Radboud University Nijmegen

Science Faculty

Exam **Complexity IBC028** June 21, 2021, 8.30 – 10.30
The maximum number of points per question is given in the margin. (Maximum 100 points in total.)
When using well-known results that we have seen in the course, clearly state the result you are using; you don't have to prove it again.

---

(20) 1. We have a recursive algorithm whose time complexity $T(n)$ satisfies

$$T(n) = 7T(n-2) + 5T(n-3) + f(n),$$

with $f(n) = \Theta(n^3)$. Prove that $T(n) = \mathcal{O}(3^n)$.
TestedTopics:
Computing complexity of an exponential algorithm using the Substitution Method

(20) 2. We have a recursive algorithm that, on an input of size n, does $3i$ recursive calls on input of size $\frac{n}{3}$ plus additional computations of time complexity $\Theta(n^2)$. Determine the time complexity of this algorithm for $i = 1, 2, 3, 4$.
TestedTopics:
Computing the complexity of algorithms using the Master Theorem

3. Suppose we have two algorithms A_1 and A_2 for which we have bounds on the running time, given by T_1 and T_2, respectively for which we know the following (for some constants c and d).

$$\begin{aligned} T_1(n) &= T_1(\left\lfloor \frac{n}{7} \right\rfloor) + T_1(\left\lfloor \frac{2n}{7} \right\rfloor) + T_1(\left\lfloor \frac{3n}{7} \right\rfloor) + cn \\ T_2(n) &= T_2(\left\lfloor \frac{n}{2} \right\rfloor) + T_2(\left\lfloor \frac{n}{3} \right\rfloor) + T_2(\left\lfloor \frac{n}{6} \right\rfloor) + dn \end{aligned}$$

(10) (a) Use the recursion tree method to compute an f_1 such that algorithm A_1 is $\Theta(f_1(n))$. (Subtleties due to rounding may be ignored.)

(10) (b) Use the recursion tree method to compute an f_2 such that algorithm A_2 is $\Theta(f_2(n))$. (Subtleties due to rounding may be ignored.)

TestedTopics:

Computing the complexity for algorithms using the Recursion Tree Method

See next page

4. We have defined the problem *not-all-equal*-3CNF-SAT, Neq3CNF-SAT(φ), as the problem of deciding for a formula $\varphi \in$ 3CNF whether there is an assignment such that in every clause in φ, **at least one literal is true and at least one literal is false**.
Similarly, we have Neq4CNF-SAT: the problem of deciding for a formula $\varphi \in$ 4CNF whether there is an assignment such that in every clause in φ, **at least one literal is true and at least one literal is false**.

(10) (a) Describe a procedure to transform a disjunction of 4 literals $\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4$ into a 3CNF, φ, such that

$$\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4 \text{ is Neq4-satisfiable if and only if } \varphi \text{ is Neq3-satisfiable.}$$

Prove that your procedure satisfies this property.

(10) (b) It is given that Neq4CNF-SAT is NP-complete. Prove that Neq3CNF-SAT is NP-complete.

TestedTopics:

Polynomial Reduction, SAT-related problems, proving NP-completeness of a SAT-related problem

5. Define, for $G = (V, E)$ an undirected graph, the problem "relaxed 3Color", r3Color(G), as the problem to decide whether G can be 3-colored where **at most one edge can have both endpoints of the same color** and each other edge has two endpoints with a different color.

(5) (a) Draw a graph that can be "relaxed-3-colored", but not 3-colored.

(15) (b) Prove that r3Color is NP-complete.
<u>*Hint*</u> Use the NP-hardness of 3Color; add a simple graph to your graph.

TestedTopics:

NP-completeness, NP-Hardness, proofs of these properties for Graph-problems

END