Week 7 Exercises

HamiltonEuler.txt

Consider the following graphs.

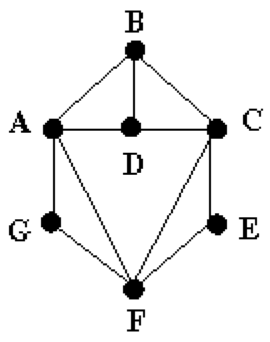

Figure 1

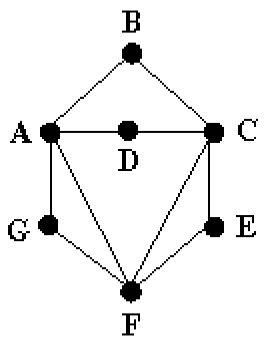

Figure 2

a. In Figure 1:
    i. Is there an Eulerian cycle?
    ii. Is there an Eulerian path?
    iii. Is there a Hamiltonian cycle?
    iv. Is there a Hamiltonian path?
b. In Figure 2:
    i. Is there an Eulerian cycle?
    ii. Is there an Eulerian path?
    iii. Is there a Hamiltonian cycle?
    iv. Is there a Hamiltonian path?

Missing code in the Linked-List ADT LL.c

1. ***function_putHead.c***

2. ***function_getHead.c***

3. ***function_getTail.c***

These 3 functions are missing from the linked-list ADT LL.c in the Linked List lecture.

i. First write **putHead.c** as it is similar to, but simpler than, **putTail**, which is shown in the lecture notes

ii. Then write **getHead.c**, which conceptually at least, reverses what **putHead** has done, which you've just written
  - a 'design decision' needs to be made about what to do if the linked list is empty. There are basically 3 options:
    1. report it and exit the program
    2. report it and return 0 (say). This is the option I chose.
    3. just ignore it and return 0 (say)

iii. Finally, write **getTail.c**, which reverses the action of **putTail**
  - the same 'design decision' needs to be made here as ii.

iv. You can simplify **testLL.c** in the lecture notes to test the functions as you write them, but you will need to remove references to the functions that you have not yet written.
  - The complete **testLL.c** should work when you are finished

Missing code in the Graph Adjacency Matrix ADT GraphAM.c

4. ***function_AM_freeGraph.c***

5. ***function_AM_isEdge.c***

There are 2 functions missing from the graph ADT GraphAM.c in the week 7 Graph lecture.

- *int isEdge(Graph g, Edge e)* returns *true* if the edge *e* exists in the graph
  - as it is an adjacency matrix, this is very simple to determine
- *Graph freeGraph(Graph g)* returns the pointer to graph after freeing all its memory
  - the adjacency matrix is two dimensional, so in effect, each row of data in the matrix must be freed

Missing code in the Graph Adjacency List ADT GraphAL.c

6. ***function_AL_freeGraph.c***

7. ***function_AL_isEdge.c***

The same 2 functions are missing from the graph ADT GraphAL.c in the week 7 Graph lecture.

- *int isEdge(Graph g, Edge e)* for the *List* version is more complicated because each vertex has a linked-list of vertices it is attached to.
  - ... so determining whether an edge <v w> exists requires you to search for *w* in vertex v's linked list
- *Graph freeGraph(Graph g)* is also more complicated for the *List* version as all the linked lists for all the nodes need to be freed: so you need a double loop, one for the vertices, the other for the linked lists attached to the nodes.

8. ***function_AL_removeV.c***

This is a more challenging exercise.

Some observations:

- Notice *removeV()* is *static*, which mean it is allowed to be called only from within the ADT.

- The name of the function suggests the aim is to remove a vertex.
    - We see it is called by the interface function *removeE()*, which removes an edge.
        - For the adjacency matrix ADT, the function *removeE()* does not call such a function.
            - The first question you should ask yourself is *why?*: *what is special about the matrix ADT?*
                - If you can answer this question you should be ready to write *removeV()* for the list ADT.

More observations:

- The function is called twice: is it obvious why?
- It returns *success* if it is successful, and we see that the number of edges *nE* is then decremented.
    - If it is not successful, then there is no change to the graph.

Week7Exercises (2019-07-17 11:51:42由AlbertNymeyer编辑)