


程序代写代做CS编程辅导

Assignment 2

Cohesive Subgraphs, Distributed Graph Processing, and Graph Feature



Summary

Submission	Submit an electronic submission on Moodle (Only the last submission will be used).
Required Files	A .pdf file is  should be ass2_zid.pdf
Deadline	9pm Friday 4 August (Sydney Time)
Marks	20 marks (10% towards your total marks for this course)

WeChat: estutorcs

Assignment Project Exam Help

Email: ~~tutors@163.com~~ **START OF QUESTIONS**

START OF QUESTIONS

Q1 (4 marks)

QQ: 749389476

<https://tutorcs.com>

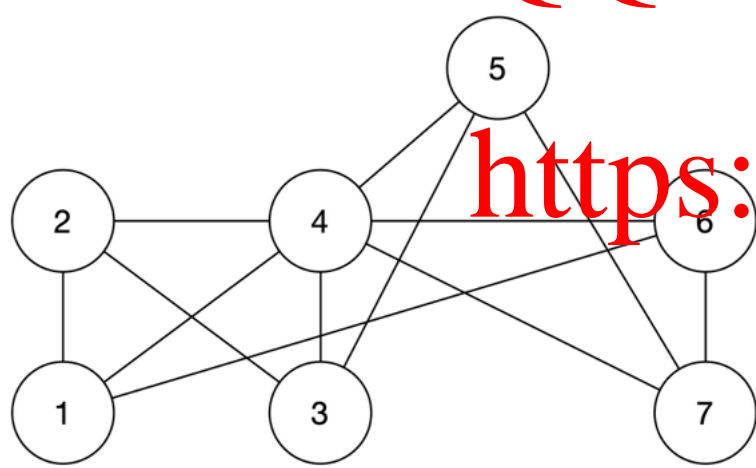


Figure 1

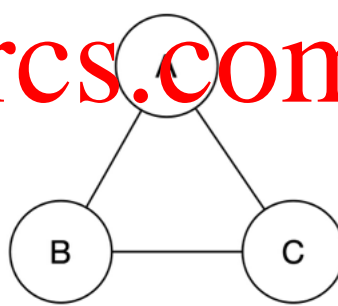


Figure 2

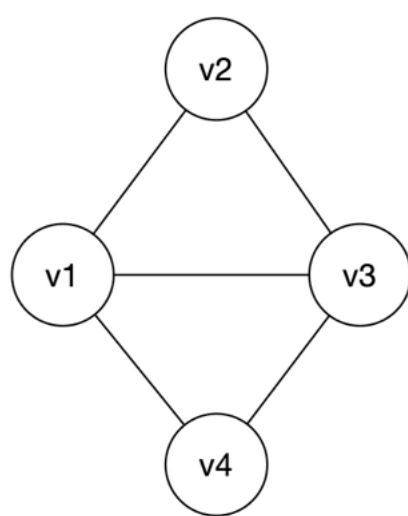


Figure 3

1.1 Are the graphs in Figure 1 and Figure 2 homomorphic? If so, demonstrate a matching instance from vertices in Figure 1 to those in Figure 2. (2 marks)

1.2 Present all unique subgraphs in Figure 1 that are isomorphic to the graph in Figure 3. (2 marks)

Marking for Q1.1: 2 marks are given for the correct answer. 0 mark is given for all other cases.

Marking for Q1.2: 2 marks are given if the result subgraphs are correct, complete, and not redundant. If incorrect subgraphs exist or some correct subgraphs are missing, some points may be given by considering the proportion of correct subgraphs.

Q2 (5 marks)

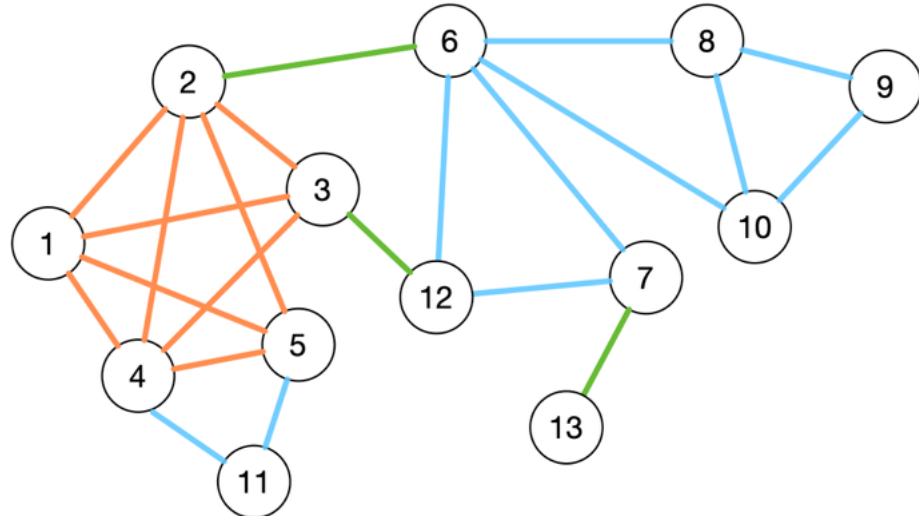


Figure 4

id	edge	Truss Number	id	edge	Truss Number
0	(1,2)	4	11	(6,7)	3
1	(1,3)	4	12	(6,8)	3
2	(1,4)	4	13	(6,10)	3
3	(1,5)	4	14	(6,12)	3
4	(2,3)	4	15	(7,12)	3
5	(2,4)	4	16	(8,9)	3
6	(2,5)	4	17	(8,10)	3
7	(3,4)	4	18	(9,10)	3
8	(4,5)	4	19	(2,6)	2
9	(4,11)	3	20	(3,12)	2
10	(5,11)	3	21	(7,13)	2

Table 1

Consider a graph G with n vertices and m edges. We have a precomputed array L storing the truss number of each edge in G . Note that the truss number of an edge e is the largest integer k such that there exists a k -truss containing e . Each item in the array is a tuple including two vertex IDs of the edge and the truss number. Items in the array is already sorted in non-increasing order of the truss number.

Design an algorithm that finds the vertex set of every (connected) k -truss. You may show your algorithm by presenting the **pseudocode**. The input of your pseudocode includes a graph G represented by adjacency lists, the precomputed array L , and an integer k . Analyze the **time complexity** of your algorithm.

Take the graph G shown in Figure 4 and the corresponding array L shown in Table 1 as an example. With input $k=3$, your algorithm should return a two-dimensional array: $[[1,2,3,4,5,11],[6,7,8,9,10,12]]$, which indicates two resulting 3-truss. Note that both k -truss vertex sets and vertices in each k -truss can be in any order. For example, $[[10,6,7,8,9,12],[11,3,1,2,4,5]]$ is also a valid result.

Marking for Q2: Two factors are evaluated in marking: (1) How good is your algorithm; (2) Does your time complexity match your algorithm. Full marks are given if your algorithm achieves our expected efficiency and your time complexity corresponds your algorithm. To identify the efficiency of your algorithm, the marking tutor will read your pseudocode and calculate a corresponding time complexity. Then, the marking tutor will compare the calculated time complexity with your submitted time complexity. For the pseudocode, you need to clearly present operations in each step. For complexity analysis, you need to provide correct, simplified, and tight time complexity (refer to [Topic 3.2](#)).

Q3 (5 marks)

Consider the basic distributed algorithm to compute connected components in Pregel that has been mentioned in [lecture \[topics 4\]](#):

```
virtual void Compute(MessageIterator* msgs) {
    int minID = GetValue();
    for (; !msgs->done(); msgs->Next())
        minID = min(minID, msg->Value());
    }
    if (minID < GetValue()){
        *MutableValue() = minID;
        SendMessageToAllNeighbors(minID);
    }
    VoteToHalt();
}
```

Write the pseudocode of a combiner implementation to optimize the message passing.

Marking for Q3: 5 marks are given for the correct answer. 0 mark is given for all other cases.

Q4 (6 marks)

Consider the graph in Figure 5,

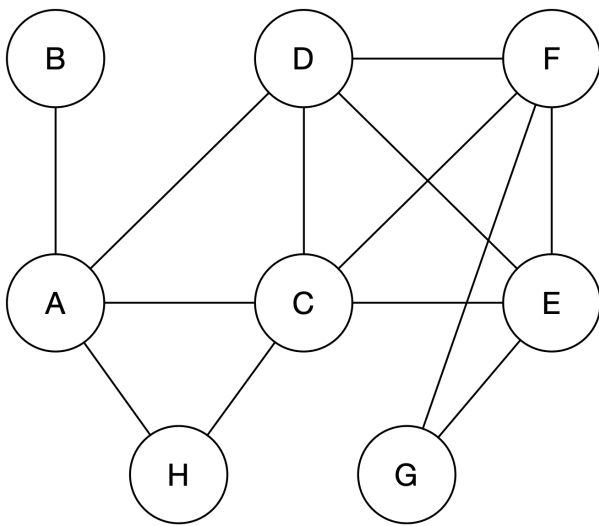


Figure 5

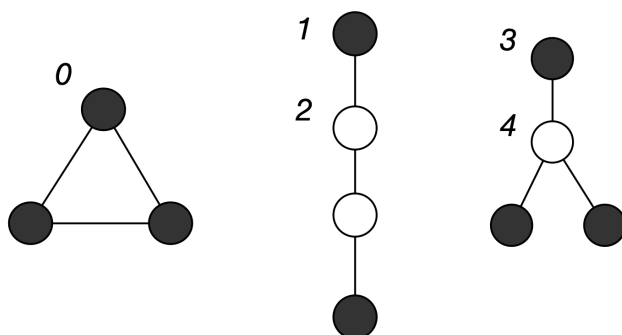


Figure 6

4.1 Compute the betweenness centrality and closeness centrality of nodes C and D. (3 marks)

4.2 Given the graphlets in Figure 6, derive the graphlet degree vector for nodes E and A. Only consider the induced matching instances. (3 marks)

Marking for Q4.1: 1.5 marks are given for correct betweenness centrality values. 1.5 marks are given for correct closeness centrality values.

Marking for Q4.2: 3 marks are given if the vector is correct. 2 marks are given if 4/5 values in the vector are correct. 1 mark is given if 2/5 values in the vector are correct.

END OF QUESTIONS