# FIT2004 S1/2017: Assignment 5 (Weight: 6 Marks)

THIS PRAC IS ASSESSED! Interview during Week 12 Lab

**Submission deadline:** Midnight 11:59pm Sun 21 May 2017

**CLASS:** This programming exercise has to be prepared in advance, and submitted via unit's Moodle page by 11:59pm Sunday 21 May 2017. Implement this exercise strictly using **Python programming language**. During your **assigned** Week 12 lab, your demonstrator will interview you based on your **submitted** version by downloading it from Moodle. This practical work is marked on the performance of your program **and also on your understanding of the program**. A perfect program with zero understanding implies you will get **zero** marks! "Forgetting" is not an acceptable explanation for lack of understanding. You must write all the code yourself, and may not use any external library routines that will invalidate the assessment of your learning. The usual input/output and other basic routines are exempted

**DEMONSTRATORS** are not obliged to mark programs that do not compile or that crash. Time allowing, they will try to help in tracking down errors, but they are not required to mark programs in such a state, particularly those that do not compile. Therefore keep backup copies of working partial solutions.

**OBJECTIVE:** This assignment has two **independent** tasks. The first task will test your ability to implement a solution to a common path problem that arises in many graphs. The second task deals with a combinatorial enumeration problem (with many practical uses), and will test your ability to solve new algorithmic problems.

## Background on Task 1

### Eulerian cycles

An Eulerian cyle is a path in a graph that **start and ends at the same vertex** and goes through each edge in the given graph **exactly once**. Note, the constraint is **only** on the edges being visited exactly one. The vertices can however be visited multiple times.

For a graph $G(V, E)$ containing an Eulerian cycle, Carl Hierholzer in 1873 proposed a simple algorithm to find such a cycle that can be implemented to run in $O(|E|)$-time. His approach is based on the observation that an Eulerian cycle can potentially be decomposed into smaller cycles that contain disjoint edges (i.e., no edges are common between the decomposed cycles). Hierholzer's method of finding an Eulerian cycle in $G$ uses the following steps:

(I) Start from any arbitrary vertex $s$ in the given graph $G$.

(II) Find a path (i.e. cycle) that starts **and** ends in $s$ by traversing along unvisited edges. If all edges in the graph have been visited along this current cycle/path, print that path and terminate.

(III) However, if the graph $G$ still has unvisited edges:

- Choose any vertex $v$ on the current cycle/path with unvisited edges.
- Traverse along the unvisted edges in $G$ starting from $v$ and find another path (i.e. cycle) starting and ending in $v$. Note that this cycle will contain a set of edges that is disjoint from the set of edges in the earlier cycle.
- Merge the two paths/cycles at the common vertex $v$ to get a larger (current) cycle.

(IV) Iterate the above step (III) until all the edges in $G$ have been visited (traversed). Print the Eulerian cycle and terminate.

## $\mathcal{D}$-Graph

In Task 1, you will construct a special graph, which we will call the $\mathcal{D}$-graph, which always contains an Eulerian cycle. This special $\mathcal{D}$-graph is defined using the integer parameters $m$ and $n$. The vertices of this $\mathcal{D}$-graph are **all possible** strings of size $n$ that can be generated from an alphabet of size $m$. Clearly, there will be $m^n$ such vertices.

Any two vertices $u$ and $v$ in the $\mathcal{D}$-graph have a **directed edge** (from $u$ to $v$) **if and only if** the **last** $n-1$ letters of the string represented by the vertex $u$ is exactly same as the **first** $n-1$ letters represented by the vertex $v$. Note, since both $u$ and $v$ vertices represent $n$-sized strings that overlap in their $n-1$ letters (affixes), the traversal along the (directed) edge $\langle u, v\rangle$ extends the string in $u$ by the last character of the string in $v$. In the same way, any path along edges in this $\mathcal{D}$-graph results in a string that gets bigger by one at each traversal. To see all this more clearly, consider the following example:

```
Let m = 2 (assume the first two letters of English alphabet, {A,B})
Let n = 3 (we are looking at 3 letter strings from the above alphabet)
Therefore, number of vertices in this special graph = m^n = 2^3 = 8.
These vertices are:
               v1 = "AAA", v2 = "AAB", v3 = "ABA", v4 = "BAA",
               v5 = "ABB", v6 = "BAB", v7 = "BBA", v8 = "BBB".
Notice that (among other such edges) there is a directed edge from
vertex v5 = "ABB" to vertex v7 = "BBA". The last  n-1 (i.e., 2) letters
of v5 is exactly the same as the first n-1 (i.e., 2) letters of v7:
                      v5 = A B B
                      v7 =   B B A
                           -------
                   string  A B B A
Based on the above, we can see that the edge between v5 and v7 implicitly
represents the string  "ABBA" which comes from extending v5 by the last
letter in v7.  Consequently, any path in the D-graph also corresponds to
a string. For example, v5 --> v7 --> v4 --> v2 corresponds to the string:
                      v5 = A B B
                      v7 =   B B A
                      v4 =     B A A
                      v2 =       A A B
                           -----------
                   string  A B B A A B
```

## Background for Task 2

A set containing $n$ elements has $2^n$ subsets. For example, a set of 4 elements $\{a, b, c, d\}$ has $2^4 = 16$ subsets:

```
subset  1: {},
subset  2: {a},
subset  3: {b},
subset  4: {c},
subset  5: {d},
subset  6: {a,b},
subset  7: {a,c},
subset  8: {a,d},
subset  9: {b,c},
subset 10: {b,d},
subset 11: {c,d},
subset 12: {a,b,c},
subset 13: {a,b,d},
subset 14: {a,c,d},
subset 15: {b,c,d},
subset 16: {a,b,c,d}
```

The above listing is one of the many orders in which these subsets can be enumerated and listed. Another order of listing these subsets would be an enumeration under the constraint that any two successively listed subsets differ by **exactly one element**. Note, in the above listing, the subsets 1 to 5 obey this constraint, but the difference between the successive subsets 5 and 6 violates this constraint. Other voilations are: subsets 8 and 9, subsets 11 and 12, subsets 16 and 1 (if you consider these subsets in a cyclic order, where subsets 16 and 1 are successive). An example of such an enumeration that obeys the above constraint is given below:

```
subset  1: {},
subset  2: {a},
subset  3: {a,b},
subset  4: {b},
subset  5: {b,c},
subset  6: {a,b,c},
subset  7: {a,c},
subset  8: {c},
subset  9: {c,d},
subset 10: {a,c,d},
subset 11: {a,b,c,d},
subset 12: {b,c,d},
subset 13: {b,d},
subset 14: {a,b,d},
subset 15: {a,d},
subset 16: {d}
```

Clearly, each `subset` $i$ differs from `subset` $i + 1$ in exactly one element, including (cyclically) the `subset` 16 and `subset` 1.

## Assignment

Based on the background information, complete the following two independent tasks: (please strictly conform to the file naming specifications stated below)

**Task 1** Write a program entitled `task1.py` that accepts the graph parameters $m$ and $n$ as arguments. Using these parameters your program must:

- Construct the corresponding $\mathcal{D}$-graph. (Given the combinatorial nature of the vertex and edge sets of any specified $\mathcal{D}$-graph, constrain your input program parameters to $2 \leq m \leq 5$ and $2 \leq n \leq 3$.) Assume your alphabet is the first $m$ letters in the English alphabet.
- Implement the Hierholzer's approach (introduced in the background) to find an Eulerian cycle in the above constructed $\mathcal{D}$-graph.
- Output to a file entitled outputTask1.txt containing just one line giving the full string corresponding to that Eulerian cycle of the input $\mathcal{D}$-graph. (No special formatting required for output of this task. Simply write the string corresponding to the Eulerian cycle/path.)

**Task 2** Write a program entitled `task2.py` that accepts a parameter $n$ $(2 \leq n \leq 10)$ as its argument. Using this parameter, your program must:

- Enumerate all subsets of a set containing $n$ elements in the order prescribed (see second listing of the subsets) in the background for task 2.
- Output this enumeration to a file outputTask2.txt. Follow the same format as shown in the background.

```
-------===============oOo==========-------
                     END.
BEST OF LUCK FOR ALL YOUR FUTURE ENDEAVOURS
-------===============oOo==========-------
```