Lecture 04
The Chinese Postman Problem and Graphic Sequences

Dr. Rajiv Misra
Associate Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Preface

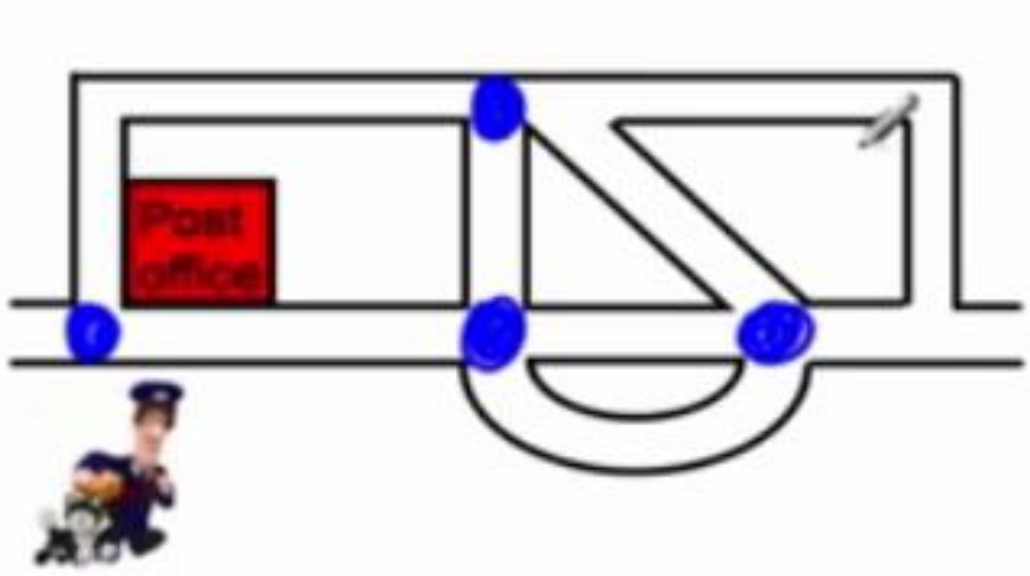
Recap of previous Lecture:

- In the previous lecture, we have discussed eulerian circuits, fundamental parameters of a graph i.e. vertex degrees, counting and extremal problems.

Content of this Lecture:

- In this lecture, we will discuss the application of eulerian circuit i.e. chinese postman problem and graphic sequences .

The Chinese Postman Problem



Overview

- Routing problems are concerned with finding ways to route the delivery of goods or services to an assortment of destinations; like mail or garbage collection.

- A weighted graph is a graph in which the edges are assigned positive numbers called weights. The weights can represent distances, times, or costs.

- The Chinese postman problem is the problem of finding a trip that covers all the edges of a weighted graph and that is optimal. These problems are a generalization of Euler Circuit problems.

Real-life Applications

- There could be a road network which must be travelled by a mail carrier delivering mail to buildings along the streets, a snowplow which must clear snow from each lane of the streets, a highway department crew which must maintain each street, a police patrol car which makes its rounds through all streets several times a day.
- In each case, the person or vehicle must travel each street at least once. In the best situation, where every road intersection (i.e. vertex) has even degree, no retracing is needed. In such a case, any Euler Circuit solves the problem.
- But it is rarely the case that every vertex in a road network is even.

Application: Chinese Postman Problem

- A mail carrier must traverse **all edges** in a road network, starting and ending at the Post Office. The edges has nonnegative **weights representing distance** or time.
  - We seek ***a closed walk of minimum total length*** that uses all the edges.
  - This is the **Chinese Postman Problem**, named in honor of the Chinese mathematician Guan Meigu [1962], who proposed it.

Application: Chinese Postman Problem continue

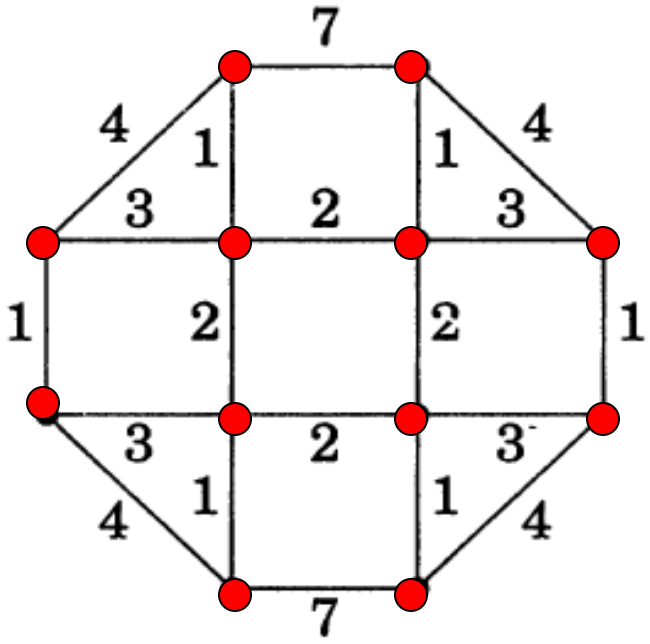
- **If every vertex is even**, then the graph is Eulerian and the answer is the sum of the edge weights.
- **Otherwise**, we must repeat edges. Every traversal is an Eulerian circuit of a graph obtained by duplicating edges.
- Finding the shortest traversal is equivalent to **finding the minimum total weight of edges whose duplication will make all vertex degrees even .**

Application: Chinese Postman Problem continue

- We say “duplication” because we need not use an edge more than twice.
- If we use an edge three or more times in making all vertices even, then deleting two of those copies will leave all vertices even.
- There may be many ways to choose the duplicated edges. ■

Example

- In the example below, the eight outer vertices have odd degree. If we match them around the outside to make the degrees even, the extra cost is 4 + 4 + 4 + 4 =16 or 1 + 7 + 7 + 1 = 16. We can do better by using all the vertical edges, which total only 10.



Contd…

- Adding an edge from an odd vertex to an even vertex makes the even vertex odd.
- We must continue adding edges until we complete a trail to an odd vertex.
- The duplicated edges must consist of a collection of trails that pair the odd vertices.

Contd…

- **Edmonds and Johnson [1973]** described a way to solve the **Chinese Postman Problem**. If there are only two odd vertices, then
  - We can use Dijkstra's Algorithm to find the shortest path between them and solve the problem
- If there are ***2k*** odd vertices, then
  - Use **Dijkstra's algorithm** to find the shortest paths connecting each pair of odd vertices
  - Use these lengths as weights on the edges of K_{2k}
  - Find the minimum total weight of k edges that pair up these ***2k*** vertices. This is a weighted version of the maximum matching problem.

Chinese Postman Algorithm

Step 1 List all odd vertices.

Step 2 List all possible pairings of odd vertices.

Step 3 For each pairing find the edges that connect the vertices with the minimum weight.

Step 4 Find the pairings such that the sum of the weights is minimised.

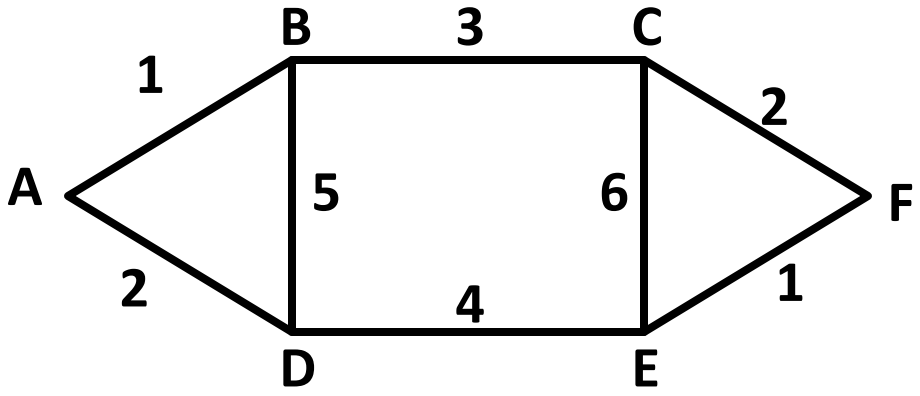
Step 5 On the original graph add the edges that have been found in Step 4.

Step 6 The length of an optimal Chinese postman route is the sum of all the edges added to the total found in Step 4.

Step 7 A route corresponding to this minimum weight can then be easily found.

Example

- A mail carrier has to deliver mail to all streets in a subdivision. The mail carrier has to start and end at point A, where the post office is. The weight of each edge represents the length of each street. What is the most efficient route for the mail carrier?



Solution

Step 1 The odd vertices are BCDE

Step 2 There are three ways of pairing these odd vertices

Step 3 The shortest way of joining pairs is using the paths.

Odd Nodes

BCDE

Pairs

BC-DE

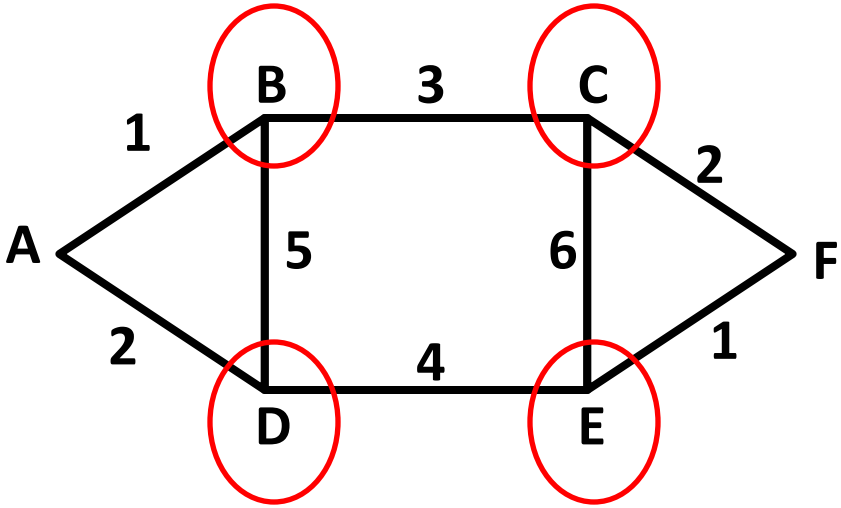
3 4 7

BD*-CE*

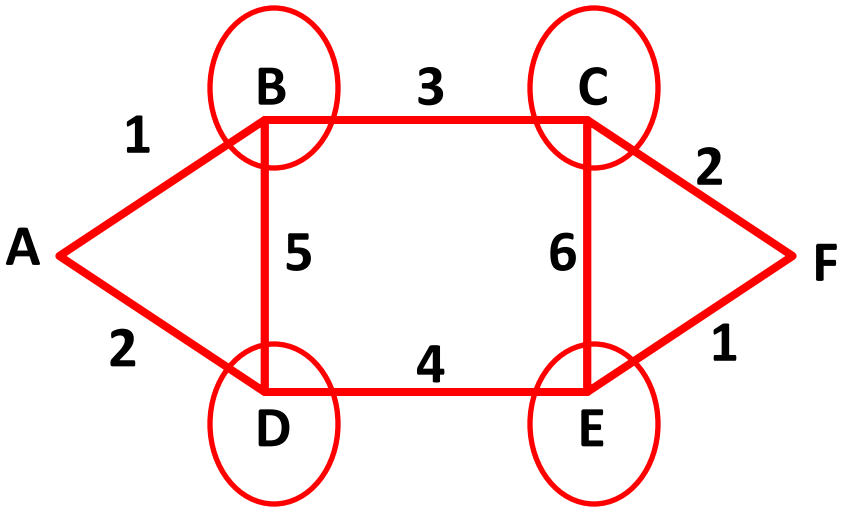
3 3 6

BE*-CD*

6 6 12



Solution



Find a route round all points from A with least cost
Smallest + Total
6 + 24 = 30

A B C F C E F E D B A D A

Step 4 Draw these edges onto the graph

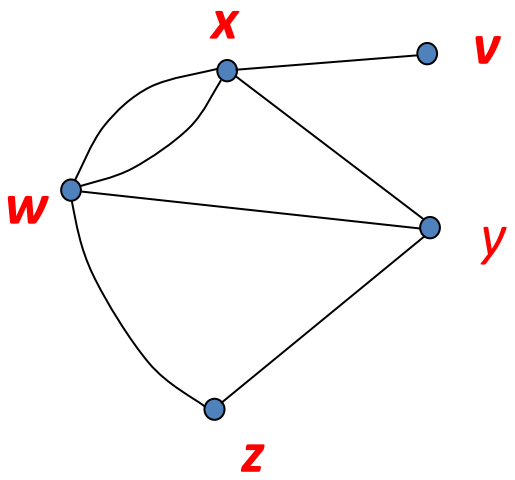
Step 5 The length of the optimal Chinese postman route is the sum of all the edges in the original graph, which is 24, plus the answer found in step 4, which is 6. So the length is 30.

Graphic Sequences

Degree sequence 1.3.27

- The ***Degree Sequence*** of a graph is the list of vertex degrees, *usually* written in non-increasing order, as $d_1 \geq \ldots. \geq d_n$.

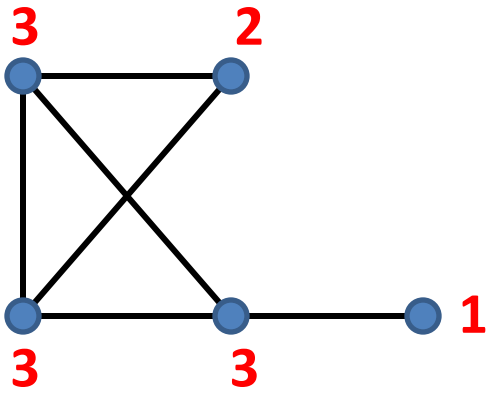
- Example:



Degree sequence:

$d(w), d(x), d(y), d(z), d(v)$

Example



Degree sequence:

1, 3, 3, 3, 2

Or 2, 3, 3, 3, 1

Or 1, 2, 3, 3, 3

etc

Given a graph G, it is easy to find a degree sequence

But given a sequence of non-negative integers, when is it a degree sequence of some graph ?

Proposition: The nonnegative integers $d_1, \ldots, d_n$ are the vertex degrees of some graph if and only if $\sum d_i$ is even. 1.3.28

Proof: ***Necessity*** **(only if)**

- When some graph G has these numbers $d_1, \ldots, d_n$ as its vertex degrees, the degree-sum formula implies that $\sum d_i = 2e(G)$, which is even.

Proposition: The nonnegative integers $d_1, \ldots, d_n$ are the vertex degrees of some graph if and only if $\sum d_i$ is even. 1.3.28

Proof: *Sufficiency* (if)

- Suppose that $\sum d_i$ is even.
- We construct a graph with vertex set $v_1, \ldots, v_n$ and $d(v_i) = d_i$ for all i.
- Since $\sum d_i$ is even, the number of odd values is even.
- First form an arbitrary pairing of the vertices in $\{v_i : d_i \text{ is odd}\}$.
- For each resulting pair, form an edge having these two vertices as its endpoints
- The remaining degree needed at each vertex is even and nonnegative; satisfy this for each i by placing $[d_i/2]$ loops at v_i. ∎

Graphic Sequences 1.3.29

- A ***graphic sequence*** is a list of nonnegative numbers that is the degree sequence of some **simple** graph.

- A simple graph "realizes" *d*.
  - means: A simple graph with degree sequence *d*.

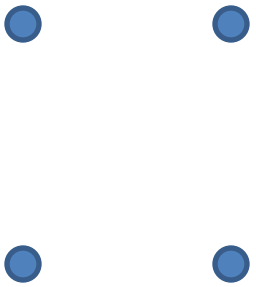
Example:

(i) 1 1 3 – Not graphical sequence since every graph has an even number of odd degree vertices

(ii) 3 3 – Not graphical sequence since on 2 vertices the maximum degree should be n-1 i.e. 1

Obvious Condition

- $\sum d_i$ *must be even*
- $d_i \leq n-1$ *for all* $i=1,\ldots,n$
- *Although these conditions are necessary, they are not sufficient.*
- *Ex: 3,3,3,1 n= 4*

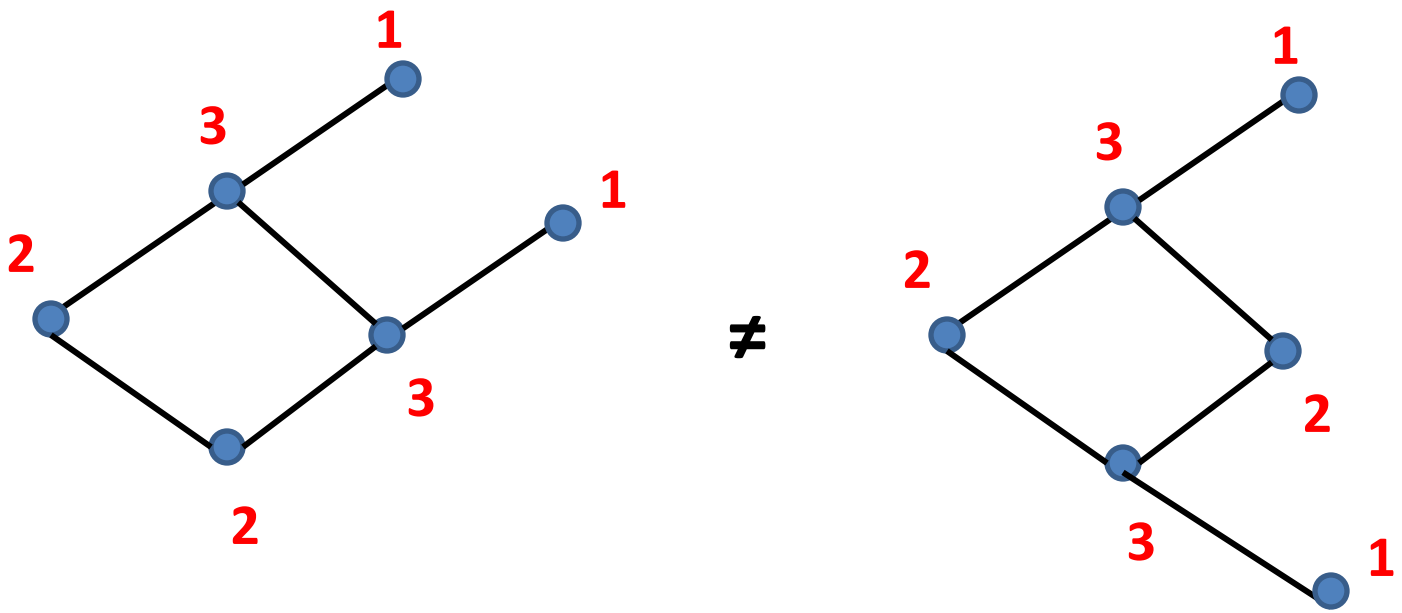


$n = 4$
$\sum d_i = 10$
$d_i \leq 3$

But the sequence is not graphical

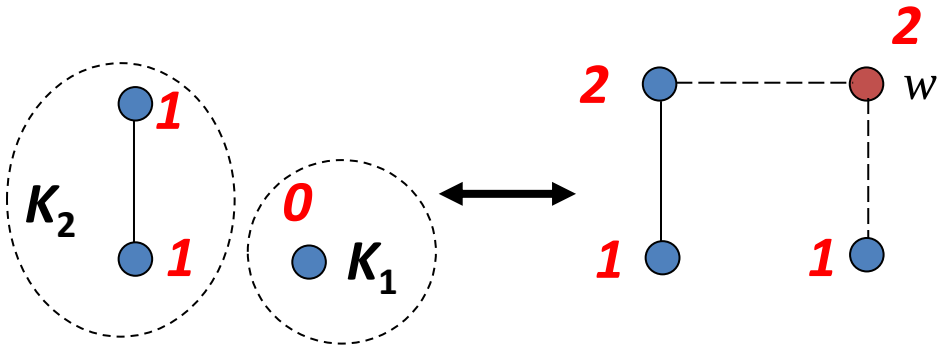
Observation

- A graphic sequence may be a degree sequence for more than one graph.
- Example: 3 3 2 2 1 1



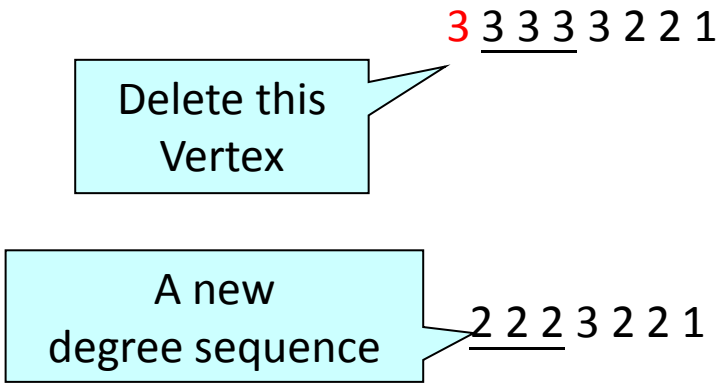
Recursive condition 1.3.30

- The lists (2, 2, 1, 1) and (1, 0, 1) are graphic. The graphic K_2+K_1 realizes 1, 0, 1.
- Adding a new vertex adjacent to vertices of degrees 1 and 0 yields a graph with degree sequence 2, 2, 1, 1, as shown below.
- Conversely, if a graph realizing 2, 2, 1, 1 has a vertex w with neighbors of degrees 2 and 1, then deleting w yields a graph with degrees 1, 0, 1.



Recursive condition 1.3.30

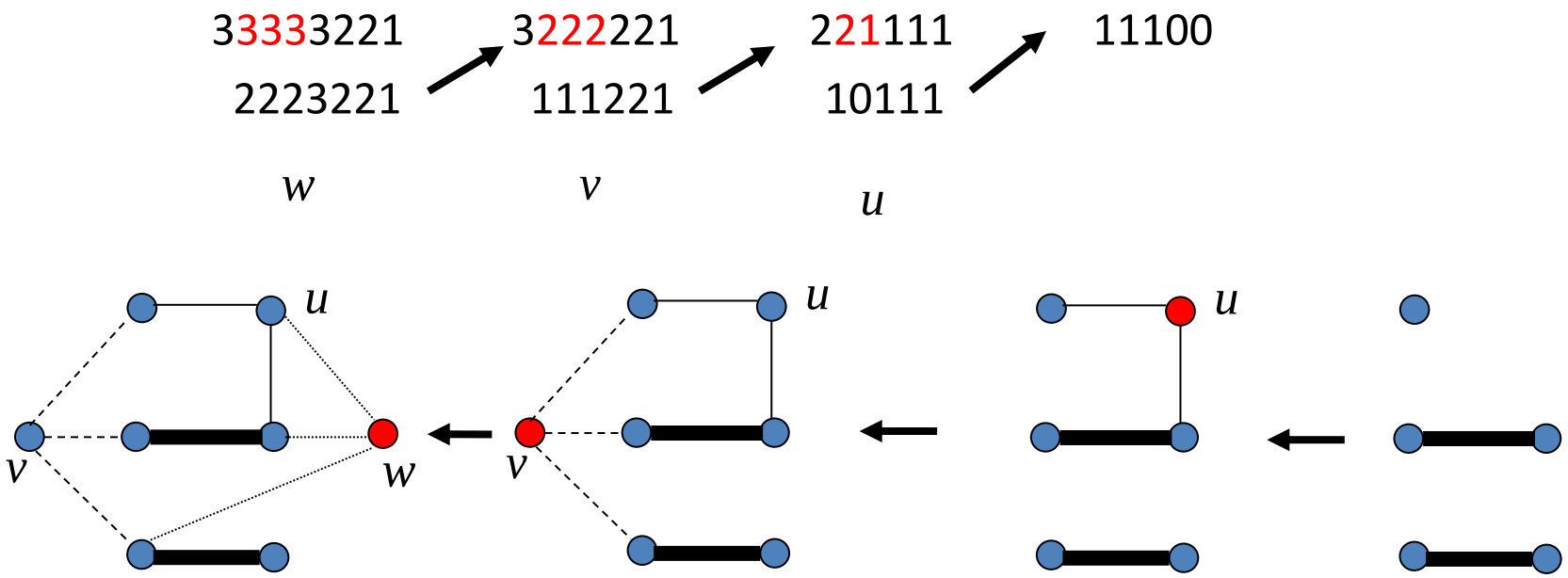
- Similarly, to test 33333221, we seek a realization with a vertex ***w*** of degree 3 having three neighbors of degree 3.



Recursive condition 1.3.30

- This exists if and only if 2223221 is graphic.
  - We reorder this and test 3222221.
  - We continue deleting and reordering until we can tell whether the remaining list is realizable.
  - If it is, then we insert vertices with the desired neighbors to walk back to a realization of the original list.
  - The realization is not unique.
- The next theorem implies that this recursive test works.

Recursive condition 1.3.30

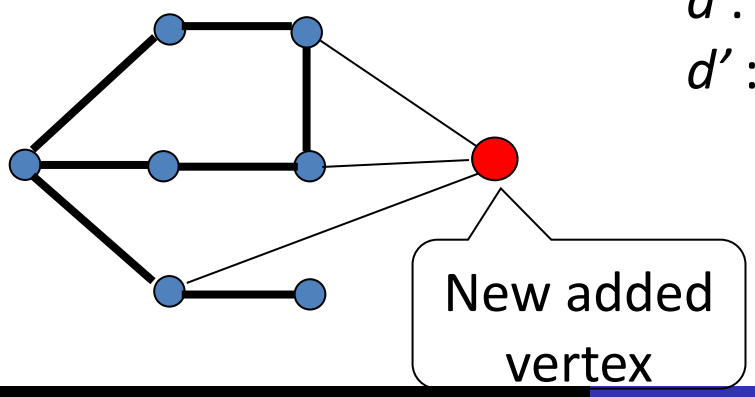


Havel [1955], Halimi [1962]

Theorem. For $n>1$, an integer list d of size n is graphic if and only if d' is graphic, where d' is obtained from d by deleting its largest element Δ and subtracting 1 from its Δ next largest elements. The only 1-element graphic sequence is $d_1=0$. 1.3.31

Proof: For $n=1$, the statement is trivial.

- For $n>1$, **we first prove that the condition is sufficient**.
  - Given d with $d_1 \geq \ldots\ldots \geq d_n$ and a simple graph G' with degree sequence d', we add a new vertex adjacent to vertices in G' with degrees $d_2-1,\ldots\ldots,d_{\Delta+1}-1$.
  - These d_i are the Δ largest elements of d after (one copy of) Δ itself, but $d_2-1,\ldots\ldots,d_{\Delta+1}-1$ need not be the Δ largest numbers in d'



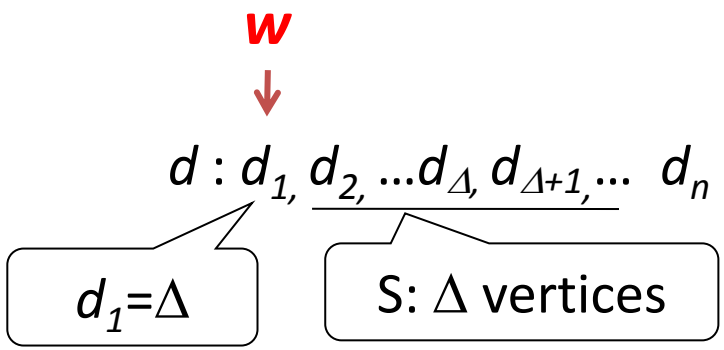
$d : d_1, d_2, \ldots d_n$

$d' : d_2-1,\ldots\ldots,d_{\Delta+1}-1,\ldots d_n$

May not be the Δ largest numbers

Theorem 1.3.31 *continue*

- **To prove necessity,**
  - We begin with a simple graph ***G*** realizing ***d*** , we produce a simple graph ***G'*** realizing ***d'*** .
  - Let ***w*** be a vertex of degree Δ in ***G***, and let ***S*** be a set of Δ vertices in ***G*** having the "desired degrees" $d_2,\ldots..,d_{\Delta+1}$
    If $N(w)=S$, then we delete w to obtain G'.



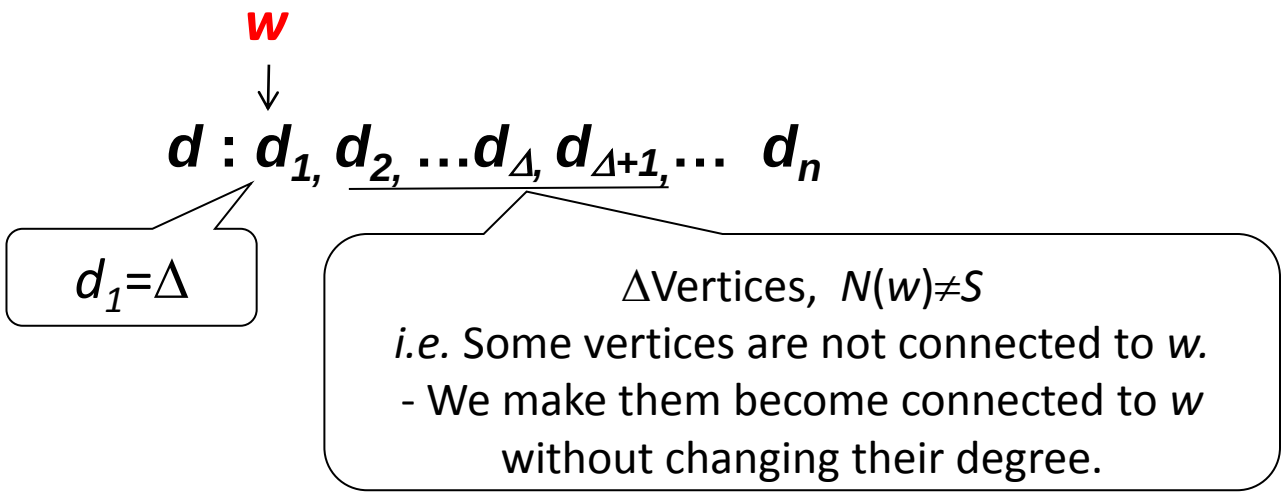
Delete w than we have G'

d': d_2-1,.....,$d_{\Delta+1}$-1,... d_n

Theorem 1.3.31

Proof: *continue*

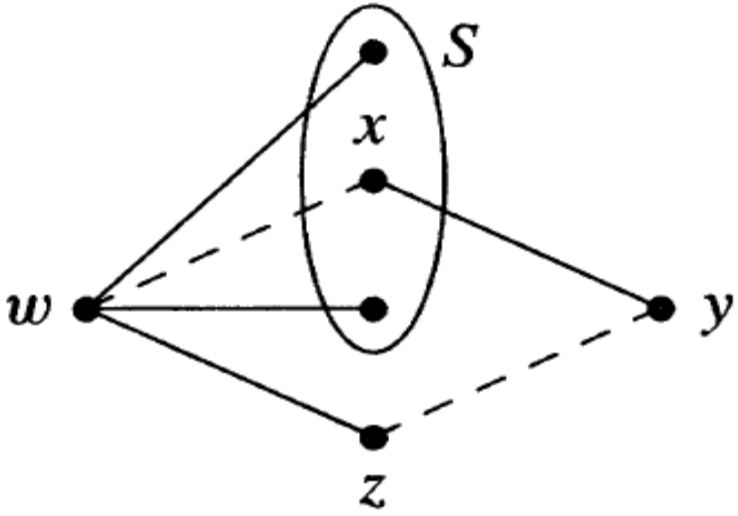
- Otherwise, Some vertex of S is missing from $N(w)$.
  - In this case, we modify G to increase $|N(w) \cap S|$ without changing any vertex degree.
  - Since $|N(w) \cap S|$ can increase at most Δ times, repeating this converts G into another graph G^* that realizes d and has S as the neighborhood of w.
  - From G^* we then delete w to obtain the desired graph G' realizing d'.



Theorem 1.3.31

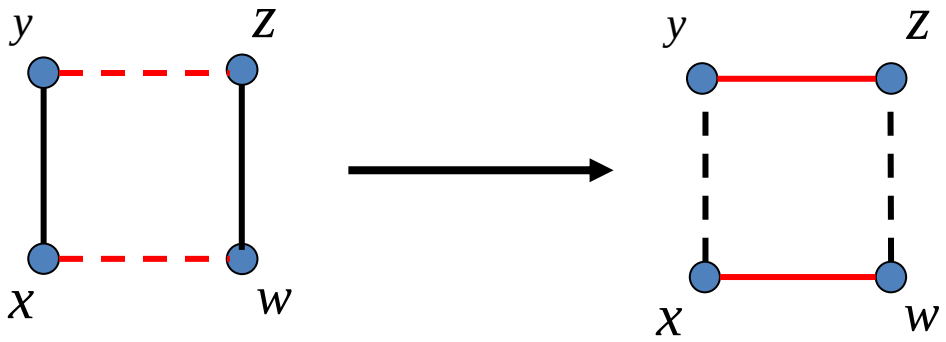
Proof: *continue*

- To find the modification when $N(w) \neq S$, we choose $x \in S$ and $z \notin S$ so that $w \leftrightarrow z$ and $w \not\leftrightarrow x$.
- We want to add wx and delete wz, but we must preserve vertex degrees. Since $d(x) \geq d(z)$ and already w is a neighbor of z but not x, there must be a vertex y adjacent to x but not to z. Now we delete $\{wz, xy\}$ and add $\{wx, yz\}$ to increase $|N(w) \cap S|$.



2-switch 1.3.32

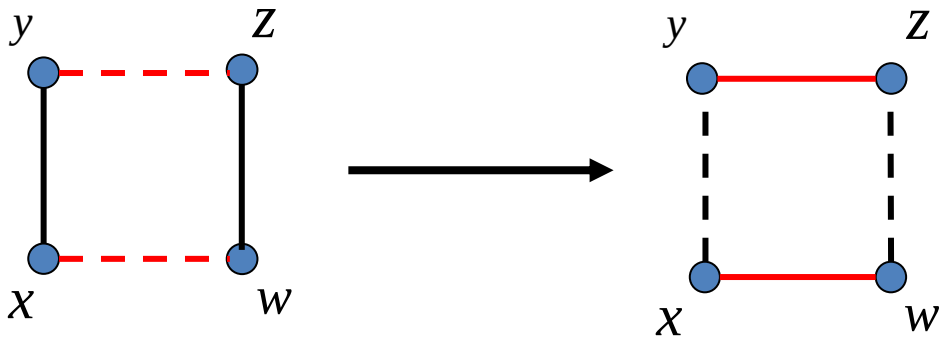
- A ***2-switch*** is the replacement of a pair of edges ***xy*** and ***zw*** in a simple graph by the edges ***yz*** and ***wx***, given that ***yz*** and ***wx*** did not appear in the graph originally.



The dashed lines above indicate nonadjacent pairs.

2-switch 1.3.32

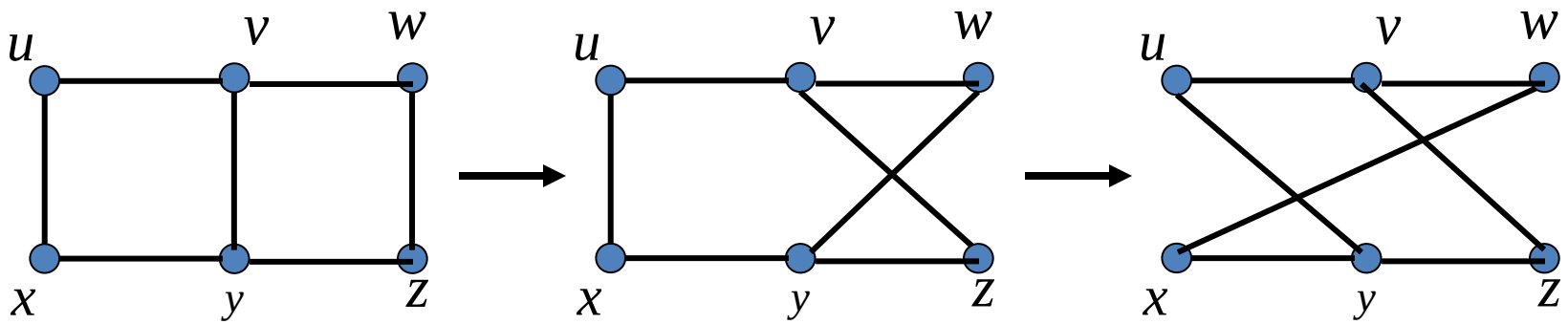
- If y $\leftrightarrow$ z or w $\leftrightarrow$ x then the 2-switch cannot be performed, because the resulting graph would not be simple. A 2-switch preserves all vertex degrees. If some 2-switch turns H into H*, then a 2-switch on the same four vertices turns H* into H.



The dashed lines above indicate nonadjacent pairs.

2-switch 1.3.32

- Below we illustrate two successive 2-switches.



Conclusion

- In this lecture, we have discussed the application of eulerian circuit i.e. chinese postman problem and graphic sequences, .

- In upcoming lectures, we will discuss the concepts and properties of trees and distance.