CSE 551: Quiz 7 Solutions

Jamison Weber

November 5, 2021

1 Problem 1

Which of the following problems does not likely have a polynomial-time algorithm? Select all that apply.

- Vertex Cover
- Shortest Path
- Knapsack
- Minimum Cut

1.1 Rationale

VERTEX-COVER and KNAPSACK are both NP-Complete problems. If one were to find a polynomial time algorithm that solves either of these, it would prove P=NP.

2 Problem 2

What is a decision problem?

- A problem in which the answer is always yes or no.
- A problem that requires any algorithm for it to make a decision.
- A problem that is in P or NP.
- A problem that requires any algorithm for it to return a result that is not a boolean value.

2.1 Rationale

A decision problem is a problem that can be answered either yes or no by a Turing machine given an instance and a parameter. More specifically, the Turing Machine decides whether an input string is an element of the language it recognizes.

3 Problem 3

Why, informally, is P a subset of NP?

- **Because the certificate could be the algorithm itself, and the certifier could be the execution of the algorithm.**
- Because an NP algorithm would be to simulate the P algorithm.
- Because any polynomial-time algorithm obviously runs in polynomial-time.
- Because it follows directly from the definition.

3.1 Rationale

For any problem that can be solved in polynomial time we can demonstrate that a certificate is correct simply by running the full algorithm, which will, of course, execute in polynomial time.

4 Problem 4

Which of the following is a valid statement about problem Y for when ($X \leq_P Y$) and (X) is guaranteed to be polynomial-time solvable?

- Nothing can be guaranteed about Y.
- Y is in P.
- Y is not in P.
- Y is equivalent to X.

4.1 Rationale

If X polynomially reduces to Y, then we know that problem Y is at least as hard to solve as problem X. If problem X can be solved in polynomial time then one can make problem X arbitrarily more complex to reduce it to problem Y. Therefore, knowing that problem X is solvable in polynomial time does not tell us whether or not Y is in P.

5 Problem 5

If problems X and Y are equivalent (i.e. $X \leq_P Y$ and $Y \leq_P X$), what can we conclude about the worst case running times of the best algorithms that solve them?

- They are at most a polynomial factor different.
- They are identical.
- They are at most an exponential factor different.
- They are within a constant factor.

5.1 Rationale

If problem X polynomially reduces to problem Y and problem Y polynomially reduces to problem X, then we know that both X is at least as hard to solve as Y and Y is at least as hard to solve as X. If this is the case, then they are both in the same complexity class and each of their complexities may only differ from each other by at most a polynomial factor.

6 Problem 6

Which of the following is a valid set of constraints on the inputs to the VERTEX-COVER problem?

- A graph, and a nonnegative integer k.
- A directed graph, and a nonnegative integer k.
- A graph, and an arbitrary integer k.
- A directed graph, and an arbitrary integer k.

6.1 Rationale

We require a graph instance G and a non-negative integer k to be able to decide whether G has a cover of size at least k.

7 Problem 7

In the SAT problem, what does conjunctive normal form mean?

- The formula is a conjunction of clauses, which are disjunctions of literals.
- The formula is a disjunction of clauses, which are disjunctions of literals.
- The formula is a conjunctions of clauses, which are conjunctions of literals.
- The formula is a disjunction of clauses, which are conjunctions of literals.

7.1 Rationale

This is defined carefully in the slides.

8 Problem 8

Consider the following algorithm for VERTEX-COVER: given the graph G and integer k, test all $O(n^k)$ sets of k vertices, and for each determine if it is a vertex cover. Is this a polynomial-time algorithm? Why/why not?

- No, since k is part of the input and it is not constrained to be a constant.
- Yes, because n^k is a polynomial.
- Yes, because there are polynomially many subsets of the vertices.
- No, because no polynomial time algorithm exists for VERTEX-COVER.

8.1 Rationale

As the highlighted answer suggests, if the problem is framed this way, then the size of the input is potentially exponential on n. Thus, this is not a polynomial time solution to the VERTEX-COVER problem, since $p(exp(n))$ is exponential.

9 Problem 9

What was the clause gadget in the 3-SAT to INDEPENDENT-SET reduction?

- A triangle (cycle of length 3).
- A path of length 3.
- A complete graph of size $O(n)$ (n is the number of variables).
- A single vertex with no edges.

9.1 Rationale

This gadget is described clearly and is shown to be effective in lecture.

10 Problem 10

Why is the $(\leq_P)$ relation transitive? (i.e., $(X \leq_P Y)$ and $(Y \leq_P Z)$ implies that $(X \leq_P Z)$)

- **Because the reductions can be composed together into a polynomial time reduction from problem X to problem Z.**
- Because a polynomial plus another is also a polynomial.
- Because a polynomial times another is also a polynomial.
- None of the other answers is correct.

10.1 Rationale

If problem X polynomially reduces to problem Y and problem Y polynomially reduces to problem Z, then the total amount of computations required to reduce problem X to problem Z is the sum of two polynomials. The set of all polynomials is closed under addition.

11 Problem 11

Suppose that in the 3-SAT problem you were allowed to have at most 3 literals per clause(the original problem specified exactly 3). Please mark which of the following are correct.

- **The modified version of the 3-SAT problem is polynomially equivalent to the original version of 3-SAT.**
- The modified version of the 3-SAT problem is in P, and 3-SAT is not in P.
- The modified version of the 3-SAT problem is in P, and 3-SAT is NP-complete. will not change when Dijkstra's algorithm is applied.
- The modified version of the 3-SAT problem is the same as the 2-SAT problem.

11.1 Rationale

Suppose I have a clause $(x_1 \vee x_2)$. To reduce this to a 3-SAT instance, I can create a constant literal $x_3 = 0$. Now $(x_1 \vee x_2 \vee x_3)$ returns true iff $(x_1 \vee x_2)$ does. Creating dummy literals will require time linear on the number of clauses.

12 Problem 12

Which of the following is not an example of a known NP-complete problem?

- PRIMES
- TRAVELING SALESMAN PROBLEM
- SUBSET-SUM.
- KNAPSACK.

12.1 Rationale

In 2001, an algorithm was discovered that can take an integer k and decide whether it is prime or not in $p(\log k)$ computations. Therefore, PRIMES is in P.

13 Problem 13

My friend is trying to show $X \leq_P Y$ for problems X, Y. He has given a polynomial time function that transforms an instance I of X into an instance $poly(I)$ of Y. Lastly, he has shown that if the decision is yes for $poly(I)$, then the decision is always yes for I. Is my friend's proof complete?

- No, your friend must show a decision of no for $poly(I)$ corresponds to a decision of no for I.
- No, your friend must show a decision of no for $poly(I)$ corresponds to a decision of yes for I.
- Yes, this completes all the steps to show a polynomial time reduction from X to Y since yes decisions of Y are mapped to yes decisions for X.
- Yes, this completes all the steps to show a polynomial time reduction from X to Y since yes decisions of Y are mapped to no decisions for X.

13.1 Rationale

In order to show that a problem X polynomially reduces to problem Y, we must create an instance of Y in polynomial time from any arbitrary instance of problem X. To show that problem Y is at least as hard as problem X, we need that the algorithm used to solve Y is effectively simulating an algorithm that solves X. For this simulation to map, it is critical that both decisions of yes and no are mapped.

For those of you who want a more rigorous perspective, a Turing machine recognizes languages, which we tend to think of as "problems". A language L is simply a set of strings, and in this analogy, an instance of a problem is simply a string s. A string s is said to be in L if and only if there exists a Turing machine that recognizes language L for which the decision for input s is "accept". Then the formal definition of a polynomial time reduction is as

follows: $X \leq_P Y \iff$ there exists function $poly(\cdot)$ such that $I \in X$ if and only if $poly(I) \in Y$. Critically, $poly(s)$ must terminate in time polynomial in the size of s.

14 Problem 14

The statements below concern the classes P and NP. Mark all correct statements.

- $|P| \leq |NP|$
- P and NP consist only of decision problems
- The version of the Knapsack problem wherein one must find a subset of maximum profit subject to weight constraints is in NP
- The version of the Traveling Salesman problem wherein one must determine whether a tour of weight at least W exists is in NP.

14.1 Rationale

Regarding the first statement, in lecture we saw that P is a subset of NP, thus the size of P cannot be greater than the size of NP. Regarding the second statement, the sets P and NP are indeed defined as sets of decision problems. Regarding statement three, this version of the Knapsack problem is not framed as a decision problem, but rather a combinatorial optimization problem. Thus it cannot be in NP. Regarding statement four, this version of TSP is indeed a decision problem. A certificate for the problem is simply the tour itself, which is a sequence of vertices. We need only check that there exist edges in the graph instance such that the sequence is possible, which can clearly be done in polynomial time. Thus, this problem is in NP.

15 Problem 15

If an instance I of a problem X undergoes a transformation to an instance I' of a problem Y, and this transformation terminates in time polynomial in the size of I , then must the size of I' be polynomial in the size of I?

- Yes, a Turing machine that runs for $|I|^c$ transitions for some constant c can only access at most $|I|^c$ tape cells.

- Yes, since the size of I' must be equal to the size of I.
- No, even it the transformation terminates in polynomial time, it could have used an exponential amount of space in its computation.
- No, problem X might be NP-complete, and thus any transformation of I will produce a result exponential in the size of I.

15.1 Rationale

The first answer is correct. Recall that I is manipulated to I' in polynomial time, or $|I|^c$ steps for some constant c. In the worst case, the string instance I' could then only occupy $|I|^c$ cells of the Turing machine, since there wasn't enough time for the TM to enter the $(|I|^c + 1)^{st}$ cell. The remaining answer choices are all false statements.