¡Felicitaciones! ¡Aprobaste!

Calificación recibida 100 % **Para Aprobar** 80 % o más

Ir al siguiente elemento

Problem Set #2

Calificación de la entrega más reciente: 100 %

1. Which of the following statements *cannot* be true, given the current state of knowledge? **1 / 1 punto**

[✓ Some NP-complete problems are polynomial-time solvable, and some NP-complete problems are not polynomial-time solvable.]

There is an NP-complete problem that is polynomial-time solvable.

There is an NP-complete problem that can be solved in $O(n^{\log n})$ time, where n is the size of the input.

There is no NP-complete problem that can be solved in $O(n^{\log n})$ time, where n is the size of the input.

Correcto
A polynomial-time algorithm for a single NP-complete automatically yields polynomial-time algorithms for all NP-complete algorithms (i.e., implies that P=NP).

2. Let TSP1 denote the following problem: given a TSP instance in which all edge costs are positive integers, compute the value of an optimal TSP tour. Let TSP2 denote: given a TSP instance in which all edge costs are positive integers, and a positive integer T, decide whether or not there is a TSP tour with total length at most T. Let HAM1 denote: given an undirected graph, either return the edges of a Hamiltonian cycle (a cycle that visits every vertex exactly once), or correctly decide that the graph has no such cycle. Let HAM2 denote: given an undirected graph, decide whether or not the graph contains at least one Hamiltonian cycle. **1 / 1 punto**

[✓ If TSP2 is polynomial-time solvable, then so is TSP1. If HAM2 is polynomial-time solvable, then so is HAM1.]

Polynomial-time solvability of TSP2 does not necessarily imply polynomial-time solvability of TSP1. Polynomial-time solvability of HAM2 does not necessarily imply polynomial-time solvability of HAM1.

If TSP2 is polynomial-time solvable, then so is TSP1. But, polynomial-time solvability of HAM2 does not necessarily imply polynomial-time solvability of HAM1.

Polynomial-time solvability of TSP2 does not necessarily imply polynomial-time solvability of TSP1. But, if HAM2 is polynomial-time solvable, then so is HAM1.

Correcto
Given a polynomial-time algorithm for TSP2, use it repeatedly while binary searching over the choice of T to solve TSP1 (the number of iterations necessary is logarithmic in the sizes of the input numbers, which is polynomial in the input size). Use a polynomial-time algorithm for HAM2 to solve HAM1 as follows. First, run HAM2 on the graph. If it says "no", report "no". Otherwise, delete some edge (in effect, guessing that there is also a Hamiltonian cycle without it) and run HAM2 again. If it says "yes", iterate on the new smaller graph. If it says "no", restore the edge (it lies on every Hamiltonian cycle, do you see why?) and move on to the next edge. If HAM2 originally said "yes", you will identify (in $O(m)$ iterations) a sequence of successful edge deletions, and the remaining edges will constitute a Hamiltonian cycle of the original graph.

3. Assume that $P \neq NP$. Consider undirected graphs with nonnegative edge lengths. Which of the following problems can be solved in polynomial time? **1 / 1 punto**

Hint: The Hamiltonian path problem is: given an undirected graph with n vertices, decide whether or not there is a (cycle-free) path with $n-1$ edges that visits every vertex exactly once. You can use the fact that the Hamiltonian path problem is NP-complete. There are relatively simple reductions from the Hamiltonian path problem to 3 of the 4 problems below.

[✓ For a given source s and destination t, compute the length of a shortest s-t path that has exactly $n-1$ edges (or $+\infty$, if no such path exists). The path is allowed to contain cycles.]

Amongst all spanning trees of the graph, compute one with the smallest-possible number of leaves.

Amongst all spanning trees of the graph, compute one with the minimum-possible maximum degree. (Recall the degree of a vertex is the number of incident edges.)

For a given source s and destination t, compute the length of a shortest s-t path that has exactly $n-1$ edges (or $+\infty$, if no such path exists). The path is not allowed to contain cycles.

Correcto

Use a Bellman-Ford-type recurrence.

4. Choose the strongest true statement. 1 / 1 punto

○ If the minimum-size vertex cover problem can be solved in time $O(T(n))$ in bipartite graphs, then the maximum-size independent set problem can be solved in time $O(T(n))$ in bipartite graphs.

[✓ All three of the other assertions are true.]

○ If the maximum-size independent set problem can be solved in time $O(T(n))$ in general graphs, then the minimum-size vertex cover problem can be solved in time $O(T(n))$ in general graphs.

○ If the minimum-size vertex cover problem can be solved in time $O(T(n))$ in general graphs, then the maximum-size independent set problem can be solved in time $O(T(n))$ in general graphs.

Correcto
Vertex covers and independent sets are complements (take the complement of one and you get the other). Thus solving one problem allows you to solve the other with $\Theta(n)$ postprocessing, where n is the number of vertices.

5. Which of the following statements is true? 1 / 1 punto

[✓ Consider a TSP instance in which every edge cost is the Euclidean distance between two points in the place (just like in Programming Assignment #5). Deleting a vertex and all of its incident edges cannot increase the cost of the optimal (i.e., minimum sum of edge lengths) tour.]

○ Consider a TSP instance in which every edge cost is negative. The dynamic programming algorithm covered in the video lectures might not correctly compute the optimal (i.e., minimum sum of edge lengths) tour of this instance.

○ Consider a TSP instance in which every edge cost is negative. Deleting a vertex and all of its incident edges cannot increase the cost of the optimal (i.e., minimum sum of edge lengths) tour.

○ Consider a TSP instance in which every edge cost is either 1 or 2. Then an optimal tour can be computed in polynomial time.

Correcto
Take the optimal tour in the original instance. Now, instead of visiting the deleted vertex v, skip straight from v's predecessor to its successor on the tour. Because Euclidean distance satisfies the "Triangle Inequality", this shortcut only decreases the overall distance traveled. The best tour can of course only be better.Take the optimal tour in the original instance. Now, instead of visiting the deleted vertex v, skip straight from v's predecessor to its successor on the tour. Because Euclidean distance satisfies the "Triangle Inequality", this shortcut only decreases the overall distance traveled. The best tour can of course only be better.