Instructor: Rocco A. Servedio

Computer Science 4236: Introduction to Computational Complexity
Problem Set #2 Spring 2023

Due 11:59pm Wednesday, April 5, 2023

See the course Web page for instructions on how to submit homework.
Important: To make life easier for the TAs, **please start each problem on a new page.**

Problem 1 Let BIPARTITE be the language BIPARTITE := $\{G : G$ is an undirected bipartite graph$\}$. (Recall that a graph is bipartite if its vertex set can be partitioned into two disjoint sets L and R such that every edge has one L-vertex and one R-vertex.)

Show that BIPARTITE $\in$ NL. (Hint: Think about cycles.)

Problem 2 The *outdegree* of a directed graph is the maximum number of directed edges $(i \to j)$ coming out of any node i in G.

(a) Show that Generalized Geography is still PSPACE-complete even if restricted to directed graphs G with outdegree 2. An equivalent but more formal phrasing of this problem would be: "Show that the language $GG_2 = \{(G, v) : G$ is a digraph with outdegree 2 such that player 1 has a winning strategy for Generalized Geography played on G starting at node $v\}$ is PSPACE complete."

(b) Your friend claims that he can prove that that Generalized Geography is PSPACE-complete even if restricted to directed graphs G with outdegree 1. Why should you be skeptical of his claim?

Problem 3 (The point of this problem is that it's not important that our model for randomized computation uses binary randomness; a range of reasonable alternatives would do just as well.)

Show that it is possible to efficiently simulate a fair N-sided die using coin tosses. In more detail, show that for any positive integer N and any $\delta > 0$, there is a probabilistic Turing machine M (which "tosses fair coins" as described in our Wed March 8 lecture) running in worst-case time $\mathrm{poly}(\log N, \log 1/\delta)$) which always produces an output in $\{1, 2, \ldots, N, \perp\}$, and satisfies the following: (i) conditioned on not outputting $\perp$, the output of M is uniformly distributed over $[N] = \{1, \ldots, N\}$, and (ii) the probability that M outputs $\perp$ is at most δ.

Is it possible to achieve a stronger simulation, with $\delta = 0$, that runs in worst-case time $\mathrm{poly}(\log N)$? Explain why or why not.

Problem 4 Show that if the language SAT of satisfiable Boolean formulas is in BPP, then SAT is in RP.

Problem 5 In this problem we'll see a randomized polynomial-time algorithm for 2-CNF satisfiability which has a similar flavor to the $\text{poly}(n) \cdot (3/2)^n$-time randomized algorithm that we did in class for 3-CNF satisfiability.

Recall that a 2-CNF formula is an AND of clauses each of which has at most two literals; for example,

$$\phi = (x_1 \vee \overline{x}_4) \wedge (\overline{x_2} \vee \overline{x}_3) \wedge (x_3 \vee x_4)$$

is a 2-CNF. Consider the following randomized algorithm which attempts to find a satisfying assignment of an input 2-CNF formula ϕ:

Input: $\phi = C_1 \wedge \cdots \wedge C_m$ a 2-CNF on n vars

[1] Let $z \in \{0,1\}^n$ be any initial assignment to variables

[2] If $\phi(z) = 1$ stop and output "satisfiable"

[3] If $\phi(z) = 0$ choose any clause C which is not satisfied by z. Pick a random literal of C and flip that bit of z.

[4] Repeat Steps (2) and (3) $r = 2n^2$ times; if you still haven't found a satisfying assignment, stop and output "probably unsatisfiable."

It's clear that this algorithm always outputs "probably unsatisfiable" if ϕ is indeed unsatisfiable. Below you'll argue that if ϕ is satisfiable then the above algorithm succeeds in finding a satisfying assignment with probability at least 1/2.

Similar to the analysis in class, fix a satisfying assignment $z^* \in \{0,1\}^n$. Let $t(i)$ denote the max, over all n-bit strings z that differ from z^* in at most i bit positions, of the expected number of "random flips" (steps like Step (3) in the algorithm) which would be required until a patient version of the algorithm (which doesn't "time out" after r trials, but keeps trying forever) would reach a satisfying assignment for ϕ, given that the current assignment is z.

(a) Explain why $t(\cdot)$ satisfies the following conditions: $t(0) = 0$; $t(n) \leq 1 + t(n-1)$; for $i \in \{1, \ldots, n-1\}$, $t(i) \leq 1 + (1/2)[t(i-1) + t(i+1)]$.

(b) Let $t'(\cdot)$ be obtained by relaxing the above inequalities to equalities, i.e. $t'(\cdot)$ satisfies $t'(0) = 0$; $t'(n) = 1 + t'(n-1)$; for $i \in \{1, \ldots, n-1\}$, $t'(i) = 1 + (1/2)[t'(i-1) + t'(i+1)]$. Prove that $t'(n) \leq n^2$.

(c) It can be argued (you don't need to do this) that $t(i) \leq t'(i)$. Use this to justify the claim that the above algorithm finds a satisfying assignment, when one exists, with probability at least 1/2.