A simple, universal model of computation.
(similar to a DFA, but can read/write to tape, which is arbitrarily long)



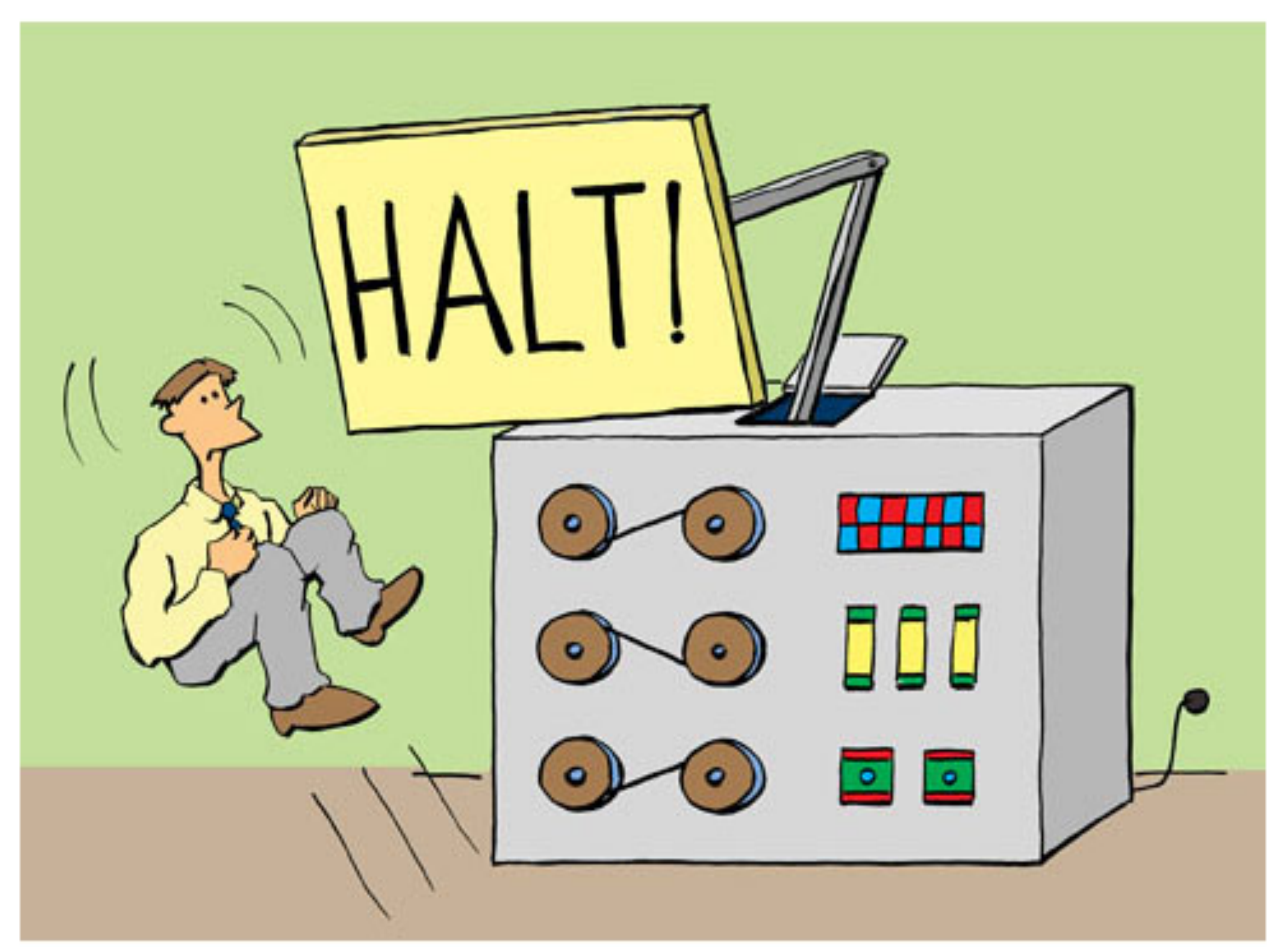


Relevance. Provides rigorous definition of a computer program; enables study of computation.

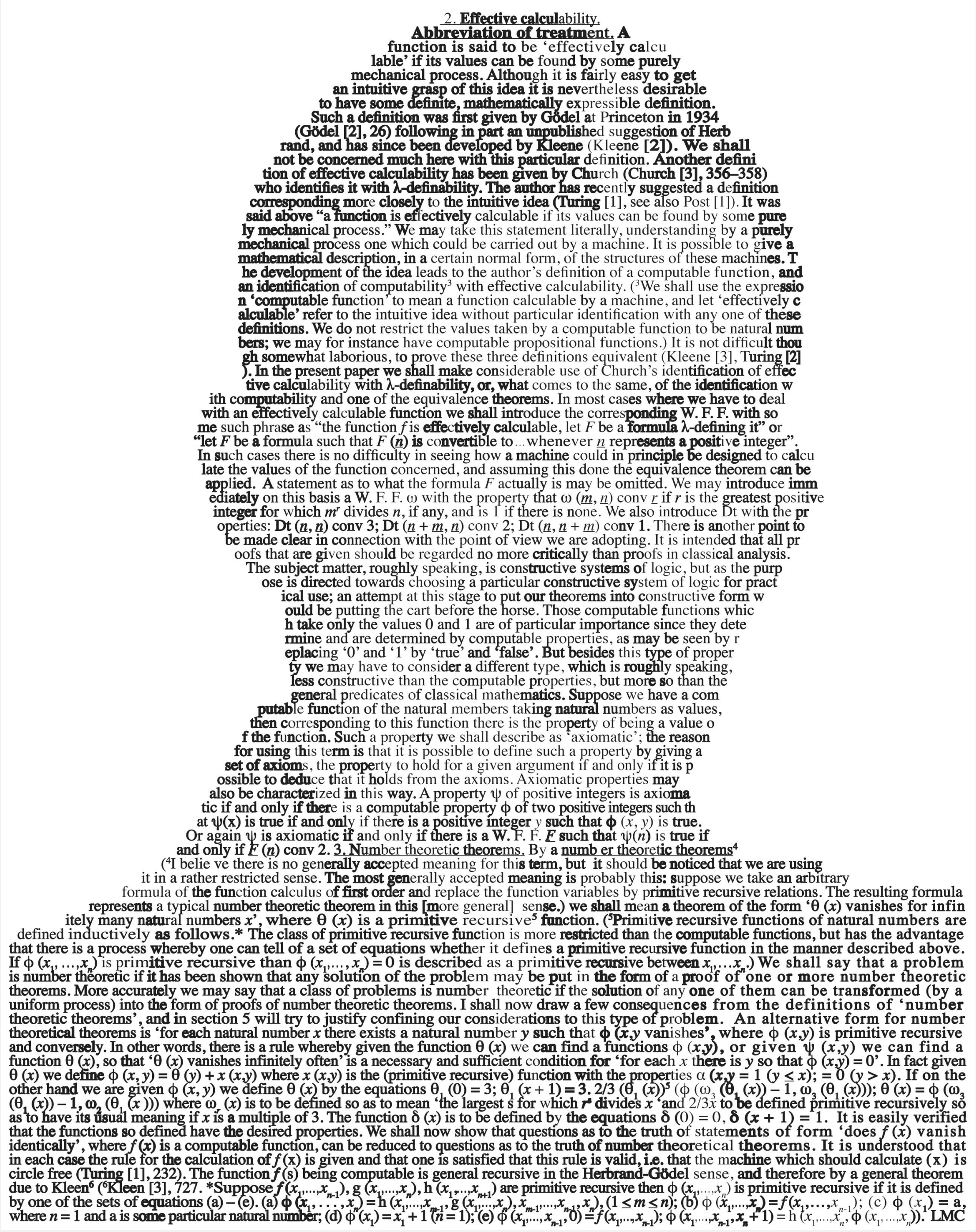
Big ideas

Halting problem. Given a Java program (or TM) and its input, does that program halt when run on that input?

Computability. The halting problem is undecidable—it's impossible to write a Java program (or TM) to solve.



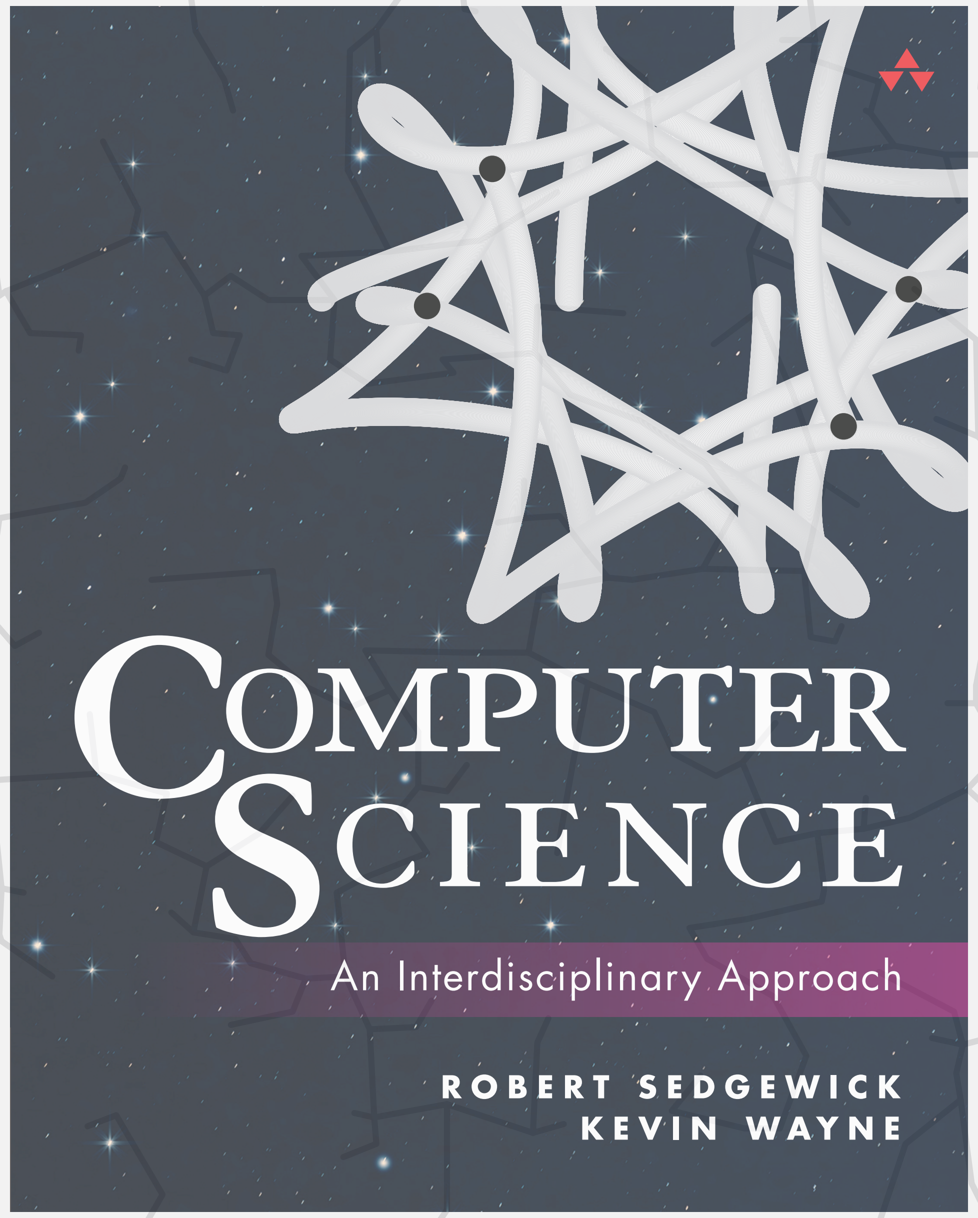
Alan designed the perfect computer



Church–Turing thesis (modern interpretation). A TM can perform any computation (decide a language or compute a function) that can be described by any physically realizable computing device.

Spring 2012, Question 8 and Fall 2012, Question 9

	TRUE	FALSE	UNKNOWN
For any language that some TM decides, there exists a DFA that recognizes the same language.	○	○	○
For any language that some RE describes, there exists a TM that decides the same language.	○	○	○
For any function that some TM computes, there exists a Java program that computes the same function.	○	○	○
Most computer scientists believe that the Church–Turing thesis will eventually be proved true.	○	○	○
No future computer can solve the halting problem.	○	○	○



http://introcs.cs.princeton.edu

All Questions Theory

- REs, DFAs, and NFAs
- Universality and computability
- *P*, *NP*, *NP*-complete

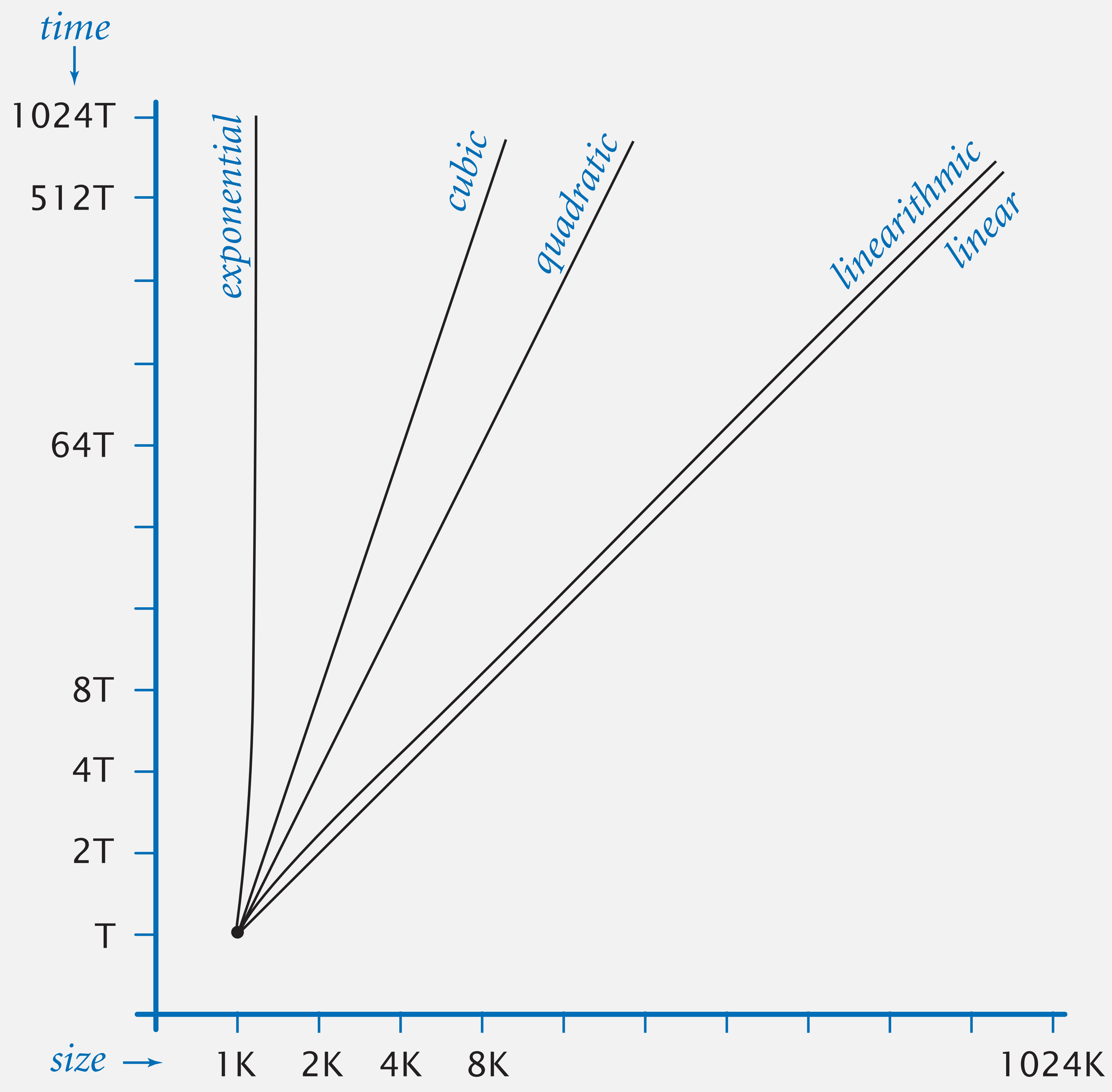
Big ideas (polynomial time vs. exponential time)

Polynomial-time algorithm. Running time $\le a\, n^b$ for all inputs of size n.

Exponential-time algorithm. Running time $\ge 2^n$ for infinitely many inputs.

or $1.1^{\sqrt{n}}$ or $n!$

Relevance. "Efficient in practice."



Orders of growth (log–log plot)

Polynomial time vs. exponential time

Classify each algorithm.

	POLYNOMIAL	EXPONENTIAL
mergesort (to sort an array of n elements)	○	○
insertion sort (to sort an array of n elements)	○	○
recursive H-tree program (to draw an H-tree of order n)	○	○
smallest increase heuristic (to build a TSP tour of n points)	○	○
brute-force TSP algorithm (that tries all $n!$ permutations)	○	○

Big ideas (search problems)

Search problem. There exists a poly-time algorithm that checks whether a given solution solves a given instance of the problem.

FACTOR. Given an n-digit integer x, find a factor (other than 1 and x).

Q. How to show that FACTOR is a search problem?

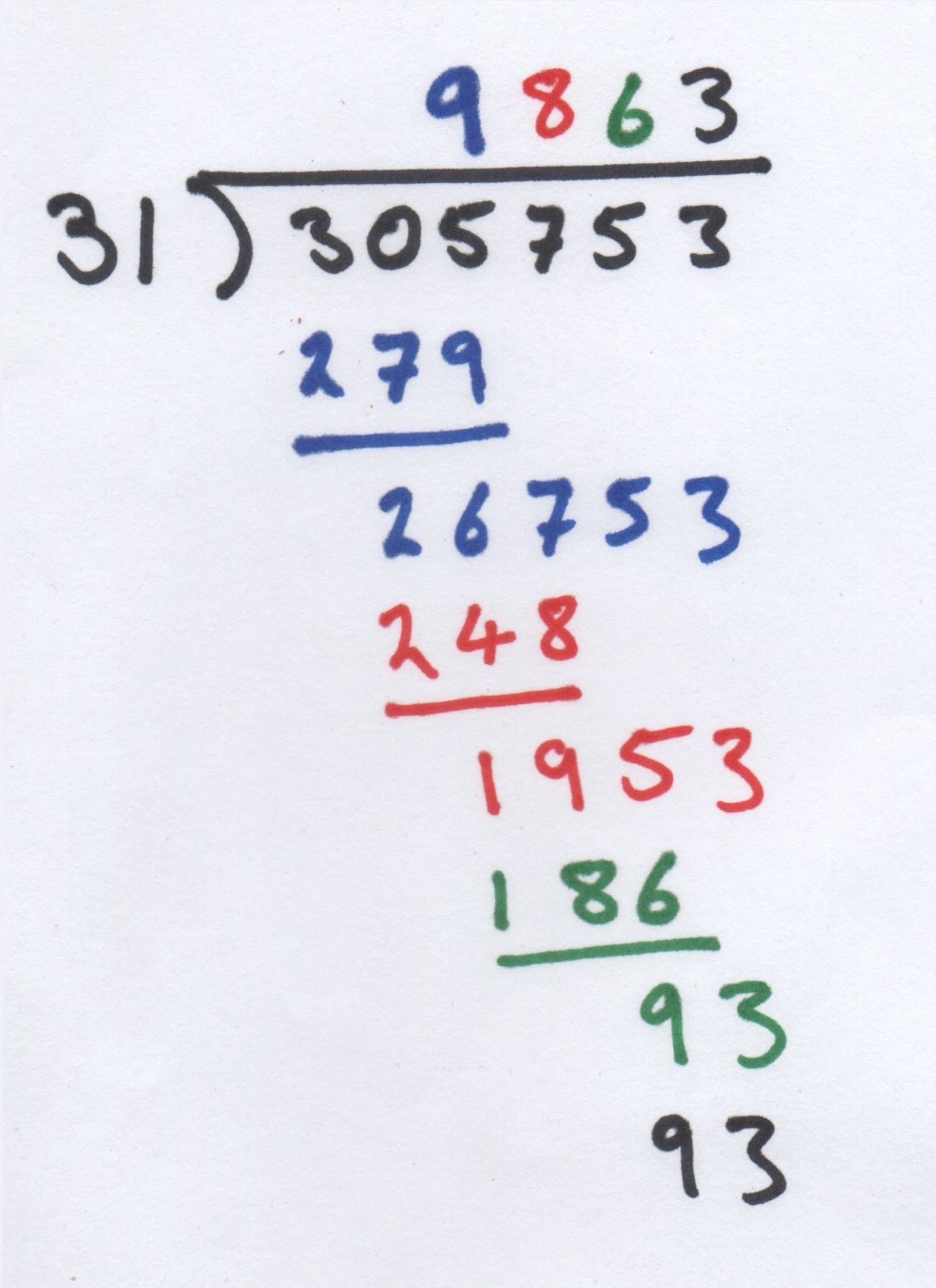
A. Given a purported factor d, need a poly-time algorithm to check that

- d is not equal to 1 or x.
- d is a divisor of x.

grade-school division is n^2
(faster algorithms are known)

Instance. $x = 305753$

Factor? $d = 31$



Big ideas (search problems)

Search problem. There exists a poly-time algorithm that checks whether a given solution solves a given instance of the problem.

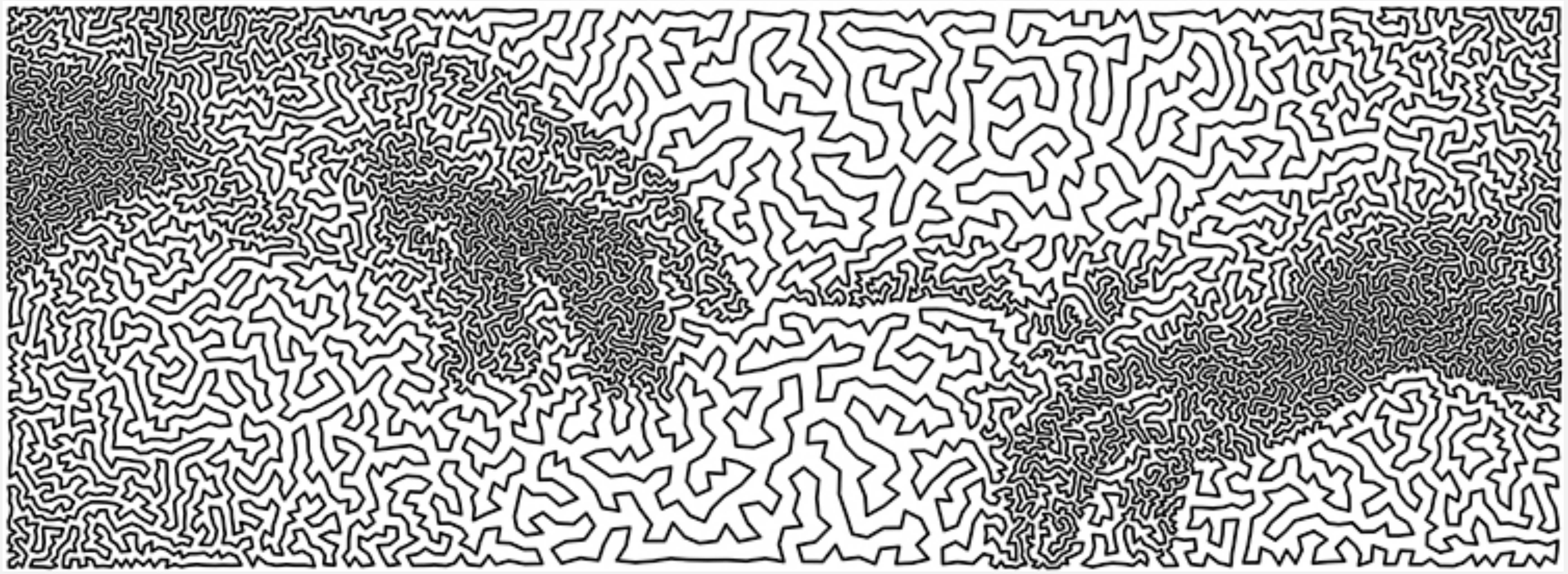
TSP. Given n points in the plane and an integer L, find a tour of length at most L.

distance between two points is Euclidean distance, rounded to the nearest inch

Q. How to show that TSP is a search problem?

A. Given a purported tour x, need a poly-time algorithm to verify that

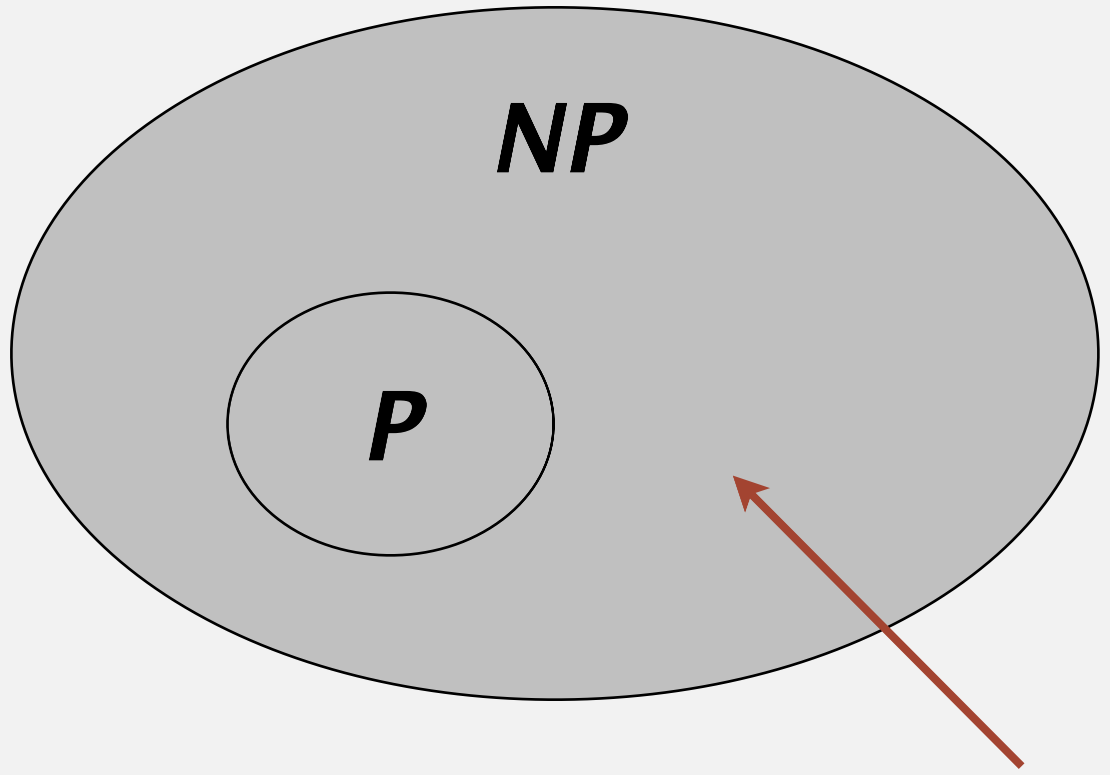
- x is a tour (i.e, a permutation of the n points).
- The length of x is at most L.



Big ideas (P vs. NP)

NP. The set of all search problems.

P. The set of all search problems that can be solved in polynomial time.



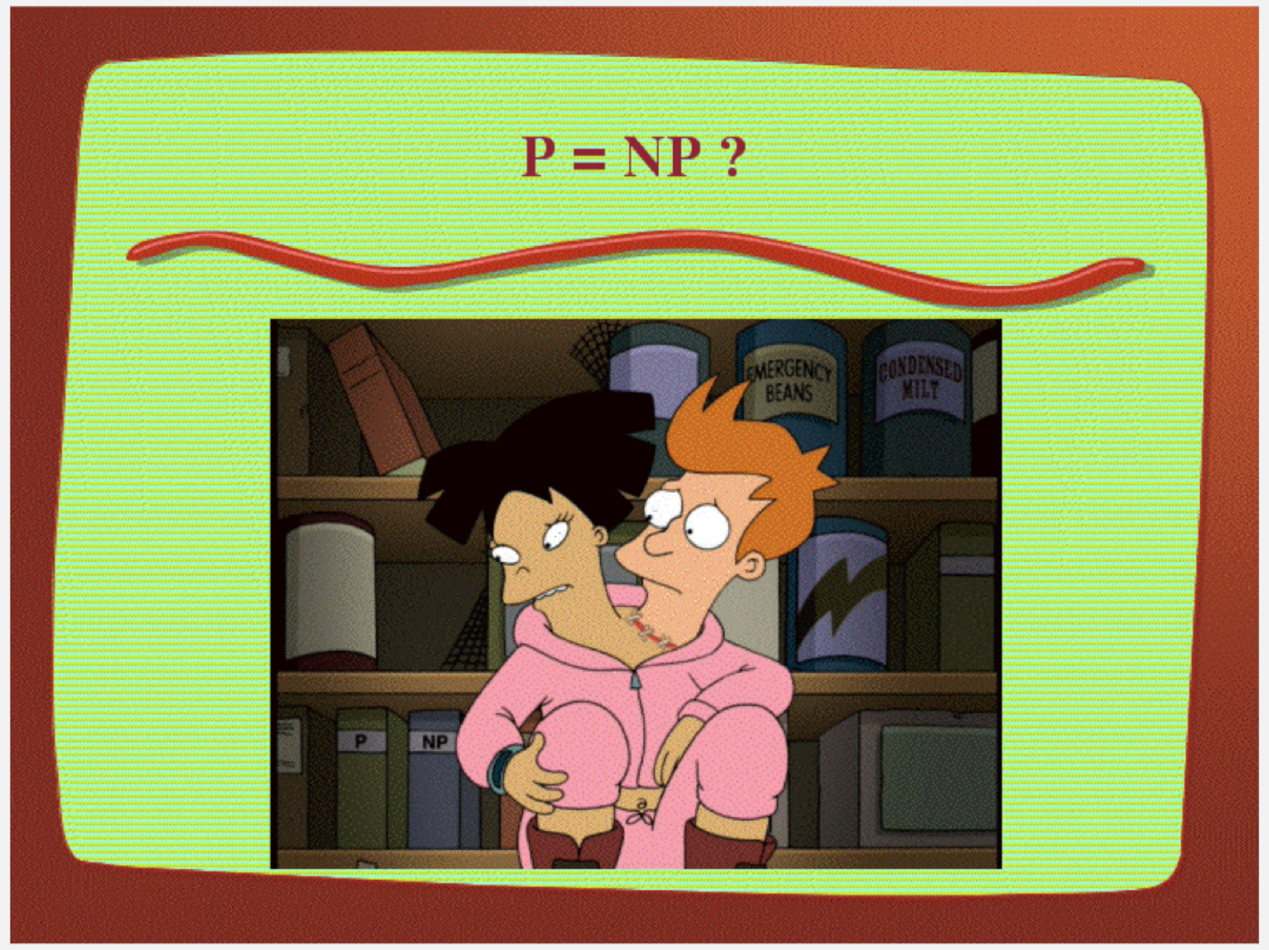
$P \neq NP$

intractable search problems

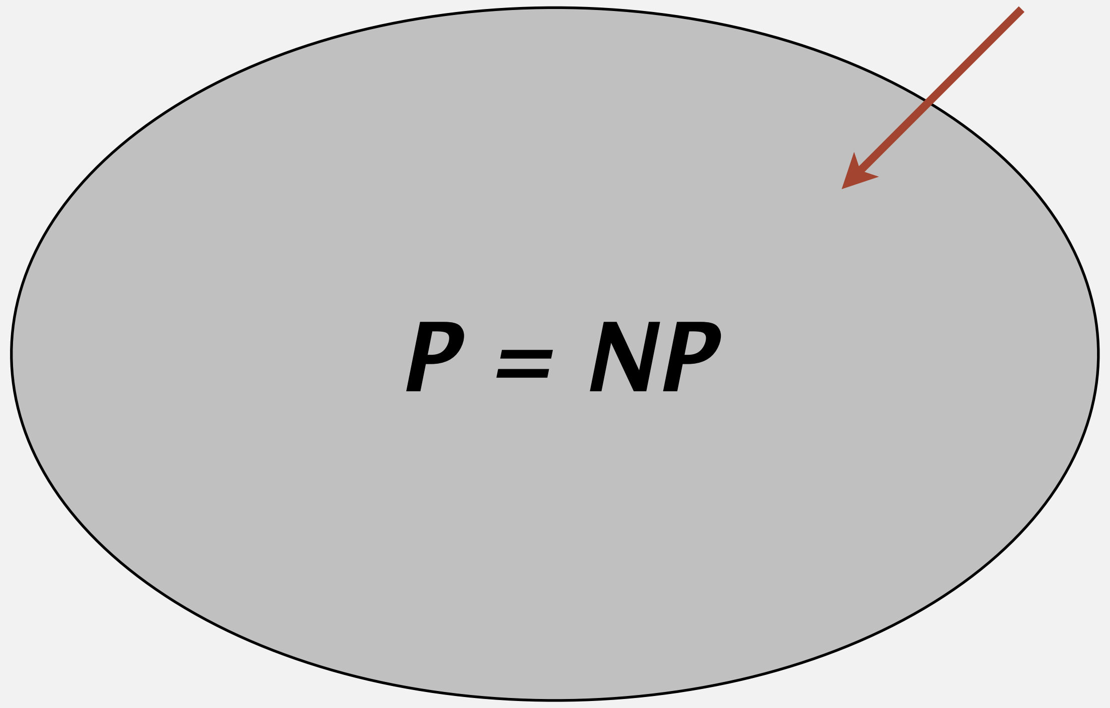
Relevance. ***NP*** contains problems that we aspire to solve in practice;
P contains problems that we can solve in practice.

Definition. A problem is intractable if there exists no poly-time algorithm to solve it.

Famous conjecture. $P \neq NP$ [There exist intractable search problems]



no intractable search problems



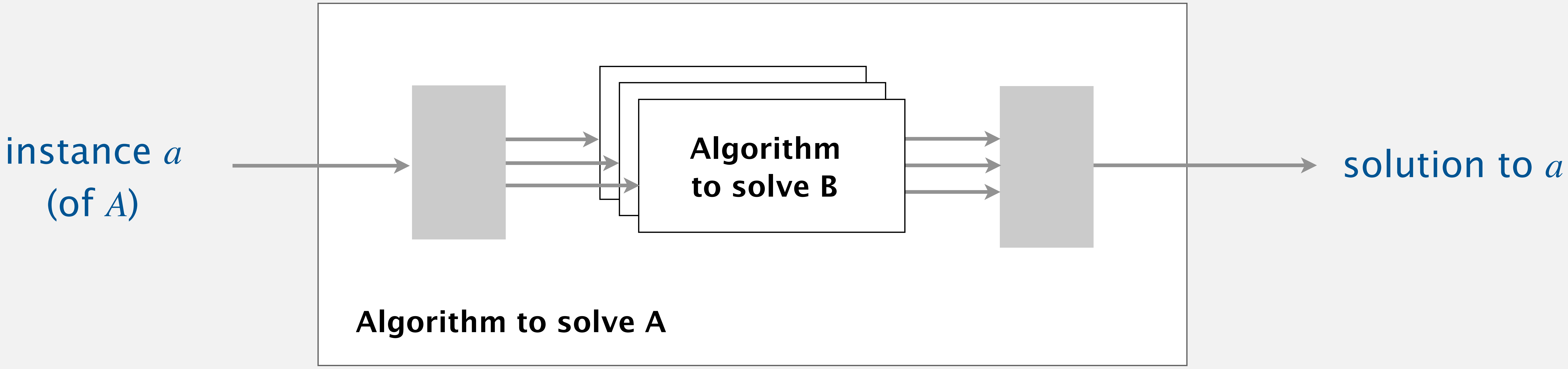
$P = NP$

Fall 2014, Question 8

	TRUE	FALSE	UNKNOWN
TSP is in ***NP***.	○	○	○
Every problem in ***P*** is also in ***NP***.	○	○	○
Every problem in ***NP*** is also in ***P***.	○	○	○
Every problem in ***NP*** can be solved with an exponential-time algorithm.	○	○	○
The halting problem is in ***NP***.	○	○	○
If ***P*** = ***NP***, then there exists a poly-time algorithm for FACTOR.	○	○	○

Big ideas (poly-time reductions)

Definition. A problem A poly-time reduces to a problem B if there exists an algorithm for A that uses a polynomial number of calls to a subroutine for B, plus polynomial time outside of those subroutine calls.



A poly-time reduces to B
(A can be solved by B)

Intuition. Can solve B efficiently $\Rightarrow$ can also solve A efficiently.

Contrapositive. Can't solve A efficiently $\Rightarrow$ can't solve B efficiently either.

Definition. A search problem B is ***NP***-complete if every search problem A poly-time reduces to B.

Spring 2015, Question 5a

Suppose that problem A poly-time reduces to problem B. Which of the following statements can we infer?

	TRUE	FALSE
If B is in ***P***, then A is in ***P***.	○	○
If A is in ***P***, then B is in ***P***.	○	○
If A is ***NP***-complete, then B is ***NP***-complete.	○	○
If A is ***NP***-complete and B is in ***NP***, then B is ***NP***-complete.	○	○
A and B cannot both be ***NP***-complete.	○	○