Theory of Computation, Spring 2026
Solutions to Test 3

Problem 1. Explain where and how, in the analysis of how much parallelization can speed up computations, we use the two physical assumptions: that all speeds are bounded by the speed of light c, and that the volume of a sphere is proportional to the cube of its radius.

Solution. These two assumptions are used to prove that for each parallel computation that takes time $T_{\rm par}$, the number N of processors involved in this computations cannot exceed $C \cdot T_{\rm par}^3$ for some constant C. Indeed, the all processors that can affect our location at moment $\le T_{\rm par}$ are located inside a sphere of radius $R = c \cdot T_{\rm par}$ with center in this cell. The volume of this sphere is $V = (4/3) \cdot \pi \cdot R^3 = (4/3) \cdot \pi \cdot c^3 \cdot T_{\rm par}^3$, so within this sphere, we can have no more than $V/\Delta V$ processors, where ΔV is the volume of the smallest processor. Thus, $N \le C \cdot T_{\rm par}^3$.

We can simulate N-processor computations on a sequential computer by first performing the first step of all N processors, then the second step of all N processors, etc. The overall time increases by N, so the overall computation time on a sequential processor is $N \cdot T_{\rm par}$. Here, $N \le C \cdot T_{\rm par}^3$, so the overall sequential computation time is $\le C \cdot T_{\rm par}^3 \cdot T_{\rm par} = C \cdot T_{\rm par}^4$. Thus, the smallest possible sequential computation time $T_{\rm seq}$ cannot exceed $C \cdot T_{\rm par}^4$, where $T_{\rm par}$ is the time for parallel computations.

Problem 2. Use the general algorithm to translate the formula

$$(p \vee q \vee r \vee \neg s) \,\&\, (s \vee t)$$

into 3-CNF.

Solution: According to the general algorithm, we can replace the first clause with the following two clauses by introducing an auxiliary variable x:

$$(p \vee q \vee x) \,\&\, (\neg x \vee r \vee \neg s).$$

Thus, the whole formula takes the following 3-CNF f0orm:

$$(p \vee q \vee x) \,\&\, (\neg x \vee r \vee \neg s) \,\&\, (s \vee t).$$

Problem 3–4. Reduce the satisfiability problem for the formula

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

to:

- 3-coloring,
- clique,
- subset sum problem, and
- interval computations.

In all these reductions, explain what will correspond to $a = T$, $b = F$, and $c = T$.

Solution: For *3-coloring*, we do the following:

- We start with the palette, i.e., with vertices T, F, and U which are connected to each other.
- We form vertices a and $\neg a$, and we connect them to each other and to U.
- We form vertices b and $\neg b$, and we connect them to each other and to U.
- We form vertices c and $\neg c$, and we connect them to each other and to U.
- We form a new vertex $\neg a \vee \neg b$ and connect it to U.
- We form new vertices $(\neg a \vee \neg b)_1$ and c_1, connect them to each other and to T, connect $(\neg a \vee \neg b)_1$ to $\neg a \vee \neg b$, and connect c_1 to c.
- We form new vertices $\neg a_1$ and $\neg b_1$, connect them to each other and to $\neg a \vee \neg b$, connect $\neg a_1$ to $\neg a$, and connect $\neg b_1$ to $\neg b$.
- We form new vertices a_2 and $\neg b_2$, connect them to each other and to T, connect a_2 to a, and connect $\neg b_2$ to $\neg b$.

When $a = T$, $b = F$, and $c = T$, we can color the vertices in the following colors:

- The vertex T is colored T, the vertex F is colored F, and the vertex U is colored with color U.
- a is colored with color T, $\neg a$ is colored with color F.
- b is colored with color F, $\neg b$ is colored with color T.
- c is colored with color T, $\neg c$ is colored with color F.
- The vertex $\neg a \vee \neg b$ is colored with color T.
- For vertices $(\neg a \vee \neg b)_1$ and c_1 we have a choice: one of them is colored F, another one is colored U.

- the vertex $\neg a_1$ is colored F, the vertex $\neg b_1$ is colored U.
- For the vertices a_2 and $\neg b_2$, we also have a choice: one of them is colored F, another one is colored U.

One can check that in this case, no two connected vertices have the same color.

For *clique*, we add the following vertices: $\neg a_1$, $\neg b_1$, c_1, a_2, and $\neg b_2$. We then:

- connect $\neg a_1$ with $\neg b_2$ (but not with a_2;
- connect $\neg b_1$ with a_2 and $\neg b_2$; and
- connect $\neg c_1$ with a_2 and with $\neg b_2$.

The values $a = T$, $b = F$, and $c = T$ correspond to the four possible 2-cliques:

- $\neg b_1$ and a_2,
- $\neg b_1$ and $\neg b_2$,
- c_1 and a_2, and
- c_1 and $\neg b_2$.

For *subset sum*, by applying the general algorithm, we get the following table:

	a	b	c	C_1	C_2
a	1	0	0	0	1
$\neg a$	1	0	0	1	0
b	0	1	0	0	0
$\neg b$	0	1	0	1	1
c	0	0	1	1	0
$\neg c$	0	0	1	0	0
C_1'	0	0	0	1	0
C_1''	0	0	0	1	0
C_2'	0	0	0	0	1
C_2''	0	0	0	0	1
S	1	1	1	3	3

The values $a = T$, $b = F$, and $c = T$ correspond to selecting coins $a = 10001$, $\neg b = 01011$, and $c = 00110$. To get the desired sum 11133, we also need to add coins $C_1' = 00010$, $C_2' = 00010$, and $C_2'' = 00001$.

For *interval computations*, we get the polynomial

$$(1 - A \cdot B \cdot (1 - C)) \cdot (1 - (1 - A) \cdot B),$$

where A, B, and C are from the interval $[0, 1]$. This polynomial attains the value 1 when $A = 1$, $B = 0$, and $C = 1$, so its largest possible value is 1.

Problem 5. Show how to compute the minimum of 9 integer values in parallel if we have an unlimited number of processors and we can ignore communication time. Why do we need parallel processing in the first place? If we take communication time into account, how much time do we need to compute the minimum n values? What is NC? Give an example of a P-complete problem.

Solution:

- At moment $t = 1$, Processor 1 computes $r_1 = \min(x_1, x_2)$, Processor 2 computes $r_2 = \min(x_3, x_4)$, Processor 3 computes $r_3 = \min(x_5, x_6)$, and Processor 4 computes $r_4 = \min(x_7, x_8)$.
- At moment $t = 2$, Processor 1 computes $r_5 = \min(r_1, r_2)$, and Processor 2 computes $r_6 = \min(r_3, r_4)$. As a result, we get $r_5 = \min(x_1, x_2, x_3, x_4)$ and $r_6 = \min(x_5, x_6, x_7, x_8)$.
- At moment $t = 3$, Processor 1 computes

$$r_7 = \min(r_5, r_6) = \min(x_1, \ldots, x_8).$$

- At moment $t = 4$, Processor 1 computes the desired result $\min(r_7, x_9)$. One can easily check that this value is equal to $\min(x_1, \ldots, x_9)$.

For this computation, we need 4 processors and 4 moments of time.

For computing the the minimum of n values, sequential computations require at least $T_{\text{sequential}}(n) \geq n - 1$ steps. Since $T_{\text{parallel}}(n) \geq \text{const} \cdot (T_{\text{sequential}}(n))^{1/4}$, we thus need $T_{\text{parallel}}(n) \geq c \cdot (n-1)^{1/4}$.

NC is the class of all problems that can be computed on polynomial number of processors ($N_{\text{processors}} \leq P(n)$ for some polynomial $P(n)$ of the length n of the input) in polylog time, i.e., in time bounded by $P(\log(n))$ for some polynomial $P(n)$.

An example of P-complete problem is *linear programming*: checking whether a given set of linear inequalities $a_{i1} \cdot x_1 + \ldots + a_{in} \cdot x_n \geq b_i$, where a_{ij} and b_i are known, and x_j are unknowns, has a solution.

Problem 6. What can you say about the Kolmogorov complexity of the following string: 01100110... in which 0110 is repeated 1,000 times.

Solution: A possible program for producing this sequence x is as follows:

```
for(int i = 1; i <= 1000; i++)
     System.out.print('0110');
```

This program has $30 + 25 = 55$ symbols, so the Kolmogorov complexity $K(x)$ of this string – which is the length of the shortest program for computing this string – is smaller than or equal to 55: $K(x) \leq 55$.

Problem 7. Suppose that we have a probabilistic algorithm that gives a correct answer 3/4 of the time. How many times do we need to repeat this algorithm to reduce probability of error to at most 5%? Give an example of a probabilistic algorithm. Explain why we need probabilistic algorithms in the first place.

Solution: The probability of error is $1-3/4 = 1/4$. We want the probability of error to be smaller than $5\% = 1/20$. So, we need to select the smallest number of iterations k for which

$$\frac{1}{4^k} \le \frac{1}{20},$$

i.e., equivalently, for which $4^k \ge 20$.

- For $k = 1$, we get $4^1 = 4 < 20$.
- For $k = 2$, we get $4^2 = 16 < 20$.
- For $k = 3$, we get $4^3 = 64 > 20$.

Thus, we need 3 iterations.

An example of a probabilistic algorithm is an algorithm for checking program correctness. If someone proposes a program $f(x)$ that, given a real number $x \in [0, 1]$, supposed to compute the desired expression $g(x)$, then a natural way to check whether the program is correct is run a random number generator several times and check that for all the resulting random values $r_1, \ldots, r_k$, we get $f(r_i) = g(r_i)$.

We need probabilistic algorithms since many practical problems are NP-complete, which means that we cannot have a feasible algorithm that always produced a correct answer. It is thus reasonable to try to find a feasible algorithm that produces a correct answer with some probability.

Problem 8. Use the variable-elimination algorithm for checking satisfiability of the following 2-SAT formula:

$$(\neg a \vee \neg c) \,\&\, (\neg b \vee c) \,\&\, (a \vee \neg b) \,\&\, (\neg b \vee \neg c).$$

Find all solutions. Start by eliminating a, then, if needed, eliminate b.

Solution: Let us first eliminate the variable a. Clauses containing a or $\neg a$ lead to the following inequalities:

$$b \le a \le \neg c.$$

The condition that every lower bound must be smaller than or equal than every upper bound leads to

$$b \le \neg c.$$

This is equivalent to the clause $\neg b \vee \neg c$. So, for remaining variables b and c, we have the following clauses:

$$(\neg b \vee c) \,\&\, (\neg b \vee \neg c) \,\&\, (\neg b \vee \neg c).$$

Let us now eliminate the variables b. Clauses containing b or $\neg b$ lead to the following inequalities:

$$b \le \neg c, b \le c.$$

Here, we do not have any lower bounds, so we do not have any restrictions on c. Let us consider both cases $c = 0$ and $c = 1$.

If $c = 0$, then from $b \le c = 0$ and $b \le \neg c = 1$, we conclude that $b = 0$. In this case, the inequality $b \le a \le \neg c$ leads to $0 \le a \le 1$, i.e., we can have both $a = 0$ and $a = 1$. Thus, in this case, we have three satisfying tuples:

$$(a, b, c) = (0, 0, 0) \text{ and } (a, b, c) = (1, 0, 0).$$

If $c = 1$, then from $b \le \neg c = 0$ and $b \le c = 1$, we conclude that $b = 0$. In this case, the inequality $b \le a \le \neg c$ leads to $0 \le a \le 0$, i.e., to $a = 0$. Thus, in this case, we have only one satisfying triple:

$$(a, b, c) = (0, 0, 1).$$

So, overall, we have three satisfying tuples:

$$(a, b, c) = (0, 0, 0), \quad (a, b, c) = (1, 0, 0), \text{ and } (a, b, c) = (0, 0, 1).$$

Problem 9. How is the NOR operation $f(x_1, x_2) = \neg(x_1 \vee x_2)$ represented in quantum computing? Provide a general formula and explain it on the example when x_1 is true and both x_2 and the auxiliary variable are false.

Solution: the general way to represent a function $f(x_1, \ldots, x_n)$ is as a mapping

$$|x_1, \ldots, x_n, y\rangle \to |x_1, \ldots, x_n, y \oplus f(x_1, \ldots, x_n)\rangle.$$

In our case, $n = 2$ and $f(x_1, x_2) = \neg(x_1 \vee x_2)$, so we have

$$|x_1, x_2, y\rangle \to |x_1, x_2, y \oplus \neg(x_1 \vee x_2)\rangle.$$

In particular, when $x_1 = 1$ and $x_2 = y = 0$, we get

$$|1, 0, 0\rangle \to |1, 0, 0 \oplus \neg(1 \vee 0)\rangle = |1, 0, 0 \oplus \neg(1)\rangle = |1, 0, 0 \oplus 0\rangle = |1, 0, 0\rangle.$$

Problem 10. Why do we need to study recursively enumerable (r.e.) sets? Is the union of three r.e. sets still r.e.? If yes, prove it, if no, provide a counterexample.

Solution. In most of the class, we used computing intuition, but in some cases, it is useful to also use set-theoretic intuition, e.g., the intuitive notions of union and intersection.

If we have three r.e. sets A, B, and C, this means that we have three algorithms that will eventually generate all the elements of each set and only then. To get the union:

- first, we run all three algorithms for 1 hour;
- then, we run all three algorithms for 1 more hour;
- etc.