Iniziato	Friday, 21 June 2024, 09:35
Stato	Completato
Terminato	Friday, 21 June 2024, 10:33
Tempo impiegato	58 min. 7 secondi

Domanda **1**

Completo

Punteggio max.: 6,00

Construct a deterministic TM of the kind you prefer, which decides the following language:

$\square = \{w \in \{0,1\}^* \mid$ w contains an even number of 0s and an even number of

Study the complexity of the TM you have defined.

It will be considere a TM machine with 2 tapes, the first one is the input tape (read only) while the second one is the output tape.

(q_init, (start,start)) --> (q_scan, start, (R,R))
(q_scan, (0, blank)) --> (q_0_odd_1_even, blank, (R,S))
(q_scan, (1, blank)) --> (q_0_even_1_odd, blank, (R,S))
(q_scan, (blank, blank)) --> (q_halt, 1, (S,S))
(q_0_odd_1_even, (0, blank)) --> (q_0_even_1_even, blank, (R,S))
(q_0_odd_1_even, (1, blank)) --> (q_0_odd_1_odd, blank, (R,S))
(q_0_odd_1_even, (blank, blank)) --> (q_halt, 0, (S,S))
(q_0_even_1_odd, (0, blank)) --> (q_0_odd_1_odd, blank, (R,S))
(q_0_even_1_odd, (1, blank)) --> (q_0_even_1_even, blank, (R,S))
(q_0_even_1_odd, (blank, blank)) --> (q_halt, 0, (S,S))
(q_0_odd_1_odd, (0, blank)) --> (q_0_even_1_odd, blank, (R,S))
(q_0_odd_1_odd, (1, blank)) --> (q_0_odd_1_even, blank, (R,S))
(q_0_odd_1_odd, (blank, blank)) --> (q_halt, 0, (S,S))
(q_0_even_1_even, (0, blank)) --> (q_0_odd_1_even, blank, (R,S))
(q_0_even_1_even, (1, blank)) --> (q_0_even_1_odd, blank, (R,S))
(q_0_even_1_even, (blank, blank)) --> (q_halt, 1, (S,S))

This TM works clearly in polynomial time since it iterates just once on every element of the input string (O(n), with n length of the string).

Domanda **2**

Completo

Punteggio max.: 7,00

You are required to prove that the following problem □ is in $\mathbf{NP}$. To do that, you can give a TM or define some pseudocode. The language □ includes precisely those binary strings which are encodings of pairs in the form (G, n) where G is an undirected graph having a path which goes through n distinct edges of G, going through each of them exactly once. Can you say anything else about the complexity of this problem?

To prove that the problem belongs to NP class we must consider the certificate and find a verifier and check that can verify in polynomial time.

INPUT:
  G = (V,E)
  n

CERTIFICATE:
  L -> ordered list of edges of the path

VERIFIER:

```
  edges = V
  if L.length() != n or V.legth() < n:         # If the length of the certificate doesn't correspond to n, the certificate it's wrong for sure.
      return 0                                  # Even if the graph doesn't have enough edges to have a path at least long n it must return 0.

  for (u,v) in L:
    if (u,v) in edges:                # When we find the edge me remove it from the list edges
      edges.remove(u,v)
    else:                             # If we can't find it, either the certificate it's wrong or the edges has already been removed and this means that it pass through that edge at least twice
      return 0
```

The certificate length is O(|L|) and the algorithm of the verifier is O(|L|) as well since it visits every single edge contained in L. In this way we have proved that for sure the given problem is in NP.

Commento:
I understand, but you do not check whether the input is really a path!

Domanda **3**

Completo

Punteggio max.: 7,00

The SubsetSum problem is a well-known NP-complete problem defined as follows: given a finite sequence $n_1, \ldots, n_k)$ of natural numbers and a natural number m, determine if there is a subset I of $\{1, \ldots, k\}$ with $\sum_{i \in I} n_i = m$.

Now, consider the following variation to the problem: given a finite sequence $n_1, \ldots, n_k$ of natural numbers and natural numbers m and p, determine if there is a subset I of $\{1, \ldots, k\}$ with $\sum_{i \in I} n_i = m$ and $\sum_{i \notin I} n_i = p$.

To which complexity class does this new problem belong? Prove your claim.

The variation to the problem is still NP-complete, this can be proved by proving that it belongs to NP class and by reduction of SubsetSum.

First of all it must be proved to be in NP. To do that we need to evaluate the length of the certificate and the verifier must run in polynomial time:

```
INPUT:
  L -> list of integers
  p, m  --> integers
CERTIFICATE:
  I -> the subset
  P -> the subset I-P (all the index not considered in I)

VERIFIER:
  sum_I = 0
  sum_P = 0
  for 0 <= i <= L.length():
     if i in I:
        sum_I = sum_I + L[i]
     else:
        sum_P = sum_P + L[i]
  if sum_I == m and sum_P == p:
     return 1
  return 0
```

The certificate size is O(|L|) and the verifier runs in polynomial time O(|L|) as well since iterates on the list checking once every value.

This variation of SubsetSum requires at least the same complexity of SubsetSum to find a valid set I of indexes whose correspondents integer sum it's equal to m, and then it adds complexity forcing the sum of all the integers not considered in I to be equal to p. This force to find the right combination between I and P and it's like running the SubsetSum more than once to find the right I and its complementary set P. So we are sure that SubsetSum can be reduced to its variation SubsetSum <p SubsetSumVar, thus since SubsetSum belong to NP-hard class and its variation has been proved NP, we can say that also its variation it's a NP-complete problem.

Commento:
The proposed solution is correct, but the proof of NP-hardness is a bit too sketchy.

◄ LAAI - 21062024 - QUESTIONS - LONG

Vai a...

LAAI - 21062024 - PROBLEMS - LONG ►