Discrete optimization
Prof. F. Eisenbrand

Thursday, May 16, 2019
EPFL

The problem can be submitted until May 24, 12 :00 noon, either at the exercise session or into the box in front of MA C1 563.

Student(s) [1] :

Question 1 : *The question is worth 5 points.*

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 *Reserved for the corrector*

Consider a directed graph $D = (V, A)$ where every vertex $v \in V$ is reachable by any other vertex. A walk in D is called *Euler tour* if it traverses every edge exactly once and it starts and ends at the same vertex. Design an algorithm which checks if D admits an Euler tour and, if so, finds it in time $O(|V| + |A|)$.

Hint : You might want to prove that $D = (V, A)$ has an Euler tour if and only if for every $v \in V$ the number of arcs in A which have v as the head is equal to the number of arcs with v as the tail.

Sol.: *We first prove the above statement.*

($\Rightarrow$) *We can represent an Euler tour as $T = (v_0, a_1, v_1, \ldots, v_{m-1}, a_m, v_m)$. Since every arc is traversed exactly once in T, the statement follows from a simple fact. For every appearance of a vertex v in T we have one arc with v as head and one with v as tail.* [2]

($\Leftarrow$) *Use induction on $|A|$. For $|A| = 0$ and $|A| = 2$ the statement is trivial. Let $|A| > 2$, start with an arbitrary vertex v and traverse an arbitrary outgoing arc reaching some u. Continue from u and repeat the process until closing a cycle, denote it with C. By removing the edges of C from $|A|$, one obtains a graph D' where every vertex has the number of ingoing arcs equal to the number of outgoing arcs. This implies that every connected component in the underlying simple graph of D' induces a strongly connected component in D'.*

By applying the inductive hypothesis to each of those components we get a set of Euler tours $T_1, T_2, \ldots, T_l$. Those tours are merged with C into an Euler tour on D in the following way. Represent C as $T(C) = (v_0, a_1, v_1, \ldots)$. For each $i \in [l]$, $T(C)$ and T_i share a vertex, denote it with $w(i)$. Represent each T_i so that $w(i)$ is its starting and ending vertex. For $i \in [l]$, replace an appearance of $w(i)$ in $T(C)$ with T_i.

Checking, whether the number of ingoing and outgoing arcs is equal for every vertex, can be easily done in $O(|A|)$ time. Then, the second part of the proof can be used as an algorithm to construct an Euler tour. Finding a cycle C is done in time $O(|A|)$, and by induction building each of T_i can be done in $O(|T_i|)$ operations, where $|T_i|$ is the number of arcs in T_i. Finding a common vertex $w(i)$ and replacing it with $|T_i|$ in $T(C)$ can also be done in time $O(|T_i|)$. Thus the total complexity is $O(|A|) + \sum_{i \in [l]} O(|T_i|) = O(|A|)$.

1. You are allowed to submit your solutions in groups of at most three students.
2. Specially, $v_0 = v_m$ is considered as a single appearance.