GRAPH THEORY and APPLICATIONS

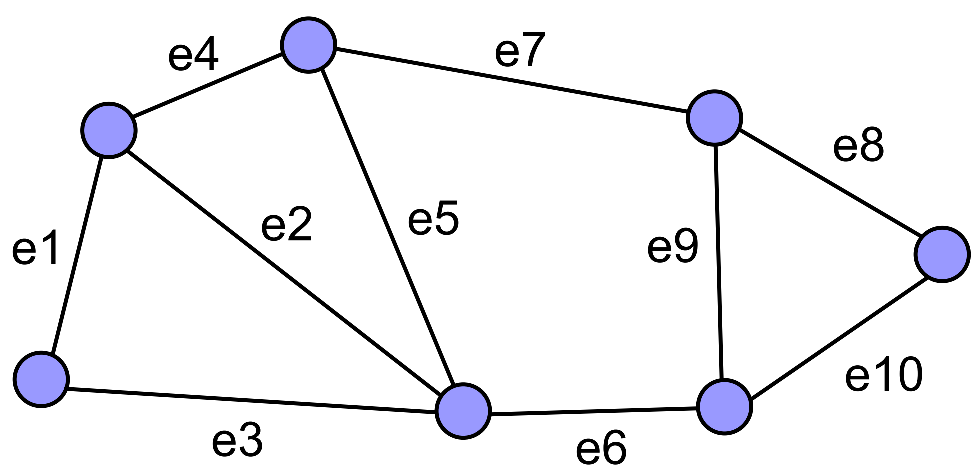
Matchings

Definition

- Matching of a graph G: Any subset of edges M ⊆ E such that no two elements of M are adjacent.

Example:

{e1} {e1,e5,e10} {e2,e7,e10} {e4,e6,e8}



Definition

- Maximum-cardinality matching: A matching which contains a maximum number of edges.
- Perfect matching: A matching in which every vertex of the graph is an end point of an edge in matching.
  - Every graph may not contain a perfect matching.
  - If a graph contains a perfect matching M, then M is a maximum-cardinality matching.

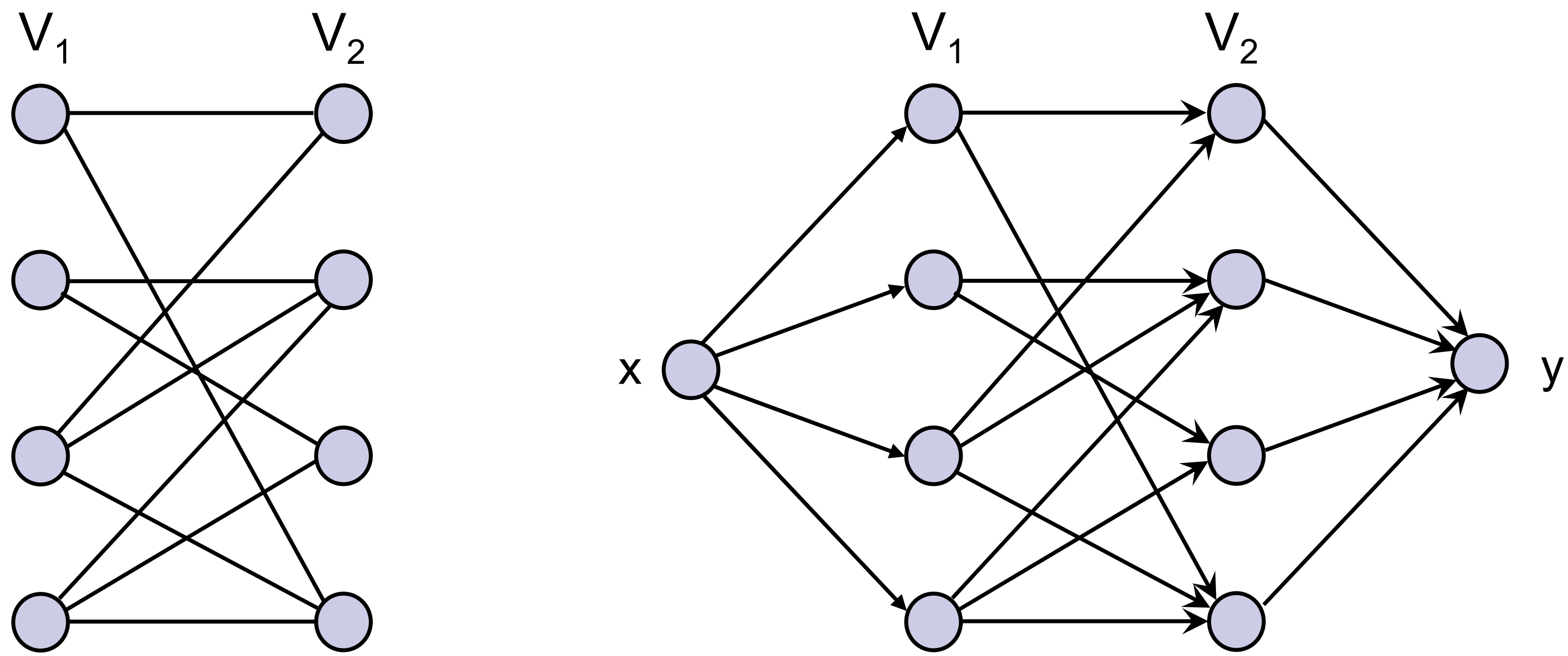
Definition

- In a bipartite graph G with bipartition (V',V"): a complete matching of V' into V", is:
  - a matching M
  - every element of V' is an end-point of an edge of M.
- If a bipartite graph contains a complete matching M, then M is maximum cardinality matching.
- In a weighted graph, a maximum-weight matching is a matching, where:
  - the sum of edge-weights is maximum.

Maximum-cardinality matching

- Consider bipartite graphs.
- Using a simple method (flow techniques), we can find a maximum-cardinality matching.
- $G = (V,E)$, a bipartite graph with bipartition (V_1,V_2). Construct G' as follows:
  - Direct all edges from V_1 to V_2
  - Add a source x, and a directed edge from x to each vertex in V_1.
  - Add a sink y, and a directed edge from each vertex in V_2 to y.
  - Let each edge (u,v) have a capacity $c(u,v) = 1$

Example



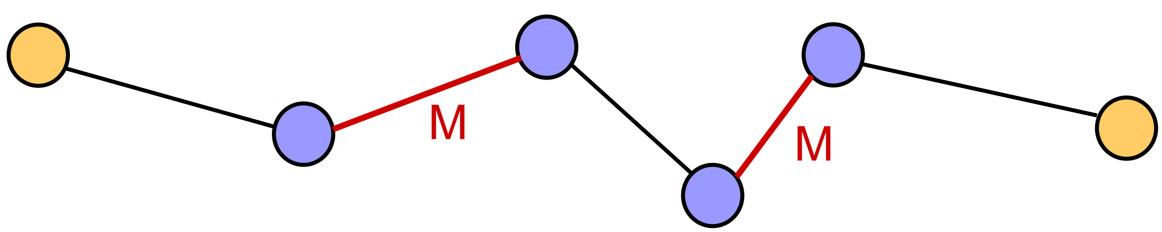
- Given such construction, we can find a maximum-cardinality matching M, by maximizing the flow from x to y.

Bipartite maximum-cardinality matching

- M consists of edges linking V_1 to V_2 which carry a flow of one unit.
- If some matching M' exists such that |M'| > |M|
  - then we could construct a flow of value |M'|
  - sending one unit of flow along each path: ((x,u),(u,v),(v,y)) for all $(u,v) \in M'$

General maximum-cardinality matching

- Consider general graphs.
- If $M \subseteq E$ is a matching for G, then:
  - any vertex v is called a free vertex, if it is not an end-point of any element of M.
- An alternating path: A simple path in G whose edges alternately belong to M, and to (E – M).
- An augmenting path with respect to M: An alternating path between two free vertices.



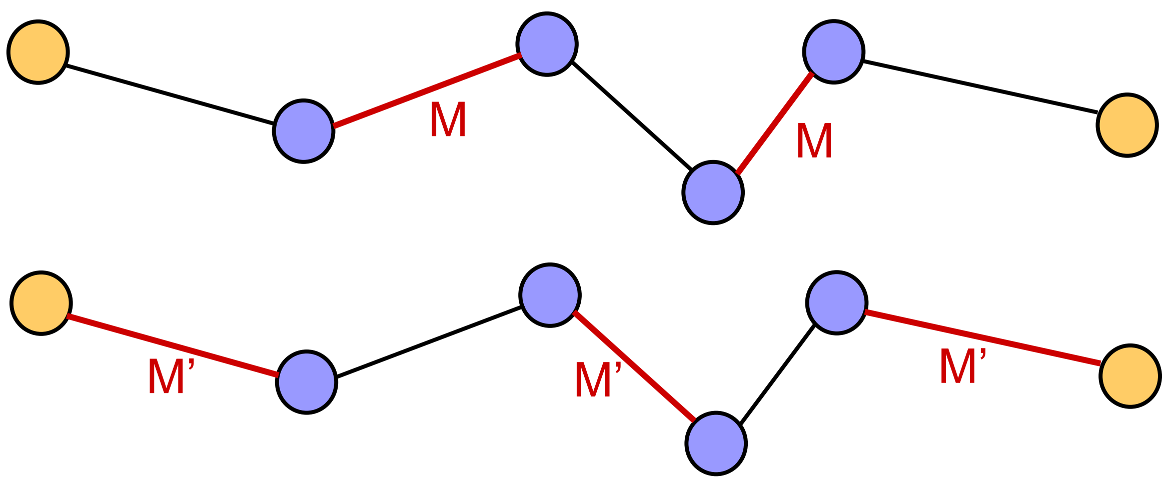
Augmenting path

Notice:

- If G contains an augmenting path P, then a matching M’ can be found, such that:

$$|M'| = |M| + 1$$

by reversing the rôles of the edges in P.

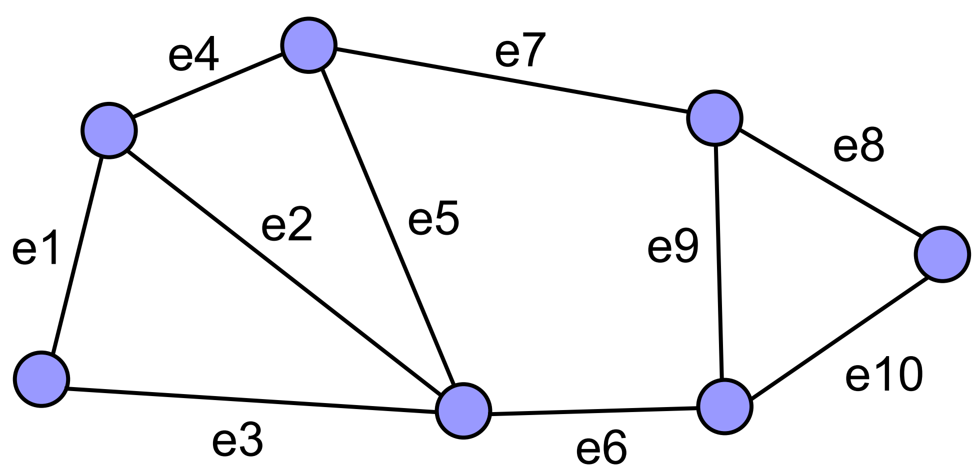


Augmenting paths

Example:

- If M = {e3,e8},

  An augmenting path can be traced along: (e1,e3,e5)
- Reversing edge rôles, we obtain: M’ = {e1,e5,e8}



Algorithm

Theorem: There is an M-augmenting path if and only if M is not a maximum-cardinality matching.

- The theorem suggests an algorithm to find a maximum-cardinality matching.
- Start with an arbitrary matching.
  - Might be a null matching.
- Repeatedly carry out augmentations along M-augmenting paths, until no such path exists.

Algorithm

- The process is bound to terminate.
  - A maximum matching has finite cardinality.
  - Each augmentation increases the cardinality of the current matching by one.
- Problem: Specifying a systematic search for M-augmentations.
- Solution inspired by Edmonds.

M-augmenting path search – MAPS

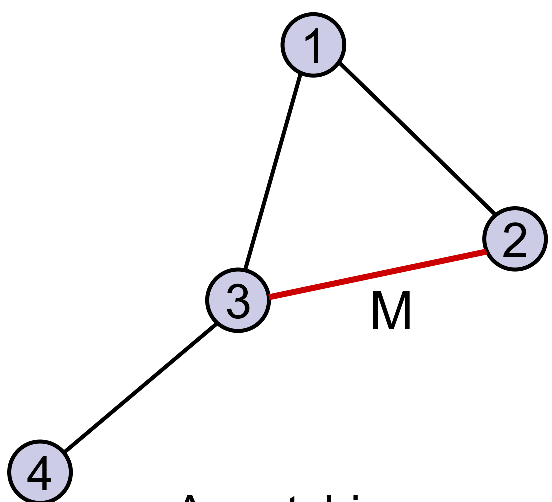
- A search tree T is constructed.
  - T is rooted at some free vertex v.
  - Any path in T starting at v is an alternating path:
    - The vertices are alternately labeled outer and inner.
    - The root v is labeled outer.
- At start, T is initialized to be v.
  - v is labeled outer.
- There are three possible exits from the search.
  - to exits A, B, and H.
  - only exit to A indicates an augmenting path.

MAPS

```
1.Choose an outer vertex x∈T and some edge (x,y)
  not previously explored;
  Label (x,y) to be explored;
  If no such edge exists goto H;
2.If y is free and unlabeled, add (x,y) to T;
  goto A;
3.If y is outer add (x,y) to T;
  goto B;
4.If y is inner goto 1;
5.Let (y,z) be the edge in M with endpoint y;
  Add (x,y) and (y,z) to T;
  Label y inner;
  Label z outer;
  goto 1;
```

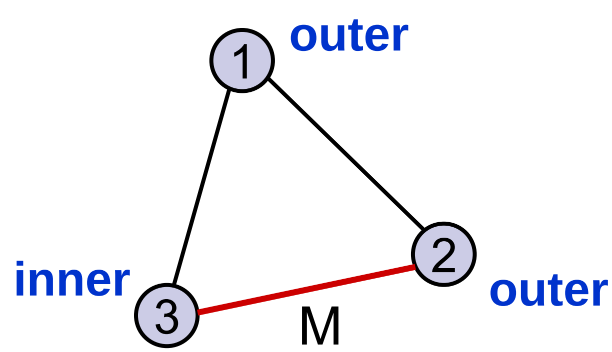
Odd-length circuits

- If y is found to be labeled outer:
  - An odd-length circuit has been found, jump to B.
  - Why does the procedure terminates on detecting odd-length circuits?



A matching

Call MAPS:
T initialized at 1.
If line 2 is executed with y = 3:



An augmenting path cannot be found.

Blossom

- Odd length cycles introduces ambiguities in alternating path search.
  - A new graph is constructed by shrinking the cycle C to form a single vertex.
  - Those vertices are called blossom.
  - This vertex is labeled outer.
  - MAPS is called again.
  - Previous labels are carried forward.
  - An odd-length cycle itself may contain blossoms.

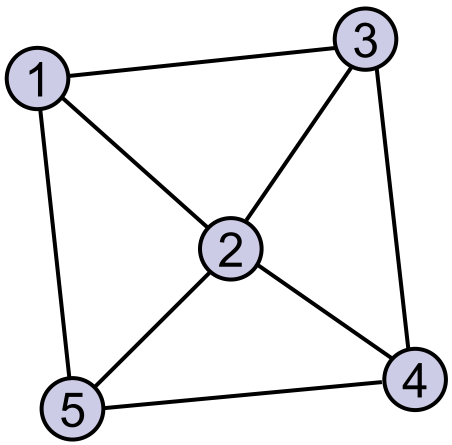
Hungarian tree

- If y is inner:
  - An even-length circuit is detected.
  - (x,y) is not added to T, extend the tree from some other outer vertex.
- Consider exit to H:
  - T cannot be extended.
  - Each path from the root of the tree is stopped at some outer vertex.
  - The only free vertex is the root.
  - T is called a Hungarian tree.
  - In this case, the tree is removed from G.
  - The search of path continues with G – T.

Expanding blossoms

- An alternating path may contain one ore more blossoms.
- The even-length side of each odd-length cycle is interpolated in the path.
- This procedure is repeated until no blossoms are left in the path.

Example

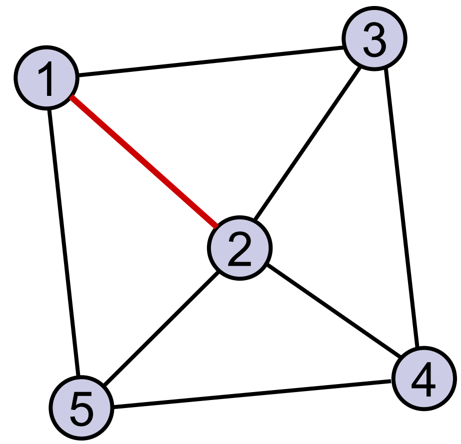


First iteration

Free vertices: 1 2 3 4 5

P = (1,2)

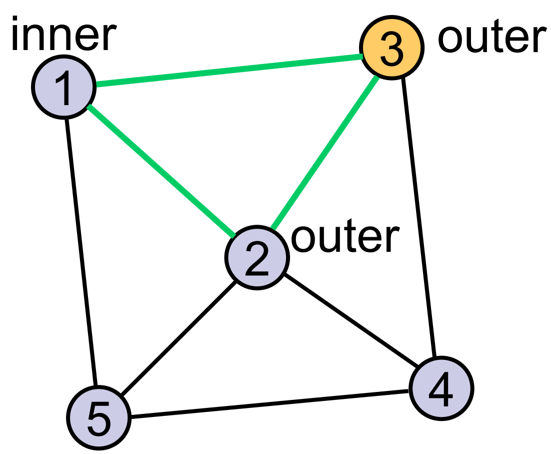
First iteration discovers the first edge as a path between free vertices.



Second iteration

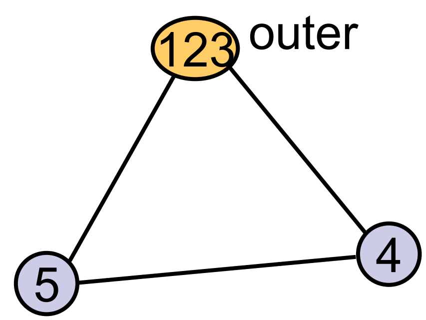
Free vertices: 3 4 5

Root: 3 (outer)
MAPS:
- Choose (3,1)
- Label 1 inner, 2 outer
- Choose (2,3)
- 3 is outer: blossom!

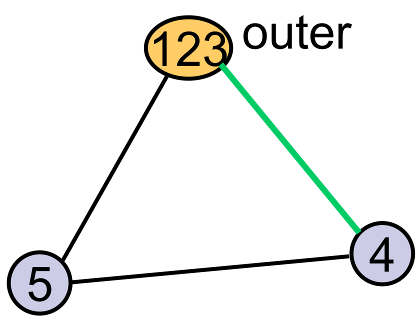


Second iteration

B: Shrink the blossom (1,2,3)
Label new vertex 'outer'



Example



MAPS:

Choose (123,4)

4 is free: add (123,4) to tree.

A:

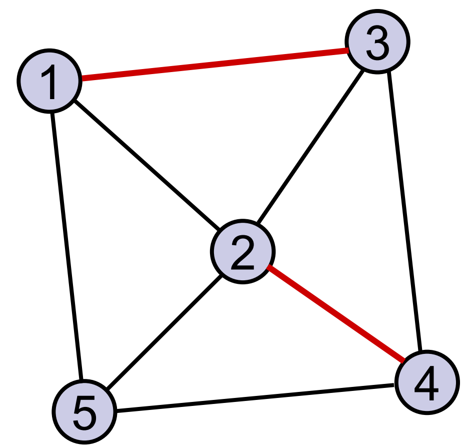
Identify augmenting path:

(123,4)

(3,1),(1,2),(2,4)

Augmentation:

M = {(3,1),(2,4)}



Examine the final iteration from page 133 of the textbook.

Perfect Matching

- Every vertex of a graph is the end point of an edge in a matching.
- If a perfect matching exists, then the result of the algorithm to find maximum cardinality matching will be a perfect matching.
- A necessary and sufficient condition for G to have a perfect matching:

Theorem: G(V,E) has a perfect matching if and only if:

$$\Phi(G - V') \leq |V'| \quad \text{for all } V' \subset V$$

$\Phi(G - V')$: number of components of $(G - V')$ containing odd number of vertices.

Max-weight/min-weight matching

- Maximum-weight matchings or minimum-weight matchings can be found by polynomial-time algorithms (due to Edmonds).
- However, they are somewhat complicated.
- Approximation algorithms are designed to obtain near-optimal results with lower complexity.

TSP approximation by matching

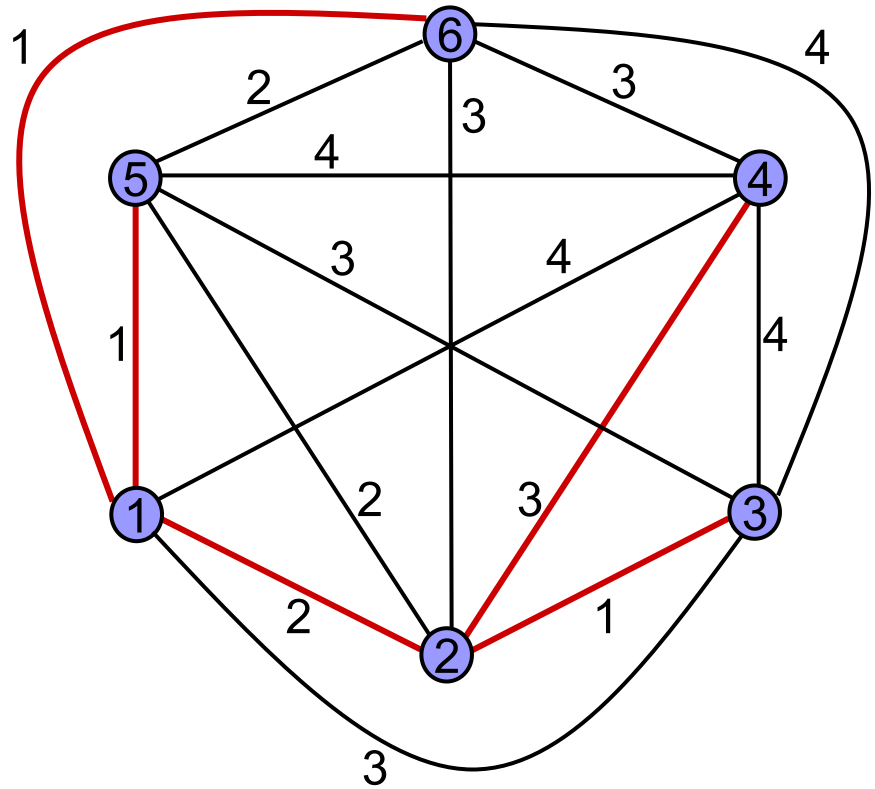
- The twice-around-the-MST heuristic can be improved:
  - Using perfect matching idea
  - Approximation: $\alpha \leq 3/2$

  $\Rightarrow$ Minimum-weight matching algorithm for TSP

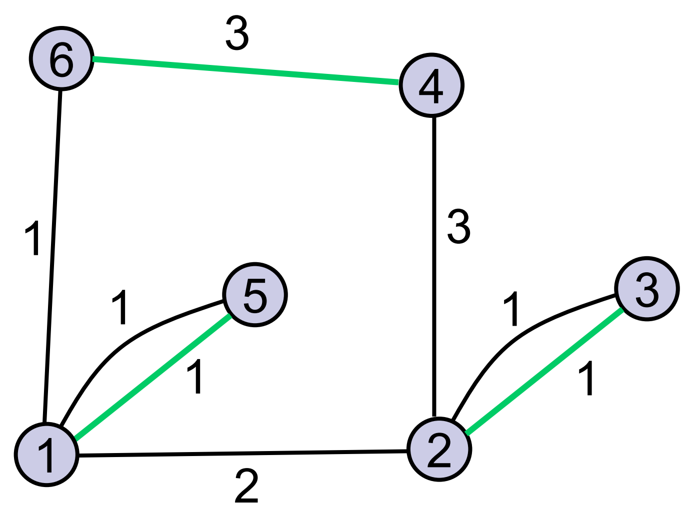
An improved approximation for TSP

```
1.Find a minimum-weight spanning tree T of G;
2.Construct the set V′ of vertices of odd degree
  in T;
  Find a minimum-weight perfect matching M of V′;
3.Construct the Eulerian graph G′
    by adding the edges of M to T;
4.Find an Eulerian circuit C0 of G′;
  Index each vertex according to the order, L(v),
    where v is first visited in a trace of C0;
5.Output the following minimum-weight Hamilton
  cycle:
    C = (v1, v2, …, vn, v1) where
    L(vj) = j;
```

Example



A minimum-weight perfect matching:
(1,5), (2,3), (4,6)



Eulerian circuit: (1,5,1,2,3,2,4,6,1)
Hamiltonian circuit: (1,5,2,3,4,6,1)