Homework 2

Joni Vrapi

09/25/2022

Statement of Integrity: I, Joni Vrapi, attempted to answer each question honestly and to the best of my abilities. I cited any, and all, help that I received in completing this assignment.

Problem 1. A language is *decidable* if a Turing machine accepts strings that are in the language, and rejects strings that are not in the language. In other words, the Turing machine will halt on all inputs [1]. From this, we can see that an *undecidable* language is one in which a Turing machine will not halt. Suppose we have a Turing machine M which can decide a language L. We can therefore create another Turing machine M' by running M and switching its accepts to rejects and vice versa, that decides L^C. M will always halt if it decides L, therefore we do not need to worry L^C will cause M to never halt. If decidable languages are closed under complementation, then undecidable languages must be as well.

Problem 2. A transitive relation R on a set X occurs when $\forall a, b, c \in X \mid (aRb) \wedge (bRc) \implies (aRc)$ [2]. We can define $f(x)$ to be the polynomial time reduction function such that $x \in L_1 \iff f(x) \in L_2$. We can then define $g(x)$ to be the polynomial time reduction function such that $x \in L_2 \iff g(x) \in L_3$. Finally, we can then compute, in polynomial time, $g \circ f \mid x \in L_1 \iff g(f(x)) \in L_3$. $\therefore L_1 \leq_P L_3$, so $\leq_P$ is transitive.

Problem 3a. $L_1 = \{a^n b^n \mid 0 \leq n \leq 1000\}$
Since $n \leq 1000$, L_1 is finite. All finite languages are regular. All regular grammars belong to Type 3.

Problem 3b. $L_2 = \{a^n b^n \mid n \geq 0\}$
Since $n \geq 0$, L_2 is infinite. This is not a regular language as it does not satisfy the pumping lemma. There is, however, a context free grammar. Context free grammars are of Type 2.

Problem 3c. $L_3 = \{a^n b^m \mid n, m \geq 0\}$
L_3 can be written as a regular expression in the form $a^* b^*$, therefore this is regular and regular grammars are of Type 3.

Problem 4. By contradiction, assume that there exists some maximal clique K of graph G such that $\chi(G) < |K|$. Let's also assume a proper coloring [3] C of G which takes $\chi(G)$ colors. Using the pigeonhole principle [4], let the vertices of K be the pigeons, while the colors assigned to them are the holes. At least two vertices in K should be the same color, and we can label them V_1 and V_2. $\therefore C(V_1) = C(V_2)$. From the definition of a clique, we know that every pair of vertices in K are adjacent, therefore V_1 and V_2 are adjacent in K as well as G. But, as stated before, we also have $C(V_1) = C(V_2)$, which is a contradiction to the fact that C is a proper coloring. Therefore, we can conclude that $\chi(G)$ is no less than the size of any maximal clique of G.

Problem 5. Starting with the adjacency matrix representation [5], because it is easier, we can use an integer x to represent the number of vertices and encode this integer in the normal way integers are encoded in binary. Following this, there will be x^2 bits. The value of bit y will be 1 if there is an edge from vertex $\lfloor m/n \rfloor$ to vertex $m\%n$, and 0 if there is not an edge there.

For the adjacency list representation [5], we should use a different encoding for integers that pads each bit with a 0. This will allow us to ensure that the string 11 will never be found in integers that are encoded in this way. Additionally, since we are only doubling the size of the first binary string, this is still polynomially related to the adjacency matrix representation. We can then use 11 to delimit the combination of vertices and edges. If we define a function $f(x)$ that will then encode an edge (represented as an integer) in the way mentioned above, we can concatenate the strings, starting with the vertex and followed by the edges, to each other. If we represent edges as being indexed by a variable i_k, and a vertex that has been encoded by $f(x)$ as V (for clarity) we will then end up with a string like $V11f(i_1)11f(i_2)11...f(i_k)1111$. Finally, encoding the entire graph will just be the concatenation of all the strings of the above single-vertex representation.

Problem 6. We begin by showing that ER is in NP. If we have a list of $m, (k < m)$, counselors who are qualified for all the sports, this would suffice as a "yes" answer to ER. We can check the validity of this answer in polynomial time by:

1. Check if $k < m$.
2. Then, for each sport n, search all m counselors to see if any are qualified for it.
3. If we find a counselor for all sports, accept, otherwise reject.

This runs in $O(mn)$ time, making it an NP problem.

Next we must show that ER $\leq_P$ of some known NP-complete problem. The following will show that ER $\leq_P$ Set Cover [6] [7]. If we have an instance A of SC which consists of n subsets $S_1...S_n$ of a set $S = \{a_1...a_n\}$, we can construct an instance B of ER as follows. For each of the elements $a_n \in S$ we can create a sport x_i, and for each subset of S_j we can create a counselor y_j. The counselor y_j is qualified for the sport a_i iff $a_i \in S_j$. It is obvious then that this reduction can be performed in polynomial time.

Now we can show that there are k subsets in A that cover S iff there are k counselors in B that cover all the sports. If we assume that the k sets $S_{j_1}...S_{j_k}$ cover S, we can claim that the corresponding k counselors $c_{j_1}...c_{j_k}$ cover all the sports. Consider any sport x_i, the corresponding element a_i must belong to some subset S_{j_i} that contains a_i. Then, the counselor y_i is qualified to teach sport a_i.

If we also assume that the k counselors $c_{j_1}...c_{j_k}$ cover all the sports, we can claim that the corresponding k subsets $S_{j_1}...S_{j_k}$ cover S. Any element $e_i \in S$, corresponding to sport b_i must be covered by some counselor c_{j_i}. Then, S_{j_i} contains e_i. Thus, ER is NP-complete.

Problem 7. First, we can show that *DS* is in NP because, for each customer, we can check all products, and can, in polynomial time, confirm that no two customers purchased the same product. Given a graph $G = (V, E)$, an $m \times n$ array A, and a number of customers $k \leq m$, we construct a customer for each node V and a product for each edge E. We then build an array of customers V who bought product E iff E is incident on V. Finally, we ask whether this array is a *DS* of size k. We claim that this holds iff G has an *IS* of size k.

If there is a *DS* of size k, then the set V has the property that no two V_i are incident to the same E_j. Therefore, it is an *IS* of size k. Conversely, if it is true that there exists an *IS* of size k, then V has the property that no two V_i have anything in common, so it is diverse. Thus, *DS* is NP-complete.

References

[1] "Turing machines." https://brilliant.org/wiki/turing-machines/. Accessed on 2022-09-27.

[2] "Transitive relation." https://en.wikipedia.org/wiki/Transitive_relation. Accessed on 2022-09-27.

[3] "Definition:proper coloring." https://proofwiki.org/wiki/Definition:Proper_Coloring. Accessed on 2022-09-28.

[4] "Pigeonhole principle." https://en.wikipedia.org/wiki/Pigeonhole_principle#Uses_and_applications. Accessed on 2022-09-28.

[5] "Ch34." https://sites.math.rutgers.edu/~ajl213/CLRS/Ch34.pdf. Accessed on 2022-09-28.

[6] "Set cover problem." https://en.wikipedia.org/wiki/Set_cover_problem. Accessed on 2022-09-29.

[7] "sol3.pdf." https://courses.cs.washington.edu/courses/csep531/09wi/handouts/sol3.pdf. Accessed on 2022-09-29.