POLITECNICO DI TORINO

e-Lite

Graphs: Cycles

Tecniche di Programmazione – A.A. 2010/2020

Summary

- Definitions
- Algorithms

Definitions

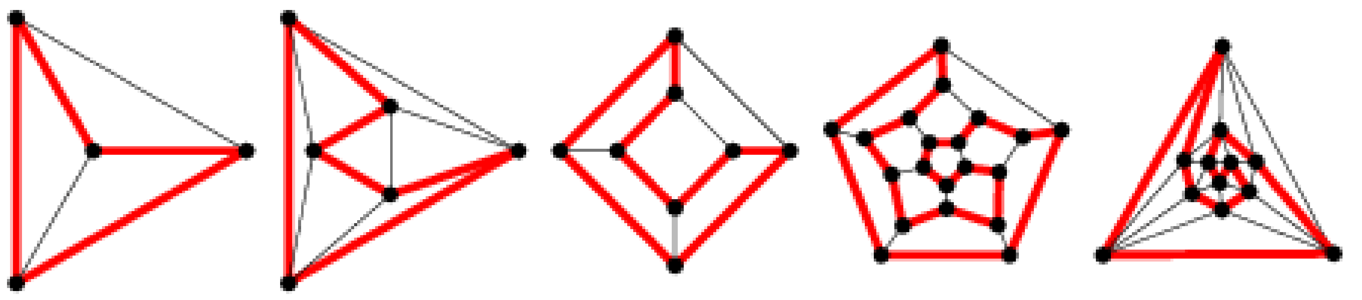
Graphs: Cycles

Cycle

- A **cycle** of a graph, sometimes also called a circuit, is a subset of the edge set of that forms a path such that the first node of the path corresponds to the last.

Hamiltonian cycle

- A cycle that uses each graph vertex of a graph exactly once is called a Hamiltonian cycle.



Hamiltonian path

- A Hamiltonian path, also called a Hamilton path, is a path between two vertices of a graph that visits each vertex exactly once.
  - N.B. does <u>not</u> need to return to the starting point

Eulerian Path and Cycle

- An **Eulerian path**, also called an Euler chain, Euler trail, Euler walk, or "Eulerian" version of any of these variants, is a walk on the graph edges of a graph which **uses each graph edge** in the original graph **exactly once**.
- An **Eulerian cycle**, also called an Eulerian circuit, Euler circuit, Eulerian tour, or Euler tour, is a trail which starts and ends at the **same** graph vertex.

Theorem

- A connected graph has an Eulerian **cycle** if and only if it **all vertices have even degree**.
- A connected graph has an Eulerian **path** if and only if it has **at most two graph vertices of odd degree**.

  - …easy to check!

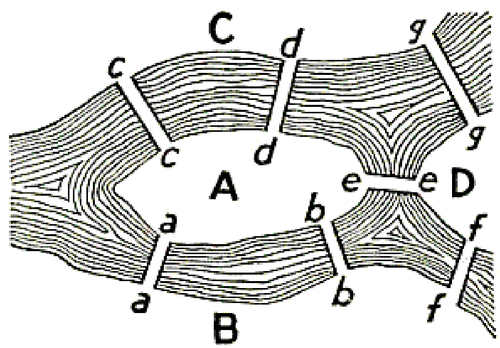
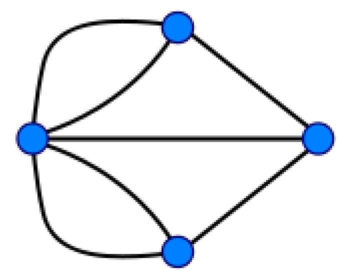


FIGURE 98. *Geographic Map: The Königsberg Bridges.*



Königsberg Bridges

Weighted vs. Unweighted

- Classical versions defined on Unweighted graphs
- Unweighted:
  - Does such a cycle exist?
  - If yes, find at least one
    - Optionally, find all of them
- Weighted
  - Does such a cycle exist?
    - Often, the graph is complete ☺
  - If yes, find at least one
  - If yes, find **the best one** (with **minimum** weight)

Algorithms

Graphs: Cycles

Eulerian cycles: Hierholzer's algorithm (1)

- Choose **any** starting vertex *v*, and **follow a trail** of edges from that vertex until returning to *v*.
  - It is **not** possible to get stuck at any vertex other than *v*, because the even degree of all vertices ensures that, when the trail enters another vertex *w* there must be an unused edge leaving *w*.
  - The tour formed in this way is a **closed** tour, but may **not** cover all the vertices and edges of the initial graph.

Eulerian cycles: Hierholzer's algorithm (2)

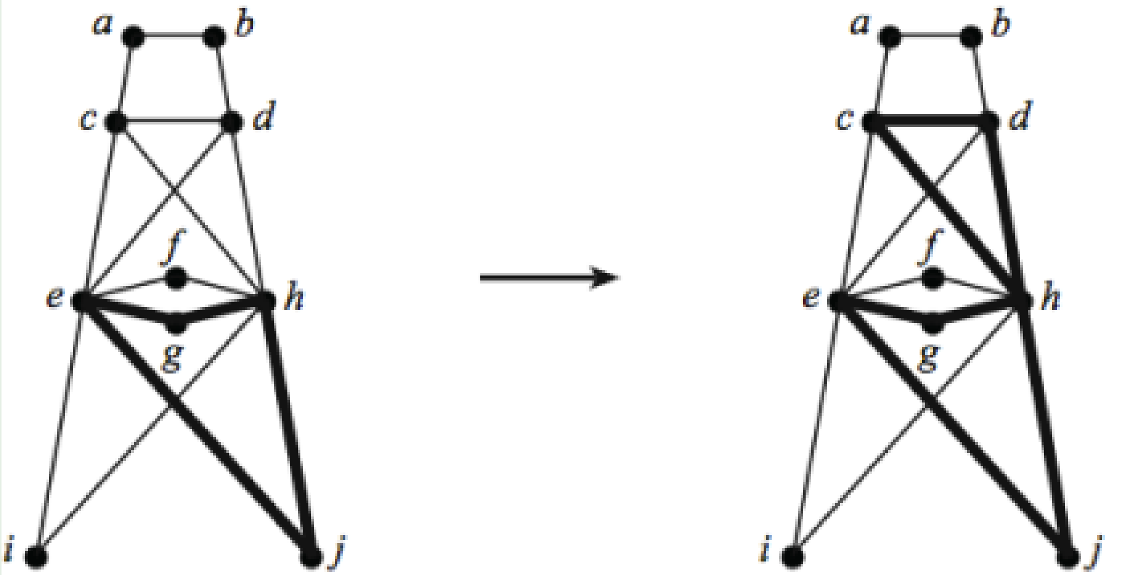
- As long as there exists a vertex *v* that belongs to the current tour but that has adjacent edges not part of the tour, **start another trail** from *v*, following **unused** edges until returning to *v*, **and join** the tour formed in this way to the previous tour.

Given: an Eulerian graph G

Find an Eulerian circuit of G.

1. Identify a circuit in G and call it R_1. Mark the edges of R_1. Let $i = 1$.
2. If R_i contains all edges of G, then stop (since R_i is an Eulerian circuit).
3. If R_i does not contain all edges of G, then let v_i be a node on R_i that is incident with an unmarked edge, e_i.
4. Build a circuit, Q_i, starting at node v_i and using edge e_i. Mark the edges of Q_i.
5. Create a new circuit, R_{i+1}, by patching the circuit Q_i into R_i at v_i.
6. Increment i by 1, and go to step (2).

Example



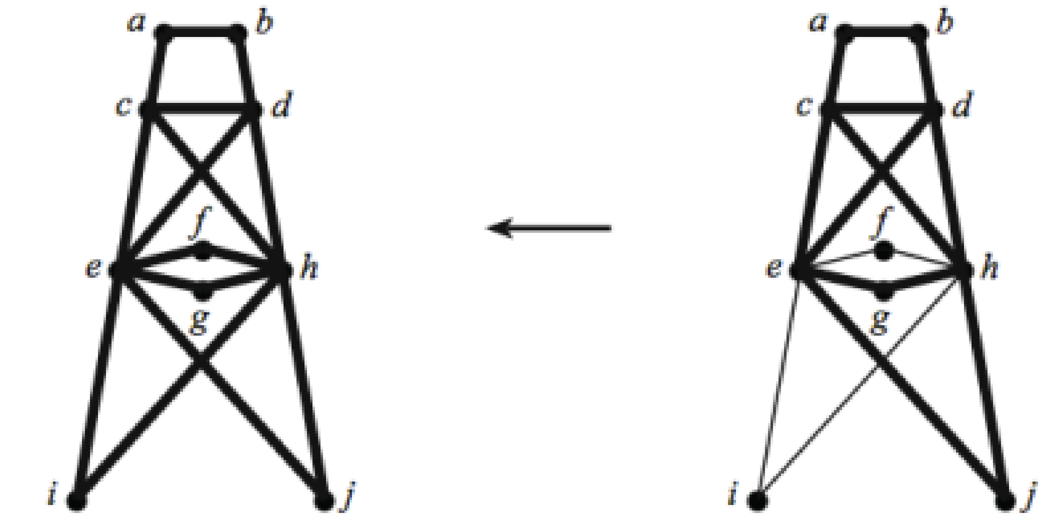
R_1: e, g, h, j, e

Q_1: h, d, c, h

R_2: $e, g, \mathbf{h}, \mathbf{d}, \mathbf{c}, \mathbf{h}, j, e$

Q_2: d, b, a, c, e, d

Example (continued)



R_4: $e, g, \mathbf{h}, \mathbf{f}, \mathbf{e}, \mathbf{i}, \mathbf{h}, d, b, a, c, e, d, c, h, j, e$

R_3: $e, g, h, \mathbf{d}, \mathbf{b}, \mathbf{a}, \mathbf{c}, \mathbf{e}, \mathbf{d}, c, h, j, e$

Q_3: h, f, e, i, h

Eulerian Circuits in JGraphT

org.jgrapht.alg.cycle

All Classes

Packages

org.jgrapht
org.jgrapht.alg
org.jgrapht.alg.clique
org.jgrapht.alg.color
org.jgrapht.alg.connectivity
org.jgrapht.alg.cycle
org.jgrapht.alg.decomposition
org.jgrapht.alg.flow

HawickJamesSimpleCycles
HeavyPathDecomposition
HeavyPathLCAFinder
HeldKarpTSP
HelloJGraphT
HierholzerEulerianCycle
HopcroftKarpMaximumCardinalityBipartiteMat
HyperCubeGraphGenerator
ImmutableDoubleValueGraphAdapter
ImmutableGraphAdapter
ImmutableNetworkAdapter
ImmutableValueGraphAdapter
ImportException
IndependentSetAlgorithm
IndependentSetAlgorithm.IndependentSet
IndependentSetAlgorithm.IndependentSetImpl
IntegerComponentNameProvider
IntrusiveEdgesSpecifics
IsomorphicGraphMapping
IsomorphismInspector
IsomorphismUndecidableException
JGraphXAdapter
JGraphXAdapterDemo
JohnsonShortestPaths
JohnsonSimpleCycles
KleinbergSmallWorldGraphGenerator
KolmogorovMinimumWeightPerfectMatching
KolmogorovMinimumWeightPerfectMatching.D
KolmogorovMinimumWeightPerfectMatching.St
KosarajuStrongConnectivityInspector

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

org.jgrapht.alg.cycle

Class HierholzerEulerianCycle<V,E>

java.lang.Object
org.jgrapht.alg.cycle.HierholzerEulerianCycle<V,E>

Type Parameters:

`V` - the graph vertex type

`E` - the graph edge type

All Implemented Interfaces:

`EulerianCycleAlgorithm<V,E>`

```
public class HierholzerEulerianCycle<V,E>
extends Object
implements EulerianCycleAlgorithm<V,E>
```

An implementation of Hierholzer's algorithm for finding an Eulerian cycle in Eulerian graphs. The algorithm works with directed and undirected graphs which may contain loops and/or multiple (parallel) edges. The running time is linear, i.e. $O(|E|)$ where $|E|$ is the cardinality of the edge set of the graph.

See the Wikipedia article for details and references about Eulerian cycles and a short description of Hierholzer's algorithm for the construction of an Eulerian cycle. The original presentation of the algorithm dates back to 1873 and the following paper: Carl Hierholzer: Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Mathematische Annalen 6(1), 30–32, 1873.

Author:

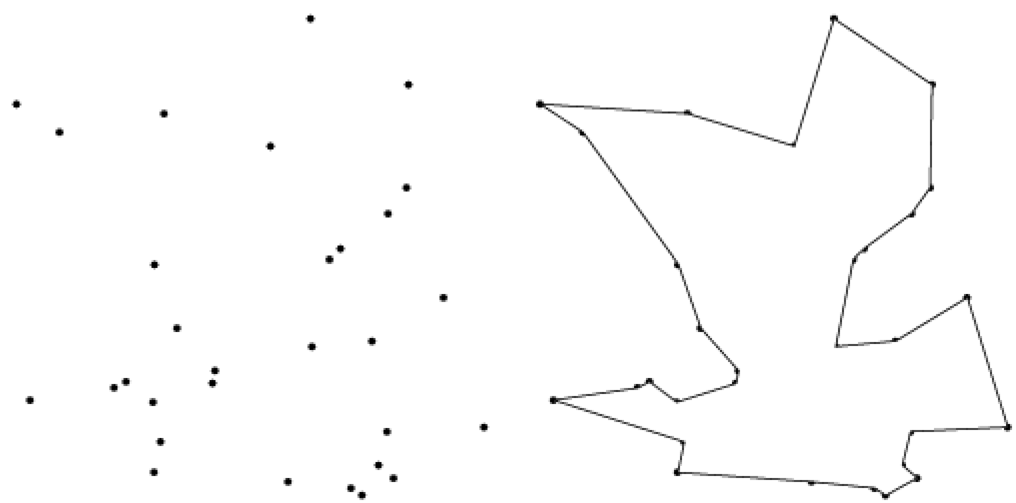
Dimitrios Michail

Constructor Summary

Constructors

Hamiltonian Cycles

- There are theorems to identify **whether** a graph is Hamiltonian (i.e., whether it contains at least one Hamiltonian Cycle)
- **Finding** such a cycle has **no** known efficient solution, in the general case
- Example: the **Traveling Salesman Problem** (TSP)



The Traveling Salesman Problem (TSP)

Weighted or unweighted

Given a collection of cities connected by roads

Find the shortest route that visits each city exactly once.

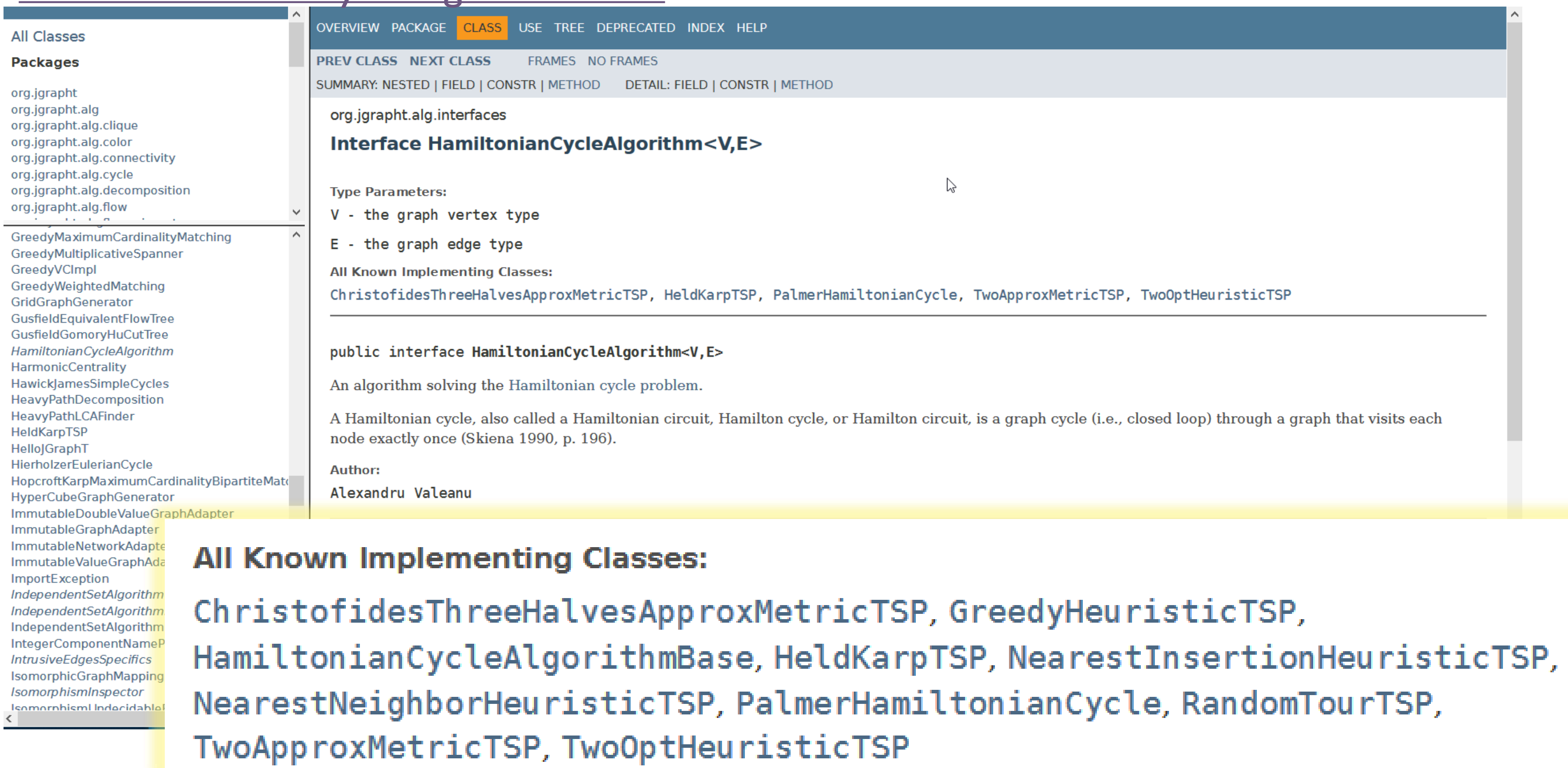
About TSP

- Most notorious NP-complete problem.
- Typically, it is solved with a backtracking algorithm:
  - The best tour found to date is saved.
  - The search backtracks unless the partial solution is cheaper than the cost of the best tour.

Hamiltonian Cycles in JGraphT

https://jgrapht.org/javadoc/org/jgrapht/alg/interfaces/HamiltonianCycleAlgorithm.html

org.jgrapht.alg.**interfaces**


OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

All Classes

Packages

org.jgrapht
org.jgrapht.alg
org.jgrapht.alg.clique
org.jgrapht.alg.color
org.jgrapht.alg.connectivity
org.jgrapht.alg.cycle
org.jgrapht.alg.decomposition
org.jgrapht.alg.flow

GreedyMaximumCardinalityMatching
GreedyMultiplicativeSpanner
GreedyVCImpl
GreedyWeightedMatching
GridGraphGenerator
GusfieldEquivalentFlowTree
GusfieldGomoryHuCutTree
HamiltonianCycleAlgorithm
HarmonicCentrality
HawickJamesSimpleCycles
HeavyPathDecomposition
HeavyPathLCAFinder
HeldKarpTSP
HelloJGraphT
HierholzerEulerianCycle
HopcroftKarpMaximumCardinalityBipartiteMat
HyperCubeGraphGenerator
ImmutableDoubleValueGraphAdapter
ImmutableGraphAdapter
ImmutableNetworkAdapte
ImmutableValueGraphAda
ImportException
IndependentSetAlgorithm
IndependentSetAlgorithm
IndependentSetAlgorithm
IntegerComponentNameP
IntrusiveEdgesSpecifics
IsomorphicGraphMapping
IsomorphismInspector

org.jgrapht.alg.interfaces

Interface HamiltonianCycleAlgorithm<V,E>

Type Parameters:

`V - the graph vertex type`

`E - the graph edge type`

All Known Implementing Classes:

`ChristofidesThreeHalvesApproxMetricTSP, HeldKarpTSP, PalmerHamiltonianCycle, TwoApproxMetricTSP, TwoOptHeuristicTSP`

`public interface HamiltonianCycleAlgorithm<V,E>`

An algorithm solving the Hamiltonian cycle problem.

A Hamiltonian cycle, also called a Hamiltonian circuit, Hamilton cycle, or Hamilton circuit, is a graph cycle (i.e., closed loop) through a graph that visits each node exactly once (Skiena 1990, p. 196).

Author:

`Alexandru Valeanu`

All Known Implementing Classes:

`ChristofidesThreeHalvesApproxMetricTSP, GreedyHeuristicTSP, HamiltonianCycleAlgorithmBase, HeldKarpTSP, NearestInsertionHeuristicTSP, NearestNeighborHeuristicTSP, PalmerHamiltonianCycle, RandomTourTSP, TwoApproxMetricTSP, TwoOptHeuristicTSP`

Limitations…

- No exact solution (Approximate algorithms)
  - Class TwoApproxMetricTSP<V,E>
  - Class ChristofidesThreeHalvesApproxMetricTSP<V,E>
  - Class TwoOptHeuristicTSP<V,E>
- Or complete under extra conditions
  - Class PalmerHamiltonianCycle<V,E>
- Or complete but O(2^N)
  - Class HeldKarpTSP<V,E>

ASSUMPTION: G is a metric graph.

1. Compute a minimum weight spanning tree T for G.
2. Perform a depth-first traversal of T starting from any node, and order the nodes of G as they were discovered in this traversal.

   $\Rightarrow$ a tour that is at most twice the optimal tour in G.

Class TwoApproxMetricTSP<V,E>

Resources

- http://mathworld.wolfram.com/
- http://en.wikipedia.org/wiki/Euler_cycle
- Mircea MARIN, Graph Theory and Combinatorics, Lectures 9 and 10, http://web.info.uvt.ro/~mmarin/

Licenza d'uso