Kent Mark

4/15/2021

Cs 311 – Assignment 4

Assignment 4

1. I do not know how to answer this question.

2. To prove that P is closed under the star operation let us consider a language $A \in P$. The following procedure decides A*:

   M = "On input assume $y = y_1 y_2 \dots y_n \in \Sigma$.
   1) If $y = \varepsilon$, output Accept and halt.
   2) Initialize the table $T[i,j] = 0\ for\ i \leq j$
   3) For $i = 1\ to\ n$
      a) $Run\ M\ on\ y_i\ if\ y_i \in L\ then\ set\ T[i,i] = 1$
   4) For $k = 2\ to\ n$
      For i = 1 to n – k + 1
      a) Assume $j = i + k - 1$
      b) Run M on $y_i \dots y_j\ if\ y_i \dots y_j \in L\ then\ set\ T[i,j] = 1$
      c) For $i = 1\ to\ j - 1$
         Set $T[i,j] = 1, if\ T[i,l] = 1\ and\ T[l,j] = 1$
   5) Output Accept if $T[1,n] = 1$ otherwise output Reject

   The algorithm explained above is an example of a polynomial time algorithm. Therefore, P is closed under the star operation.

3. Given the information about the triangle in the prompt we know that r = 3. Let the graph be G = (V, E) where V means the number of vertices and E means the number of edges. We will also let n = abs(V) and m = abs(E). So, enumerating all triplets (x, y, z) where x, y, z ∈ V. If G contains all the edges (x, y), (y, z), (x, z) of the triplet in the edge set E, return True, otherwise return False. There are n 3 = θ(n3) triplets that can be formed from vertices in the graph G. These triplets can be enumerated in O(n3) time. In order to check if all the edges of a particular triplet belong to the set E or not, it will O(m) time. As such, the overall time in order to run the algorithm is O(n3m) or O(|V|3|E|). This time bound is polynomial. Therefore, 3 – ANGLE ∈ P.

4. Consider the language L = {(n, a, b)|n has a factor p in the range a ≤ p ≤ b}. L is in NP since the factor can serve as a certificate. Since the problem prompt indicates that P = NP a polynomial algorithm can be used to solve the language described above. Repeated use of the algorithm allows us to divide our search space in half each time a search is made by figuring out whether or not there is a factor in the range (a, a + b/2), and the algorithm

determines that if there is not there is a factor in another range. The total number of times we have to apply the algorithm is equal to log(n), or in other words O(k) if k is the number of bits of n. So, if we run the program a polynomial number of times the program's algorithm allows us to isolate one factor. From this, since there are O(k) factors as well we can find all the factors in polynomial time.