University of Nevada, Las Vegas Computer Science 456/656 Fall 2024

Answers to Assignment 5: Due Saturday October 19, 2024, 11:59 PM

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet.

1. Give a $\mathcal{P}$–TIME reduction of the Subset Sum problem to Partition.

   An instance of SS, the Subset Sum problem is a sequence of n positive numbers $\sigma = x_1, x_2, \ldots x_n$ followed by a single positive number K. The solution to that instance is **true** if the terms of some subsequence of σ add up to K.

   An instance of the Partition problem is a sequence of m positive numbers $\tau = y_1, y_2, \ldots y_m$. The solution to that instance is **true** if there is some subsequence of τ whose sum is half the sum of τ.

   We define a reduction from the Subset Sum problem to Partition.

   Let $(x_1, x_2, \ldots x_n, K)$ be an instance of SS. Define $S = \sum_{i=1}^n x_i$. Without loss of generality, $S \geq K$, since otherwise (trivially) there is no solution to that instance. Our method is to define two additional numbers, and we reduce to an instance which contains all of σ, together with those two numbers.

   The obvious choice is to let the two extra numbers be K and $S - K$. The sum of τ is then $2S$, and we can see that if σ has a subsequence whose total is K, the subsequence of τ consisting of that subsequence together with $S - K$ will sum to exactly half the sum of τ.

   But this solution, although nice, fails, because we could pick the subsequence of τ to be σ, whose sum is S, exactly half the sum of τ. So even if there is no solution to the original SS instance, there is a solution to the Partition instance, which is not allowed.

   What we do is add 1 to each of our new numbers. Thus, $\tau = x_1, x_2, \ldots x_n, K + 1, S - K + 1$. The sum of τ is $2S+2$, and the two extra numbers add up to $S+2$, which is more than half $2S+2$, hence cannot both be in the subsequence of τ.

   Here is a proof that our construction is a reduction of SS to Partition.

   (a) Suppose σ has a subsequence whose sum is K. Then append $S - K + 1$ to that subsequence, and its total is now $S + 1$, which is exactly half the sum of τ

   (b) Conversely, suppose that there is a subsequence τ_1 of τ whose total is $S+1$. Then the remaining terms of τ form a subsequence τ_2 whose sum is also $S + 1$. Since $K + 1$ and $S - K + 1$ total $2S+2$, they cannot both be in either τ_1 or τ_2; thus $S-K+1$ is in just one of those subsequences, say τ_1. The remaining terms of τ_1 form a subsequence of σ whose total is K.

   That is, the given instance of SS is **true** if and only if the instance of Partition that we constructed is **true**. That is the definition of a reduction, and our contruction takes linear time.

2. Give a $\mathcal{P}$–TIME reduction of 3-SAT to the Independent Set problem.

   An instance of IND, the independent set problem, is an ordered pair $\langle G \rangle, \langle k \rangle$ where G is a graph and k is a positive number. The solution to that instance is **true** if there are k vertices of G which are *independent*, meaning that no two of them are neighbors.

   An instance of 3-SAT is a Boolean expression E in 3-CNF form. That is, E is the conjunction of clauses, each of which is the disjunction of three terms, each of which is either a variable or the negation of a variable. That is, $E = C_1 \cdot C_2 \cdot C_3 \cdots C_k$ where $C_i = (t_{i,1} + t_{i,2} + t_{i,3})$ where each $t_{i,j}$ is either a variable or the negation of a variable. Then $E \in$ 3-SAT if E has a satisfying assignment.

   Given a 3-CNF expression E, our reduction consists of constructing an instance $\langle G \rangle k$ which is a member of IND if and only if E is satisfiable. Let $\{v_{i,j} \,:\, i \in \{1, \ldots k\},\ j \in \{1,2,3\}\}$ be the vertices of G. We say that the vertex $v_{i,j}$ *corresponds to* the term $t_{i,j}$. Two terms $t_{i,j}$ and $t_{i',j'}$ are *contradictory* if one of them is a variable and the other the negation of that variable, and we also call their corresponding vertices in G contradictory. There is an edge from connecting vertex $v = v_{i,j}$ to vertex $v' = v_{i',j'}$ if either $i = i'$ or v and v' are contradictory. Thus, the vertices corresponding to the terms of one clause form a clique.

   $\langle G \rangle \langle k \rangle$ is an instance of 3-CNF. Construction of that instance is clearly $\mathcal{P}$–TIME. We need to show that the construction is a reduction from 3-SAT to IND, namely that $E \in$ 3-SAT if and only if $\langle G \rangle \langle k \rangle \in$ IND.

   Suppose $\mathcal{I}$ is an independent set k vertices of G. Then $\mathcal{I}$ must consist of exactly one member of each clique. Let $\mathcal{T}$ be the set of terms corresponding to $\mathcal{I}$. Assign truth values to each variable of E such that each member of $\mathcal{T}$ is true. Variables which do not occur in $\mathcal{T}$ can be assigned arbitrary values. The assignment is consistent, since $\mathcal{I}$ can contain no two contradictory vertices. Since at least one term of each clause is assigned true, the assignment is satisfying.

   Conversely, suppose E has a satisfying asignment. At least one term of each clause must be assigned true. Let $\mathcal{T}$ be a set consisting of exactly one term of each clause, where that term is assigned true. Let $\mathcal{I}$ be the set of k vertices of G corresponding to $\mathcal{T}$. We need to show that $\mathcal{I}$ is independent. Two members of $\mathcal{I}$ are in different cliques, hence are not connected by a clique edge. Since two members of $\mathcal{I}$ in different cliques cannot be contradictory, since their corresponding terms are both assigned true, and hence cannot be connected by an edge between two cliques. Thus $\mathcal{I}$ is an independent set of order k.

3. Use the pumping lemma to prove that $L = \{a^n b^n \,:\, n \geq 0\}$ is not regular.

Claim 1 *L is not regular.*

Proof: By contradiction. Assume L is regular. Then L must have a pumping length, a positive number p. Let $w = a^p b^p$, which is a member of L of length greater than p. Therefore, by the pumping lemma, there exist string x, y, z such that: 1. $w = xyz$

2. $|xy| \leq p$
3. $|y| > 0$
4. For any $i \geq 0$ $xy^i z \in L$

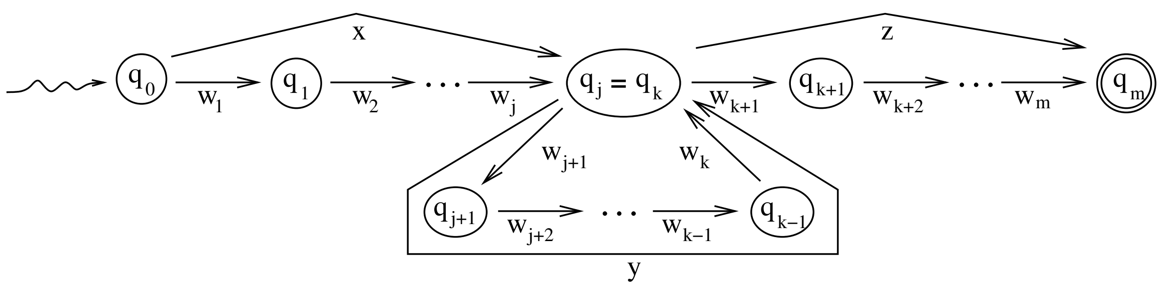
By 1, xy is a prefix of w, and by 2, that prefix has length at most p. Thus, xy is a substring of the prefix a^p of w. It follows that $y = a^k$ for some k. By 3, $k > 0$. Let $i = 0$. By 4, $xy^0 z = xz \in L$. But xz is obtained by deleting a^k from w, hence $xz = a^{p-k} b^p$, which is not in L, contradiction. ∎

4. Prove that any language accepted by a DFA with p live (not dead) states has pumping length p.

Let $M = (\Sigma, Q, q_0, F, \delta)$ be a DFA, where Q consists of p live states, and possibly a dead state. Recall that $F \subseteq Q$ is the set of final states of M, and $\delta : Q \times \Sigma \to Q$ is the transition function of M. As is standard, we extend the second parameter of δ by concatenation, that is, for any $q \in Q$, $\delta(q, \lambda) = q$ and $\delta(q, uv) = \delta(\delta(q, u), v)$ for any $u, v \in \Sigma^*$. Let $L = L(M)$, and let $w \in L$ be of length $m \geq p$, and write $w = w_1 w_2 w_3 \cdots w_m$. During the accepting computation of M with input w, let $q_t \in Q$ be the state of M after reading $w_1 w_2 \cdots w_t$ for any $t \leq m$, thus $\delta(q_{t-1}, w_t) = q_t$. The list of states of M during that computation is $S = (q_0, q_1, \ldots q_m)$, where $q_m \in F$, and q_t must be live. S has length $m + 1$, which is greater than p. The first $p + 1$ terms of S must contain a duplication, since there are only p live states of M. That is, $q_j = q_k$ for some $0 \leq j < k \leq p$,

We let $x = w_1 w_2 \cdots w_j$. (Note that $x = \lambda$ if $j = 0$.) Let $y = w_{j+1} \cdots w_k$, and let $z = w_{k+1} \cdots w_m$. Then

1. $w = xyz$
2. $|xy| = j \leq p$
3. y has length $k - j > 0$.
4. Finally, suppose $i \geq 0$; we need to show $xy^i z \in L$. Note that $\delta(q_j, y) = q_k = q_j$



We illustrate computations of M in the figure. When M reads x its state changes from q_0 to q_j When M reads y from q_j, the computation loops from q_j to q_k, but those are the same state, so that computation is a cycle. When M starts at q_k and reads z its state ends at q_m, a final state.

Now suppose, starting from q_0, M reads $xy^i z$ for some i. The path of the computation through the figure starts at q_0, moves to $q_j = q_k$, loops i times around the cycle, and finally ends at q_m, an accepting state. Thus, $xy^i z \in L$.