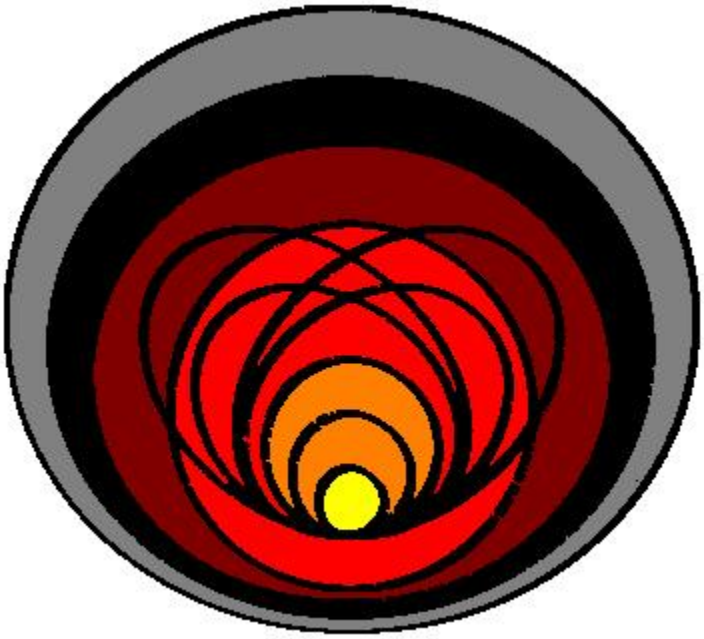
Theoretical Computer Science

CS Games 2018

I'm sorry, Dave. I'm afraid I can't do that.



Solutions

The central computer of the spaceship, ALAN, was broken during the crash. As you repair it, it becomes more and more powerful.

Instructions :

- **Teams of 2**
- **3 hours**
- You cannot use any handwritten notes nor a computer, except for a simple **calculator**.
- Answer the questions as best as you can. Some are easier than they appear while others are harder than they seem.
- You are not expected to have the time to answer every question. Choose carefully!
- Points will be awarded for partial answers and good ideas.
- Whenever the question starts with *True or False*, you need not justify your answer, but a wrong answer will make you **lose two points**.
- Whenever the question starts with *Choices*, you need not justify your answer, but a wrong answer will make you **lose one point**.

Contents

1 Just a regular computer

You somehow manage to boot ALAN, but he suffers from amnesia. He has no memory, thus he can only recognize regular languages.

1.1 High Among the Stars (3pts)

Consider the following regular expression R over $\{0,1\}$:

$$R = (0^*1)^*$$

To make ALAN's coming to life a little easier, you decide to rewrite this expression. Give a regular expression R' such that $L(R') = L(R)$ and whose star height is 1 (i.e. there can be no nested Kleene star). The only operations allowed are union, concatenation and the Kleene star, and you can use the empty string ϵ.

Average score: 1.34 / 3
Solution:
$R' = \epsilon \cup (0 \cup 1)^*1$

1.2 Boring Poems (4pts)

Although ALAN was designed for space travel applications, he always wished he were an artist. Therefore, his very first action is to write a poem.

His style is very traditional : he only uses unmixed (AABB), alternate (ABAB) and envelope (ABBA) rhymes. Formally, the set of poem structures he may use is :

$$\{s_0 s_1 ... s_{4n-1} | n \in \mathbb{N}^+ \wedge$$

$$\forall i \in \{0, ..., n-1\}, s_{4i}s_{4i+1}s_{4i+2}s_{4i+3} \in \{AABB, ABAB, ABBA\}\}$$

1. Show that this set is **regular**.
2. Since regular poems are boring, he would like to add palindromic rhymes :

   $$\{ww^r | w \in \{A,B\}^*,\ r \text{ denotes the reverse string}\}$$

   Use the pumping lemma to show that the set is not regular anymore.

Average score: 1.00 / 4
Solution:

1. A simple regular expression is sufficient :

   $$(AABB|ABAB|ABBA)(AABB|ABAB|ABBA)^*$$

2. Suppose the set is regular. Then there exists a pumping length $p > 0$. Let $w = A^pBBA^p$ be a word in the set. Since $|w| > p$, we can write $w = xyz$ with $|xy| \leq p$ and $|y| > 0$, such that for each $i \in \mathbb{N}$, $w' = xy^iz$ is also in the set. Then y must contain only As and $xy^2z = A^{p+|y|}BBA^p$, which should not belong to the set. This provides a contradiction to the assumption that the set was regular, so it is not regular.

1.3 True or False: NFA (2pts)

True or false : In general, it is impossible to describe the transition relation of a nondeterministic finite automaton with a function.

Average score: 0.27 / 2
Solution:
False. It can be expressed as $S \times \Sigma \to \mathcal{P}(S)$ where $\mathcal{P}(S)$ is the power set of S.

2 ALAN the pushdown automaton

You manage to enhance ALAN by equipping him with a stack memory. He is very excited because he is now able to distinguish words of the same length and parse simple arithmetic expressions.

2.1 Not the Same (4pts)

Show that

$$L = \{xy \mid x \neq y \wedge x, y \in \{a, b\}^n \text{ with } n \in \mathbb{N}^+\}$$

is context-free by designing either a context-free grammar (strongly recommended) or a pushdown automaton for L.
Hint : Any $w = xy \in L$ with $|x| = |y| = n$ has an index $i \leq n$ for which, without loss of generality, $x[i] = a$ and $y[i] = b$, while the other symbols can be **anything**.

Average score: 0.48 / 4
Solution:
We can write w as $x_l a x_r y_l b y_r$ where $x_l, y_l \in \{a, b\}^{i-1}$, $x_r, y_r \in \{a, b\}^{n-i}$. We can then rewrite $x_r y_l$ as $x'_r y'_l$ where $x'_r \in \{a, b\}^{i-1}$ and $y'_l \in \{a, b\}^{n-i}$. Then $w = x_l a x'_r y'_l b y_r$, where $|x_l| = |x'_r|$ and $|y'_l| = |y_r|$. The context-free grammar is straightforward from this point :
S → XY | YX
X → aXa | aXb | bXa | bXb | a
Y → aYa | aYb | bYa | bYb | b

2.2 Parsing Arithmetic Expressions (4pts)

Recall the following definitions, where $A, B, ...$ denote nonterminal symbols, S is the starting nonterminal symbol, $a, b, ...$ denote terminal symbols, \$ is the end of input marker, $\alpha, \beta, ...$ are strings of nonterminal and terminal symbols, and ϵ is the empty string :

- $FIRST$ set
  - $FIRST(\epsilon) = \{\epsilon\}$
  - $FIRST(a\alpha) = \{a\}$
  - $FIRST(A) = FIRST(\alpha_1) \cup FIRST(\alpha_2) \cup ... \cup FIRST(\alpha_n)$, where $A \to \alpha_1|\alpha_2|...|\alpha_n$ are the A-productions.
  - $FIRST(A\alpha) = \begin{cases} (FIRST(A) - \{\epsilon\}) \cup FIRST(\alpha) & \text{if } \epsilon \in FIRST(A) \\ FIRST(A) & \text{otherwise} \end{cases}$
- $FOLLOW$ set
  - \$ $\in FOLLOW(S)$
  - If $A \to \alpha B \beta$ is a production rule, then $FIRST(\beta) - \{\epsilon\} \subseteq FOLLOW(B)$.
  - If either $A \to \alpha B$, or $A \to \alpha B \beta$ where $\epsilon \in FIRST(\beta)$, then $FOLLOW(A) \subseteq FOLLOW(B)$

The following grammar G parses expressions containing additions and multiplications over integers in some programming language. S and T are nonterminal symbols, while +, *, **int**, (and) are terminal.

S → T | T + S | T * S
T → **int** | (S)

1. Compute the $FIRST$ and $FOLLOW$ sets for each of G's nonterminal symbols.
2. Answer the following questions. Justify briefly.
   - Is G ambiguous?

- Is G $LL(1)$?
- Is $L(G)$ $LL(1)$?

Average score: 0.89 / 4
Solution:

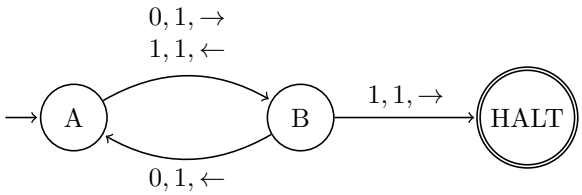
1. $FIRST(S) = FIRST(T) = \{int, (\}$
   $FOLLOW(S) = \{\$,)\}$, $FOLLOW(T) = \{+, *,), \$\}$
2. - No, there is always only one derivation tree for an expression.
   - No, it is not left-factored.
   - Yes, left-factoring G yields an LL(1) grammar which recognizes the same language.

3 Turing-complete At Last

You find an old strip of tape on the floor and decide to hand it to ALAN. Strangely, he is now able to compute any computable function. Of course, the first thing ALAN does with his new computing power is play games.

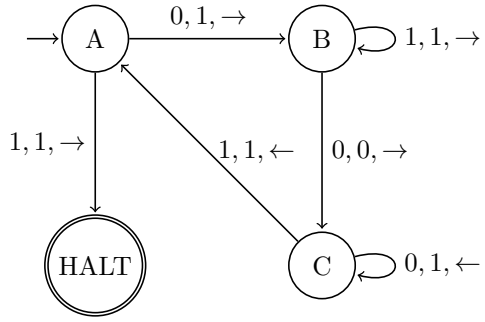
3.1 Busy Beaver Game (4pts)

In the n-states Busy Beaver game, the goal is to design a Turing machine, with at most n states (apart from the halting one), alphabet $\{0, 1\}$ and initial tape filled with 0s, which writes as many 1s as possible *before halting*.
As an example, the following machine for the 2-states Busy Beaver writes 4 1s and then halts :



You are asked to design a **3-states** Turing machine for the Busy Beaver game. The more 1s on the tape, the more points! But an invalid or non halting Turing Machine won't get you any points.

Average score: 1.03 / 4
Solution:
The following **3-states** Turing machine for the Busy Beaver game is optimal. It writes a total of 6 1s on the tape before halting.



3.2 True or False: Logic (2pts)

$$\Big((\neg r \Rightarrow q) \vee \neg(q \vee s) \vee s\Big) \wedge \Big(\big(q \wedge r \wedge \neg(t \wedge s)\big) \Rightarrow \big((q \wedge \neg t) \vee (r \wedge \neg s)\big)\Big)$$

Average score: 1.07 / 2
Solution:

True.

$$
\begin{aligned}
& \Big((\neg r \Rightarrow q) \vee \neg(q \vee s) \vee s\Big) \wedge \Big(\big(q \wedge r \wedge \neg(t \wedge s)\big) \Rightarrow \big((q \wedge \neg t) \vee (r \wedge \neg s)\big)\Big) \\
= \quad & \Big(\neg\neg r \vee q \vee (\neg q \wedge \neg s) \vee s\Big) \wedge \Big(\neg\big(q \wedge r \wedge \neg(t \wedge s)\big) \vee (q \wedge \neg t) \vee (r \wedge \neg s)\Big) \\
= \quad & \Big(q \vee r \vee s \vee (\neg q \wedge \neg s)\Big) \wedge \Big(\neg q \vee \neg r \vee \neg\neg(t \wedge s) \vee (q \wedge \neg t) \vee (r \wedge \neg s)\Big) \\
= \quad & \Big(q \vee r \vee ((s \vee \neg q) \wedge (s \vee \neg s))\Big) \wedge \Big(\neg q \vee \neg r \vee (t \wedge s) \vee (q \wedge \neg t) \vee (r \wedge \neg s)\Big) \\
= \quad & \Big(q \vee r \vee ((s \vee \neg q) \wedge True)\Big) \wedge \Big(((\neg q \vee q) \wedge (q \vee \neg t)) \vee \neg r \vee (t \wedge s) \vee (r \wedge \neg s)\Big) \\
= \quad & \Big(q \vee r \vee s \vee \neg q\Big) \wedge \Big((True \wedge (q \vee \neg t)) \vee \neg r \vee (t \wedge s) \vee (r \wedge \neg s)\Big) \\
= \quad & \Big(r \vee s \vee True\Big) \wedge \Big(q \vee \neg t \vee \neg r \vee (t \wedge s) \vee (r \wedge \neg s)\Big) \\
= \quad & True \wedge \Big(q \vee \neg t \vee \neg r \vee (t \wedge s) \vee (r \wedge \neg s)\Big) \\
= \quad & True \wedge \Big(q \vee ((\neg t \vee t) \wedge (\neg t \vee s)) \vee \neg r \vee (r \wedge \neg s)\Big) \\
= \quad & True \wedge \Big(q \vee \neg t \vee s \vee \neg r \vee (r \wedge \neg s)\Big) \\
= \quad & True \wedge \Big(q \vee \neg t \vee s \vee ((\neg r \vee r) \wedge (\neg r \vee \neg s))\Big) \\
= \quad & True \wedge \Big(q \vee \neg t \vee s \vee \neg r \vee \neg s\Big) \\
= \quad & True \wedge \Big(q \vee \neg t \vee \neg r \vee True\Big) \\
= \quad & True \wedge True \\
= \quad & True
\end{aligned}
$$

4 Deterministic and Polynomial

You realize ALAN is limited in resources, not a perfect machine. In fact, he can only run deterministic polynomial algorithms.

4.1 Horn-Satisfiability (2pts)

Give a satisfying assignation of $x_1, ..., x_6$ for the following clauses :
$\overline{x_1} \vee x_2 \vee \overline{x_3}$
$\overline{x_2} \vee \overline{x_4} \vee \overline{x_5}$
x_4
$\overline{x_3} \vee \overline{x_4} \vee x_6$
$x_3 \vee x_5$
$\overline{x_1} \vee \overline{x_2} \vee x_3$
$\overline{x_1}$
$x_1 \vee \overline{x_5}$
$x_2 \vee \overline{x_6}$

Average score: 1.45 / 2
Solution :
Since there is at most one positive litteral in each clause, this SAT instance is in fact a Horn-SAT instance, for which there exists an efficient algorithm named unit propagation. Notice first that $x_1 = 0$ and $x_4 = 1$ are necessary conditions. All clauses containing the litteral $\overline{x_1}$ or x_4 are true and can therefore be totally erased, while clauses with x_1 or $\overline{x_4}$ cannot be satisfied by this litteral so we erase the litteral. This yields :
$\overline{x_2} \vee \overline{x_5}$
$\overline{x_3} \vee x_6$
$x_3 \vee x_5$
$\overline{x_5}$
$x_2 \vee \overline{x_6}$
Similar reasoning on these few clauses yields this final assignment :
$x_1 = 0$
$x_2 = 1$
$x_3 = 1$
$x_4 = 1$
$x_5 = 0$
$x_6 = 1$

4.2 Choices: Boolean algebra (3pts)

Only one of these statements about boolean algebra is false. Which one?

1. $(\mathbb{Z}_2, \oplus)$ is an abelian group.
2. $(\mathbb{Z}_2, \oplus, \vee)$ is a ring.
3. $(\mathbb{Z}_2, \oplus, \wedge)$ is a field.
4. $(\mathbb{Z}_2, \vee, \wedge)$ is a lattice.

Average score: 0.10 / 3
Solution:
$(\mathbb{Z}_2, \oplus, \vee)$ **is not a ring** because it does not have the distributivity property : $a \vee (b \oplus c) \neq (a \vee b) \oplus (a \vee c)$.

4.3 True or false: $\mathcal{O}(1)$ (2pts)

The following problem is solvable in $\mathcal{O}(1)$:

4-COLORING OF PLANAR GRAPHS	
Input :	Any *planar* graph (V, E)
Question :	Is there a function $f : V \to \{Red, Green, Blue, Yellow\}$ such that for each $(v_1, v_2) \in E$, $f(v_1) \neq f(v_2)$?

Average score: 0.24 / 2
Solution :
True. It is a well-known result of graph theory that planar graphs can *always* be 4-colored (it is always possible to color a world map with only four colors). Therefore, an algorithm for this problem can simply always *return True*.

5 I'm feeling lucky

One of ALAN's pieces of hardware begins to act very strangely and outputs gibberish. By routing these random bits to ALAN's CPU, you allow him to run probabilistic algorithms.

5.1 The Musical Box (5pts)

With his new device, ALAN emits random melodies in the key of A minor. Each note is taken randomly from the scale A, B, C, D, E, F, G according to the following probability table :

Note	Probability
A	0.30
B	0.08
C	0.15
D	0.12
E	0.20
F	0.05
G	0.10

1. At some point, what is the probability that the next 4 symbols give the melody ABCD?
2. At some point, what is the probability that, in the next 20 symbols, F appears at least once?
3. What is the entropy of the source in bits/note?
4. ALAN wants to share the his melody with the world by sending each note as a binary word. Help him by giving a **uniquely decodable binary code** for this source, then calculate the average length of your code. The closer the average length is to the entropy, the more points you score!

Average score: 1.98 / 5
Solution:

1. $P(A) \cdot P(B) \cdot P(C) \cdot P(D) = 0.30 \cdot 0.08 \cdot 0.15 \cdot 0.12 = 0.000432$
2. $1 - 0,95^{20} = 0,6415$. Approximation $1 - 1/e = 0.6321$ also accepted.
3. $\sum_{x \in Notes} -P(x) \log_2 P(x) = -0.30 \log_2 0.30 - 0.08 \log_2 0.08 - 0.15 \log_2 0.15 - 0.12 \log_2 0.12 - 0.20 \log_2 0.20 - 0.05 \log_2 0.05 - 0.10 \log_2 0.10 = 2.6029$
4. In that kind of situation, the Huffman algorithm gives an optimal code.
   - Order by probability:

Note group	Probability	Note	Code
A	0.30	A	
E	0.20	B	
C	0.15	C	
D	0.12	D	
G	0.10	E	
B	0.08	F	
F	0.05	G	

- Merge B with F:

Note group	Probability	Note	Code
A	0.30	A	
E	0.20	B	0
C	0.15	C	
B,F	0.13	D	
D	0.12	E	
G	0.10	F	1
		G	

- Merge D with G:

Note group	Probability	Note	Code
A	0.30	A	
D,G	0.22	B	0
E	0.20	C	
C	0.15	D	0
B,F	0.13	E	
		F	1
		G	1

- Merge C with B,F:

Note group	Probability	Note	Code
A	0.30	A	
B,C,F	0.28	B	10
D,G	0.22	C	0
E	0.20	D	0
		E	
		F	11
		G	1

- Merge D,G with E:

Note group	Probability	Note	Code
D,E,G	0.42	A	
A	0.30	B	10
B,C,F	0.28	C	0
		D	00
		E	1
		F	11
		G	01

- Merge A with B,C,F:

Note group	Probability	Note	Code
A,B,C,F	0.58	A	0
D,E,G	0.42	B	110
		C	10
		D	00
		E	1
		F	111
		G	01

- Merge A,B,C,F with D,E,G:

Note group	Probability	Note	Code
A,B,C,D,E,F,G	1.00	A	00
		B	0110
		C	010
		D	100
		E	11
		F	0111
		G	101

- Average length : $0.30 \cdot 2 + 0.08 \cdot 4 + 0.15 \cdot 3 + 0.12 \cdot 3 + 0.20 \cdot 2 + 0.05 \cdot 4 + 0.10 \cdot 3 = 2.63$ bits/note

5.2 Gambling (3pts)

ALAN knows its random bits allow him to run Las Vegas algorithms and Monte-Carlo algorithms. He then wants to put his new computing skills to good use : making money. Since he doesn't really understand what these algorithms really are, you must explain the difference between Las Vegas and Monte-Carlo algorithms. Give an example for each type.

Average score: 0.57 / 3
Solution:
Las Vegas algorithms always give the right result, but they gamble on the computation time. *Randomized quicksort* is an example : it always sorts the list the right way, but although the average runtime is in $\mathcal{O}(n \log n)$ it will sometimes run in $\mathcal{O}(n^2)$ if unlucky.
Monte Carlo algorithms will always run in a specific, fixed time, but their result is not guaranteed to be right. *Primality* algorithms are an example : they run a fixed set of tests on a number and outputs whether it passed or fail, but there is a chance that a composite number passes every test although it shouldn't.

6 Cryptobot

Incredible as it may sound, another computer on another planet, BELLE, heard one of ALAN's melody and thought it was beautiful. ALAN wants to chat with her, but doesn't want you to eavesdrop, so he begins to encrypt his communication.

6.1 Flawed Protocol (5pts)

Thankfully, ALAN was designed years ago and uses an old, flawed, protocol. In the following key exchange protocol,

- N_a and N_b are nonces,
- K_{ab} is the key to be shared by A and B,
- $Hash$ is a public cryptographic hash function,
- k_a and k_b are respectively A and B's public keys,
- k_a^{-1} and k_b^{-1} are respectively k_a and k_b's private counterparts,
- $\{x\}_{K_j}$ means that x was encrypted using K_j and thus can only be decrypted with K_j^{-1},
- $\{x\}_{K_j^{-1}}$ means that x was signed using K_j^{-1}.

A		**B**
Generate N_a randomly	$\rightarrow N_a \rightarrow$	Generate N_b randomly
	$\leftarrow N_b \leftarrow$	
Generate K_{ab} randomly		
Compute $h = Hash(N_a, N_b, K_{ab})$	$\rightarrow \{h\}_{K_a^{-1}}\{K_{ab}\}_{K_b} \rightarrow$	
		Decrypt K_{ab}
		Verify that $h = Hash(N_a, N_b, K_{ab})$

The protocol is vulnerable to a man-in-the-middle attack. Fill the following table to show how an intruder I (with public and private keys K_i and K_i^{-1}) can intercept K_{ab} without A and B noticing.
Hint : there should be two intertwined instances of the protocol : one between A and I, and one between I and B.

A		**I**		**B**

Average score: 1.17 / 5

Solution:

A		I		B
Generate N_a randomly	$N_a \longrightarrow$			
			$N_a \longrightarrow$	Generate N_b randomly
			$\longleftarrow N_b$	
	$\longleftarrow N_b$			
Generate K_{ab} randomly				
Compute $h = Hash(N_a, N_b, K_{ab})$	$\{h\}_{K_a^{-1}}\{K_{ab}\}_{K_i} \longrightarrow$	Decrypt K_{ab}		
			$\{h\}_{K_a^{-1}}\{K_{ab}\}_{K_b} \longrightarrow$	Decrypt K_{ab} Verify that $h = Hash(N_a, N_b, K_{ab})$

6.2 Breaking RSA (4pts)

ALAN realizes you are listening to his conversation and switches to the RSA cryptosystem, which goes like this :

- **Key generation :**
  - Choose two primes p and q then compute $n = pq$.
  - Choose an e such that $1 < e < \lambda(n)$ and $gcd(e, \lambda(n)) = 1$, where $\lambda(n) = lcm(p-1, q-1)$, lcm means *least common multiple* and gcd means *greatest common divisor*.
  - Public key : (n, e)
- **Encryption :** $c = m^e \mod n$, where c is the ciphertext and m the original message.
- **Decryption :** $m = c^d \mod n$, where $d = e^{-1} \mod \lambda(n)$.

This protocol is pretty secure, but you recall seeing in the spaceship a small, black-box looking computer, which could efficiently compute the prime factors of an arbitrarily long number. Maybe this could help?

Suppose the following *polynomial* algorithm exists :

FACTORIZE	
Input :	Any $n \in \mathbb{N} \setminus \{0, 1\}$
Output :	The list of prime numbers $p_1, p_2, ..., p_m$ such that $n = \prod_{i=1}^{m} p_i$

Show how a spy knowing (n, e), c and FACTORIZE can find m in polynomial time.
Hint : there exist polynomial algorithms to compute *greatest common divisors*, *least common multiples*, *modular inverses* and *modular exponentiations*.

Average score: 2.00 / 4
Solution:
The following algorithm finds m and runs in polynomial time assuming FACTORIZE is polynomial.

Algorithm Plaintext finder

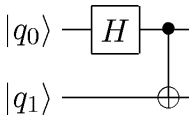
Input: $(n, e), c$
$p, q =$ FACTORIZE(n)
Compute $\lambda(n) = lcm(p-1, q-1)$
Compute $d = e^{-1} \mod \lambda(n)$
Return $c^d \mod n$

7 QuALANtum

The small computer you found was in fact a device capable of quantum computation. When ALAN realizes that, he takes it away from you and uses it to enhance his own computations and thus impress BELLE.

7.1 Belle State (5pts)

Consider the following simple quantum circuit :



- In the above circuit, $|q_0\rangle$ goes through the gate H, then both the result of $|q_0\rangle$ through H and the unaltered $|q_1\rangle$ go through the $CNOT$ gate, whose output is the output of the whole circuit.
- In the following definitions, if A and B are matrices, $\otimes$ denotes the tensor product :

$$A \otimes B = \begin{bmatrix} A_{11}B & \dots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \dots & A_{mn}B \end{bmatrix}$$

- $|q_0\rangle$ and $|q_1\rangle$ are two qubits :
  $|q_0\rangle = \alpha_0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta_0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \beta_0 \end{bmatrix}$, with $\alpha_0, \beta_0 \in \mathbb{C}$
  $|q_1\rangle = \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}$, with $\alpha_1, \beta_1 \in \mathbb{C}$
  Their composition is given by $|q_0\rangle \otimes |q_1\rangle = \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{bmatrix}$
- H denotes the Hadamard gate :

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

-  ($CNOT$) is the controlled-not gate :

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- The result of one qubit $|q\rangle$ going through a one-input gate G_1 is given by $G_1 \cdot |q\rangle$. Similarly, two qubits $|q_0\rangle$ and $|q_1\rangle$ going through a two-inputs gate G_2 yield $G_2 \cdot (|q_0\rangle \otimes |q_1\rangle)$.

Compute the output of the quantum circuit for the input $|1\rangle \otimes |0\rangle$, i.e. when $\alpha_1 = \beta_0 = 1$ and $\beta_1 = \alpha_0 = 0$.

Average score: 1.48 / 5

Solution:
We must compute $CNOT((H \otimes I_2)(|q_0\rangle \otimes |q_1\rangle))$, or more explicitly :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right)$$

which is equal to $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$ or, equivalently, $\frac{|00\rangle - |11\rangle}{\sqrt{2}}$.

7.2 Choices: Cryptography Problems (3pts)

Only one of these computational problems used in cryptography **cannot** be solved in polynomial time by a quantum computer. Which one?

1. Quadratic residuosity problem
2. Shortest vector problem
3. Discrete logarithm problem
4. Integer factorization

Average score: 0.72 / 3
Solution:
Shortest vector problem. All the others are number-theoretic problems which can be reduced to the Hidden Subgroup Problem for abelian groups, for which there exists a quantum polynomial algorithm.

8 Computational Secret

BELLE tells ALAN she will only be impressed if he can solve NP-complete problems.

8.1 Zero-Knowledge Proof (3pts)

ALAN claims he knows the solution to an instance of the 3-coloring problem.

3-COLORING	
Input :	Any graph (V, E)
Question :	Is there a function $f : V \to \{Red, Green, Blue\}$ such that for each $(v_1, v_2) \in E$, $f(v_1) \neq f(v_2)$?

Since this problem is NP-complete, BELLE is skeptical. ALAN wants to prove to her that he knows a function $f : V \to \{Red, Green, Blue\}$ such that for each $(v_1, v_2) \in E$, $f(v_1) \neq f(v_2)$, but *without handing her any bit of information about the solution.* To this end, they both agree to participate in the following **zero-knowledge proof** protocol :

- Repeat these steps N times :
- 1. ALAN randomly picks a **permutation** $p : \{Red, Green, Blue\} \to \{Red, Green, Blue\}$ and sends BELLE the set $C = \{c(p(f(v)))|v \in V\}$, where c represents a **commitment scheme** (see 8.2). In other words, he changes the color of each vertex following his permutation, then commits to the new color of each vertex.
  2. BELLE randomly picks an edge (v_i, v_j) of the graph, and asks ALAN to open his commitments $c(p(f(v_i)))$ and $c(p(f(v_j)))$.
  3. ALAN reveals $p(f(v_i))$ and $p(f(v_j))$. BELLE **rejects** if $p(f(v_i)) = p(f(v_j))$.
- BELLE **accepts**.

Suppose ALAN is cheating. In his graph, one of the edges is of the form (v_i, v_j), where $f(v_i) = f(v_j)$. Give the probability that ALAN fools BELLE as a function of the number of edges $|E|$ and the number of rounds N.

Average score: 0.52 / 3
Solution:
At each round, BELLE can find out the trickery with probability $\frac{1}{|E|}$, so ALAN fools her with probability $1 - \frac{1}{|E|}$. Each round is independant, so after N rounds the probability decreases to $(1 - \frac{1}{|E|})^N$.

8.2 Commitment Scheme (2pts)

Here is an example of a commitment scheme, where ALAN commits to some value n :

- Commitment phase : ALAN picks a random number x and computes $c = h(n||x)$, where h is a cryptographic hash function and $||$ denotes concatenation. He sends c to BELLE.
- Reveal phase : ALAN hands n and x to BELLE, who then checks that $c = h(n||x)$.

Assuming it is too hard to invert h or find a collision for h, this scheme is both **hiding** and **binding**. Explain the meaning of those properties and how they don't hold if h is not a good cryptographic hash function.

Average score: 0.74 / 2
Solution:
Hiding means that BELLE cannot find n knowing only the commitment c. But if h can be inverted, she can compute $n||x$ from $h(n||x)$.
Binding means that ALAN cannot change his n after the commitment phase. But if it is easy to find collisions for h, he might be able to find some n', x' such that $h(n||x) = h(n'||x')$.

8.3 An Insecure Pseudo-Random Number Generator (5pts)

Let a linear congruential generator be defined by the following recurrence relation :

$$x_n = (4x_{n-1} + 3) \mod 11,$$

with x_0 as the secret seed.

1. Find the general term of the recurrence, that is, give x_n as a function of x_0 and n. Simplify as much as possible. Hints: recall that $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$, and you don't have to make new modulos appear at each step.

2. Suppose you receive the 333^{th} generated number and its value is $x_{333} = 3$. Compute the seed x_0.

Average score: 2.03 / 5
Solution:

1.

$$\begin{aligned}
x_n &= 4x_{n-1} + 3 \mod 11 \\
&= 4(4x_{n-2} + 3) + 3 \mod 11 \\
&= 4^2 x_{n-2} + 4 \cdot 3 + 3 \mod 11 \\
&= \ldots \\
&= 4^i x_{n-i} + \sum_{j=0}^{i-1} 4^j \cdot 3 \mod 11
\end{aligned}$$

$\langle \text{Let } i = n \rangle$

$$\begin{aligned}
x_n &= (4^n x_0 + 3 \sum_{j=0}^{n-1} 4^j \mod 11 \\
&= 4^n x_0 + 3(\sum_{j=0}^{n} 4^j - 4^n) \mod 11 \\
&= 4^n x_0 + 3 \sum_{j=0}^{n} 4^j - 3 \cdot 4^n \mod 11 \\
&= 4^n x_0 + 3 \frac{4^{n+1} - 1}{4 - 1} - 3 \cdot 4^n \mod 11 \\
&= 4^n x_0 + 4^{n+1} - 1 - 3 \cdot 4^n \mod 11 \\
&= 4^n (x_0 + 4 - 3) - 1 \mod 11 \\
&= 4^n (x_0 + 1) - 1 \mod 11
\end{aligned}$$

2.

$$
\begin{aligned}
3 &= 4^{333}(x_0+1)-1 \mod 11 \\
&= (4^{333} \mod 11)(x_0+1)-1 \mod 11 \\
&\langle \text{Modulo 11, 4 has a period of 5} \rangle \\
&= (4^{333 \mod 5} \mod 11)(x_0+1)-1 \mod 11 \\
&= (4^3 \mod 11)(x_0+1)-1 \mod 11 \\
&= (64 \mod 11)(x_0+1)-1 \mod 11 \\
&= 9(x_0+1)-1 \mod 11 \\
4 &= 9(x_0+1) \mod 11 \\
&\langle 5 = 9^{-1} \mod 11 \rangle \\
5 \cdot 4 &= x_0+1 \mod 11 \\
20 &= x_0+1 \mod 11 \\
9 &= x_0+1 \mod 11 \\
8 &= x_0 \mod 11
\end{aligned}
$$

8.4 ZKP Properties (2pts)

A zero-knowledge proof protocol has to satisfy three properties :

- Completeness : if ALAN is honest then BELLE will always accept
- Soundness : if ALAN is malicious then BELLE will reject with overwhelming probability
- Zero-knowledge : BELLE gains no knowledge whatsoever about ALAN's solution, apart from the fact that he knows a solution

1. Suppose BELLE's random number generator is insecure; that is, ALAN can always guess in advance BELLE's random choices. Give which of the three properties he can violate and how.
2. Now suppose it is ALAN's random number generator which is insecure. Give which of the three properties BELLE can violate and how.

Average score: 0.83 / 2
Solution:

1. ALAN knows which edge BELLE is going to ask for, so he puts different colors on the two vertices of this edge. The other vertices can be litterally anything, because BELLE won't ask for their colors. This violates the *soundness* property.
2. BELLE knows which permutation of the colors ALAN chooses, so she can always compute the inverse permutation of the colors of the edge he asked for. She can then cumulate the original colors and progressively build ALAN's solution. Clearly, she learns the solution, therefore it violates the *Zero-Knowledge* property.

8.5 True or False: One-way Functions (2pts)

If P=NP, one-way functions do not exist.
Hint: A function f is one-way when computing $f(x)$ is in P but computing $f^{-1}(x)$ is not.

Average score: 0.69 / 2
Solution:
True. Computing $f^{-1}(x)$ is in NP since it is checkable in polynomial time by running $f(x)$. If P = NP then $f^{-1}(x)$ is in P, yielding a contradiction with the definition of one-way functions.

9 Fool of a Proof

You suspect ALAN cheated during the last protocol and never actually knew the solution to the 3-coloring problem. Truth is, he actually found out that $\mathsf{P} = \mathsf{NP}$ and agrees to show you the 3-steps proof :

Subset Sum	
Input :	A sequence $\langle w_1, w_2, ...w_n \rangle$ with $w_i \in \mathbb{N}$, and $s \in \mathbb{N}$.
Question :	Does there exist a subset I of $\{1, 2, ..., n\}$ such that $\sum_{i \in I} w_i = s$?

- The Subset Sum problem is NP-hard.
  Proof : We reduce from 3-SAT.
  - Let the instance of 3-SAT be n variables x_i and m clauses c_j.
  - For each variable x_i, construct numbers t_i and f_i of $n + m$ digits :
    - The i-th digit of t_i and f_i is equal to 1
    - For $n + 1 \leq j \leq n + m$, the j-th digit of t_i is 1 if $x_i \in c_{j-n}$.
    - For $n + 1 \leq j \leq n + m$, the j-th digit of f_i is 1 if $\overline{x_i} \in c_{j-n}$.
    - All other digits of t_i and f_i are 0.
  - For each clause c_j, construct numbers x_j and y_j of $n + m$ digits :
    - The $(n + j)$-th digit of x_j and y_j is 1.
    - All other digits of x_j and y_j are 0.
  - Finally, construct a sum number s of $n + m$ digits :
    - For $1 \leq j \leq n$, the j-th digit of s is 1.
    - For $n + 1 \leq j \leq n + m$, the j-th digit of s is 3.
  - **Example :** The 3-Sat instance

    $$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

    results in the Subset Sum instance

    $$(\langle t_1, f_1, t_2, f_2, t_3, f_3, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \rangle, s)$$

    with values taken from the following table :

	i			j			
Number	1	2	3	1	2	3	4
t_1	1	0	0	1	0	0	1
f_1	1	0	0	0	1	1	0
t_2	0	1	0	1	0	1	0
f_2	0	1	0	0	1	0	1
t_3	0	0	1	1	1	0	1
f_3	0	0	1	0	0	1	0
x_1	0	0	0	1	0	0	0
y_1	0	0	0	1	0	0	0
x_2	0	0	0	0	1	0	0
y_2	0	0	0	0	1	0	0
x_3	0	0	0	0	0	1	0
y_3	0	0	0	0	0	1	0
x_4	0	0	0	0	0	0	1
y_4	0	0	0	0	0	0	1
s	1	1	1	3	3	3	3

- The Subset Sum problem is solvable in polynomial time and is therefore in P.
  Proof : There is a dynamic programming algorithm for this problem, which runs in $\mathcal{O}(ns)$:

- Let M be a 2-dimensional table of booleans defined by the following recurrence :

$$M[i,w] = \begin{cases} 0 & \text{when } i < 0 \vee j < 0 \\ 1 & \text{when } i = j = 0 \\ M[i-1,w] \vee M[i-1,w-w_i] & \text{otherwise} \end{cases}$$

- Intuitively, $M[i,w]$ is true when you can find a subset I of $\{w_0, ..., w_i\}$ that sums to w.
- The algorithm is now pretty straightforward :

Algorithm SUBSET SUM solver

- **Input:** $\langle w_1, w_2, ...w_n \rangle$ with $w_i \in \mathbb{N}$, and $s \in \mathbb{N}$
  Initialize each entry of M to 0
  Set $M[0,0] = 1$
  for $i \leftarrow 1$ to n **do**
    for $w \leftarrow 0$ to s **do**
      $M[i,w] = M[i-1,w] \vee M[i-1,w-w_i]$
    end for
  end for
  Return $M[n,s]$

- Since SUBSET SUM is NP-hard and SUBSET SUM $\in$ P, then NP $\subseteq$ P. Also, since a deterministic algorithm always implies a non-deterministic one, we trivially have that P $\subseteq$ NP. Hence, P = NP.

Q.E.D.

Wait... that can't be right, can it?

9.1 Reduction (4pts)

Show that the polynomial reduction 3-SAT $\leq$ SUBSET SUM is correct, that is :

- if the original 3-SAT instance is satisfiable, then there exists a subset whose sum is s in the generated SUBSET SUM instance.
- if there exists a subset whose sum is s in the generated SUBSET SUM instance, then the original 3-SAT instance is satisfiable.
- the reduction runs in polynomial time.

Average score: 0.55 / 4 **Solution :**

- $\Rightarrow$:
  - Take t_i if x_i is true
  - Take f_i if x_i is false
  - Take x_j if the number of true literals in c_j is at most 2
  - Take y_j if the number of true literals in c_j is 1
  - One and only one of t_i and f_i is in the subset, so the first n digits each sum to one.
  - Since the instance is satisfiable, each clause is associated with at least one t_i of f_i which is in the subset, so the digit of a clause sum to at least 1 and can sum to 3 by having some x_j and y_j in the subset.
- $\Leftarrow$:

- Assign value true to x_i if t_i is in subset
- Assign value false to x_i if f_i is in subset
- Exactly one number per variable must be in the subset, otherwise one of first n digits of the sum is greater than 1
- At least one variable number corresponding to a literal in a clause must be in the subset, otherwise one of next m digits of the sum is smaller than 3
- Each clause is satisfied

- The reduction algorithm iterates over all variables and all clauses, then over all clauses; it is in $\mathcal{O}(nm+m)$, which is polynomial.

9.2 Induction (5pts)

Use mathematical induction to prove that the suggested dynamic programming indeed solves any Subset Sum instance.

Average score: 0.90 / 5 **Solution:** The induction is done on the well-ordered set

$$(0,0),(0,1),...,(0,s),(1,0),(1,1),...,(1,s),...,(n,s)$$

The predicate is $P(i,w)$: $M[i,w]$ contains the right solution (true or false) to the (i,w) sub problem.

- Base case : $(0,0)$
  The algorithm sets $M[0,0]$ to 1, and it is indeed true, because the sum 0 can always be obtained by not taking any element of the set.
- Induction step : (i,w)
  Suppose *every* previous answer of the algorithm is right. Then both $M[i-1,w]$ and $M[i-1,w-w_i]$ contain the correct answer.
  - If $M[i-1,w]=1$ then by not taking the ith number we still have a sum of w, so $M[i,w] = 1 \vee M[i-1,w-w_i]=1$ is correct.
  - If $M[i-1,w-w_i]=1$ then by taking the ith number we will have a sum of exactly w, so $M[i,w]=1$ is correct.
  - If $M[i-1,w]=M[i-1,w-w_i]=0$ then there is no way the ith number can make the sum equal to w, so $M[i,w]=0\vee 0=0$ is correct.

9.3 Flaw in the Proof (3pts)

Something *has* to be wrong with this proof. What is it?

Average score: 0.31 / 3
Solution:
In the reduction, it is not the sum s, but the number of digits of s which is polynomial in the number of clauses and variables. So the value of s can be, in the worst case, $2^{p(n)}$ where p is a polynomial function. The algorithm therefore runs in $\mathcal{O}(ns)=\mathcal{O}(n2^{p(n)})$, and so it does not imply that Subset Sum is in P.

9.4 Choices: Not NP-complete (3pts)

Only one of these graph problems is not known to be NP-complete. Which one?

1. Graph isomorphism problem
2. Longest path problem
3. 3-coloring

4. Clique problem

Average score: -0.03 / 3
Solution:
The **Graph isomorphism problem** is not known nor believed to be NP-complete, although it is not known to be in P either.

10 Almighty ALAN

While you were busy finding the flaw in ALAN's proof, ALAN found a way to have infinite resources. He claims he can now compute any function in finite time. Instead of using this power to solve important problems, though, he prefers to play his favorite game : the Busy Beaver (see section 3.1).

10.1 True or False: Computability (2pts)

The busy beaver function $BB(n)$, which returns the maximal number of 1s writable without halting with an n-state Turing machine, is computable on a Turing machine for all n.

Average score: 0.07 / 2
Solution:
False. $BB(n)$ grows faster than any computable function, and thus is uncomputable. If we could compute $BB(n)$, then we would be able to know if any Turing Machine halts by checking if it takes more or less computation steps than for $BB(n)$.

10.2 Countable Set (4pts)

Since the set of functions is uncountably infinite, you doubt it is possible to compute any function. Show that the set of Turing machines is countable by giving an injective function from *TURING_MACHINES* to $\mathbb{N}$. For simplicity, assume the input (Σ) and tape (Γ) alphabets are always $\{0, 1\}$. Recall :

$$TURING\_MACHINES = \{(Q, \Gamma, b, \Sigma, \delta, q_0, F)|$$

$$Q \text{ is a finite, non-empty set of states,}$$

$$\Gamma = \{0, 1\},$$

$$b = 0 \text{ is the blank symbol,}$$

$$\Sigma = \{0, 1\},$$

$$q_0 \in Q \text{ is the initial state,}$$

$$F \subseteq Q \text{ is the set of accepting states,}$$

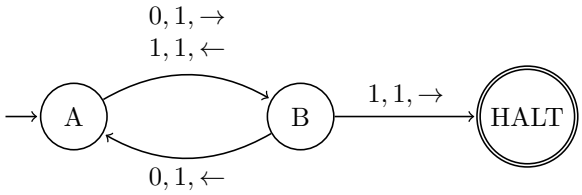
$$\delta \ : (Q \setminus F) \times \Gamma \longrightarrow Q \times \Gamma \times \{\rightarrow, \leftarrow\}\}$$

Hint : Come up with different numbers for each important element of the tuple, then concatenate these numbers with delimiters.

Average score: 0.55 / 4
Solution:
There are many ways to achieve such a function. Here's a possibility :

- For each state :
  - Write 2 if it's the starting state, 3 if it's a simple state, and 4 if it's an accepting state.
  - Write in binary the number of the state to go to when reading 0. Write a 5 as a delimiter. Write 0 if the head must go left and 1 if it must go right. Write a 6 as a delimiter. If the transition does not exist, write only a 6 at this step.
  - Write in binary the number of the state to go to when reading 1. Write a 5 as a delimiter. Write 0 if the head must go left and 1 if it must go right. Write a 6 as a delimiter. If the transition does not exist, write only a 6 at this step.

For instance, the following Turing machine if mapped to the number 21516150630506105164 66 $\in \mathbb{N}$.



10.3 Hypercomputation (5pts)

You have come up with an explanation on how ALAN may compute uncomputable functions : he must have become a Zeno machine. A Zeno machine is a Turing machine, except that it takes 2^{-i} units of time to execute its ith step, where $i \in \mathbb{N}$. Write an algorithm that runs on a Zeno machine and solves the halting problem in finite time.

Halting Problem	
Input :	A Turing machine M and an input w for M
Question :	Does M halt on input w?

Average score: 0.54 / 5
Solution:
Since $\sum_{i=0}^{\infty} \frac{1}{2^i} = 2$, the following algorithm takes 2 units of time in the worst case.

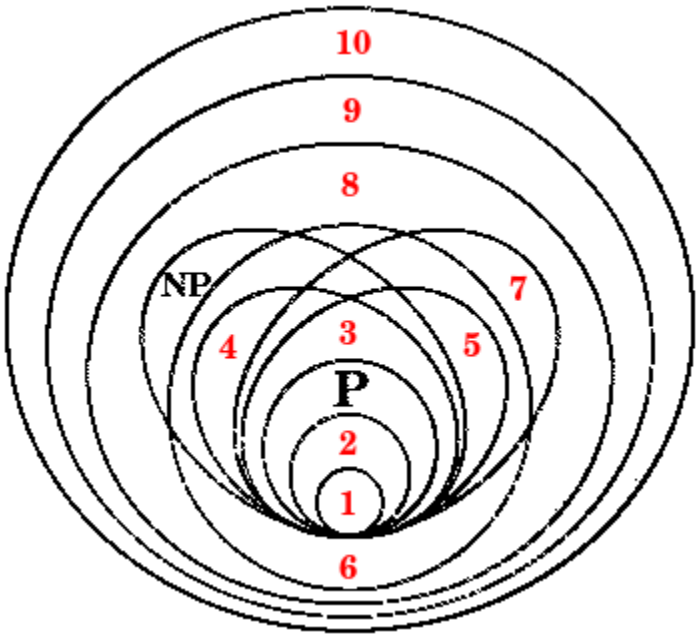
Algorithm Halting Problem solver

```
Input: M, w
Write false on the output tape.
for i = 0 to ∞ do
    Run the ith step of M on w
    if M halts then
        Write true on the output tape.
        Break out of the loop
    end if
end for
```

11 Final boss

ALAN has become way too powerful and must be shut down. You realize its eye must be its only weak point. By looking at the pattern in its eye, you have a flashback of your computational complexity course. In a last hope to save humanity from artificial intelligence, you strike it with complexity classes.

11.1 The Eye of Complexity (5pts)



Here is a list of some important complexity classes along with their definition. You are asked to place them at the right place in the above diagram by assigning them the right number.

Class	Definition	Number
BPP	Solvable in polynomial time by randomized algorithms (Answer is right with probability at least 2/3)	
coNP	"NO" answers checkable in polynomial time	
coRP	Solvable in polynomial time by randomized algorithms ("YES" answer is right with probability at least 1/2, "NO" is certainly right)	
EXP	Solvable in exponential time	
EXPSPACE	Solvable with exponential space	
L	Solvable with logarithmic space	
NL	"YES" answers checkable with logarithmic space	
NP	"YES" answers checkable in polynomial time	-
P	Solvable in polynomial time	-
PSPACE	Solvable with polynomial space	
RP	Solvable in polynomial time by randomized algorithms ("NO" answer is right with probability at least 1/2, "YES" is certainly right)	
ZPP	Solvable by randomized algorithms (answer is always right, average running time is polynomial)	

Average score: 1.71 / 5
Solution:

Class	Definition	Number
BPP	Solvable in polynomial time by randomized algorithms (Answer is right with probability at least 2/3)	6
coNP	"NO" answers checkable in polynomial time	7
coRP	Solvable in polynomial time by randomized algorithms ("YES" answer is right with probability at least 1/2, "NO" is certainly right)	5
EXP	Solvable in exponential time	9
EXPSPACE	Solvable with exponential space	10
L	Solvable with logarithmic space	1
NL	"YES" answers checkable with logarithmic space	2
NP	"YES" answers checkable in polynomial time	-
P	Solvable in polynomial time	-
PSPACE	Solvable with polynomial space	8
RP	Solvable in polynomial time by randomized algorithms ("NO" answer is right with probability at least 1/2, "YES" is certainly right)	4
ZPP	Solvable by randomized algorithms (answer is always right, average running time is polynomial)	3