CPT_S 580 HW4

Yang Zhang

11529139 (graduate)

6.1.2

$K_{m,n}$ is eulerian when m != n and both m and n are bigger than 1. If one of m or n is one then $K_{m,n}$ is a star graph, if m = n, $K_{m,n}$ is a wheel graph. Neither wheel graph nor star graph is eulerian.

6.1.3

n-vertex wheel graph is not eulerian.

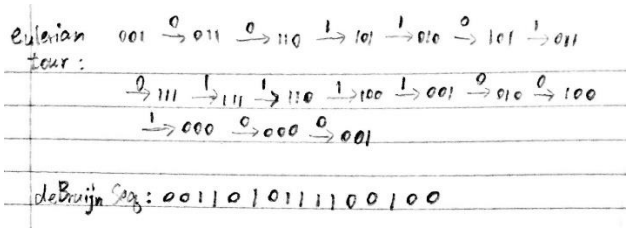
6.1.20

The graph is connected and its vertices all have even degrees, which guarantees that if visit a vertex by an edge in that graph there must be another edge can get out the vertex without repeating the same edge. Therefore, no matter what vertex to start with, let's say start from vertex **a,** any other vertex of the graph can be added to the existing eulerian tour by connecting the vertex **a** by the edge incident on it and then gets out from **a** and back to original eulerian tour by a different edge that also incident on a. We can repeat those steps until there is no unused edge that incident on **a**.

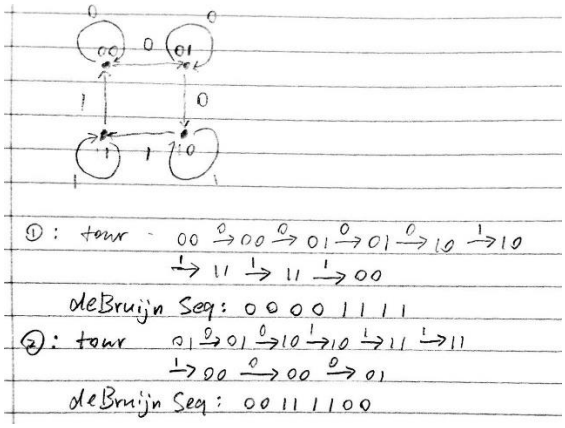
6.1.21

If there were no such edge, then the graph is not connected.

6.2.1



6.2.2



6.3.1

K_n is Hamiltonian

6.3.2

$K_{m,n}$ is Hamiltonian when m > 1 and n > 1

6.3.3

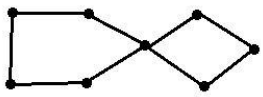
Any wheel graph is Hamiltonian

6.3.4

Tree is not Hamiltonian, because tree doesn't contain cycle.

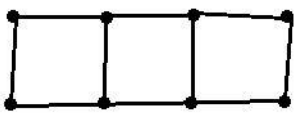
6.3.7

Eulerian but not Hamiltonian

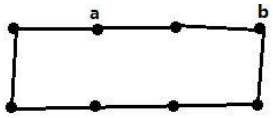


6.3.8

Hamiltonian but not Eulerian



6.3.9



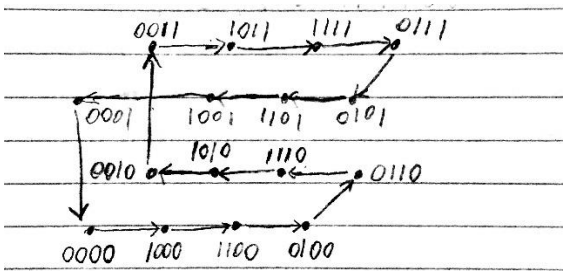
Vertices a and b are not adjacent.

deg(a) + deg(b) = 4, #vertices = 8

deg(a) + deg(b) < 8

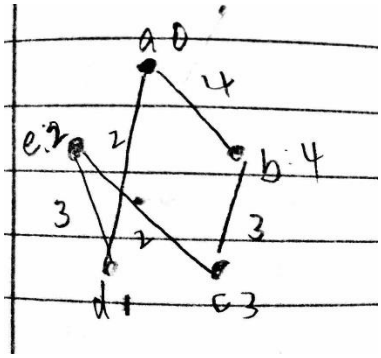
The graph is Hamiltonian but it does not satisfy ore's theorem.

6.4.1



6.4.2

Nearest neighbor:

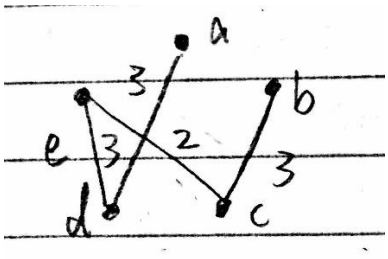


Visiting sequence: {a, d, e, c, b, a}

Total weight: 2 + 3 + 2 + 3 + 4 = 14

Double the tree and tree matching have the same result:

MST:

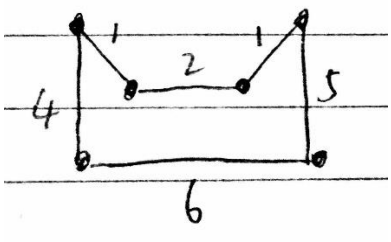


Visiting sequence: {a, d, e, c, b, a}

Total weight: 2 + 3 + 2 + 3 + 4 = 14

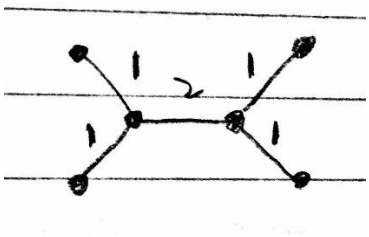
6.5.1

The minimum-length postman tour:



Total cost: 4 + 1 + 2 + 1 + 5 + 6 = 19

All minimum cost deadhead paths have the same pattern:



All those paths will travel each edge twice

The minimum cost = 2 * (1 + 1 + 2 + 1 + 1) = 12

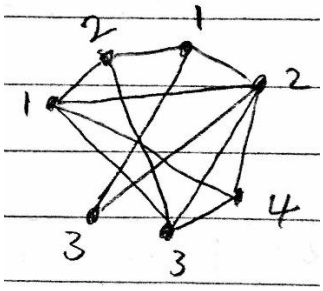
6.5.14

It is not Hamiltonian, because no Hamiltonian cycle can be found in the graph.

9.1.2

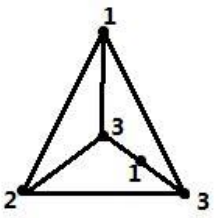
4 frequencies needed, because the graph has a 4 clique.

The frequency assignments:



9.1.4

The graph can be assigned with 3 different color, and since it has a 3 clique the vertex-coloring number cannot be smaller than 3. Therefore, 3 is the minimum vertex-coloring.

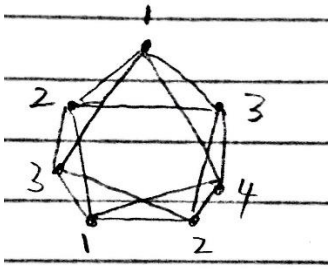


9.1.8

The graph can be assigned with 4 different color, because by proposition 9.1.4 for any graph

$X(G) >= |V_G| / a(G)$

For the following graph, X(G) = 7/2 = 4

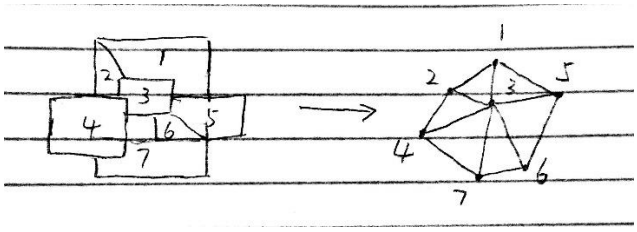


9.1.28

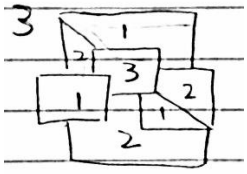
Any odd graph W_{2n+1} is composed by the join of odd cycle graph C_{2n+1} with K1.
By proposition 9.1.6,
X(G+H) = X(G) + X(H)
Thus, $X(W_{2n+1}) = X(C_{2n+1}) + X(K_1) = 3 + 1 = 4$

Therefore, all odd wheels are chromatically 4-critical.

9.2.3

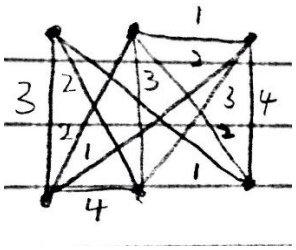


After transferring the map to a graph, we can see the resulting graph is even wheel graph. Since any even wheel graph is chromatically 3-critical, and since the exterior region only connected to the outlier of the wheel, the original map can be colored in 3 colors minimum.



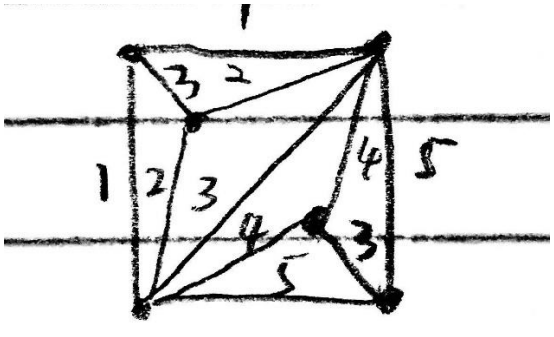
9.3.3

Since the maximum degree is 4, the graph can be edge colored in 4 color minima.



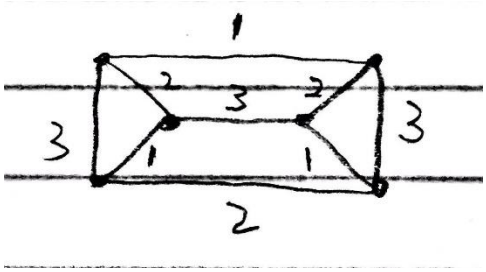
9.3.5

Since the maximum degree is 5, the graph can be edge colored in 5 color minima.



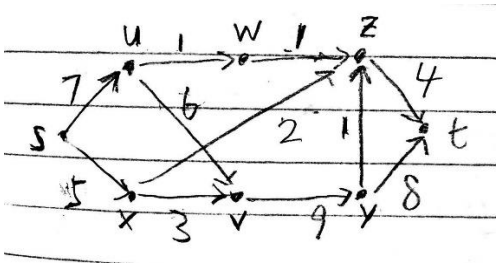
9.3.8

Since the maximum degree is 3, the graph can be edge colored in 3 color minima.



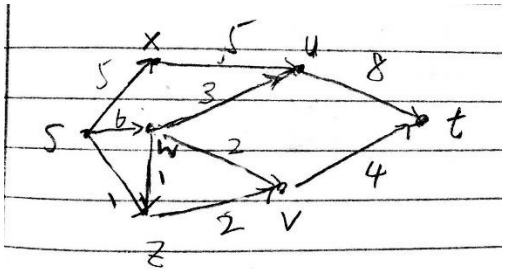
13.1.2

The arc set < {s, x}, {u, x, w, v, z, y, t}> is an s-t cut with capacity 12, and the flow shown in the figure below has value 12. Hence, by Corollary 13.1.7 both are optimal. The number on each arc in the figure indicates the flow across that arc.



13.1.3

The arc set < {s, w, v, z}, {x, u, t}> is an s-t cut with capacity 12, and the flow shown in the figure below has value 12. Hence, by Corollary 13.1.7 both are optimal. The number on each arc in the figure indicates the flow across that arc.



6.5.16

The graph with 4 vertices is Hamiltonian:

The graph with 4 vertices is non-Hamiltonian:

All the vertex-deleted subgraphs of both of the graphs are non-Hamiltonian

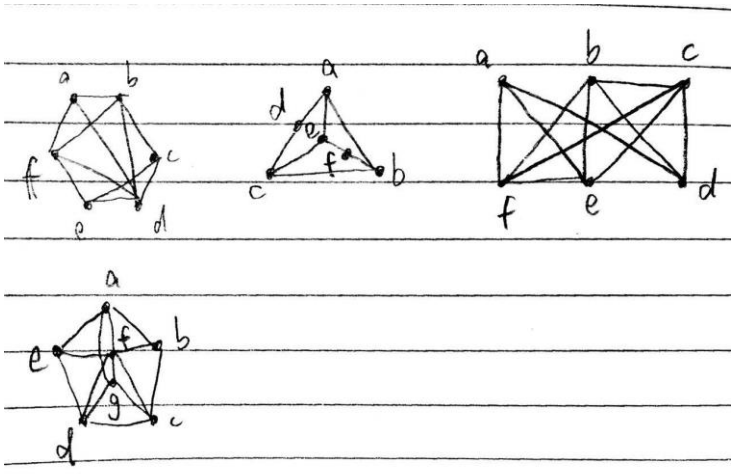
9.3.18

Suppose the $X(T) < \delta_{MAX}(T)$, and the node v in the tree has degree number = $\delta_{MAX}(T)$. Then the minimum edge-chromatic number at node v is $\delta_{MAX}(T)$ which contradicts with $X(T) < \delta_{MAX}(T)$. Therefore $X(T) >= \delta_{MAX}(T)$

Suppose the $X(T) > \delta_{MAX}(T)$, and the node v in the tree has degree number = $\delta_{MAX}(T)$. Then the minimum edge-chromatic number at node v is $\delta_{MAX}(T)$, and for any set of siblings there is no edge between them, the only restriction is to have a different color from their parents. Therefore $\delta_{MAX}(T)$ number of colors is required for the tree, which contradicts with $X(T) < \delta_{MAX}(T)$. Therefore $X(T) = \delta_{MAX}(T)$

Programming part

The 4-testing graph will be labeled as following:



1. **Sequential Coloring**

The question is asked to implement the Sequential Coloring algorithm, the way I implement it is to iterate through every vertex of the graph based on alphabetical order, assign each vertex by a lowest possible color number.

Input: adjacent matrix of the input graph
Output: the sequence of coloring
Results:

9.1.3

```
Vertex a --->  Color 0
Vertex b --->  Color 1
Vertex c --->  Color 0
Vertex d --->  Color 2
Vertex e --->  Color 1
Vertex f --->  Color 2
```

9.1.4

```
Vertex a --->  Color 0
Vertex b --->  Color 1
Vertex c --->  Color 0
Vertex d --->  Color 1
Vertex e --->  Color 1
Vertex f --->  Color 0
```

9.1.5

```
Vertex a --->  Color 0
Vertex b --->  Color 1
Vertex c --->  Color 2
Vertex d --->  Color 3
Vertex e --->  Color 3
Vertex f --->  Color 4
```

9.1.6

```
Vertex a --->  Color 0
Vertex b --->  Color 1
Vertex c --->  Color 0
Vertex d --->  Color 1
Vertex e --->  Color 2
Vertex f --->  Color 3
Vertex g --->  Color 2
```

```
Conclusion: from the result we can see that the coloring may not
be optimal, but is closed to best solution.
```

2. Largest Degree First Sequential Coloring

The question is asked to implement the Largest Degree First Sequential Coloring algorithm, the way I implement it is to iterate through every vertex of the graph based on decreasing degree number order, assign each vertex by a lowest possible color number.

Input: adjacent matrix of the input graph

Output: the sequence of coloring

The result:

```
9.1.3
Vertex a --->  Color 2
Vertex b --->  Color 1
Vertex c --->  Color 2
Vertex d --->  Color 0
Vertex e --->  Color 1
Vertex f --->  Color 0

9.1.4
Vertex a --->  Color 2
Vertex b --->  Color 1
Vertex c --->  Color 2
Vertex d --->  Color 0
Vertex e --->  Color 1
Vertex f --->  Color 0

9.1.5
Vertex a --->  Color 2
Vertex b --->  Color 3
Vertex c --->  Color 2
Vertex d --->  Color 0
Vertex e --->  Color 1
Vertex f --->  Color 0

9.1.6
Vertex a --->  Color 2
Vertex b --->  Color 0
Vertex c --->  Color 3
Vertex d --->  Color 2
Vertex e --->  Color 0
Vertex f --->  Color 1
Vertex g --->  Color 0
```

Conclusion: from the result we can see that the coloring may not be optimal, but is closed to best solution.