CSCI 6470 Quiz #8 Questions Answers

Monday November 20, 2023 (12:40pm-1:10pm EST)

Student Name ______________________________ **Student ID**____________________________

There are 4 questions and 60 points in total. Good luck!

1. (**10 points**) The exhaustive search algorithm `MaxIS-Solver` for **Maximum Independent Set** has time complexity $O(1.619^n)$, which is derived from recursive formula for the time $T(n) = T(n-1) + T(n-2) + O(n)$, where n is the number vertices in the input graph.

   (1) Explain where the term $T(n-1)$ comes from:
   vertex v is discarded from the chosen independent set $\boxed{4\,points}$

   (2) Explain where the term $T(n-2)$ comes from:
   since isolated vertices are all included in the chosen independent set, $\boxed{2\,points}$
   vertex v has at least one neighbor $\boxed{2\,points}$

   (3) The algorithm can be improved to achieve $T(n) = T(n-1) + T(n-3) + O(n)$. Explain how term $T(n-2)$ is reduced to $T(n-3)$:
   degree ≤ 1 vertices are all included in the chosen independent set, $\boxed{1\,points}$;
   vertex v has at least 2 neighbor $\boxed{1\,points}$

2. (**15 points**) Let **MST-D** be a decision version of **MST** such that, on the input graph G and input weight threshold w, **MST-D** has the "yes" answer if and only if G has a minimum spanning tree of weight $\leq w$. Now any algorithm A for **MST-D** can be used to construct an algorithm A_{mst} for problem **MST**. Answer the following questions:

   (1) On input graph H of **MST**, how does A_{mst} use A to find the minimum weight w_0 of a spanning tree for H?
   A_{mst} runs on H and $w = 1, 2, \dots$ to w_{max} or $w = w_{max}$ down to 1, (where w_{max} is the sum of edge weights), until answers are changed from "no" to "yes" (or from "yes" to "no") $\boxed{6\,points}$

   (2) Once w_0 is found, how does A_{mst} use A again to find the corresponding spanning tree of weight w_0?
   For every edge $e \in H$, A_{mst} runs on $H - \{e\}$ and w_0. If answer remains "yes" remove e from H, otherwise, keep e. $\boxed{5\,points}$

   (3) How to guarantee that, if A has time complexity $O(n^d)$, the constructed A_{mst} has time complexity $O(n^{d+2})$?
   Use binary search on w in step (1), with complexity $O(n) \times O(n^d)$ $\boxed{2\,points}$
   step (2) goes through $O(n^2)$ edges, has complexity $O(n^2) \times O(n^d)$ $\boxed{2\,points}$

3. (**15 points**) Answer the following questions regarding polynomial-time verifiable problems. Assume V_{SAT} to be a verification algorithm for **SAT**.

   (1) What should the input to V_{SAT} be? (ϕ, A), where ϕ is a boolean formula $\boxed{3\,points}$ and A is a truth value assignment for variables in ϕ $\boxed{3\,points}$

   (2) What should V_{SAT} check to fulfill its verification duty?
   (a) verifies that A has assignments for all variables in ϕ; $\boxed{3\,points}$
   (b) check that ϕ can be evaluated to TRUE; $\boxed{3\,points}$
   (c) output "yes" if and only if both (a) and (b) turn out true. $\boxed{3\,points}$

4. (**20 points**) The class $\mathcal{NP}$ is defined as follows according to the lecture note:

   Definition: $\mathcal{NP}$ is the class of decision problems whose answers can be verified in polynomial time.

   That is,

   For every decision problem $D \in \mathcal{NP}$, which decides on input x to answer "yes" or "no", there exists a verifier V_D such that

   $$\forall x, D(x) \begin{cases} = \text{"}yes\text{"} & \exists y, V_D(x, y) = \text{ TRUE} \\ = \text{"}no\text{"} & \forall y, V_D(x, y) = \text{ FALSE} \end{cases}$$

   where V_D can be computed in polynomial time, and y is called a *certificate* or *witness* to an "yes" answer.

   Let problem **Independent Set** be decision problem D in the definition.

   (1) What is x specific to the **Independent Set** problem?
   (G, k), where G is a graph and $k \geq 0$ is an integer; $\boxed{4\,points}$

   (2) What is y specific to the **Independent Set** problem?
   a set of vertices (independent set) of size k; $\boxed{4\,points}$

   (3) What is V_D specific to the **Independent Set** problem?
   verification algorithm to verify if y is an independent set of size k for G; $\boxed{4\,points}$

   (4) V_D runs in polynomial time $O(N^c)$ for some constant c, where $N = |(x, y)|$ $\boxed{4\,points}$

   (5) Does D runs in a polynomial time also? Explain
   D may or may not run in a polynomial time. A polynomial-time solvable problem, like **Reachability**, can have a polynomial-time verifier, while a more difficult problem, like **SAT**, can also have a polynomial-time verifier. $\boxed{4\,points}$

[The following space will not be graded.]