Problem Set 10

December 3, 2015

34.1-2

Give a formal definition for the problem of finding the longest simple cycle in an undirected graph. Give a related decision problem. Give the language corresponding to the decision problem.

Solution

Formal Definition: For a given undirected unweighed graph G = $< V, E >$ where V is the set of vertexes and E is the set of edges, find the number of vertexes which the longest simple cycle passes within the graph.
Related Decision Problem:
For a given input x = $< V, E, k >$, algorithm A returns true if there's a cycle with length at least k
Language corresponding to the decision problem:
L = $< V, E, k >$: V is the set of vertex, E is the set of edges, and k ¿= 0 is an integer; and there exists a simple cycle in G = ¡V, E¿ consisting of at least k edges

34.1-3

Give a formal encoding of directed graphs as binary strings using an adjacency matrix representation. Do the same using an adjacency-list representation. Argue that the two representations are polynomially related.

Solution

V = {1, 2, 3}
E = {1-2, 1-3, 2-1, 3-1}
Adj-Matrix:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Adjacency-Matrix:
From the above example: 11 000000 011 100 100
Explanation:
11 is v = no. of vertex = 3
then the separator
then the 3*3 array.
The space is just added for better understanding; in the real case there's no space
Adjacency-List:
(use ceil(log(v)) bits to represent one number, output the adj-matrix)
From the above example: 11 000000 10 1011 01 01 01 01
Explanation: 11 is v = no. of vertex = 3
then separator
then 10 = 2 is the number of edges that the first node is connecting to
then following the 2 index of node (1011) that the first element is connecting to.
Same logic follows for the rest.
Proof that the two encoding are polynomially related:
using transfer algorithm it's easy to show that one from can be transformed to another form in polynomial time.
Formally define as below:
E1 = encoding of the $< V, E >$ to adj-matrix as above
E2 = encoding of the $< V, E >$ to adj-list as above
f12 = transforming function transforms E1 form into E2 form
f21 = transforming function transforms E2 form into E1 form
f12 and f21 have been done in the chapter on Graph, that they're O(v^2). Therefore, the encodings are polynomially related.

34.1-4

Is the dynamic-programming algorithm for the 0-1 knapsack problem that is asked for in Exercise 16.2-2 a polynomial-time algorithm? Explain your answer.

Solution

This isn't a polynomial-time algorithm. Consider an encoding of the problem. There is a polynomial encoding of each item by giving the binary representation of its index, worth, and weight, represented as some binary string of length $a = \Omega(n)$. We then encode W, in polynomial time. This will have length $\Theta(\lg W) = b$. The solution to this problem of length $a + b$ is found in time $\Theta(nW) = \Theta(a * 2^b)$. Thus, the algorithm is actually exponential.