Summary on Lecture 9, January 22, 2016

Finding an Euler Circuit

We repeat the algorithms from the previous lecture. Let $H = (V(H), E(H))$ be a graph with all verices of even degree and let $v \in V(H)$ be a vertex with positive even degree. For a graph G and an edge e, we define a graph $G \setminus \{e\}$ which has exactly the same vertices as G and the same edges except given edge e. We say that the graph $G \setminus \{e\}$ is given by removing e from $E(G)$. Here is the algorithm:

Circuit (H, v)

> `Choose an edge` e `with endpoint` v
> `Let` $P := (e)$ `and remove` e `from` $E(H)$
> `while there is an edge at the terminal vertex of` P `do`
>   `Choose such an edge` e `and add it to the path:`
> $P := (P, e)$ `and remove it from` $E(H)$`,`
> `return` P

Here we repeat the algorithm which produces and Euler circuit.

EulerCircuit $G = (V, E)$ ($\deg v$ is even for each $v \in V$)

> `Choose a vertex` $v \in V(G)$
> `Let` C `:=`**Circuit** (G, v)
> `while length`$(C) < E(G)$ `do`
>   `Choose a vertex` w `in` C `of positive degree in` $G \setminus C$`.`
>   `Attach` **Circuit**$(G \setminus C, w)$ `to` C `at` w `to obtain a longer circuit` C`.`
> `return` C

Proof that EulerCircuit $G = (V, E)$ **works.** We consider the statement:

"The path C is a closed path in G with no repeated edges"

We claim that this statement is a loop invariant, i.e., if this statement holds before executing the loop, then it will remain true after executing the loop.

Indeed, let C be a closed path in G with no repeated edges, and $w \in C$ be a vertex with positive degree in $G \setminus C$, and C' be a closed path in $G \setminus C$ with no repeated edges, then attaching C' to C at w gives new closed path in G with no repeated edges:

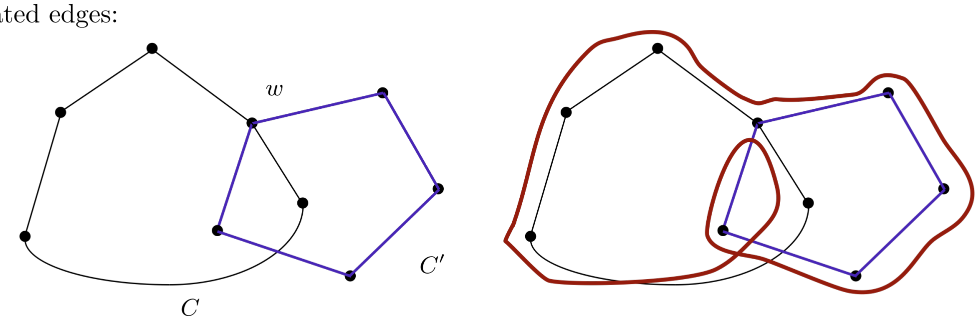

Fig. 8. Attaching C' to C at w

Now it is also clear that if the algorithm does not break down somewhere, then this algorithm will produce an Euler circuit for G, because the path C will be closed at the end of each pass through the loop, the number of edges remaining will keep going down, and the loop will terminate with all edges of G in C.

Of course, we have to show that there always be a place to attach another closed path to C, i.e., we have to explain why there exists a vertex w on C of positive degree in $G \setminus C$? In other words, can the instruction

"Choose a vertex w on C of positive degree in $G \setminus C$"

be executed?

The answer is yes, unless the path C contains **all the edges of** G, in which case the algorithm stops. Here's why. Suppose that e is an edge not in C and that u is a vertex of e. If C goes through u, then u itself has positive degree in $G \setminus C$, and we can attach at u. So suppose that u is not on C. Since G is connected, there is a path in G from u to the vertex v on C.[1] Let w be the first vertex in such a path that is on C (then $w \neq u$, but possibly $w = v$). Then the edges of the part of the path from u to w don't belong to C. In particular, the last one (the one to w) does not belong to C. So w is on C and has positive degree in $G \setminus C$.

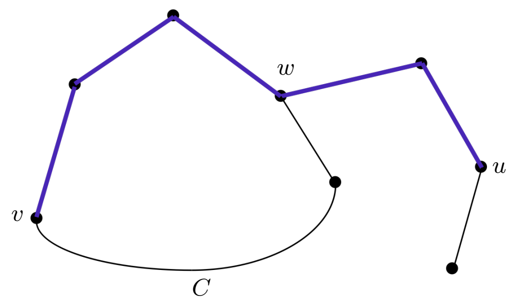

Fig. 9. Finding w with positive degree in $G \setminus C$

Now we also have to show that the instruction

"Construct a simple closed path in $G \setminus C$ through w"

can be executed. Thus the proof will be complete once we show that the following algorithm works to construct the necessary paths. Now we have to show that the algorithm **Circuit** (H, v) works as well. We write it again:

Circuit (H, v)

Input: A graph H in which every vertex has even degree, and a vertex v of positive degree
Output: A simple closed path P through v

```
Choose an edge e of H with endpoint v
Let P := (e) and remove e from E(H)
while there is an edge at the terminal vertex of P do
     Choose such an edge e and add it to the path:
     P := (P, e) and remove it from E(H),
return P
```

Proof that Circuit (H, v) **works.** We want to show that the algorithm produces a simple closed path from v to v. Simplicity is automatic, because the algorithm deletes edges from further consideration as it adds them to the path P. Since v has positive degree initially, there is an edge e at v to start with. Could the algorithm get stuck someplace and not get back to v? When P passes through a vertex w other than v, it reduces the degree of w by 2 since it removes an edge leading into w and one leading away. Thus the degree of w stays an even number.[2] Hence, whenever we have chosen an edge leading into a w, there's always another edge leading away to continue P. The path must end somewhere, since no edges are used twice, but it cannot end at any vertex other than v. □

[1] Here's where we need connectedness!
[2] Here's where we use the hypothesis about degrees.