MT7143_Problem Q_2

AKSHITA SAWHNEY

2] For a connected multi-graph G, G is Eulerian if and only if every vertex has even degree. Prove this statement.

Proof :

We need to know the following definitions before we prove:

1) A walk from let say a vertex u to a vertex v will be closed if they start and end at the same vertex that is u = v

2) A trail is a walk with no repeated edges e. It is possible that a trail repeat vertices but cannot repeat edges.

3) Euler trail → It is a trail that visits every edge only once.

4) Euler tour → Its an Euler trail that is closed.

This graph with Euler tour is called Eulerian.

Proof:

Now lets assume:

G is a connected Euler graph, so that means that it has Euler tour / Euler circuit.

Let w be an Euler tour which goes from a vertex u goes back to same vertex u ~~as u~~.

→ Let $w: u \xrightarrow{*} u$ be an Euler tour.

Lets assume v to be any vertex not equal to u of the graph G that occurs k times in w.

→ Let $v \neq u$ of G that occurs k times in w.

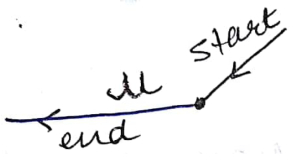
Now if we have a vertex that occurs k times in the tour then that means that every time an edge arrives at edge v we have to leave ~~to~~ an another edge.

Every time we arrive on one edge to a vertex we have to leave at another edge from the vertex as this vertex is a part of an Eulerian circuit.

$\therefore\ d(V) = 2k$

↓
degree of the vertex

Now vertex u is still having even degree because if you consider a tour let that comes accross the u. Then that W will have to enter the vertex and exit. Here we started and ended at u.



So also $d(u) = \text{even}$.

Now suppose degree is even we will show that there is an Euler tour / circuit by induction.

→ for a Graph with two vertices and 2 edges within them will obviously be a Eulerian.

→ Let G be a non-trivial connected graph whose vertices all have even degree.

→ Let W be <u>a longest trail</u> in G (cannot be further extended)

$$W: v_0 e_1 v_1 e_2 v_3 e_3 \ldots \ldots e_{l-1} v_{l-1} e_l v_l$$

$v_0 \xrightarrow{l} v_l$ → so W is the longest trail from v_0 to v_l of length l.

→ Now we know that W is the longest ~~tt~~ so it cannot extended further that means that when we have reached v_l all of the edges that exit out of vertex v_l have to be used in W else could be extend. Also we cannot simply end at v_l

Because if we end at V_e it will have odd degree which is not possible.

→ Since it has an even degree and since we have used up all the edges that come out of it ~~it~~ has to be the V_e to be equal to V_0.

$V_e = V_0$ → since V_e have evendegree and all incident edges are used in W.

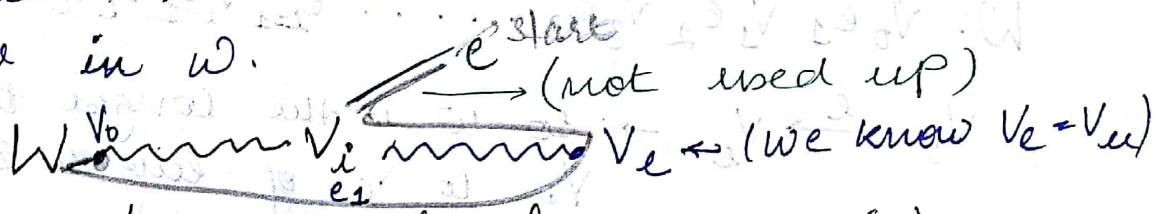
If $V_e \neq V_0$ and V_e occurs k times in W then

$d(V_e) = 2(k-1) + 1$

<u>Thus W is a closed trail.</u>

Suppose W is <u>not</u> a Euler tour/circuit

Since G is connected and W is not a Euler tour means that there will still be edges that touch W and are not used/traversed in W.



Let $V_i - e$ be an edge $f = V_i u \in E(G)$ but $\notin E(W)$

Then if we use edge f to traverse the path we get → $f e_{i+1} \ldots e_e e_1 \ldots e_i$ we get a <u>longer walk</u>

This is a contradiction ~~that~~ as we cannot have a longer walk.

[2] A connected graph G is Eulerian if and only if its edge set can be decomposed into cycles.

⇒ Let us suppose there is a cycle decomposition of G

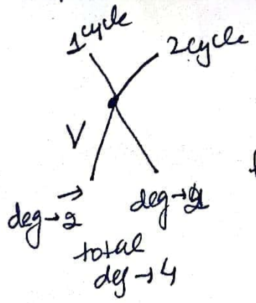
Take any vertex in the graph and we look at the degree of the vertex $v \in V(G)$

Now the cycle decomposition we took could be disconnected as well

1. So either v is isolated ⇒ so for this $\underline{d(v) = 0}$

2. If not isolated then it will lie on some cycle, it could also lie on more than 1 cycle.

→ Let the ~~# of~~ cycles be k.



we observe that regardless of how many cycle there will be twice as many edges which are incident with v

$$\underline{d(v) = 2k}$$

for all $v \in V(G)$ → $d(v)$ is even

and as proved before that a graph with all vertices of even degree is Eulerian.

Now let us suppose that the graph is Eulerian and prove that it will have its edge set decomposed as cycles.

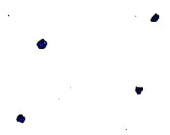
→ For this we use <u>strong induction</u>

If we know that the graph is Eulerian, we can say that all the vertices will have even degree.

<u>Base case</u>:

If we consider that there are no edges:

$|E(G)| = 0$



The decomposition will be the trivial empty decomposition as we do not have any edges.

$\mathcal{F} = \{ \}$

<u>Inductive Hypothesis</u>:

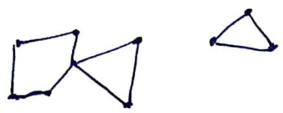
suppose that for all Eulerian graph on $< m$ edges there exist a cycle decomposition.

So now we prove that even for m edges will hold true.

⇒ Take G to be even $|E(G)| = m$.

G may not necessarily be connected so

 → isolated vertices $d(v) = 0$

 → not isolated vertices different components.

We will mainly consider

$X =$ the set of $V(G)$ with degree > 0

(not isolated vertices)

Let F be subgraph of G induced on the set X

$F = G[X]$

So we know that because G is even F is also having even degrees.

$F \rightarrow$ even graph

but all vertices have degree ≥ 2

We know that if in graph every vertex has degree ≥ 2, then there exists a cycle in the graph.

Therefore, there is a cycle in F, which we call to be C.

We now prove our hypothesis by removing the cycle C

Take $G' = G \setminus E(C)$

Since G is even graph so G' also is even graph as Even − Even = Even

degree of original graph G

degree of cycle will be $0 \mid \geq 2$

~~[illegible]~~.

G' is even graph and edges will be $< m$.

By the inductive hypothesis G' has cycle decomposition and we can call those family of cycles to be C'.

So now

$C = C \cup C'$ $\Longrightarrow$ is the cycle decomposition of original graph G.

C: cycle we removed from the original graph

C': the cycles we considered in G' (any no. of cycles)

Hence here we prove that the Graph with m edges is also decomposed into cycles.

Therefore, we finally prove that a Graph admits a cycle decomposition if and only if all its vertices have even degree ~~and~~ which means that the graph is eulerian.

MT17143
Problem - 1

The complexity of the code given for the shortest superstring is

$\boxed{O(n^3)}$

This is because there are 3 ~~each~~ nested loop: 1 to pick the k-mer from the input list and the next two are nested to find the overlap.

We <u>will not</u> get the input as the output for any k value.

This is because we are finding the shortest superstring which will vary according to the value of k. As ~~if~~ the varying the value of k will give overlaps leading to the change in output.

Also if we take a string say: ~~AAAAAAAA~~
and the value of k = 5
o/p → AAAAA.

Also if we take the same i/p as ATACGATATTTAC
and k = 3
o/p ATACGATTTA