NATIONAL UNIVERSITY OF SINGAPORE

# CS 5230: Computational Complexity

Semester 2; AY 2023/2024; Midterm Test

Time Allowed: 60 Minutes

---

## INSTRUCTIONS TO CANDIDATES

1. Please write your Student Number. Do not write your name.
2. This assessment paper consists of FOUR (4) questions and comprises NINE (9) printed pages.
3. Students are required to answer **ALL** questions.
4. Students should answer the questions in the space provided.
5. This is a **CLOSED BOOK** assessment with one helpsheet.
6. You are not permitted to communicate with other people during the exam and you are not allowed to use additional material beyond the helpsheet.
7. Every question is worth FIVE (5) marks. The maximum possible marks are 20.

STUDENT NO: ______________________________

---

This portion is for examiner's use only

| Question | Marks | Remarks |
|---|---|---|
| Question 1: | | |
| Question 2: | | |
| Question 3: | | |
| Question 4: | | |
| Total: | | |

**Question 1 [5 marks]** 

Let $F(x) = (x+1)^3$ modulo $x^2$ and $G(x) = 2^x$ (plain number, no modulo). Here the input $x$ is an integer with $x > 0$. How much computation time needs the computation in the worst case, here the computation time is measured in the size $n$ of the input and $size(x) = \min\{n \geq 1 : x \leq 2^n\}$.

- $F$: ☐ Constant Time; ☐ Linear Time; ☐ Polynomial Time; ☐ Exponential Time or more;
- $G$: ☐ Constant Time; ☐ Linear Time; ☐ Polynomial Time; ☐ Exponential Time or more.

For both, $F$ and $G$ make a choice and provide an Addition Machine program witnessing this complexity class; the choice should be the best possible time complexity. If better classes do not apply, explain in few lines why.

For info: Addition machines can do the following operation in $O(1)$ as by the machine model: Adding, Subtracting, Comparing $(=, <, >)$, Assignment, Goto, Conditional Goto, Read, Write. A sample program computing the logarithm:

```
Line 1: Read x;
Line 2: y = 1; z=0;
Line 3: If y >= x Then Goto Line 7;
Line 4: y = y+y;
Line 5: z = z+1;
Line 6: Goto Line 3;
Line 7: Write z;
Line 8: End.
```

This sample program runs in time linear of the input size, as the program just computes the logarithm, that is, the first $z$ with $2^z \geq x$. The value of $y$ is $2^z$ throughout the program.

Write down your programs and comments starting from here.

**Solution.** As $(x+1)^3 = x^3 + 3x^2 + 3x + 1$, the result modulo $x^2$ is $3x + 1$. This result can be computed by an addition machine in constant time as witnessed by the following program:

```
Line 1: Read x;
Line 2: y = x+x;
Line 3: y = y+x;
Line 4: y = y+1;
Line 5: If x = 3 then y = 1;
Line 6: If x = 2 then y = 3;
Line 7: If x = 1 then y = 0;
Line 8: Write y;
Line 9: End.
```

The case distinction at the end is to capture the cases where $3x+1 \geq x^2$. They need an adjustment of the result; for a constant time program, it must be made this way, as computing the square takes linear time and is thus not possible.

For the second program, one needs exponential time, as one needs to double up a number from at most logarithmic size to size $x$ and each addition or subtraction can increase the size (number of binary digits) of the largest register or constant in the program by at most 1. As the output is a number of exponential size, the run time is at least exponential. The following program does this:

```
Line 1: Read x;
Line 2: y = 1;
Line 3: If x < 1 Then Goto Line 7;
Line 4: y = y+y;
Line 5: x = x-1;
Line 6: Goto Line 3;
Line 7: Write y;
Line 8: End.
```

This sample program runs in time exponential of input size, as the size $n$ of $x$ satisfies $x \in \Theta(2^n)$.

Which of the following problems is known to be in LOGSPACE:

1. ☐ Given a context-free grammar and a word, check whether the grammar generates the word;

2. ☐ Given a binary word, decide whether it is in the fixed context-sensitive language $\{uvuw \in \{0,1\}^*$ and $u, v, w$ all have the same length$\}$;

3. ☐ Given a 4SAT formula, decide whether it has a satisfying assignment.

Tick one option and explain how a LOGSPACE algorithm works for this problem.

**Solution.** The second choice is in logarithmic space. The idea is to count the length $n$ of the word and divide it by 4 and then, for each of the positions $\ell = 0, 1, \ldots, n/4-1$, compare whether the symbol there is the same as at the position $\ell + n/2$. If so, then accept else reject; furthermore, reject all inputs whose length is not a multiple of 4. These checks can be done in LOGSPACE and the algorithm can be implemented with few variables tracking the current and the target positions when moving around in the input word. The variables can be stored on the work tape in logarithmic space, as they take values between 0 and $n$.

The first problem is P-complete and the third is NP-complete, see slides number 69 for first problem and slide 161 for 4SAT where the best algorithm known runs in time $O(1.4986^n)$ and thus not in LOGSPACE; if they would be in LOGSPACE then P or NP, respectively, would collapse to LOGSPACE what is unlikely.

Recall that X3SAT is the problem to find to an instance where all clauses have at most 3 literals a solution where in every clause exactly one instance is true.

Consider the following X3SAT algorithm. It always takes the first case which applies.

1. If there is a one-literal clause, then set the corresponding literal true and simplify accordingly.
2. If one of the clauses of the form $x \vee x \vee y$, $x \vee x$ or $x \vee y \vee \neg y$ appears or if two clauses of the form $x \vee y \vee w$, $x \vee \neg y \vee w$ appear then set $x = 0$ and simplify accordingly.
3. If there is a two-literal clause $x \vee y$ then replace $x$ by $\neg y$ and $\neg x$ by $y$ everywhere and remove the clause $x \vee \neg x$.
4. If there is a clause sharing at most one variable with other clauses, then remove it. Also remove clauses of the form $x \vee \neg x$.
5. If there are clauses $x \vee y \vee v$, $x \vee y \vee w$ then set $v = w$ and simplify accordingly.
6. If there are clauses $x \vee y \vee v$, $\neg x \vee \neg y \vee w$, then set $x = \neg y$, $v = 0$, $w = 0$ and simplify accordingly.
7. If there is a clause $x \vee y \vee z$ such that the variables of the literals $x, y$ appear also in other clauses, then branch $x = 1, y = 0, z = 0$ versus $x = 0$, $y = \neg z$.

Assume that when branching $x \vee y \vee z$ according to case 7, the second clause where $x$ appears is either of the form $x \vee v \vee w$ or $\neg x \vee v \vee w$. The further clause where $y$ appears is either $y \vee r \vee s$ or $\neg y \vee r \vee s$. Note that every clause shares at least two variables with other clauses when branching occurs.

Explain why other cases (like $x, y$ jointly appearing in another clause) do not arise when the algorithm goes into case 7 for branching and find the best branching factor according to the table below (only the cases in the table are relevant) for the branching. Case $x = 1$ and Case $x = 0$ can, according to different cases, lead to the elimination of the number of variables listed in a row.

| Case $x = 1$ | Case $x = 0$ | Branching Factor |
|---|---|---|
| 5 | 2 | 1.2365 |
| 5 | 3 | 1.1938 |
| 5 | 4 | 1.1673 |
| 5 | 5 | 1.1486 |
| 6 | 2 | 1.2106 |
| 6 | 3 | 1.1739 |
| 6 | 4 | 1.1509 |
| 6 | 5 | 1.1347 |

Provide your branching factor for the algorithm (the $c$ with $O(c^n)$ performance) and explain the underlying cases considered.

**Solution.** Note that the case that $x$ once appears with $y$ and once appears with $\neg y$ in a clause is eliminated by the second rule. The sixth rule eliminates the case that $x \vee y$

appears in two clauses, the seventh rule eliminates the case that $x \vee y$ and $\neg x \vee \neg y$ appear both in a clause. Thus if $x, y$ appear in one clause together, then they do not do it in any other clause.

Now consider the case where $x$ appears in this form in two clauses. If $x = 1$ then in both clauses four further variables are set to 0. At least one of them, $y$, appears in a further clause, so setting $y = 0$ causes either $r = \neg s$ or both $r, s$ being 0 due to the corresponding subcase; one assumes the worse case and so 6 variables are eliminated when $x = 1$. When $x = 0$ then $y = \neg z$ and $v = \neg w$, thus three variables are eliminated and the branching factor is 1.1739 in this case.

The other case is that $x$ occurs once as $x$ and once as $\neg x$. If $x = 1$ then $y, z$ are set to 0, $v$ is set to $\neg w$ and $y$ causes also in some other clauses a further elimination. So 5 variables are eliminated. When $x = 0$ one can argue symmetrically as the other clause is $\neg x \vee v \vee w$ and one of $v, w$ appears elsewhere in a clause. So both cases $x = 1$ and $x = 0$ eliminate five variables and the branching factor is 1.1486. So the overall performance of the algorithm is $O(1.1739^n)$, as the worse branching factor has to be taken.

Additional explanation (not required for the solution): The clause in item 4 can be removed, as every assignment making all other clauses true by choosing the variables occurring there accordingly can be extended to an assignment also making this clause true, as this assignment fixes at most one variable in the clause and, due to the other elimination cases before, this clause contains at least three variables so that the remaining variables can be chosen such that exactly one literal is true.

The Chomsky hierarchy consists of four levels: Those of languages generated by regular, context-free, context-sensitive and unrestricted grammars, respectively.

Which of the first three coincide with space complexity classes? And which of the following classes are it: CONSTANTSPACE, LOGSPACE, NLOGSPACE, POLYLOGSPACE, LINSPACE, NLINSPACE, PSPACE. Provide these equialvalences and explain which class or classes are not equivalent to any of the given complexity classes.

**Solution.** Regular languages can be recognised by deterministic finite automata. These are Turing machines which read the word from the left to the right and do not use any worktape – all memory is in the form of a state of the Turing machine. Once the Turing machine has read the complete word, it says ACCEPT or REJECT.

All context-free languages are in the class POLYLOGSPACE. However, there are even LOGSPACE decidable language which are not context-free. An example is the language of all words $0^n 1^n 0^n$ which is in LOGSPACE by counting the number of digits of the three subwords and comparing them, but the language is a standard example of a language which cannot be generated by a context-free grammar.

NLINSPACE is sufficient to simulate a context-sensitive grammar which step by step (on the work tape) generates a word until it reaches the length of the input. If it overshoots or differs from the input word, the input word is not accepted. However, if the word generated coincides with the input word, the input word is accepted.

By the Theorem of Kuroda from 1964, NLINSPACE also consists entirely of context-sensitive languages; that is, context-sensitve languages are exactly the NLOGSPACE recognisable languages. It is unknown whether the class of all context-sensitive languages also coincides with the deterministic class LINSPACE.

Explanation for grammars: Unrestricted grammars are those given by rules where the only requirement is that in a rule $v \to w$ at least one of the symbols in $v$ is a nonterminal and this condition is also satisfied by all other types of grammars. Context-sensitive grammars are those where each rule $v \to w$ also satisfies that $|v| \leq |w|$, that is, $w$ is at least as long as $v$ (except for the case that the empty word is in the language and some additional rules apply to handle that). Context-free grammars have in rules on the left side exactly one symbol which is a nonterminal. Regular grammars are context-free grammars with the additional property that the right side $w$ has at most one nonterminal and, if this exists, it is always the last symbol of all the symbols in $w$.

END OF MIDTERM TEST.